E-learning Tool For Education in Cryptography

Paraschos Panagiotis

SID: 3301100003

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in ICT Systems

SEPTEMBER 2011

THESSALONIKI – GREECE

E-learning Tool For Education in Cryptography

Paraschos Panagiotis

SID: 3301100003

Supervisor: Assist. Prof. Vasilis Katos

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in ICT Systems

SEPTEMBER 2011

THESSALONIKI – GREECE

DISCLAIMER

This dissertation is submitted in part candidacy for the degree of Master of Science in ICT Systems, from the School of Science and Technology of the International Hellenic University, Thessaloniki, Greece. The views expressed in the dissertation are those of the author entirely and no endorsement of these views is implied by the said University or its staff.

This work has not been submitted either in whole or in part, for any other degree at this or any other university.

Signed: ..Panagiotis.Paraschos.....

Name: ..Panagiotis.Paraschos.....

Date: .....26/09/11.....

Abstract

This dissertation was written as a part of the MSc in ICT Systems at the International Hellenic University. Electronic communication is a vast piece of modern technology which is constantly evolving and creating more and more, problems. For instance, in e-commerce and more generally to electronic communications, one of the biggest issues is the security of transactions. The information security issues are contemporary problems faced by all systems and directly affect the IT infrastructure. The issues of information security in virtually all areas of applications such as: corporations - organizations, the government and individual privacy, can only be ensured with information security systems and cryptography.

The goal of this current dissertation is to develop in an existing application an e-learning tool for education in cryptography. In Chapter 1, security, cryptography and e-learning are introduced and defined. In Chapter 2, a presentation of the history of cryptography is made from the ancient times until today. In Chapter 3, the problems and difficulties that appeared over the years in the sector of cryptography are discussed. In Chapter 4, a detailed description of the development of the e-learning tool upon an existing instant messaging application is presented. Lastly, in Chapter 5, the results of the application are concluded, how helpful the application is in the education sector in understanding the different methods of cryptography and if it can be extended to other sectors e.g. enterprise (employee training against security threats).

Initially, I would like to thank my supervisor, Assistant Professor Vasilis Katos, for giving me the opportunity to work with the specific and interesting subject. His support, guidance and patience played a significant role to the development and completion of this thesis. Special thanks to Assistant Professor and PhD candidate, Konstantinos Mokos, for his valuable assistance and counseling on the development of the e-learning tool for education in cryptography. Finally, I am deeply grateful for the support of my family during the research and writing of this dissertation.

Paraschos Panagiotis

26/09/2011

Contents

ABSTRACT ..... **IV**

CONTENTS ..... **V**

1 INTRODUCTION **1**

1.1 SECURITY 1

1.2 CRYPTOGRAPHY 2

1.3 CRYPTANALYSIS 5

1.4 E-LEARNING 8

2 HISTORY OF CRYPTOGRAPHY **11**

2.1 ANCIENT TIMES 11

2.1.1 Origins *11*

2.1.2 The Bible *12*

2.1.3 The Greeks *12*

2.1.4 The Romans *14*

2.2 MIDDLE AGES – 20TH CENTURY 15

2.2.1 The Persian World *15*

2.2.2 The Europeans *16*

2.3 20TH CENTURY – WORLD WAR I & II 18

2.3.1 The Zimmermann Telegram *18*

2.3.2 The Jefferson Disk *20*

2.3.3 The Hebern Cipher Machines *21*

2.3.4 The Enigma Machine *22*

2.3.5 The Kryha Device *24*

2.3.6 M-94/CSP-488 Cylindrical Device *25*

2.3.7 TYPEX and SIGABA *26*

2.3.8 Urkryptografen - The Clock Cryptograph *26*

2.3.9 NEMA *28*

2.3.10 Operation VENONA *30*

2.4 ERA OF MODERN CRYPTOGRAPHY 31
2.4.1 DES – Data Encryption Standard 31
2.4.2 Diffie – Hellman Key Exchange 32
2.4.3 IDEA – International Data Encryption Algorithm 34
2.4.4 PGP – Pretty Good Privacy 34
2.4.5 RSA 34
2.4.6 RC5 36
2.4.7 RC6 37
2.4.8 SSL 38
2.4.9 S/MIME 38
2.4.10 AES – Advanced Encryption Standard 38
2.4.11 Blowfish 39
2.4.12 Quantum cryptography 41
2.4.13 Elliptical curve cryptography (ECC) 45
3 PROBLEMS IN CRYPTOGRAPHY 49
3.1 METHODS OF ENCRYPTION 50
3.2 SYMMETRIC CRYPTOGRAPHY 52
3.3 ASYMMETRIC CRYPTOGRAPHY 54
3.3.1 El Gamal 57
3.3.2 Diffie–Hellman key exchange algorithm 60
4 TURTLE CHAT V1.0 WITH VIGENERE 63
5 CONCLUSIONS 71
BIBLIOGRAPHY 75
APPENDIX 77

Figures

Figure 1.1: Security Objectives 1
Figure 1.2: Schemes of Cryptography 3
Figure 1.3: List of primitives 4
Figure 2.1: Disk of Faistos and its table of signs 13

Figure 2.2: Skytale 14
Figure 2.3: Polibios association of letters to numbers 14
Figure 2.4: Jefferson's Disk Cipher 21
Figure 2.5: Hebern Cipher Machine 21
Figure 2.6: Enigma Machine and its wiring diagram 22
Figure 2.7: Kryha Cipher Device 25
Figure 2.8: M-94/CSP-488 Cylindrical Device 25
Figure 2.9: TYPEX and SIGABA 26
Figure 2.10: Urkryptografen 27
Figure 2.11: NEMA 29
Figure 2.12: Ron Rivest Adi Shamir and Leonard Adleman 36
Figure 2.13: Blowfish encryption algorithm 41
Figure 3.1: Keyspaces 51
Figure 3.2: Symmetric Cryptosystem 53
Figure 3.3: Asymmetric Cryptosystem 55
Figure 3.4: Encryption Methods 56
Figure 3.5: The message is encrypted twice with the sender and receiver's keys. 57
Figure 4.1: Chat Server 63
Figure 4.2: Chat Client 64
Figure 4.3: Room and Emoticons tabs 64
Figure 4.4: Send Keyword window 65
Figure 4.5: The encrypted chat of Panos-George 66
Figure 4.6: The encrypted chat of Panos-George (Michael's side) 66
Figure 4.7: Users selection 67
Figure 4.8: Extracted Messages tab 67
Figure 4.9: Vigenere Application 68
Figure 4.10: Index of Coincidence 69
Figure 4.11: Extracted Messages tab (Results) 70
Figure 5.1: Application Questionnaire 71
Figure 5.2: Application Use 72
Figure 5.3: Application Steps 72
Figure 5.4: Functionality of Vigenere 73

1 Introduction

1.1 Security

What is security?

"Treat your password like your toothbrush. Don't let anybody else use it, and get a new one every six months." - Clifford Stoll

"If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology." - Bruce Schneier

Security is all about protecting assets (hardware, software, data) from threats (internal, external). The objective of security is to provide in any social sector confidentiality, integrity, availability (Figure 1.1: Security Objectives).

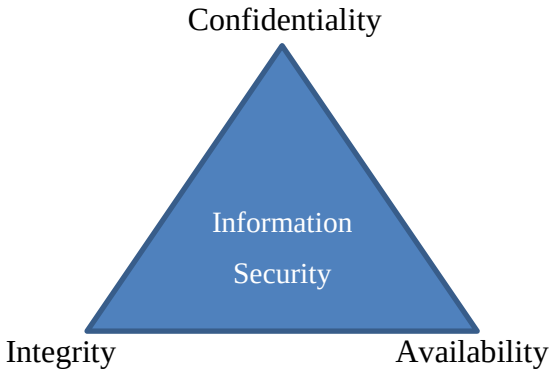


Figure 1.1: Security Objectives

- Confidentiality/Privacy: A service used to keep the content of information from all but those authorized to have it. Secrecy is a term synonymous with confidentiality and privacy. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible.

- Data Integrity: A service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes such things as insertion, deletion, and substitution.

- Availability: A service related to the availability of information and system resources to the legitimate users. To ensure availability, the service must include redundancy and fault tolerance system design.

1.2 Cryptography

During this time when the Internet provides essential communication between tens of millions of people and is being increasingly used as a tool for commerce, security becomes a tremendously important issue to deal with. There are many aspects to security and many applications, ranging from secure commerce and payments to private communications and protecting passwords. One essential aspect for secure communications is that of cryptography.

"Cryptography is about communication in the presence of adversaries" - Ron Rivest, 1990 (Figure 2.12 – left person)

"Cryptography is the science of writing in secret code and is an ancient art (Ancient times)" - Gary C. Kessler

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication [5].

In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet. Therefore in any application-to-application communication, some specific security requirements are included such as:

- Authentication: A service related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak). Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. For these reasons this aspect of cryptography is usually subdivided into two major classes: entity authentication and data origin authentication. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed).

- Non-repudiation: A service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary. A procedure involving a trusted third party is needed to resolve the dispute.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There is a number of basic cryptographic tools (primitives) used to provide information security. Examples of primitives include, in general, three types of cryptographic schemes typically used to accomplish these goals [2]: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography and hash functions, each of which is described below (Figure 1.2: Schemes of Cryptography).

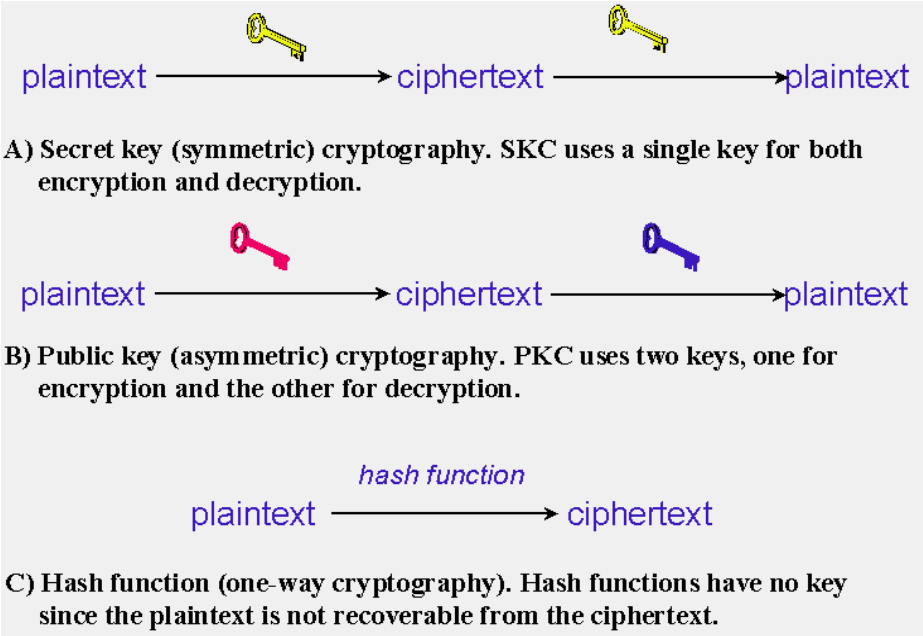


Figure 1.2: Schemes of Cryptography

Figure 1.3: List of primitives provides a schematic listing of the primitives considered and how they relate:

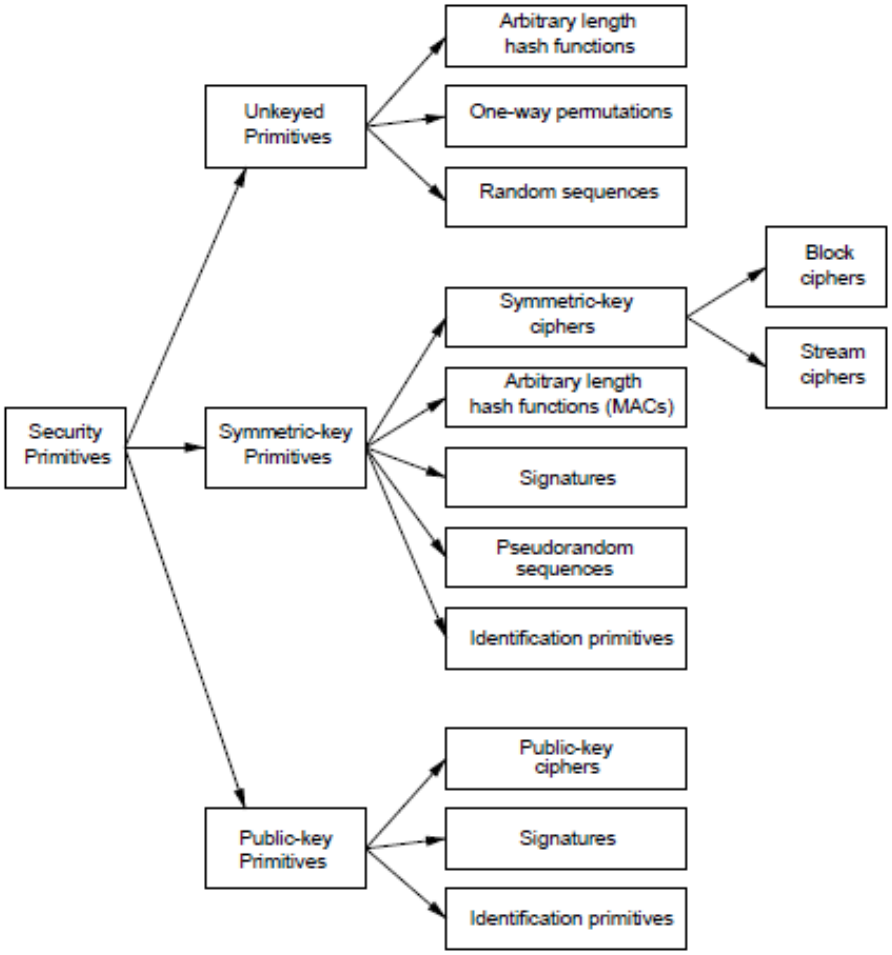


Figure 1.3: List of primitives

These primitives should be evaluated with respect to various criteria such as:

(1) Level of Security: This is usually difficult to quantify. Often it is given in terms of the number of operations required (using the best methods currently known) to defeat the intended objective. Typically the level of security is defined by an upper bound on the amount of work necessary to defeat the objective. This is sometimes called the work factor.

(2) Functionality: Primitives will need to be combined to meet various information security objectives.

(3) Methods of Operation: Primitives, when applied in various ways and with various inputs, will typically exhibit different characteristics: thus, one primitive could provide very different functionality depending on its mode of operation or usage.

(4) Performance: This refers to the efficiency of a primitive in a particular mode of operation. For example, an encryption algorithm may be rated by the number of bits per second which it can encrypt.
(5) Ease of Implementation: This refers to the difficulty of realizing the primitive in a practical instantiation. This might include the complexity of implementing the primitive in either a software or hardware environment.

The relative importance of various criteria is very much dependent on the application and resources available. For example, in an environment where computing power is limited one may have to trade off a very high level of security for better performance of the system as a whole. More detailed discussion in Chapter 3.

1.3 Cryptanalysis

Cryptanalysis is the study of attacks against cryptographic schemes in order to reveal their possible weaknesses [7]. It is the art and science of recovering the plaintext of a message without access to the key. Successful cryptanalysis may recover the plaintext or the key. The loss of a key through non-cryptanalytic means is called a *compromise* and an attempted cryptanalysis is called an *attack*.

The study of a cryptographic system for the purpose of finding weaknesses in the system and breaking the code used to encrypt the data without knowing the code's key. There are 4 basic steps in a typical cryptanalysis:

(1) Determine the language being used
(2) Determine the system being used: this can be a time-consuming stage in the process and involves counting character frequency, searching for repeated patterns and performing statistical tests.
(3) Reconstruct the system's specific keys
(4) Reconstruction of the plain text: this step typically takes place at the same time as the keys are reconstructed.

Cryptanalysis often is used by governments in military and diplomatic surveillance, by enterprises in testing the strength of security procedures and by malicious hackers in exploiting weaknesses in Web sites. The attacks performed on block ciphers are based on a threat model called Kerckhoff's assumption whereby the attacker knows all the de-

tails of the cipher except the secret key. Based on this assumption, attacks are classified according to adversary's capabilities:

- Ciphertext-only: a passive attack whereby an adversary is assumed to possess a set of ciphertext to recover the plaintext of secret key.

- Known plaintext: a passive attack whereby an adversary knows some plaintext-ciphertext pairs to find the unknown portion of the plaintext or secret key.

- Chosen plaintext: an active attack whereby an adversary has the ability to choose plaintexts and obtained the corresponding ciphertexts.

- Chosen ciphertext: an active attack whereby an adversary has the ability to choose ciphertexts and obtained the corresponding plaintexts.

- Related-key: an adversary is assumed to choose some relation between the secret key used in encryption and decryption, but not the value of the key. A cipher vulnerable to a ciphertext-only attack is considered weak while a cipher which is secure against a chosen ciphertext attack is deemed secure [8].

Attacks can also be classified based on the required effort that the adversary needs to solve:

- Brute-force: in this attack, every possible value of the key is tried until the plaintext is recognized. The attack is also called an exhaustive key search.

- Shortcut attacks: an attack that has the complexity less than that of brute-force.

- Side channel attacks: an attack based on information obtained from physical implementation of a cipher.

- Fault analysis: an attack based on systematically inducing faults in particular hardware components used to protect or to store keys or algorithms.

Attacking a cipher does not necessarily mean to find the secret key. Based on recovered information, Knudsen described a hierarchical classification of the outcomes of an attack:

- Total break: an adversary recovers the secret key.

- Global deduction: an adversary discovers an algorithm which is functionally equivalent to the encryption and decryption process without knowledge of the key.

- Instance (local) deduction: an adversary recovers the plaintext (or ciphertext) from an intercepted ciphertext (or plaintext) which was not acquired from the legitimate sender.

- Information deduction: an adversary obtains information about the secret key, plaintexts or ciphertexts which was not directly came from the legitimate sender and which was not known before the attack.

- Distinguishing algorithm: an attacker is able to tell whether the attacked cipher is a randomly chosen permutation or one of the 2k permutations indicated by the secret key.

The success of an attack can be measured by its complexity as follows:

- Data complexity: the amount of data (plaintexts or ciphertexts) required to execute the attack under a certain threat model.

- Time complexity: the number of encryption / decryption needed to perform the attack.

- Memory complexity: the amount of memory needed to hold all data during the attack.

- Success probability: measures the frequency of a successful attack when repeated in a number of times.

A range of interesting mathematical problems arise in attempting to cryptanalysis pseudorandom number generators. Basically, there are two kinds of cryptanalysis problems: reconstruction problems, which attempt to reconstruct the parameters of the generator from some output of the generator, and predicting problems, which attempts to predict future output of the generator from some observed output.

In recent years, methods, which are based on lattice basis reduction or juts lattice reduction or the so called LLL technique, have been used repeatedly for the cryptanalytic attack of various cryptosystems. Lattice reduction techniques seem inherently linear. The general idea of this technique is to relate the nonlinear problem to a lattice problem by building a lattice from the nonlinear equation and translate the problem to finding a vector with smallest Euclidean norm possible in the lattice, the Shortest Vector Problem (SVP).

1.4 E-learning

In a world that is constantly changing, the enhancement of knowledge plays an increasing role in the acquisition of useful skills. The training/education does not stop, but continues in other stages of our life.

Since the new economy and technology require more and more people to gain new knowledge and skills in a timely and effective method, the evolution of technologies and computer networks provide different tools to support learning with a personalized, flexible, portable and on demand manner. These radical changes, in the needs of learning and technology, fueling a transition to modern learning in the Internet era, usually referred to as E-learning.

In the midst of this transition, companies, government organizations and educational institutions need to understand the phenomenon of e-learning and they need to take strategic decisions on how to adopt techniques in e-learning environments.

The Internet and multimedia technologies are reshaping the way knowledge is transmitted and e-learning is an effective alternative to traditional learning. Several definitions and terms have occasionally been proposed: e-learning, digital learning, distance learn-

ing are similar terms all of which refer to a modern education effort designed primarily to exploit Web-based technology.

So e-learning could be defined as based technology learning in which learning materials transferred electronically to remote students through a computer network [3]. Another relevant definition states e-learning as a technology-based education that includes training, Web-based training us get with the help of computers.

Compared with time communication between instructor and student, there are two forms of e-learning, modern and asynchronous:

- Modern distance learning: In the modern distance learning course becomes a designated time and place. The trainees monitor online the instructor and may participate in all procedures of the course in real time, i.e. essentially is the transfer of the traditional way electronic teaching conditions except that the instructor and called minicabs can separated by hundreds or thousands of kilometers.

Asynchronous distance learning: Trainees have access (usually through Internet) to multiple sources of information (teaching materials, video, subjects self, etc.). This material is accessible on any time and day and sometimes it can be transferred to the computer. The use of this material is usually with multimedia technology at a time suited to the trainees. This time is not necessary to coincide with the timing of other trainees who "participate" in this virtual class. Throughout the above procedure, the presence of the instructor is not required while trainees are online. This procedure is used mainly by the open universities and other providers training to specific groups, e.g. employees.

2 History of cryptography

This chapter describes the history of cryptography [9], from ancient times to our days which is divided into four phases:

(1) Ancient civilizations
(2) The first part of the twentieth century, with relatively simple algorithms, that were designed and implemented by hand.
(3) Extensive use of encrypting electro-mechanical machines, around the period of the Second World War.
(4) Ever more pervasive use of computers, about in the last fifty years, supported by solid mathematical basis.

The word "cryptography" comes from ancient Greek and is the union of two words:

- krypto(κρύπτω) = hidden
- grafo(γράφω) = write

2.1 Ancient times

Cryptography was used during ancient times mostly in three kinds of contexts:

- Private Communications
- Art and Religion
- Military and Diplomatic Use

2.1.1 Origins

In the Egyptian town of Menet Khufu , about 4000 years ago, hieroglyphic figures were written on the tomb of the nobleman Khnumhotep II in such a way to confuse those who were not familiar with this kind of writing. This is registered as the oldest known encrypted text.

2.1.2 The Bible

The Bible refers to three encryption techniques:

- Atbash: This technique reversed the alphabet by substituting the last letter (tav) with the first letter (aleph), the one before last letter (shin) with the second letter (beth), etc.
- Albam: This technique was based on the division of the alphabet in two parts where afterwards the Atbash technique was used.
- Atbah: This technique was based on an equation which contains the original letter (e.g. a) of the alphabet and its substitution letter (e.g. k):

  a + k = 10 if a is one of the first 9 letters
  a + k = 45 if a is one of the last 8 letters
  a + k = 28 otherwise

For example, in the Book of Jeremiah, "Sheshakh" is Atbash for "Bavel" (Babylon). Other examples based on Atbash technique are: "hob"="sly", "hold"="slow", "holy"="slob", "horn"="slim", "zoo"="all", "irk"="rip", "low"="old", "glow"="told" and "grog"="tilt".

2.1.3 The Greeks

The Cretan pictographic (or hieroglyphic) writing, has not revealed its source, it is known however that this is not a script that uses images as signs, but vocal writing, which is running at about two hundred gems and coexisted with the Linear A, both in time and locally, as is clear from the excavations at the palace of Malia, Crete. It appears in the Disk of Phaistos (Figure 2.1: Disk of Faistos and its table of signs) was discovered in 1908 in southern Crete. It is a circular plate, dated around 1700 BC and has writing in the form of two spirals. The symbols are handmade, but have been developed with the help of a variety of stamps, making the disk as the oldest example of typesetting. There are other similar findings, so the decryption is based on very limited information. Until now, there has been deciphered and remains the most mysterious ancient

European script.

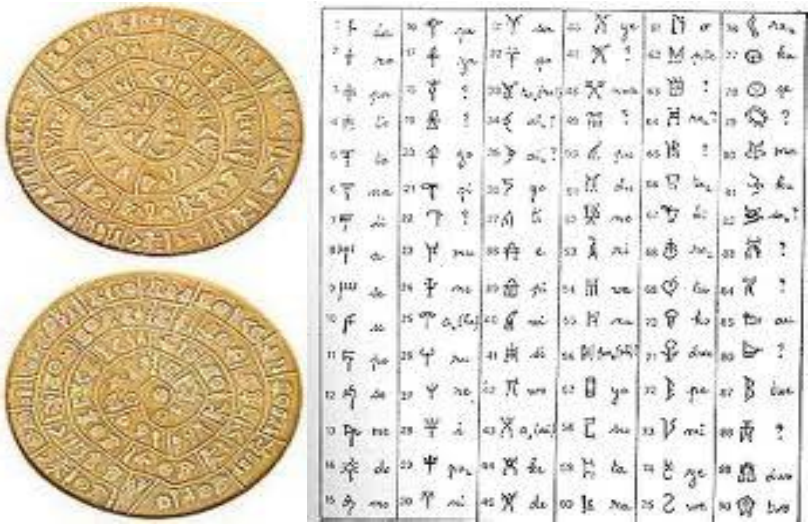


Figure 2.1: Disk of Faistos and its table of signs

Herodotus describes how transferred encrypted messages from the messengers. The Jews were encrypting some words on the parchment.

One of the earliest references to cryptography is in Omiro's Iliada, which states that an encrypted message sent by Bellerophon. In Greece (Iliada, Rapsody f), the Bellerophon went to Lycia carrying hidden text message. Furthermore, in the History of Herodotus, Istiaios of Miletus, a trusted slave of Darius, was used for sending a message to his written head. Herodotus also mentions that the Demaratos, in exile in Susa in Persia, warned the Spartans to plan the invasion of Xerxes, sending mail covered with wax. Gorgo, Leonidas' wife, told her the message and Xerxes lost the advantage of surprise.

In 500BC the Spartans invented a cryptographic device to send and receive secret messages. This was a cylinder called Scytale (Figure 2.2: Skytale). A narrow strip of parchment or leather was wrapped around the Scytale and the message was written across it. If someone unwrapped it, then that person would read on the strip a sequence of letters without having a meaning. The only way to read the message correctly was the strip to be wrapped around the Scytale of exactly the same diameter. The code, produced by unwinding the tape, was a transposition cipher, that is, one where the letters remain the same but the order is changed. This technique is used until today as a standard for many popular modern techniques.

Figure 2.2: Skytale

Another encrypted method was the Additive/Substitution cipher. The antique cipher of the Greek historian Polibios used a table, with rows and columns, to associate a letter to a pair of numbers (Figure 2.3).

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Figure 2.3: Polibios association of letters to numbers

Thus, the plaintext: "LET NONE ENTER IGNORANT OF GEOMETRY" gives ciphertext:

"31 15 44 33 34 33 15 15 33 44 15 42 24 22 3334 42 11 33 44 34 21 22 15 34 32 15 44 42 54".

2.1.4 The Romans

Julius Caesar used a substitution technique, the famous Caesar Cipher, which was first described by the Greek writer Polibios. The encoding was based on substituting the letter in the text with the one which was three positions to the right (considering the 26 letters of the English alphabet):

$m = (k + 3) \bmod 26$

A became D, B became E etc. An example of Caesar Cipher is:

"Omnia Gallia estdivisa in partestres"

"Rpqld Jdoold hvw glylvd lq sduwhv wuhv"

2.2 Middle Ages – 20th Century

From 500 – 1400 AD the "dark age of cryptography" began in Europe: During this period cryptography was considered as black magic art and a lot of knowledge was lost. By contrast cryptography flourished in the Persian world.

2.2.1 The Persian World

The first who understood well the principles of cryptography and cryptanalysis were the Arabs. Constructed and used replacement algorithms and displacement and found use of the frequency of characters and possibilities in cryptanalysis. So in 1412 the Al-Kalka-Shandi included a description of several cryptographic systems in the encyclopaedia Subh al-a'sha and gave clear instructions and examples for the cryptanalysis of encrypted texts using the frequency of characters.

Even in India cryptography was developed and practiced, for espionage services (Arthasastra), in religious context (Latilavistara, writings about Buddah) and in Kamasutra: among the 64 arts women should know, the number 45 was "mlecchita-vikalpa", the art of secret writing, advocated in order to help women conceal the details of their liaisons.

Another suggested technique is the mating of the letters of the alphabet at random and then each letter is replaced with the partner.

The most important representative of Arab cryptology is the professor of the ninth century Al-Kindi, who wrote over 290 books Mathematics - Linguistics - Astrology, Medicine and Music. In 1987, the Ottoman Archives Suleymaniye of Istanbul, discovered the treatise "On decoding encrypted messages" and says the following: "One way to read an encrypted text if you know the language, is to find a different plaintext in the same language covering approximately one sheet and then measure the frequency of occurrence of each letter. The most frequently occurring letter is called first, the next second, by characterizing all the letters of the plaintext. Then classify the same way the symbols of the desired encrypted text we wanted to decipher. We find the most frequently occurring symbol and replace it with the first letter, second in frequency symbol with the second letter, the third with the third letter and so on".

2.2.2 The Europeans

The European cryptology has its roots in the Middle Ages, developed by the Pope and the Italian city-states, but most systems were based on mere substitution of letters of the alphabet (as in Caesar's algorithm). The first algorithm is based on replacing the vowels. The first European manual cryptography (The Nomenclature Code - 1379) was a collection of algorithms by Gabriele de Lavinde of Parma, for the Pope. Due to its simplicity this nomenclature code was used over the next 450 years especially in diplomatic circles.

Coming to more recent times, an Italian Leon Battista Alberti, "father of western cryptography", described in 1466 the construction of a cipher disk, founding the concept of Polyalphabetic ciphers. This was the most important advance in cryptography in at that period.

In 1499 AD the German abbas Johanes Trithemius wrote Stenography, a book of communication with spirits. The book came to the Index Librorum Prohibitorum, but and in a similar list of Protestants. The third volume contained tables of numbers that supposedly represented codes of communication with spirits. 150 years later, in 1676 AD Heidel, an attorney of Mainz, claimed that he broke the code of Trithemius, but wrote the solution in code. In 1996 AD the codes of Trithemius and Heidel were broken which unfortunately were silly spells. Much more interesting was the Trithemiu's Polygraphy (1518 AD), the first printed book of Cryptography. It contained a table of letters where the first line contained the 26 letters of the Latin alphabet and the remaining 26 rows were cyclic permutations of the first row by 0, 1, 2, ..., 25 posts. The columns represented plaintext, ciphertext was at the intersection of row and column.

For example consider the following plaintext: DEUS

DEUS is DFWV

The first line is below the letter D on the first line, again the letter D.
The third line is below the E line first letter F.
The 4th »» »U» »» W.
The fifth »» »S» »» V.
Therefore: DEUS → DFWV

Sir Francis Bacon in 1563 described an algorithm which today bears his name. It was a coding algorithm using 5 bits. In this algorithm, developed a method of steganography using a change in the form of characters conveyed every bit of coding.

In the same year, Giovanni Battista Porta published "De Furtivis Literarum Notis", a book describing encryption methods and cryptanalysis. In it, the first digraph substitution cipher is mentioned.

In 1577, Van Marnix, the Flemish code breaker, decrypted a Spanish letter, which contained the plan, to conquer England by sending Spanish troops from the Netherlands.

In 1586, Sir Francis Walsingham, Secretary of State, proved that the Queen of Scotland, Mary Stuart was participating in the conspiracy to kill her cousin Elizabeth, Queen of England, by deciphering her messages, which were hidden inside beer barrels, with Sir Babington (The Babington plot). The conspiracy was based on using substituting and meaningless symbols for confusion. Mary Stuart was sentenced to death.

In 1586, the French diplomat Blaise de Vigenere, published his description of a polyalphabetic cipher before the court of Henry III of France. It was called "le chiffre indechiffrable" (The Unbreakable Cipher). It was practically impossible to be broken for almost four centuries until a vulnerability to the statistical attack was discovered in the later years.

The Vigenere cipher consists of several Caesar ciphers in sequence with different shift values. To encipher, a table of alphabets was used, termed a "tabula recta", "Vigenere square", or "Vigenere table".

For example, let the plaintext to be encrypted is: HELLOWORLD

The sender chooses a keyword and repeats it until it matches the length of the plaintext, for example, the keyword "STAR": STARSTARST

The first letter of the plaintext, H, is enciphered using the alphabet in row S, which is the first letter of the key. This is done with the help of the Vigenere square and so on with all the letters. At the end we obtain:

Plaintext: HELLOWORLD

Key: STARSTARST

Ciphertext: ZXLCGPOIDW

In 1600, Cardinal Richelieu used a card with holes to write an encrypted message. He was filling in the blanks with words that look like a normal letter. To decode the letter, the card, with which the letter was written, was needed.

Antoine Rissignol became the first full-time cryptanalyst being employed after his decryption of a hostile encoded message, which terminated the siege of Realmont by the Huguenots (1628). Since then, cryptanalysts have always been a fixed element in military organizations.

In 1641, the first secretary of the Royal Society of London John Wilkins introduced the words Cryptography and Cryptology in English.

In 1700, the Russian tsar used a big code table of 2000-3000 syllables and words to encrypt his messages.

In 1854, Charles Babbage, developed the method of statistical analysis by which he successfully decrypted Vigenere encrypted messages. Because of the fact that he was working for the British by decrypting Vigenere messages sent in the Crimea, it was covered until the twentieth century.

Same period the English physicist, Charles Wheatstone, invented a cipher which worked with a 5*5 matrix which was published in diplomatic and military circles by Lord Lyon Playfair, Baron of St. Andrews (The Playfair Cipher).

So the honor of developing the statistical attack technique and cracking Vigenere went to the Prussian Kasiski in 1863, named as the Kasiski Test.

In 1883, Auguste Kerckhoff von Nieuwendhoff presented "La Cryptographie militaire", a cryptographic milestone in the telegraph era. It contained the "principle of Kerckhoff", which requires to base the security of an encryption method only on the privacy of the key and not of the algorithm.

In 1891, the French major Etienne Bazeries invented the Bazeries cylinder, similar in principle to the wheel cipher. He published the design in 1901, after the French Army rejected it.

2.3 20th Century – World War I & II

2.3.1 The Zimmermann Telegram

In World War I (WWI), at the early days, the British cable ship Telconia located and cut Germany’s trans-Atlantic cables, forcing them to send all their international traffic

via Sweden or American-owned cables. Soon all German traffic ran through the UK and was routinely routed to Room 40, the Royal Navy's cipher organization. About January 16, 1917, William Montgomery and Nigel De Gray, cpyptanalysts of Room 40, were given a message encrypted in the German Foreign Office code, a Book Cipher number 0075. After a successful decryption they discovered that it was sent by the German Foreign Minister Zimmermann to the Mexican President via the German Embassies in Washington and Mexico City. The Zimmermann telegram informed the President of Mexico that Germany proposed to start unrestricted submarine warfare in February and that he should, with German support, attack the US and also convinces the Japanese to do the same. The message was presented to US President Wilson and on April 2, 1917 the US declared war on Germany and by 1918 Germany had been defeated. To obscure the source of the original intercept and to point to a security breach in Mexico, Room 40, using a British agent in Mexico, obtained a copy of the edited US/Mexico version of the original Zimmermann cable. This of course differed from the original for procedural reasons associated with its re-transmission from Washington to Mexico City. The decrypt of this was the copy released to the US press its use obscuring the fact that the British had obtained their information not from espionage in Mexico but decryption of the original telegram in London. The Germans spotted the difference and took the bait. The deception was successful and the source was safe.

Towards the end of WWI, Major Joseph Mauborgne, head of cryptographic research for the US Army introduced the concept of a code based on truly Random keys. This would take the form of two identical pads printed with lines of randomly generated letters. Using the Vigenere technique, each page is to be used to encrypt and decrypt one message and then destroyed. The setback of the Vigenere square was the repetition of the key. This new technique injected the same randomness into the ciphertext as was contained in the key and there was therefore no usable pattern or structure within the message. Attacks seeking to exploit these weaknesses such as the Babbage and Kasiski tests, would fail.

A key length of as little as 21 letters meant that a Key Exhaustion attack, the cryptographic equivalent of Custer's last stand, would require the testing of 500×10^{27} keys and even then multiple decrypts may all appear plausible. This method is still in use today, called the One Time Letter Pad (OTLP) and is used for encrypting the most secret of communications. OTLP is still the only "admitted" system to provide the perfect secrecy. The First World War showed the importance of cryptography on the battlefield,

and the danger of weak encryption and spawned the development of the "unbreakable" OTLP.

The Germans near the end of World War I, around 1918 were using the **ADFGVX** cipher which was broken by the French cryptanalyst, Lieutenant Georges Painvin. This was a 2-step cipher which first performed a substitution (each letter was substitutes by a bigram through a keyed array), and then the bigrams were fractionated in columns and the columns trans-positioned.

In 1929, Lester S. Hill invented the Hill cipher, a polygraphic substitution cipher based on linear algebra (and so matrix theory). It was the first polygraphic cipher in which it was practical to operate on more than three symbols at once. Instead of substituting one letter for another letter, a polygraphic cipher performs substitutions with two or more groups of letters. This has the benefit of masking the frequency distribution of letters, making this way much more difficult the frequency analysis attacks.

As time passed, in the history of cryptography, people began to understand that

(1) complex ciphering could be obtained by concatenating a certain number of simpler ciphering phases.
(2) ciphering operations could be made no more by hand but with the help of machines, at first simple and then more complex ones.

2.3.2 The Jefferson Disk

At the end of the eighteenth century a rotation ciphering machine, The Jefferson Disk, or Wheel Cipher as the third US President Thomas Jefferson named it, was a cipher system using 26 wheels, each with the letters of the alphabet arranged randomly around them (Figure 2.4). Once the order of wheels along the axis has been devised, the user can rotate each wheel up and down until a desired message is spelled out in one row. Then the user can copy a row of text on the wheels other than the one that contains the message. The receiver simply has to put the discs in the agreed-upon order, spell out the encrypted message by rotating the wheels and then look around the rows until he sees the plaintext message.

Figure 2.4: Jefferson's Disk Cipher

2.3.3 The Hebern Cipher Machines

In 1910, Edward H. Hebern began to develop encryption devices in the United States. In the early 1920's, he engineered what was termed the electric code machine which used a single rotor. In 1924 he designed 3-rotor and 5-rotor machines as part of bids for the U.S. Navy. Unlike the Enigma (Figure 2.6), the Hebern machines (Figure 2.5) used rotors whose internal wiring could be readily changed. The rotors could also be inserted in either their standard or reverse rotation. However, his devices were suspect to crypt-analytic weaknesses as indicated by William Friedman.

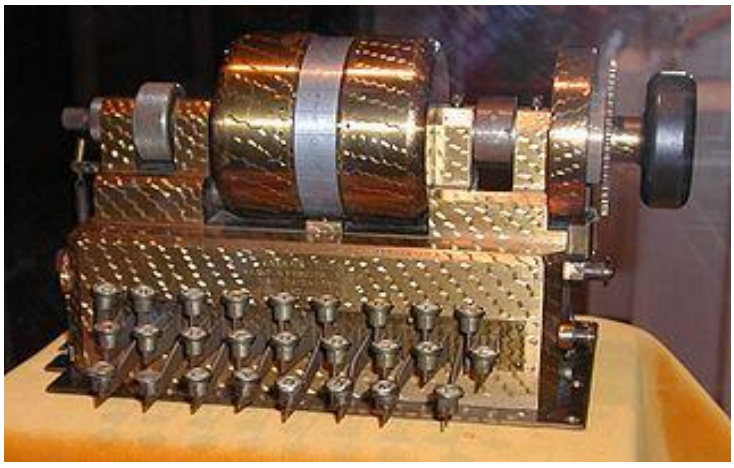


Figure 2.5: Hebern Cipher Machine

2.3.4 The Enigma Machine

The Second World War (WWII) became a defining moment in the history of cryptography and placed it squarely at the center of military and political strategy from that time to the present day. In the WWII an important role was played by another rotation device, the Enigma Machine (Figure 2.6), an encryption and decryption machine used by the German army to communicate in a secure way. When the Allies managed to break the code, it changed the entire flow of the war against the Axis. This is one out of many examples how much impact cryptography has throughout history. Enigma was a family of related electro-mechanical rotor machines. The mechanical mechanism consisted of a keyboard, a set of rotating disks called rotors, arranged adjacently along a spindle and a stepping mechanism to turn one or more of the rotors with each key press. The mechanical parts acted in such a way as to form a varying electrical circuit: the actual encipherment of a letter was performed electrically. Enigma wiring diagram with arrows and the numbers 1 to 9 showing how current flows from key depression to a lamp being lit. The A key is encoded to the D lamp. D yields A, but A never yields A: this property was due to a patented feature unique to the Enigmas and could be exploited by cryptanalysts in some situations:

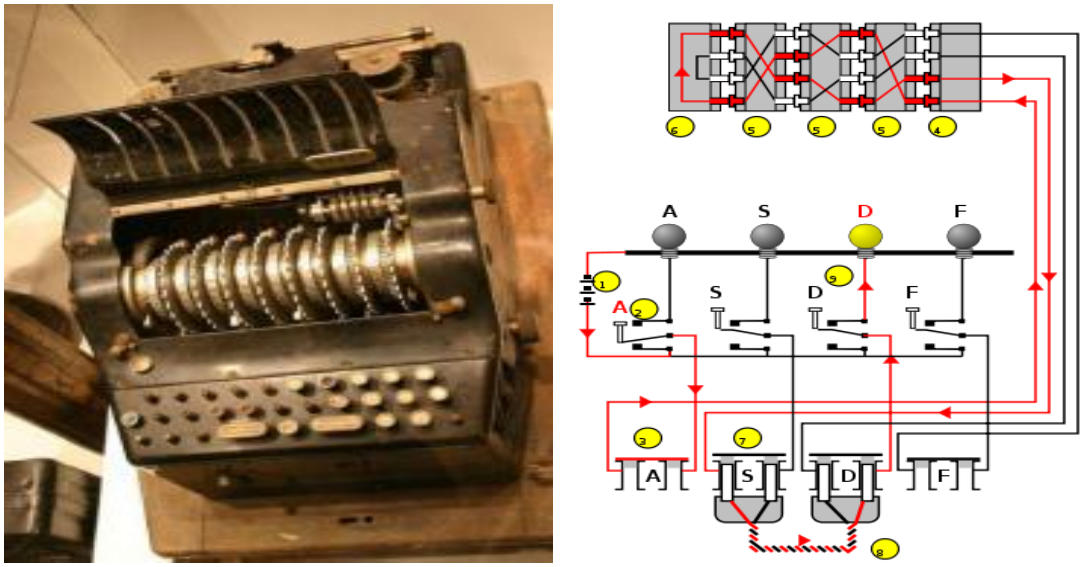


Figure 2.6: Enigma Machine and its wiring diagram

The Enigma Machine was invented from patents by the German Arthur Scherbius in 1919. After many trials and several improvements, the machine was adopted by the German Navy in 1926, the Army in 1928, the Air Force in 1935 and in other sections of the German government. In 1926, Winston Churchill published his book "World Cri-

sis", in which he refers about British decryption of German letters during World War I. Along with the achievements of Room 40, the official war history of British Royal Navy and other publications, had as a result Scherbius's Enigma to become very famous and to be used extensively. The evolved encoding and decoding we use nowadays is due to those events. The present slogan is "never give a sucker an even chance". The decryption the Allies were using during WWI, for almost all German cipher traffic, was insufficient by early 1930 by the time Enigma was introduced into service. The Enigma is an offline cipher system. Enigma's functionality was manual where each plaintext letter was typed on the Keyboard (Tastatur) and the resultant ciphertext letter appeared illuminated on the Lamp Board (Gluhlampenfeld). This letter was transcribed on a message pad and the procedure repeated until the message was complete. This ciphertext message was then transmitted by radio using Morse code. Decryption followed the same procedure with the ciphertext letter typed in and the plaintext equivalent displayed on the lamp board. This had as a result to provide high-grade ciphertext from the input of plaintext and the reverse.

The British cryptanalysts made several tries to break Enigma but without a positive result. The first step in breaking the Enigma machine was a result of the good old fashioned HUMan INTellingence (HUMINT) and became as follows: In late 1931 a French secret service agent managed to photograph two Enigma instruction manuals by bribing a German public servant. Those manuals contained sufficient information of describing the internal wiring of the machine. Between Britain and Poland was a framework agreement on sharing information. Under this agreement, British gave copies of the manuals to Poland and thus Marian Rejewski, a young mathematician began working on this issue.

The Poles, within 18 months, managed to decrypt Enigma's functionality without revealing their success and were able, by manual means, to recover a "day key" and read Enigma messsages. Meanwhile the Germans changed the transmission technique. This had as a result the British to develop a mechanical device, which contained six individual machines in total, so they can have the ability to extract the key. This was the first of many "Bombs" (keyword for code breaking) at Bletchley Park of Bakinchamsair. This success continued until 1938 when two new scrambler wheels (4&5) and 4 more plug board (Stekerbrett) connections were added. On 24.07.1939 British and French cryptanalysts arrived at the Polish Biuro Szyfrow to be told of Rejewski's success about code breaking the new Enigma and to be given one each with the accompanying blue prints.

The code breaking discovery by the Poles along with the difficult and complex decryption task the British had to undertake, played a significant role in changing the flow of World War II in favor of the Allies. This was a result of the weaknesses which were discovered both by the Poles and the British. The Enigma was, in terms of its internal architecture, a swapping machine and, as such, two machines set the same would give the same result. Key X to get C or Key C to get X. This meant that once the "setting" or "day key" was found, all messages using that setting could be decrypted. There was no internal dynamic update of the key based on the message traffic or any other variable. In addition keying X would not give X. This latter weakness was used to great effect when applying "cribs", "ordered or known text that provide clues to breaking a cypher" such as Dear Sir, or Heil Hitler! Decrypts of Enigma traffic produced many results for the Allies. Although the Allies were informed of the German airborne landing on Crete, the allied forces were defeated because they were forbidden to pre-empt the attack so their informants would not be caught by the enemy. Despite a recent (2000) American movie which attempted to rewrite history, British work on the decryption of the German naval Enigma which had more rotors than a "normal" machine and associated military operations designed to capture code books, led directly to the defeat of the U-boat offensive in the Atlantic and the saving of countless lives on both sides.

Of all the historical figures who contributed to the development of cryptography by William Frederick Friedman, founder of Riverbank Laboratories, Cryptology in American government and driver breaking the code of the Japanese Purple Machine during the Second World War, considered the father of American cryptanalysis. In 1918 he wrote the book "The Index of Coincidence and Its Applications in Cryptography" still regarded by many as the most important textbook on cryptography in the 20th century.

2.3.5 The Kryha Device

In the period of 1920, the Ukrainian Alexander Von Kryha invented the Kryha (Figure 2.7). It had a fixed semi-circle of letters against which was juxtaposed a cipher disk with gears controlling the number of spaces it turned. Both sequences of letters could be mixed alphabets. A handle served to rotate a powerful clock spring which drove the rotating platform on which the inner cipher disk was mounted. The key or setting included another wheel whose segments (open or closed) controlled the rotational stepping of the inner cipher disk.

Although Kryha was an interesting encoding device, its mechanism was similar to a simple polyalphabetic cipher with a single cipher alphabet and a period of a few hundred letters which was presented by American cryptanalysts was solvable in a matter of hours.

Figure 2.7: Kryha Cipher Device

2.3.6 M-94/CSP-488 Cylindrical Device

The U.S. Army developed in 1922 it's M-94 (Figure 2.8) which included 25 4cm in diameter aluminum disks on a 10.5cm long axis. M-94 was in use World War II. It was also used in other areas such as the Radio Intelligence Division of the Federal Communications Commission in the United States and the Coast Guard. The U.S. Navy used a similar device called CSP-488.

Figure 2.8: M-94/CSP-488 Cylindrical Device

2.3.7 TYPEX and SIGABA

During the period 1926-1935 the study of commercially available cryptographic equipment such as the Hebern, Kryha and Enigma, had as a result a British interdepartmental committee to adopt an ENIGMA-type cipher machine called the TYPEX (Figure 2.9).

The Mark III model of the TYPEX had five interchangeable rotors and a design which allowed irregular rotor movement. It was electrically powered and output was via a paper tape printer located in the back of the machine. There have been several versions of the TYPEX which were used in the British Army (most of them in Royal Air Force) and in Canada.

The U.S. Army and Navy adopted a similar device in the late 1930's, called the Electric Cipher Machine (ECM) Mark II. The U.S. Army referred to this device as the SIGABA (Figure 2.9). It was used exclusively by the Americans during World War II and beyond.

In order for the ECM and TYPEX to intercommunicate, TYPEX compatible rotor cages were designed for the ECM and gave rise to the Combined Cipher Machine (CCM).

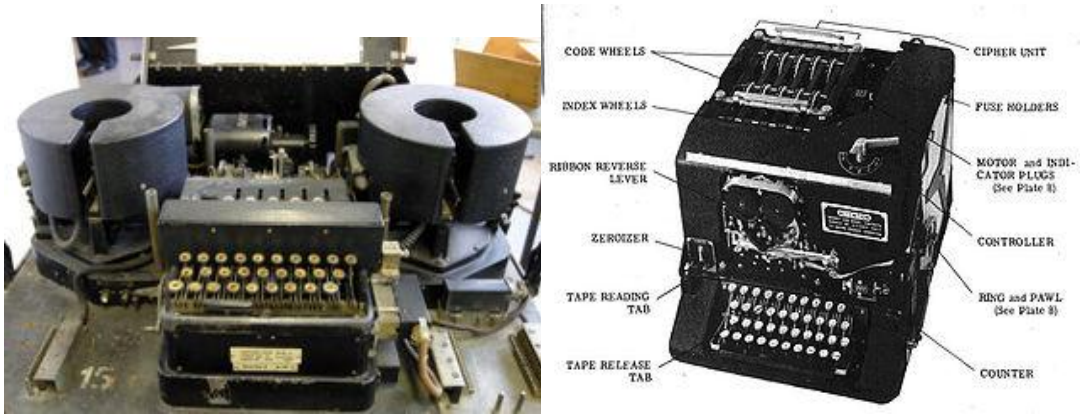


Figure 2.9: TYPEX and SIGABA

2.3.8 Urkryptografen - The Clock Cryptograph

Urkryptografen (Figure 2.10) is a Danish product, designed for use in the Danish Army [13]. Although the Ministry of War authorized its useon 23 June 1936, the Danish Army started to use it at least from 1934. Despite of the primary use of Urkryptographen was at divisional and regimental level, as the User's Manual refers to, the clock cryoptograph was used at all command levels from the General Staff and down. The Danish Navy used it mostly for joint operations. Significant numbers of Urkryptografen were pro-

duced. After WWII mechanical cryptographs (Hagelin) replaced Urkryptografen at higher command levels, and in 1947-48 Urkryptografen was replaced at the tactical level by other manual cipher devices that were easier to handle and cheaper to produce.

Urkryptografen is a substitution cipher device, identical with the Wheatstone cipher disk based on Greg Mellen. The outer ring is made of paper and carries the written mixed ciphertext alphabet. This paper ring is removable and should be changed often within a given cipher net. In the back of the device is room for a number of alphabet rings. For each 360 degree turn of the rings, the outer ring moves one step (letter) further than the inner ring. The rings must always be turned in the direction of the arrow. The engraved standard plaintext alphabet on the inner ring shown on the photograph is for training only. A number of metal disks with engraved mixed plaintext alphabets were produced and distributed, some of them reserved for wartime use.

Figure 2.10: Urkryptografen

The rings are turned until the black panel on the inner ring is exactly within the black metal frame. The brake (the small button on the side) is then pressed, stopping the inner ring. The outer ring is then turned until a message indicator letter appears in the black metal frame. This is the start position. A new message indicator letter is selected by the operator for each message and later placed in a prearranged position somewhere in the enciphered message.

Encipherment is performed by turning the rings (always in the direction of the arrow) so that each subsequent plain text letter appears in the black metal frame in the inner ring; the equivalent cipher letter is then read in the frame in the outer ring. The black panel indicates new word and Q indicates shift to figures and symbols and back again to letters. X replaces every second letter (or figure) in double-letter (or double-figure) combinations.

Decipherment is performed in the same manner, but cipher letters are read in the outer ring and the equivalent plain text letters found in the inner ring. Operation is not particularly easy. Before the actual encipherment, the plain text message must, according to the manual, be rewritten twice with the appropriate Q's", X's" and symbols inserted. The manual also prescribes that operation of Urkryptografen must be performed by a team of two operators, hardly an economical use of manpower, as seen through to-day's eyes.

2.3.9 NEMA

The NEMA (NEueMAchine) (Figure 2.11) was designed between 1941 and 1943 by the Swiss Army's Cipher Bureau and two pre-production models were produced in 1944. The device was put into active service in 1947 by the firm Zellwager A.G. in Switzerland. It was based on the Enigma principle such as the use of a reflective rotor but it included features so it could overcome the weaknesses of the German Enigma. The NEMA was designated as the T-D (Tasten-Drücker-Maschine) type and the serial numbers took the format of T-D XXX. It contained a total of 10 rotors. However, the rotor movements were much more irregular and the reflector moved during encipherment. In fact, five rotors were referred to as Fortschaltwalzen and acted as drive wheels for the other five. The rightmost, which was red while the other nine were black, controlled the movement of the two slower rotors while the other three rotors stepped more frequently.

The NEMA had several features such as exchangeable external power source (e.g. 110v, 220v, etc.), a separate lampboard for ease of use and an adapter which fit in any universal light bulb socket. The Swiss Army and the Swiss Diplomatic Service were using the NEMA but with different rotor sets each.

Figure 2.11: NEMA

An important game of life and death was the struggle under the weight of the German in the west and the Japanese in the east. The use of encryption by the allied forces and the interception and decryption of enemy ciphers played an important role: The headquarters of the Japanese 8th fleet, in the evening of 13.04.1943, sent messages concerning the itinerary for a visit by the commander in chief (CIC) of the Japanese Fleet. To protect this vital information, the message was encrypted using Japanese Naval code 25 or JN-25. This message, like many others, was intercepted by a US intercept station in Hawaii and the Royal Australian Airforce no.1 Wireless Unit in Townsville, North Queensland. Unknown to the Japanese or, as some suggest, suspected but ignored, the Americans had broken this code in late 1940 it having been a subset of a US army and navy code used in the Spanish-American war of 1898. The Allies' ability to intercept and decrypt this message led directly to the shooting down of aircraft carrying Admiral Yamamoto, over Bougainville, on the morning of 18.04.1943, by a United States P-38 Lightning piloted by Captain Thomas G. Lamphier. This resulted in the death of the most popular and capable officer in the Japanese navy robbing them of a brilliant and charismatic leader.

Moreover, on the celebration of the 50th anniversary of the Australian Defense Signals Directorate (DSD) their wartime allies, the United Kingdom, the United States of America, Canada and New Zealand made presentations. Underscoring under the importance of this war time codebreaking the US National Security Agency (NSA) presented DSD with a trophy containing one of the only 5 remaining rotors of a Japanese Purple Cipher Machine.

The postwar years Cryptography flourished thanks mainly to computer technology. The Colossus (1943) with 1500 electronic tubes and the fact that it was planned, the ENIAC (1945) with 18,000 electronic tubes could perform 5,000 calculations per sec.

2.3.10 Operation VENONA

In 1904, the British began successful SIGINT activities against Russia. In August 1920, in May 1923 and also in May 1927, the British politicians and the media made public detailed transcripts of intercepted and decrypted Soviet traffic first based on these successful activities. The Soviets finally realized the British work at the third time (in May 1927) and replaced the compromised codes on all their OGPU (KGB) and diplomatic networks with OTLP. This had as a result the British to be unable anymore to intercept Soviet traffic from 1927 to the early 1940's. This event, combined with the literary indiscretions of Churchill and the Royal Navy historians, are mistakes which had negative effects for Britain. After ceasing all work in the early 1930's because of the perceived impossibility of the task, the British began intercepting Russian traffic again in 1940.

By 1941 the intercepts were being shared with the US. This intercept work and its associated sharing agreement continued during and after the war, culminating in 1947, 1948 in the UKUSA agreement which had been made also with Canada, Australia and New Zealand. OTLP offered complete security in theory. However, in the occasion of either the original plain text or the used pages or current code books fell into the interceptors hands or even the pads were reused, this theory would had the opposite results. During the war years, for a variety of reasons, all these events occurred. Although the Soviets were participating with the allied nations against Germany and Japan, the Allies of the west were not holding friendly attitude towards them.

The US VENONA project began at Arlington Hall Virginia in February 1943 and by 1947 had broken into a considerable amount of traffic including 1945 KGB traffic between Moscow and Canberra. Supplemented by a similar UK effort, they were able, by the early 1950's, to identify Soviet agents in their respective intelligence and government services and the existence and makeup of a major Soviet spy ring in Australia. Despite preventing access for almost 20 years the Soviets had, at a most critical time, compromised the security of their global spy network by their failure to follow simple security rules.

2.4 Era of Modern Cryptography

Claude Shannon, the father of mathematical cryptography, began the era of modern cryptography with his work during World War II on communications security. In 1949, he published the paper "Communication Theory of Secrecy Systems" in the "Bell System Technical Journal" and later on the book "Mathematical Theory of Communication", with Warren Weaver. He established a solid theoretical basis for cryptography and for cryptanalysis (the study of methods for obtaining the meaning of encrypted information). Two important principles, which played a significant role to this era are confusion and diffusion:

The purpose of confusion is to make the relation between the key and the ciphertext as complex as possible. Ciphers that do not offer much confusion (such as Vigenere cipher) are susceptible to frequency analysis (a technique based on the fact that inside a text certain letters and combinations of letters occur with varying frequencies).

In contrast to confusion, diffusion spreads the influence of a single plaintext bit over many ciphertext bits. Normally we speak of data diffusion, in which changing a tiny part of the plaintext data may affect the whole ciphertext. But we can also speak of key diffusion, in which changing even a tiny part of the key should change each bit in the ciphertext with given probability.

After that, cryptography was used exclusively into secret government communications of organizations such as the NSA, GCHQ and similar other services. In the mid 1970's, when everything changed, cryptography made public where algorithms, principles and studies were published but only the encryption and decryption keys were kept secret.

2.4.1 DES – Data Encryption Standard

Two major public advances took place in the mid-1970s. First was the publication of the draft Data Encryption Standard in the U.S. Federal Register on 17 March 1975. The proposed DES was submitted by IBM, at the invitation of the National Bureau of Standards (now NIST), in an effort to develop secure electronic communication facilities for businesses such as banks and other large financial organizations. The algorithm was based on the Lucifer algorithm in the early 1970's. The proposal was accepted and the NSA tested and adjusted the algorithm and in 1976 released it as a federal standard.

DES is a Symmetric Block cipher based on a 64 bit block. The user feeds in a 64 block of plaintext and is returned 64 bits of ciphertext. The same algorithm and key are used both for encryption and decryption.

Since its release in 1976 the key was set at 56 bits (reduced from a 128 bit key as part of the NSA adjustment). It was possible also to build DES with a 128 bit key but its export was banned from the US. Recently however this key length restriction was removed by the US Government.

After modificated by the NSA, it was adopted and published in 1977 as a Federal Information Processing Standard Publication. The release of its specification by NBS created a mass public and academic interest in cryptography.

2.4.2 Diffie – Hellman Key Exchange

In 1976, the second development took place which was even more important and an important step in the history of cryptography. This was the publication of the paper "New Directions in Cryptography" by Whitfield Diffie and Martin Hellman. It introduced a radically new method of distributing cryptographic keys, which went far toward solving one of the fundamental problems of cryptography, key distribution, and has become known as Diffie-Hellman key exchange.

Whitfield Diffie was already considering the problems of e-commerce when the US defense department's Arpanet, the forerunner of the Internet, was still in its infancy. In 1974 he teamed with Martin Hellman and later Ralph Merkle to begin research into the problem of key exchange.

In 1976, Hellman had solved the key exchange problem by using one-way functions and modular arithmetic. In June 1976, at the US National Computer Conference, they demonstrated that Bob no longer had to meet Alice to exchange a secret key. While this was a fundamental breakthrough in conceptual terms, it did not offer a real world solution to the problem of key exchange. While working on the key exchange problem with Hellman and Merkle, Diffie had continued to ponder a solution for the obvious problems of the key exchange technique. In 1975, he developed the concept of the Asymmetric key which opened the possibility of operating a crypto system with a PUBLIC (published) and PRIVATE (secret) key. Before that, all useful modern encryption algorithms had been Symmetric key algorithms, in which the same cryptographic key is used with the specific algorithm by both the sender and the receiver.

In contrast, asymmetric key encryption uses a pair of mathematically related keys, each of which decrypts the encryption performed using the other. These algorithms have the advantage that one of the paired keys cannot be discovered from the other by any known method other than trial and error.

Only one key pair is needed per user in these algorithms. By designating one key of the pair as private (always secret), and the other as public (often visible), no secure channel is needed for key exchange. So long as the private key stays secret, the public key can be widely known for a very long time without compromising security. This led to the development of many important applications we use nowadays. A few examples of the asymmetric cryptography are digital signatures, PGP, use of public certificates, various authentication protocols (e.g. SSL, SSH).

In the same year Diffie published a paper on his work while continuing to look for the one way function that would make his theory a reality. Unfortunately, he never made that connection having this as a result the first known developers of an asymmetric key system by that time would be the inventors of RSA.

The 1980's and 90's have evolved into a digital world. The advent of the microprocessor and the Personal Computer (PC) and their acceptance into every-day life has meant that although our primary means of communication is the spoken word the "lingua franka" of our working lives and increasingly our private lives, is digital. In this digital era, several communication networks, such as Internet, digital (GSM) mobile phones, Automatic Teller Machines (ATM), appeared in human society, offering instant secure communication. These networks carry in vast number the most deliquate, private and sensitive messages of ordinary citizens as well as of different sectors of society (business, government, etc. even terrorism). All the electronic world, Electronic Commerce (E-Commerce), telecommunications, email messages etc. have the need of secure digital communication. The same need has and the other side of the coin such as drug rings, spy rings, people smugglers, organized crime, child porn and cyber-crime.

That is why law enforcement bodies and legitimate governments want "transparent" or Key Escrow encryption (CLIPPER) to fight crime and ensure national security. This technique was designed to allow these bodies of the law to obtain one's secret key if they established that one was involved in illegal activities. Other governments ban digital security to harass political opposition and quell dissent. Lobby groups, conspiracy theorists, oppressed minorities, distrustful and disgruntled citizens all want triple military-grade encryption fitted to their "fly-by" cards.

2.4.3 IDEA – International Data Encryption Algorithm

In the mid 1980's ROT13 algorithm used by users of USENET "to avoid seeing messages with objectionable content innocent eyes" and later in 1990 a discovery by Xuejia Lai and James Massey led to a stronger, 128-bit key cipher to replace the aging DES standard. The algorithm is IDEA (International Data Encryption Algorithm) designed by them had to be more efficient in general purpose computers like those used in businesses and households.

2.4.4 PGP – Pretty Good Privacy

PGP is a computer program which provides an encryption capability for e-mail which was covered by the S/MIME standard later on and was originally developed by Philip Zimmermann who began working on the algorithm in the late 1980's. PGP can be used also as a tamperproof digital signature system, which allows the user to prove that email messages and files have not been modified. The development of this system was as much about Zimmermann's distrust of the US Federal Government and its ability to intercept electronic communication as the development of a commercial cryptographic product. The history of this system has two interesting points: At first, an unlicensed implementation of RSA was used to provide key management while the IDEA algorithm was used to provide the actual data encryption layer. The complete package was uploaded onto the Internet so that it could be distributed as freeware because of Zimmermann's distrust of the US government. This, of course, created maximum heart-burn for the US government and led to their ill-considered use of pressure on him which in turn reinforced his position.

2.4.5 RSA

This algorithm, based on the original work of Diffie, was named after the three inventors Ron Rivest, Adi Shamir and Leonard Adleman (Figure 2.12). This is an Asymmetric cryptographic method and so it can be used for Public Key Cryptography. It was the first working public key cryptosystem (1976). In simple terms you can send Bob a message encrypted with his PUBLIC/Published key and when he has received it he can decrypt it with his PRIVATE/Secret key (from which his public key was derived). The

security of this system is based on the difficulty in factoring large numbers. The private and public keys can be functions of large (300-400 digit) prime numbers. While the process is known, recovering the plain text from the public key is considered to be the equivalent to factoring the product of the two prime numbers. With large numbers this is considered a major computational task, even by to-days standards, and is believed to be, in terms of time, beyond the capability of any existing technique/computer combination.

Example: Alice wants to send a message to Bob using the RSA algorithm.

Alice:

picks two primes, p, q

computes public modulus: n=p*q

picks encryption exponent e such that:

gcd(e, (p-1)(q-1))=1

publishes public key: (e, n)

computes private exponent d such that:

e *d=1 mod (p-1)(q-1) (a very helpful method for that is the Euclid's extended algorithm -)

keeps private key (n, p, q) secret

Bob:

retrieves Alice's public key (e, n)

to encrypt a message m:

$c=m^e \bmod n$

sends ciphertext c to Alice

Alice:

can decrypt the ciphertext by computing:

$m=c^d \bmod n$

A message m can also be encrypted by using Block RSA Encryption where the modulus n specifies the (maximum) block size.

The sender (Bob) uses a mapping, e.g.:

a-->10, b-->11, ...,z-->35

maps the message (plaintext) to the numerical representation

splits the latter to numbers less than n

Example:

encrypt "plars" with e=3 , n=1189:

plars --> 2521102728

252 1102 728

252^3 mod 1189 = 257

1102^3 mod 1189 = 203

728^3 mod 1189 = 230

c = (257,203,230)

Last, in 1994, Professor Ron Rivest (Figure 2.12 – left person), who helped develop the RSA, has published a new algorithm, the RC5 [[1], [14]].

Figure 2.12: Ron Rivest Adi Shamir and Leonard Adleman

2.4.6 RC5

RC5 is a fast symmetric block cipher suitable for hardware or software implementations. It uses mostly data dependent rotations. It is an invention of Ron Rivest developed in RSA Data Security, Inc. RC5 has a variety of parameters which can be used for word size, key size and for the number of rounds used. The algorithm uses block sizes which are usually 32, 64 or 128 bits and the key size is at maximum 2048 bits.

The encryption and decryption algorithms are exceptionally simple: There are three routines in RC5: key expansion, encryption and decryption. In the key-expansion routine, the user-provided secret key is expanded to fill a key table whose size depends on the number of rounds. The key expansion routine, the user-provided secret key is expanded to fill a key table whose size depends on the number of rounds. The key table is

then used in both encryption and decryption. The key table is then used in both encryption and decryption. The encryption routine consists of three primitive operations: integer addition, bitwise XOR and variable rotation. The encryption routine consists of three basic operations: integer addition, bitwise XOR and variable spin. RC5 makes it simple to implement and analyze. Like the RSA system, the encryption steps of RC5 are very easy to follow. The heavy use of data dependent rotations and the mixture of different operations provide the security of RC5. In particular, the use of use of data dependent rotations helps defeat differential and linear cryptanalysis.

After five years from the day RC5 was created, there have been numerous studies of RC5's security. Each study has provided a greater understanding of how RC5's structure and components function and how improve its security.

2.4.7 RC6

RC6 is a block cipher that was built upon RC5 and designed by Rivest, Sidney, and Yin for RSA Security [14]. Like RC5, RC6 makes essential use of data dependent rotaions. The main goal for the inventors has been to meet the requirements of the AES – Advanced Encryption Standard.

There are two main new features in RC6 compared to RC5: It includes integer multiplication as an additional primitive operation and includes the use of four working registers instead of two registers. The use of integer multiplication to increase the diffusion achieved per round so that fewer rounds are needed and the speed of the cipher can be increased. The reason for using four working registers instead of two is technical rather than theoretical. Namely, the default block size of the AES is 128 bits; while RC5 deals with 64-bit operations when using this block size, 32-bit operations are preferable given the intended architecture of the AES.

In May 1997, the U.S. patent office granted the RC5 patent to RSA Data Security. RC6 is proprietary of RSA Security but can be freely used for research and evaluation purposes during the AES evaluation period. We emphasize that if RC6 is selected for the AES, RSA Security will not require any licensing or royalty payments for products using the algorithm; there will be no restrictions beyond those specified for the AES by the U.S. government. Nevertheless, RC6 remains a trademark of RSA Security.

In 1994, Peter Shor devised an algorithm to let quantum computers determine the factorization of large integers. This was the first interesting problem for which quantum

computers promised a significant speed-up, and it therefore generated a lot of interest in quantum computers.

2.4.8 SSL

The encryption protocol SSL 1.0 was published by Netscape Communications in August 1994 -- only 9 months after the first release of Mosaic 1.0, the first popular web browser. Meanwhile, SSL encryption is supported by all popular web browsers. However, the transport protocol SSL (TLS) is not restricted to the application HTTPS.

2.4.9 S/MIME

A standard mechanism for secure email was published as RFC 1847 in October 1995. In the meantime it is supported by all popular email clients. S/MIME (Secure/Multipurpose Internet Mail Extensions) described a consistent way to send and receive secure (signed and/or encrypted) emails. It is based on the popular Internet MIME standard. However, S/MIME is not restricted to mail. S/MIME and SSL are the mostly used cryptographic protocols in the internet.

The EFF's DES cracker (*Deep Crack*) broke a DES key with a known-plaintext attack in 56 hours (DES challenge 2 by RSA Laboratories).Together, Deep Crack and *distributed.net* broke a DES key with a known-plaintext attack in 22 hours and 15 minutes (DES challenge 3 by RSA Laboratories).

2.4.10 AES – Advanced Encryption Standard

The aging DES was officially replaced by the Advanced Encryption Standard (AES) in 2001. DES, and more secure variants of it (such as Triple DES), are still used today, having been incorporated into many national and organizational standards. However, its 56-bit key-size has been shown to be insufficient to guard against brute force attacks.

The AES was issued as FIPS PUB 197 by NIST standard is the successor to DES. In January 1997 the AES initiative was announced and in September 1997 the public was invited to propose suitable block ciphers as candidates for the AES. The AES algorithm was selected in October 2001 and the standard was published in November 2002. NIST’s intent was to have a cipher that will remain secure well into the next century.

AES supports key sizes of 128 bits, 192 bits, and 256 bits, in contrast to the 56-bit keys offered by DES.

The AES algorithm resulted from a multi-year evaluation process led by NIST with submissions and review by an international community of cryptography experts. After an open competition, the Rijndael algorithm, invented by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, was selected by NIST as the standard.

Over time, many implementations are expected to upgrade to AES, both because it offers a 128-bit key size, and because it is a federal standard.

There are also some valid, efficient alternatives to AES, such as Blowfish.

2.4.11 Blowfish

Blowfish is a keyed, symmetric block cipher, designed in 1993 by Bruce Schneier and included in a large number of cipher suites and encryption products. Blowfish provides a good encryption rate in software and no effective cryptanalysis of it has been found to date.

Schneier designed Blowfish as a general-purpose algorithm, intended as an alternative to the aging DES and free of the problems and constraints associated with other algorithms. At the time Blowfish was released, many other designs were proprietary, encumbered by patents or were commercial/government secrets. Schneier has stated that "Blowfish is unpatented, and will remain so in all countries. The algorithm is hereby placed in the public domain, and can be freely used by anyone."

The Blowfish is a 64bit block cipher with a variable-length key. The algorithm (Figure 2.13) consists of two parts: extension-length key and data encryption. The expansion converts a key of 448 bit in various series of subkeys totaling 4168 bytes. Data encryption is a simple operation that is repeated 16 times. Each cycle consists of a key-dependent-mutation and a key with point-dependent replacement. All procedures are additions to XORs bit words. The only additional procedures are four data series compiled searches per cycle. Blowfish uses a large number of subkeys. These keys should be calculated before any data encrypted or decrypted.

The P-series consists of eighteen 32bit subkeys:

$$P_1, P_2, \ldots, P_{18}$$

Four 32bit S-boxes are 256 entries each:

$$S_{1,0}, S_{1,1}, \ldots, S_{1,255}$$
$$S_{2,0}, S_{2,1}, \ldots, S_{2,255}$$
$$S_{3,0}, S_{3,1}, \ldots, S_{3,255}$$
$$S_{4,0}, S_{4,1}, \ldots, S_{4,255}$$

Blowfish is a Feistel network consisting of 16 cycles. The introduction is a 64bit element X. To encrypt:

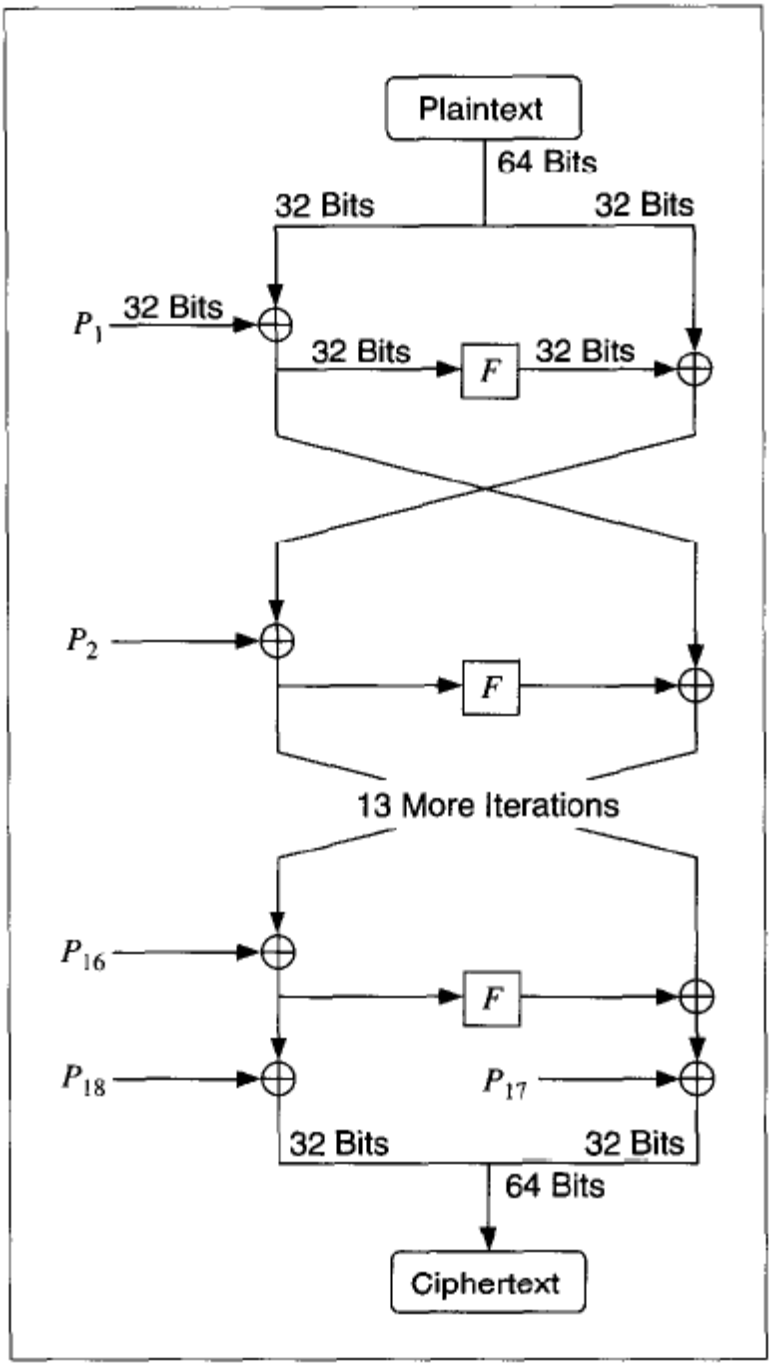


Figure 2.13: Blowfish encryption algorithm

2.4.12 Quantum cryptography

Quantum cryptography uses current knowledge of physics to develop a cryptosystem that is not able to be defeated - that is, one that is completely secure against being compromised without knowledge of the sender or the receiver of the messages [15]. Quantum cryptography takes advantage of the unique and unusual behavior of microscopic

objects to enable users to securely develop secret keys as well as to detect eavesdropping.

Quantum cryptography's security relies more on physics rather than mathematics like the traditional cryptographic systems. Basically, quantum cryptography uses individual particles or waves of light - photon and their intrinsic quantum properties to develop an unbreakable cryptosystem - essentially because it is impossible to measure the quantum state of any system without disturbing that system. It is theoretically possible that other particles could be used, but photons offer all the necessary qualities needed, their behavior is comparatively well-understood, and they are the information carriers in optical fiber cables, the most promising medium for extremely high-bandwidth communications.

In theory, quantum cryptography works in the following manner (this view is the "classical" model developed by Bennett and Brassard in 1984 - some other models do exist):

Assume that two people wish to exchange a message securely, traditionally named Alice and Bob. Alice initiates the message by sending Bob a key, which will be the mode for encrypting the message data. This is a random sequence of bits, sent using a certain type of scheme, which can see two different initial values represent one particular binary value (0 or 1).

Assume that this key is a stream of photons travelling in one direction, with each of these photon particles representing a single bit of data (either a 0 or 1). However, in addition to their linear travel, all of these photons are oscillating (vibrating) in a certain manner. These oscillations can occur in any 360-degree range across any conceivable axis, but for the purpose of simplicity (at least as far as it is possible to simplify things in quantum cryptography), assume that their oscillations can be grouped into 4 particular states: we'll define these as UP/DOWN, LEFT/RIGHT, UPLEFT/RIGHTDOWN and UPRIGHT/LEFTDOWN. The angle of this vibration is known as the polarization of the photon. A polarizer is simply a filter that permits certain photons to pass through it with the same oscillation as before and lets others pass through in a changed state of oscillation (it can also block some photons completely, but let's ignore that property for this exercise). Alice has a polarizer that can transmit the photons in any one of the four states mentioned - in effect, she can choose either rectilinear (UP/DOWN and LEFT/RIGHT) or diagonal (UPLEFT/RIGHTDOWN and UPRIGHT/LEFTDOWN) polarization filters.

Alice swaps her polarization scheme between rectilinear and diagonal filters for the transmission of each single photon bit in a random manner. In doing so, the transmission can have one of two polarizations represent a single bit, either 1 or 0, in either scheme she uses.

When receiving the photon key, Bob must choose to measure each photon bit using either his rectilinear or diagonal polarizer: sometimes he will choose the correct polarizer and at other times he will choose the wrong one. Like Alice, he selects each polarizer in a random manner. So what happens with the photons when the wrong polarizer is chosen?

The Heisenberg Uncertainty Principle states that it is unknown exactly what will happen to each individual photon, for in the act of measuring its behavior, we alter its properties (in addition to the fact that if there are two properties of a system that we wish to measure, measuring one precludes us from quantifying the other). However, there is a guess as to what happens with them as a group. Suppose Bob uses a rectilinear polarizer to measure UPLEFT/RIGHTDOWN and UPRIGHT/ LEFTDOWN (diagonal) photons. If he does this, then the photons will pass through in a changed state - that is, half will be transformed to UP/DOWN and the other half to LEFT/RIGHT. But it is unknown which individual photons will be transformed into which state (it is also a reality that some photons may be blocked from passing altogether in a real world application, but this is not relevant to the theory).

Bob measures some photons correctly and others incorrectly. At this point, Alice and Bob establish a channel of communication that can be insecure - that is, other people can listen in. Alice then proceeds to advise Bob as to which polarizer she used to send each photon bit - but not how she polarized each photon. So she could say that photon number 8597 (theoretically) was sent using the rectilinear scheme, but she will not say whether she sent an UP/DOWN or LEFT/RIGHT. Bob then confirms if he used the correct polarizer to receive each particular photon. Alice and Bob then discard all the photon measurements that he used the wrong polarizer to check. What they have, is, on average, a sequence of 0s and 1s that is half the length of the original transmission but it will form the basis for a one-time pad, the only cryptosystem that, if properly implemented, is proven to be completely random and secure.

Now, suppose there is an eavesdropper, Eve, who attempts to listen in, has the same polarizers that Bob does and must also randomly choose whether to use the rectilinear or diagonal one for each photon. However, she also faces the same problem that Bob

does, in that half the time she will choose the wrong polarizer. But Bob has the advantage of speaking to Alice to confirm which polarizer type was used for each photon. This is useless to Eve, as half the time she used the wrong detector and will misinterpret some of the photons that will form that final key, rendering it useless.

Furthermore, there is another level of security inherent in quantum cryptography - that of intrusion detection. Alice and Bob would know if Eve was eavesdropping on them. The fact that Eve is on the "photon highway" can become obvious because of the following.

Let's say that Alice transmits photon number 349 as an UPRIGHT/LEFTDOWN to Bob, but for that one, Eve uses the rectilinear polarizer, which can only measure UP/DOWN or LEFT/RIGHT photons accurately. What Eve will do is transform that photon into either UP/DOWN or LEFT/RIGHT, as that is the only way the photon can pass. If Bob uses his rectilinear polarizer, then it will not matter what he measures as the polarizer check Alice and Bob go through above will discard that photon from the final key. But if he uses the diagonal polarizer, a problem arises when he measures its polarization; he may measure it correctly as UPRIGHT/LEFTDOWN, but he stands an equal chance, according to the Heisenberg Uncertainty Principle, of measuring it incorrectly as UPLEFT/RIGHTDOWN. Eve's use of the wrong polarizer will warp that photon and will cause Bob to make errors even when he is using the correct polarizer.

To discover Eve's nefarious doings, they must perform the above procedures, with which they will arrive at an identical key sequence of 0s and 1s - unless someone has been eavesdropping, whereupon there will be some discrepancies. They must then undertake further measures to check the validity of their key. It would be foolish to compare all the binary digits of the final key over the unsecured channel discussed above, and also unnecessary.

Assume that the final key comprises 4,000 binary digits. What needs to be done is that a subset of these digits be selected randomly by Alice and Bob, say 200 digits, in terms of both position (that is, digit sequence number 2, 34, 65, 911 etc) and digit state (0 or 1). Alice and Bob compare these - if they match, then there is virtually no chance that Eve was listening. However, if she was listening in, then her chances of being undiscovered are one in countless trillions, that is, no chance in the real world. Alice and Bob would know someone was listening in and then would not use the key - they would need to start the key exchange again over a secure channel inaccessible to Eve, even though the comparisons between Alice and Bob discussed above can still be done over

an insecure channel. However, even if Alice and Bob have concluded that the their key is secure, since they have communicated 200 digits over an un-secure channel, these 200 digits should be discarded from the final key, turning it from a 4,000 into a 3,800 bit key). Thus, quantum cryptography is a way to combine the relative ease and convenience of key exchange in public key cryptography with the ultimate security of a one-time pad.

In practice, quantum cryptography has been demonstrated in the laboratory by IBM and others, but over relatively short distances. Recently, over longer distances, fiber optic cables with incredibly pure optic properties have successfully transmitted photon bits up to 60 kilometers. Beyond that, BERs (bit error rates) caused by a combination of the Heisenberg Uncertainty Principle and microscopic impurities in the fiber make the system unworkable. Some research has seen successful transmission through the air, but this has been over short distances in ideal weather conditions. It remains to be seen how much further technology can push forward the distances at which quantum cryptography is practical.

Practical applications in the US are suspected to include a dedicated line between the White House and Pentagon in Washington, and some links between key military sites and major defense contractors and research laboratories in close proximity.

2.4.13 Elliptical curve cryptography (ECC)

Elliptic curves have a rich and beautiful history, having been studied by mathematicians for over a hundred years. Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller, and more efficient cryptographic keys [16]. In 1985, Neal Koblitz and Victor Miller independently proposed using elliptic curves to design public-key cryptographic systems. Since then an abundance of research has been published on the security and efficient implementation of elliptic curve cryptography. In the late 1990's, elliptic curve systems started receiving commercial acceptance when accredited standards organizations specified elliptic curve protocols and private companies included these protocols in their security products.

The keys in ECC are generated through the properties of the elliptic curve equation rather than being the result of very large prime numbers as it is used to the traditional method of key generation. The technology can be used in conjunction with most public

key encryption methods, such as RSA and Diffie-Hellman. According to some researchers, ECC can yield a level of security with a 164 bit key that other systems require a 1,024 bit key to achieve. ECC is becoming widely used for mobile applications, like pagers, PDAs, mobile phones, because it provides the same level of security with lower cost in computing power and battery resource usage. ECC was developed by Certicom, a mobile e-business security provider which was recently licensed by Hifn, a manufacturer of integrated circuitry (IC) and network security products. RSA has been developing its own version of ECC. Many manufacturers, including 3COM, Motorola, Cylink, Siemens, Pitney Bowes, TRW and VeriFone, have designed their products to be compatible with ECC.

The properties and functions of elliptic curves have been studied in mathematics for 150 years. An elliptic curve is not an ellipse, but is represented as a looping line intersecting two axes (lines on a graph used to indicate the position of a point). ECC is based on properties of a particular type of equation created from the mathematical group (a set of values for which operations can be performed on any two members of the group to produce a third member) derived from points where the line intersects the axes. Multiplying a point on the curve by a number will produce another point on the curve, but it is very difficult to find what number was used, even if you know the original point and the result. Equations based on elliptic curves have a characteristic that benefits cryptography purposes: they are computed faster, implemented easy and they are extremely difficult to be reversed.

On the other hand, some problems have appeared on the horizon regarding ECC. Nigel Smart, a Hewlett Packard researcher, discovered a flaw in which certain curves are extremely vulnerable. However, Philip Deck of Certicom says that, while there are curves that are vulnerable, those implementing ECC would have to know which curves could not be used. He believes that ECC offers a unique potential as a technology that could be implemented worldwide and across all devices. Based on Deck, as he supported in an interview, "The only way you can achieve that is with elliptic curve".

At the *Crypto 2004* conference, Chinese researchers showed structural weaknesses in common hash functions (MD5, SHA), which make them vulnerable to practical *collision attacks*. These hash functions are still used in almost all cryptographic protocols. The Chinese researchers didn't publish all the details.

In April 2007 the wireless LAN encryption protocol *WEP* was broken by three researchers of the TU Darmstadt. Assuming enough data traffic in the network it takes only about two minutes to derive 95% of all used encryption keys.

The same year, an algorithm to break the immobilizer system, used in millions of cars, was shown at the *Crypto 2007* conference. During the presentation Eli Biham, Orr Dunkelman et al. could demonstrate an example where a corresponding car key was copied in 48 hours with the computing power of 50 PCs.

Also the proprietary encryption algorithm *A5/1*, which is used by many GSM cellular carriers, was broken by David Hulton and Joshua Lackey. This implies that in affected mobile networks even the shortest voice calls or SMS messages can be easily encrypted by a normal PC, showing that "security by obscurity" is not a good approach.

At the end of 2007 the authentication algorithm of the *Mifare chip cards*, which are used in thousands of applications by one billion issued cards, was broken. However the newest generation (MifareDESFire), which uses DES/3-DES, is not affected.

In December 2009 Chris Paget and KarstenNohl announce, that the *A5/1* Cracking Project build the 2TB time memory tradeoff attack tables for A5/1. A5/1 was broken.

Same period, Jens Franke et al., who factorized the 663 bit long number *RSA-200* in May 2005, factorized the 768 bit long number *RSA-768*.

3 Problems in Cryptography

Often the objectives of information security cannot solely be achieved through mathematical algorithms and protocols alone, but require procedural techniques and abidance of laws to achieve the desired result. For example, privacy of letters is provided by sealed envelopes delivered by an accepted mail service. The physical security of the envelope is, for practical necessity, limited and so laws are enacted which make it a criminal offense to open mail for which one is not authorized. It is sometimes the case that security is achieved not through the information itself but through the physical document recording it. For example, paper currency requires special inks and material to prevent counterfeiting. Conceptually, the way information is recorded has not changed dramatically over time.

Whereas information was typically stored and transmitted on paper, much of it now resides on magnetic media and is transmitted via telecommunications systems, some wireless. What has changed dramatically is the ability to copy and alter information. One can make thousands of identical copies of a piece of information stored electronically and each is indistinguishable from the original. With information on paper, this is much more difficult. What is needed then for a society where information is mostly stored and transmitted in electronic form is a mean to ensure information security which is independent of the physical medium recording or conveying it and such that the objectives of information security rely solely on digital information itself.

One of the fundamental tools used in information security is the signature. It is a building blocks form any other services such as non-repudiation, data origin authentication, identification and witnessing. Having learned the basics in writing, an individual is taught how to produce a handwritten signature for the purpose of identification.

At contract age the signature evolves to take on a very integral part of the person's identity. This signature is intended to be unique to the individual and serve as a means to identify, authorize and validate. With electronic information the concept of a signature needs to be redressed; it cannot simply be something unique to the signer and independent of the information signed. Electronic replication of it is so simple that append-

ing a signature to a document not signed by the originator of the signature is almost a triviality.

Analogues of the “paper protocols” currently in use are required. Hopefully these new electronic based protocols are at least as good as those they replace. There is a unique opportunity for society to introduce new and more efficient ways of ensuring information security. Much can be learned from the evolution of the paper based system, mimicking those aspects which have served us well and removing the inefficiencies.

Achieving information security in an electronic society requires a vast array of technical and legal skills. There is, however, no guarantee that all of the information security objectives deemed necessary can be adequately met. The technical means is provided through cryptography.

3.1 Methods of Encryption

Although there can be several pieces to an encryption method, the two main pieces are the algorithms and the keys.

The algorithm, the set of mathematical rules, dictates how enciphering and deciphering take place. Many algorithms are publicly known and are not the secret part of the encryption process. The way that encryption algorithms work can be kept secret from the public, but many of them are publicly known and well understood. If the internal mechanisms of the algorithm are not a secret, then something must be. The secret piece of using a well-known encryption algorithm is the key. The key can be any value that is made up of a large sequence of random bits. An algorithm contains a keyspace, which is a range of values that can be used to construct a key. The key is made up of random values within the keyspace range. The larger the keyspace, the more available values can be used to represent different keys, and the more random the keys are, the harder it is for intruders to figure them out.

A large keyspace allows for more possible keys. The encryption algorithm should use the entire keyspace and choose the values to make up the keys as random as possible. If a smaller keyspace were used, there would be fewer values to choose from when forming a key, as shown in Figure 3.1. This would increase an attacker’s chance of figuring out the key value and deciphering the protected information.

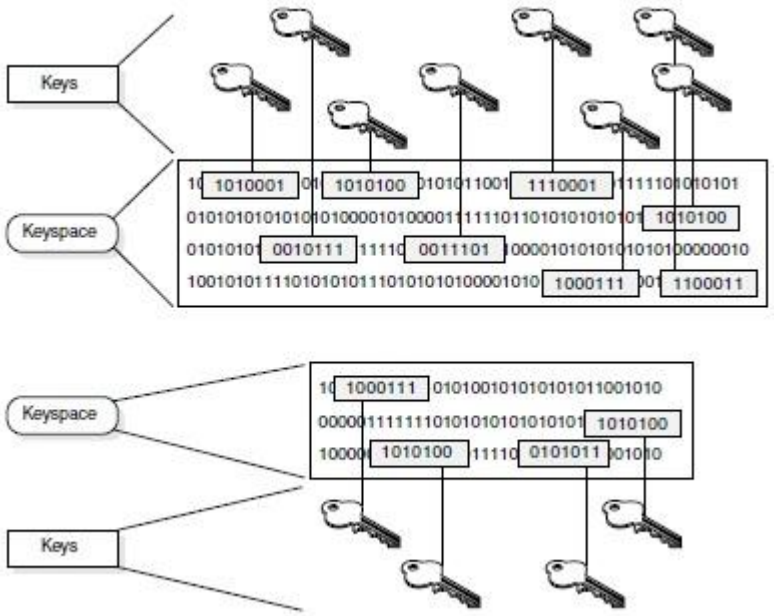


Figure 3.1: Keyspaces

As stated earlier, algorithms are usually complex mathematical formulas that dictate the rules of how the plaintext will be turned into ciphertext. A key is a string of random bits that will be inserted into the algorithm. For two entities to be able to communicate via encryption, they must use the same algorithm and, many times, the same key. In some encryption methods, the receiver and the sender use the same key and in other encryption methods, they must use different keys for encryption and decryption purposes. The strength of the encryption method comes from the algorithm, secrecy of the key, length of the key, initialization vectors, and how they all work together. When strength is discussed in encryption, it refers to how hard it is to figure out the algorithm or key, whichever is not made public. Breaking a key has to do with processing an amazing number of possible values in the hopes of finding the one value that can be used to decrypt a specific message. The strength correlates to the amount of necessary processing power and time it takes to break the key or figure out the value of the key. Breaking a key can be accomplished by a brute force attack, which means trying every possible key value until the resulting plaintext is meaningful. Depending on the algorithm and length of the key, this can be a very easy task or a task that is close to impossible. If a key can be broken with a Pentium II processor in three hours, the cipher is not strong at all. If the key can only be broken with the use of a thousand multiprocessing systems, and it takes 1.2 million years, then it is pretty darn strong.

The goal of designing an encryption method is to make compromise too expensive or too time consuming. Another name for cryptography strength is work factor, which is

an estimate of the effort it would take an attacker to penetrate an encryption method. The strength of the protection mechanism should be used in correlation to the sensitivity of the data being encrypted. Each type of encryption mechanism has its place and purpose.

Even if the algorithm is very complex and thorough, there are other issues within encryption that can weaken the strength of encryption methods. Because the key is usually the secret value needed to actually encrypt and decrypt messages, improper protection of the key can weaken the encryption strength. An extremely strong algorithm can be used, using a large keyspace, and a large and random key value, which are all the requirements for strong encryption, but if a user shares her key with others, these other pieces of the equation really don't matter.

An algorithm with no flaws, a large key, using all possible values within a keyspace, and protecting the actual key are important elements of encryption. If one is weak, it can prove to be the weak link that affects the whole process.

The following sections explain the difference between these two types of encryption methods [6].

3.2 Symmetric Cryptography

Cryptography algorithms use either symmetric keys, also called secret keys, or asymmetric keys, also called public keys.

In a cryptosystem that uses symmetric cryptography, both parties will be using the same key for encryption and decryption, as shown in Figure 3.2. This provides dual functionality.

As we said, symmetric keys are also called secret keys because this type of encryption relies on each user to keep the key a secret and properly protected. If this key got into an intruder's hand, that intruder would have the ability to decrypt any intercepted message encrypted with this key.

Each pair of users who want to exchange data using symmetric key encryption must have their own set of keys. This means if Bob and Alice want to communicate, both need to obtain a copy of the same key.

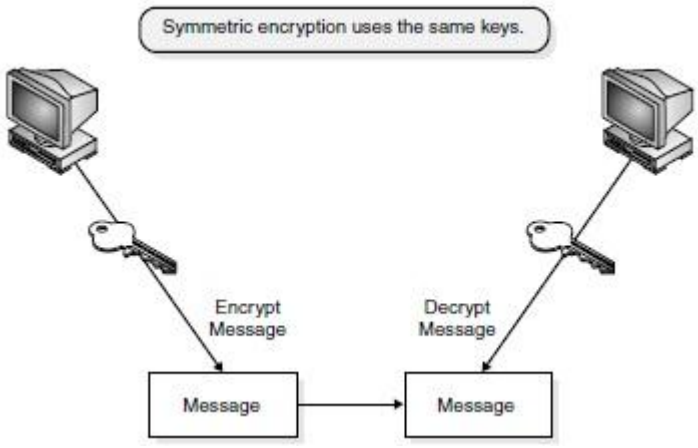


Figure 3.2: Symmetric Cryptosystem

If Bob also wants to communicate using symmetric encryption with Norm and Dave, he now needs to have three separate keys, one for each friend. This might not sound like a big deal until Bob realizes that he may communicate with hundreds of people over a period of several months, and keeping track and using the correct key that corresponds to each specific receiver can become a very daunting task. If Bob were going to communicate with 10 other people, then he would need to keep track of 45 different keys. If Bob were going to communicate with 100 other people, then he would have to maintain and keep up with 4,950 symmetric keys.

Bob does not necessarily want to spend his days looking for the right key to be able to communicate with Dave.

The security of the symmetric encryption method is completely dependent on how well users protect the key. This should raise red flags to you if you have ever had to depend on a whole staff of people to keep a secret. If a key is compromised, then all messages encrypted with that key can be decrypted and read by an intruder. This is complicated further by how symmetric keys are actually shared and updated when necessary.

If Bob wants to communicate to Norm for the first time, Bob has to figure out how to get Norm the right key. It is not safe to just send it in an e-mail message because the key is not protected and it can be easily intercepted and used by attackers. Bob has to get the key to Norm through an out-of-band method. Bob can save the key on a floppy disk and walk over to Norm's desk, send it to him via snail mail, or have a secure carrier deliver it to Norm. This is a huge hassle, and each method is very clumsy and insecure.

Because both users use the same key to encrypt and decrypt messages, symmetric cryptosystems can provide confidentiality, but they cannot provide authentication or nonrepudiation. There is no way to prove who actually sent a message if two people are using

the exact same key. Another problem is that it requires a secure mechanism to deliver keys properly. Also each pair of users needs a unique pair of keys, so the number of keys grows exponentially.

Despite of these problems, symmetric cryptosystems are used though because they are very fast and can be hard to break. Compared to asymmetric systems, symmetric algorithms are very agile and guaranteed safe. They can encrypt and decrypt large amounts of data that would take an unacceptable amount of time if an asymmetric algorithm was used instead. It is also very difficult to uncover data that is encrypted with a symmetric algorithm if a large key size was used.

Examples of symmetric key cryptography algorithms are:

- Data Encryption Standard (DES – Data Encryption Standard)
- Triple DES (3DES)
- Blowfish
- IDEA – International Data Encryption Algorithm
- RC4, RC5, and RC6

3.3 Asymmetric Cryptography

In symmetric key cryptography, a single secret key is used between entities, whereas in public key systems, each entity has different keys, or asymmetric keys. The two different asymmetric keys are mathematically related. If a message is encrypted by one key, the other key is required to decrypt the message.

In a public key system, the pair of keys is made up of one public key and one private key. The public key can be known to everyone, and the private key must only be known to the owner. Many times, public keys are listed in directories and databases of e-mail addresses so they are available to anyone who wants to use these keys to encrypt or decrypt data when communicating with a particular person. Figure 3.3 illustrates an asymmetric cryptosystem.

The public and private keys are mathematically related, but cannot be derived from each other. This means that if an evildoer gets a copy of Bob's public key, it does not mean he can now use some mathematical magic and find out Bob's private key.

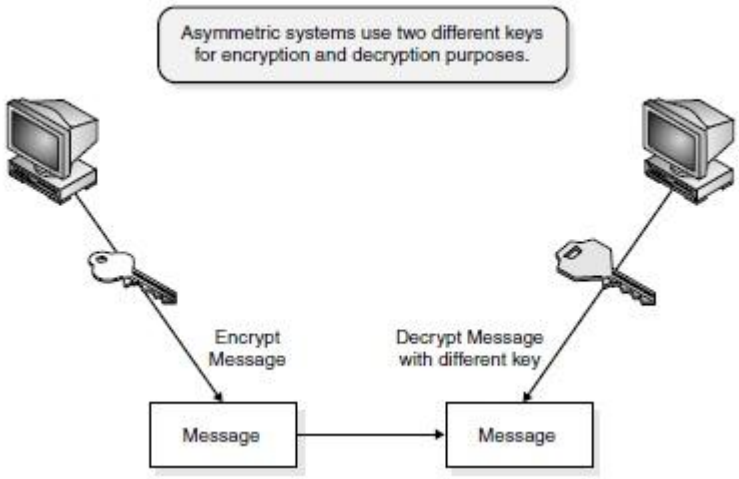


Figure 3.3: Asymmetric Cryptosystem

If Bob encrypts a message with his private key, the receiver must have a copy of Bob's public key to decrypt it. The receiver can decrypt Bob's message and decide to reply back to Bob in an encrypted form. All she needs to do is encrypt her reply with Bob's public key, and then Bob can decrypt the message with his private key. It is not possible to encrypt and decrypt using the exact same key when using an asymmetric key encryption technology.

Bob can encrypt a message with his private key and the receiver can then decrypt it with Bob's public key. By decrypting the message with Bob's public key, the receiver can be sure that the message really came from Bob. A message can only be decrypted with a public key if the message was encrypted with the corresponding private key. This provides authentication, because Bob is the only one who is supposed to have his private key. When the receiver wants to make sure Bob is the only one that can read her reply, she will encrypt the response with his public key. Only Bob will be able to decrypt the message because he is the only one who has the necessary private key.

Now the receiver can also encrypt her response with her private key instead of using Bob's public key. That way Bob will know that the message came from her and no one else. If she encrypted the response with Bob's public key, it does not provide authenticity because anyone can get a hold of Bob's public key. If she uses her private key to encrypt the message, then Bob can be sure that the message came from her and no one else. Symmetric keys do not provide authenticity because the same key is used on both ends. Using one of the secret keys does not ensure that the message originated from a specific entity.

If confidentiality is the most important security service to a sender, she would encrypt the file with the receiver's public key. This is called a secure message format because it can only be decrypted by the person who has the corresponding private key.

If authentication is the most important security service to the sender, then she would encrypt the message with her private key. This provides assurance to the receiver that the only person who could have encrypted the message is the individual who has possession of that private key. If the sender encrypted the message with the receiver's public key, authentication is not provided because this public key is available to anyone.

Encrypting a message with the sender's private key is called an open message format because anyone with a copy of the corresponding public key can decrypt the message; thus, confidentiality is not ensured.

For a message to be in a secure and signed format, the sender would encrypt the message with her private key and then encrypt it again with the receiver's public key. The receiver would then need to decrypt the message with his own private key and then decrypt it again with the sender's public key. This provides confidentiality and authentication for that delivered message. The different encryption methods are shown in Figure 3.4:

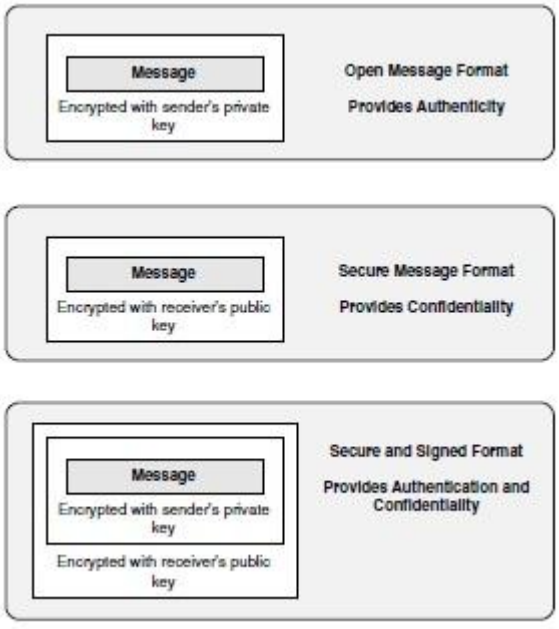


Figure 3.4: Encryption Methods

Each key type can be used to encrypt and decrypt. They both have the capability to encrypt and decrypt data. However, if data is encrypted with a private key, it cannot be decrypted with a private key. If data is encrypted with a private key, it must be decrypt-

ed with the corresponding public key. If data is encrypted with a public key, it must be decrypted with the corresponding private key. Figure 3.5 further explains the steps of a signed and secure message:

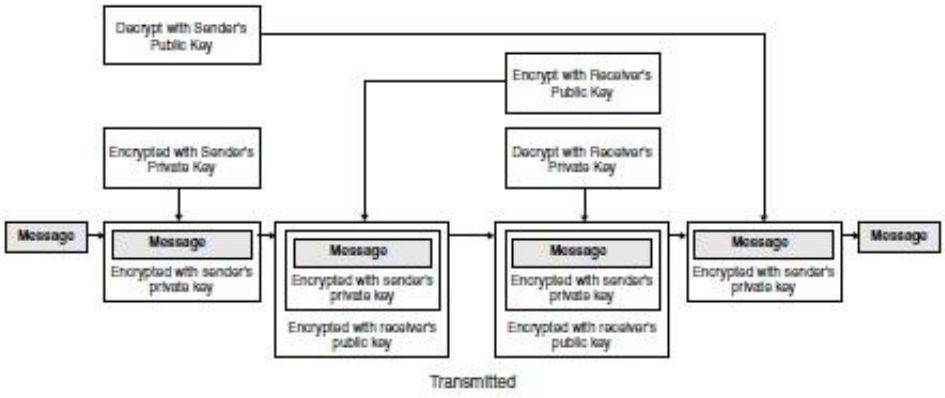


Figure 3.5: The message is encrypted twice with the sender and receiver's keys.

An asymmetric cryptosystem works much slower than symmetric systems, the length of the public key is longer, but can provide confidentiality, authentication, and nonrepudiation depending on its configuration and use. Asymmetric systems also provide for easier and more manageable key distribution than symmetric systems and do not have the scalability issues of symmetric systems.

Examples of symmetric key cryptography algorithms are:

- RSA
- Elliptic Curve Cryptosystem (Elliptical curve cryptography (ECC))
- Diffie-Hellman(Diffie – Hellman Key Exchange, Diffie–Hellman key exchange)
- El Gamal
- Digital Signature Standard (DSS)

3.3.1 El Gamal

The El Gamal cryptosystem is a rival to RSA and is widely used. El Gamal is a public key algorithm that can be used for digital signatures and key exchange [4]. Its security is based on the difficulty of the discrete logarithm in a finite field. It works as follows:

Bob chooses a prime number p and a primitive root g mod p. He also chooses an integer a ϵ {1,…, p-2} and computes h = g^a (mod p). His public key is (p, g, h). The number a is kept secret.

Alice wants to send a plaintext x to Bob, encoded as an integer in the range {1,…, p-1}. She chooses a random number k, also in this range, and computes y1 = g^k (mod p) and y2 = xh^k (mod p). The ciphertext is the pair (*y*1, *y*2).

Note that

- the ciphertext is twice as long as the plaintext
- there are p-1 different ciphertexts for each plaintext, one for each choice of the random number k.

Bob receives the message (g^k, xh^k) mod p. He knows the number a such that h = g^a mod p; so he can compute $h^k \equiv (g^a)^k \equiv (g^k)^a$ mod p without knowing Alice's secret number k. Now he can find x by dividing y2 = xh^k by h^k. More precisely, he uses Euclid's algorithm to find the inverse of h^k mod p and multiplies y2 by this inverse to get the plaintext x.

Eve, intercepting the message, is faced with the problem of finding either

- the number a for which h ≡ g^a (mod p), so that she can then use the same decrypting method as Bob; or
- the number k for which y1 ≡ g^k (mod p), so that she can find h^k directly and hence find x.

Either approach requires her to solve the Discrete Logarithm problem, and so may be assumed to be difficult. No better way of trying to break the cipher is known. Note that, if Eve does have the computational resources to solve a discrete logarithm problem, she should employ them on the first of the above problems. For if this is solved, then she knows Bob's private key and can read all his mail. Solving the second only gives her Alice's random number k, which will be different for each message, so the same job would have to be done many times.

Here is a brief example. Suppose that Bob chooses the prime p = 83, the primitive root g = 2, and the number a = 30, so that h = 2^{30} mod 83 = 40. Bob's public key is (83, 2, 40). Suppose that Alice's plaintext is x = 54 and her random number is k = 13. Then Alice's ciphertext is (g^k, xh^k) mod p = (58, 71).

Bob computes 58^{30} mod 83 = 9. By Euclid's algorithm, the inverse of 9 mod 83 is 37 and so the plaintext is 37 * 71 mod 83 = 54.

Using the El Gamal scheme for digital signatures is a bit more complicated than using, say, RSA. This is because, as we saw, the ciphertext in El Gamal is twice as long as the plaintext, and depends on the choice of a random number k. So, to sign a message, Al-

ice cannot simply pretend that the message is a cipher and decrypt it with her private key. Instead, she adds further data whose purpose is to authenticate the message.

Suppose that Alice's El Gamal public key is (p, g, h), where p is prime and g is a primitive root mod p. Then $h \equiv g^a \pmod{p}$, where the number a is known only to Alice.

To sign a message $x \in \{1,\ldots, p-1\}$, Alice chooses a random number k satisfying gcd(k, p-1) = 1. Then using Euclid's algorithm, she computes the inverse l of k mod p-1. Now she computes:

$$z_1 = g^k \bmod p,$$
$$z_2 = (x - az_1)*l \bmod p-1$$

The signed message is (x, z_1, z_2). Just as in the case of encryption, note that it is longer than (in this case, three times as long as) the unsigned message and it depends on a random number k. Alice then encrypts this message with Bob's public key and sends it to Bob.

On receipt, Bob decrypts the message, and finds three components. The first component is the plaintext *x*. The second and third components comprise the signature. Bob accepts the signature as valid if

$$h^{z1}z_1^{z2} \equiv g^x \pmod{p}$$

We have to show that

- if Alice follows the protocol correctly, this condition will be satisfied;
- Eve cannot forge the signature (i.e. produce (x, z_1, z_2) satisfying this condition) without solving a discrete logarithm problem.

The first condition is just a case of checking:

$$h^{z1}z_1^{z2} \equiv g^{az1}g^{kl(x - az1)} \pmod{p}$$

Note that $g^{p-1} \equiv 1 \pmod{p}$, so exponents of g can be read modulo p-1. Now $kl \equiv 1 \pmod{p-1}$, so $g^{kl(x - az1)} \equiv g^{x - az1} \pmod{p}$. Then:

$$h^{z1}z_1^{z2} \equiv g^{az1}g^{x - az1} \equiv g^x \pmod{p}$$

The second part is a bit more complicated and the argument will not be given here. It is clear that Eve cannot do Alice's computation without knowing a. We

have to be sure that there is no other way that she could produce a forgery.

For instance, suppose that Alice's public key is (107, 2, 15), with secret number 11, so that 2 is a primitive root mod 107, and $211 \equiv 15 \pmod{107}$. Suppose that Alice wants to send the message 10 to Bob and sign it. She chooses k = 17 (this number is co-prime to 106 and its inverse is 25. The signature is (z_1, z_2), where:

$$z_1 = 2^{17} \bmod 107 = 104,$$
$$z_2 = (10 - 11 * 104) * 25 \bmod 106 = 58$$

So she encrypts the plaintext (10, 104, 58) with Bob's public key and sends it to Bob. (Note that the one number *x* has now become six numbers in the ciphertext)

Bob, having decrypted the message, obtains (10, 104, 58). He tests whether $15^{104} \equiv 104^{58} \equiv 2^{10} \pmod{107}$ and, since this is the case, he is assured that the message is from Alice.

3.3.2 Diffie–Hellman key exchange algorithm

The functions used for the RSA cipher can also be used to implement the key exchange protocol that we discussed at the very beginning of our discussion of public key cryptography. This system of key exchange actually predates the RSA cipher.

Assume that Alice wants to send a secret message to Bob. Alice and Bob agree on a modulus p, a prime number. (They must share the prime p, so they must assume that Eve can get hold of it). Each of them chooses a number co-prime to l(p) = p-1, and computes its inverse. These numbers are not revealed. Alice chooses e_A and d_A, Bob chooses e_B and d_B. Note that our commutation condition is satisfied:

$$Te_ATe_B(x) = x^{eAeB} \bmod p = Te_BTe_A(x)$$

In terms of our analogy, Te_A is Alice putting on her padlock, while Td_A is Alice removing her padlock. Now Alice takes the message x and applies Te_A: she sends $Te_A(x)$ to Bob. Bob applies Te_B and returns $Te_BTe_A(x)$ to Alice. Alice applies Td_A and returns

$$Td_ATe_BTe_A(x) = Td_ATe_ATe_B(x) = Te_B(x)$$

to Bob, who then applies Td_B and recovers $Td_BTe_B(x) = x$, the original message.

Nobody has yet discovered a weakness in this protocol like the weakness we found using one-time pads. In other words, even if Eve intercepts all the messages $Te_A(x)$, $Te_BTe_A(x)$ and $Te_B(x)$ that pass to and fro between Alice and Bob, there is no known easy algorithm for her to discover x (even given the modulus p).

Contrast this with the standard RSA protocol: First, it allows a pair of users to communicate securely, whereas RSA allows any two users in a pool to communicate; secondly, three messages have to be sent, rather than just one; thirdly, what is secret and what is public are different in this case (the prime is public but the exponent is secret).

The security of this protocol depends on the fact that, if $y = x^e \pmod p$, then knowledge of x and y does not allow an easy calculation of e. For suppose that Eve could solve this problem. Recall that Eve knows x^{eA}, x^{eB} and x^{eAeB} (the three messages exchanged during the protocol). If she could use x^{eA} and x^{eAeB} to discover e_B, she could find its inverse d_B modulo $p-1$ and then calculate $(x^{eB})^{dB} \bmod p = x$, the secret message.

Thus, the security of this method depends on the fact that the following problem is hard:

Given x, y, and a prime p such that $y \equiv x^e \bmod p$, find e.

This is known as the discrete logarithm problem, since in a sense e is the logarithm of y to base x (where our calculations are in the integers mod p, rather than in the real numbers as usual). This problem is believed to be at least as difficult as factorization, although (like factorization) it is not known to be NP complete.

If it happens that the order of x mod p is small (so that there are only a few distinct powers of x mod p), then e can be found by exhaustive search. So, to make the problem hard, the order of x should be as large as possible. Ideally, choose x to be a primitive root mod p (an element of order $l(p) = p - 1$).

For instance, suppose that p = 30491 and x = 13. Then $x^2 = 169$, $x^3 = 2197$, $x^4 = 28561$, and $x^5 \equiv 1 \pmod p$. So the discrete logarithm problem is easily solved. On the other hand, 2 is a primitive root mod 30491, so all the powers $2^0, 2^1, \ldots, 2^{30489}$ are distinct and finding which one is a particular element y will be very laborious.

We know that $2^{30490} \equiv 1 \pmod{30491}$, by Fermat's Little Theorem. So the order of 2 must be a divisor of 30490. We factorize 30490 into prime factors: 30490 = 2 * 5 *

3049. So any proper divisor would have to divide the product of two of these primes. So we check that none of 2^{2*5}, 2^{2*3049} and 2^{5*3049} is congruent to 1 mod 30491. So in this case we only have to check three powers of 2, but it is necessary to factorize p - 1.

4 Turtle Chat v1.0 with Vigenere

The application is an e-learning tool for understanding and learning the functionality of the symmetric encryption method Vigenere. The classes which were created/modified are given in *Appendix*. The original code comes from the joining of two programs:

1) A java (AWT) chat application with customizable GUI created by a senior software engineer Jeeva S.

2) A Vigenere application developed by David Little, senior lecturer in Mathematics department of Penn State University.

The application works for all users on the same internal server. To begin the application, ChatServer (Server.jar, Figure 4.1) should be run first. Then run ChatClient (Client.jar, Figure 4.2) which allows a user to login to the chat room by giving username, server name, server port, etc.

Figure 4.1: Chat Server

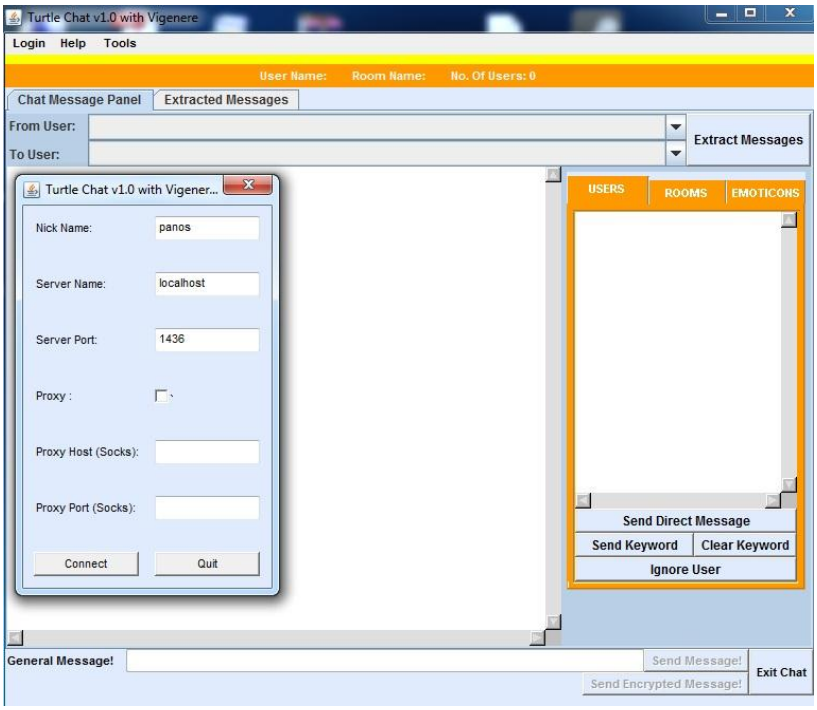


Figure 4.2: Chat Client

In the chat room the user can choose which room to join from the room tab (General, Teen, Music, Party) and can also use emoticons from the emoticons tab:

Figure 4.3: Room and Emoticons tabs

For our educational goals, we will remain in the General room, which is the default room when a user logs on, involving three users for the simulation. Each message appears on General room to all users by pressing ENTER or the "Send Message!" button.

If a user wants to speak in private with another user or not to receive messages from another user, then he selects the desired user and with the "Send Direct Message" button he sends his messages to a private chat or he uses the "Ignore User" button to ignore the other user respectively. For our educational goals, these two options will not be used but encrypted messages will be used which will appear in the General room to all users.

The encrypted chat is one-by-one conversation, where two users share and use the same key/keyword. Each user uses a unique key with each one user that he communicates (e.g. Panos-George: STAR, Panos-Michael: GOLF, George-Michael: APPLE). This is possible by selecting the desired user and using the "Send Keyword" button (e.g. Panos -> George: STAR) (Figure 4.4). For the creation of a new key for those two users, one of them should use the "Clear Keyword" button, where their last key is deleted from the hash table, and the "Send Keyword" button is reused.

Figure 4.4: Send Keyword window

Suppose there is a conversation between Panos and George where they use as key the word "STAR" (Figure 4.5). The third user (Michael), who has not the key, observes on General room their encrypted dialog (Figure 4.6).

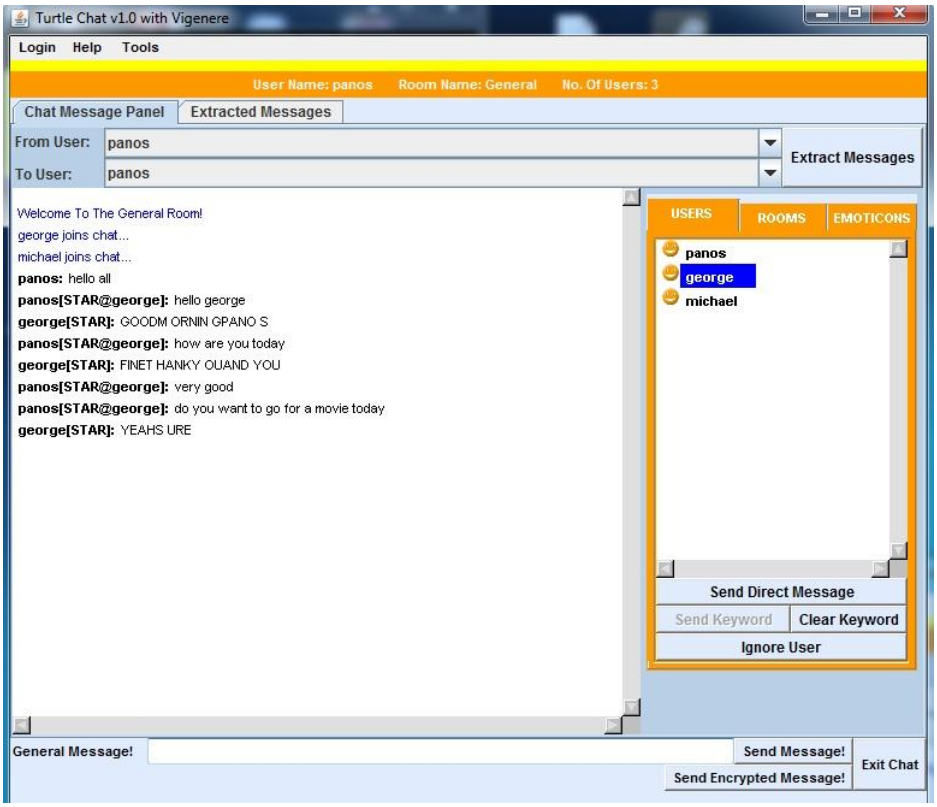


Figure 4.5: The encrypted chat of Panos-George

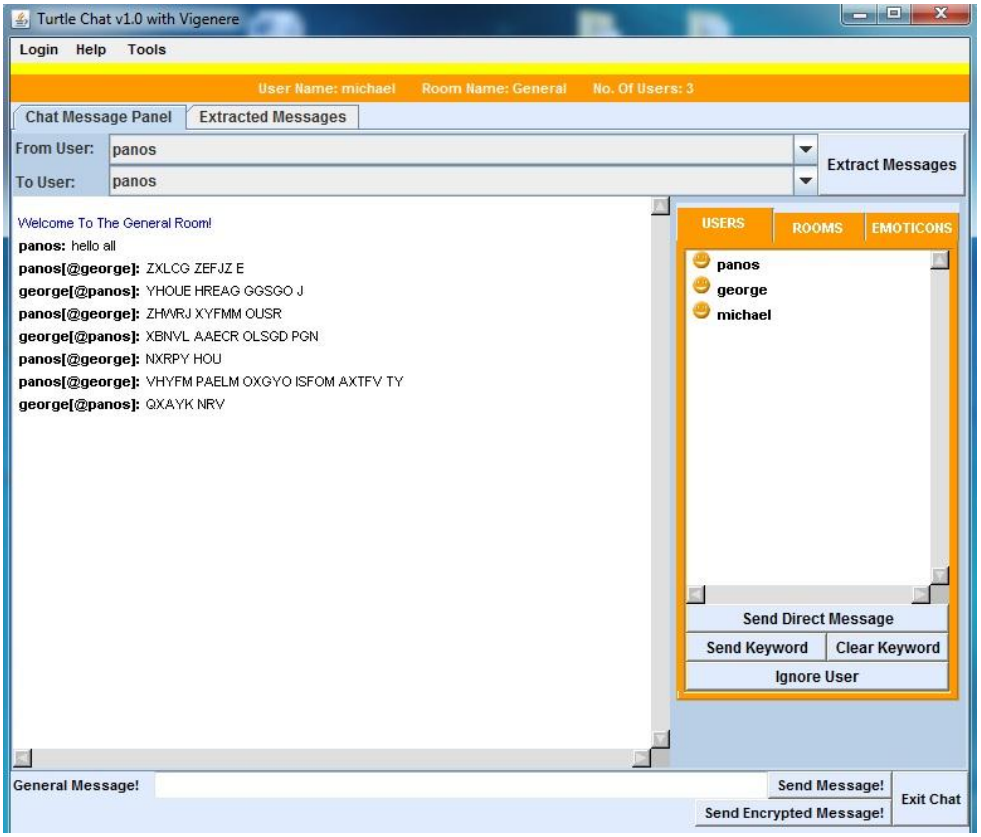


Figure 4.6: The encrypted chat of Panos-George (Michael's side)

Suppose Michael wants to decrypt this dialog. He has to follow then the following steps:

1) On the panel above the chatroom (Figure 4.7) Michael selects the desired two users and extracts their conversation to the "Extracted Messages" tab (Figure 4.8)

2) To find the key of those two users, Michael opens Vigenere program from the Tools bar

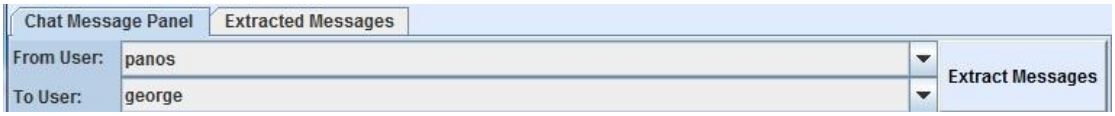


Figure 4.7: Users selection

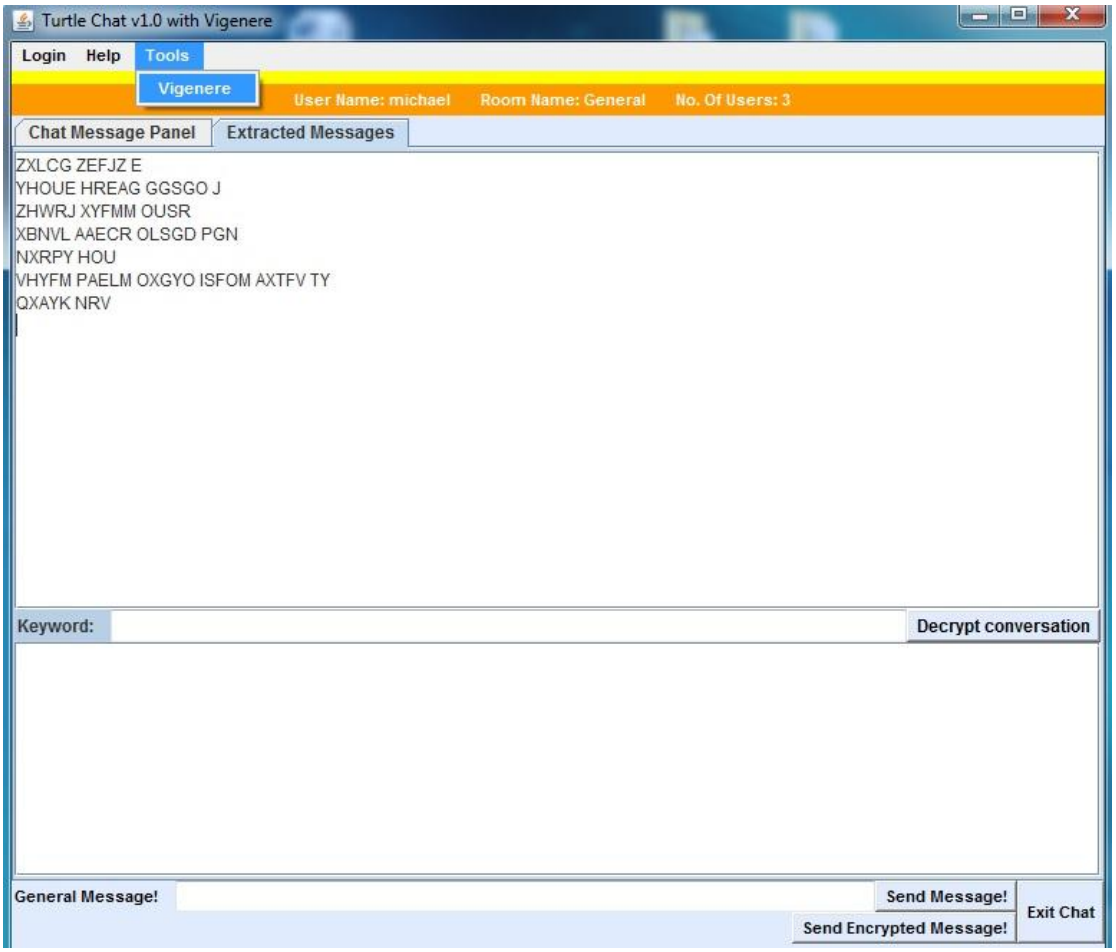


Figure 4.8: Extracted Messages tab

3) In Vigenere application he copy-paste the encrypted dialog from the "Extracted Messages" tab (Figure 4.8) and with the "Break" button, he starts the process (Figure 4.9). Noteworthy that each user can be trained and familiarized more on

Vigenere functionality by using the embedded cyphertexts, which exists in the original code, from "Random Cyphertext" button.

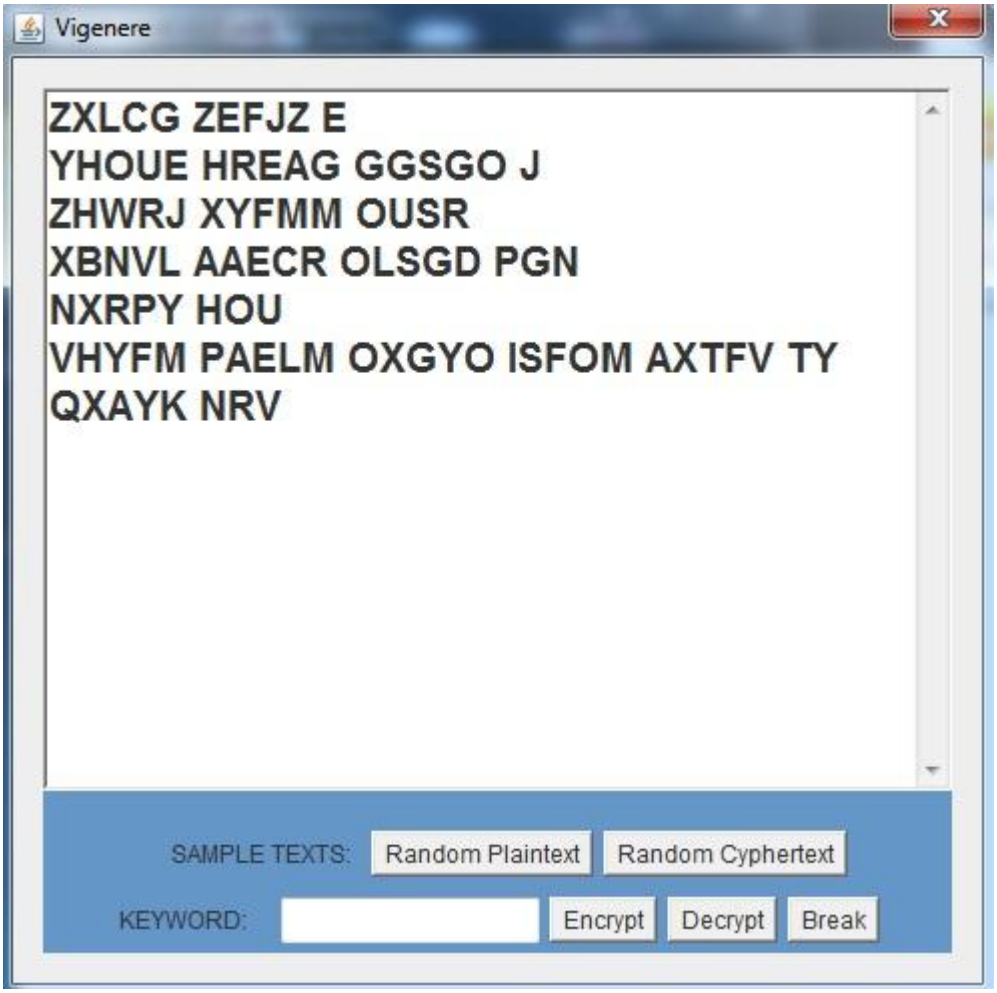


Figure 4.9: Vigenere Application

4) The following window displays (in red) the histogram of observed letter frequencies, alongside a (blue) histogram of english letter probabilities. It also displays the first 25 characters of the text, decoded according to the current keyword (Figure 4.10). The "Period +" button increases the length of the keyword and it should be pressed several times (if necessary) until the red histogram resembles a shifted version of the blue histogram. This given, using the mouse, the red histogram can be dragged to the position where it best approximates the blue histogram. With the "Position" buttons the user selects the desired keyword letter that need to be shifted. The keyboard can be used also to type the image of "E". Pressing the SPACE key will shift the histogram by one position. This way

and observing the SAMPLETEXT on the upper right corner, Michael can see if he has found the correct key.

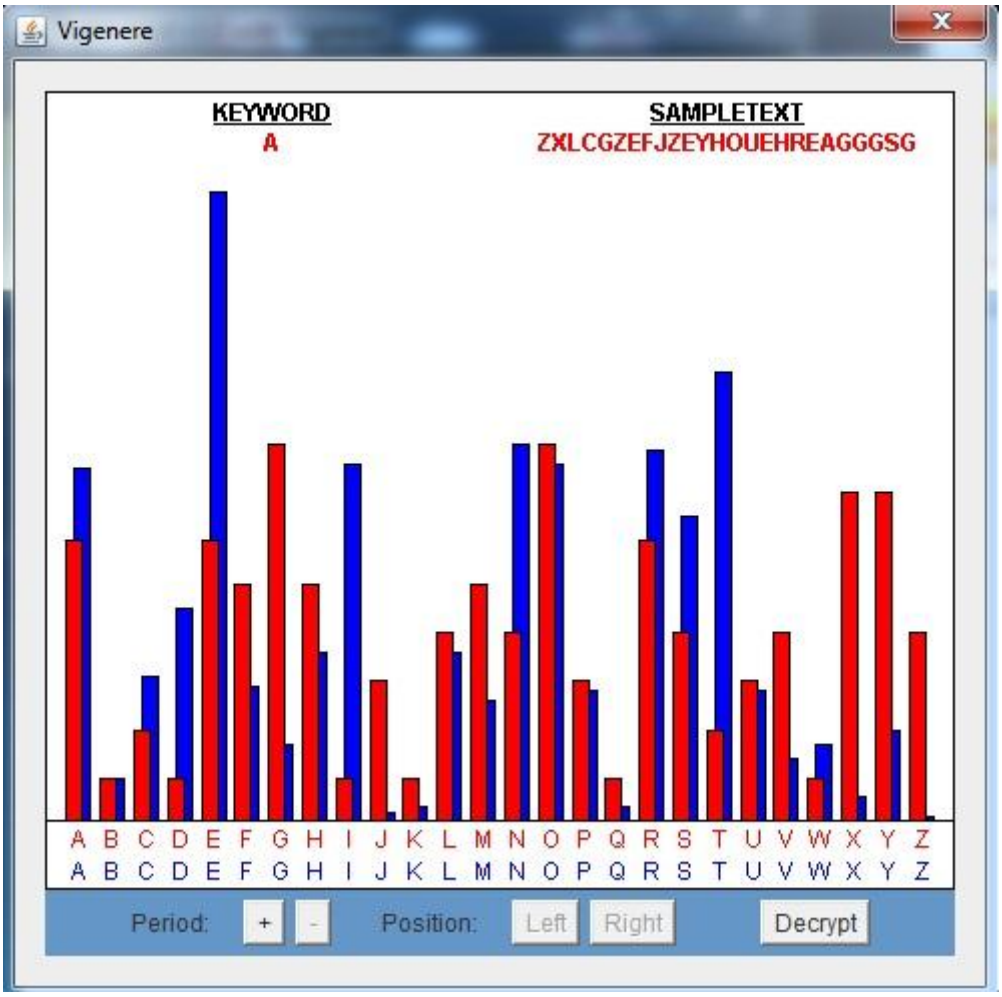


Figure 4.10: Index of Coincidence

5) When the key is found, Michael can return then to the "Extracted Messages" tab (Figure 4.8) where writing the keyword in the designated textbox, he can "Decrypt conversation" and read the decrypted now dialog of Panos and George (Figure 4.11).

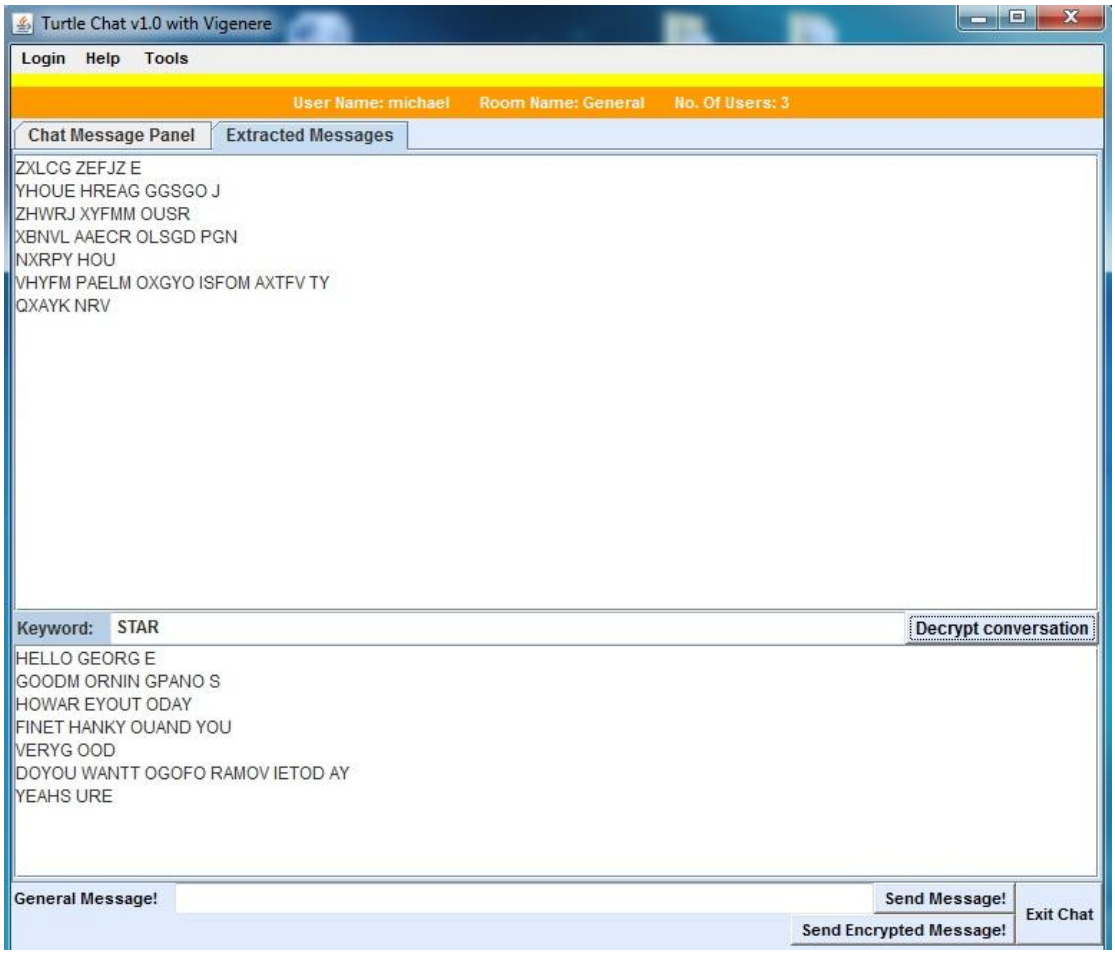
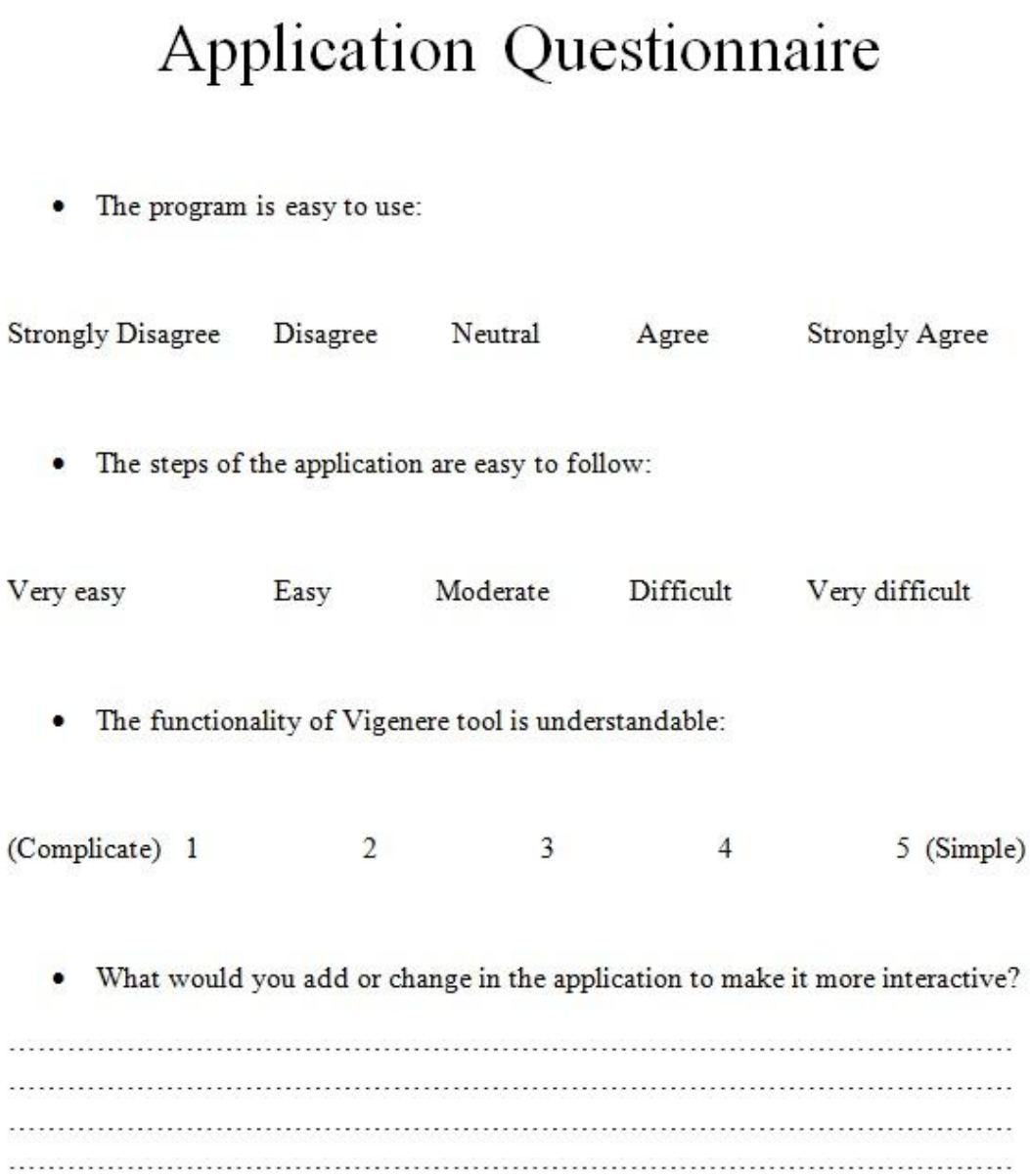


Figure 4.11: Extracted Messages tab (Results)

5 Conclusions

The application was tested by a group of people with moderate, good and very good computer knowledge respectively. After they tested it, they were given a questionnaire to evaluate and comment the application and if it achieved the educational goals:



Application Questionnaire

- The program is easy to use:

Strongly Disagree Disagree Neutral Agree Strongly Agree

- The steps of the application are easy to follow:

Very easy Easy Moderate Difficult Very difficult

- The functionality of Vigenere tool is understandable:

(Complicate) 1 2 3 4 5 (Simple)

- What would you add or change in the application to make it more interactive?

.....
.....
.....
.....

Figure 5.1: Application Questionnaire

For the first question, the majority of the group *agreed* that the program is user-friendly:

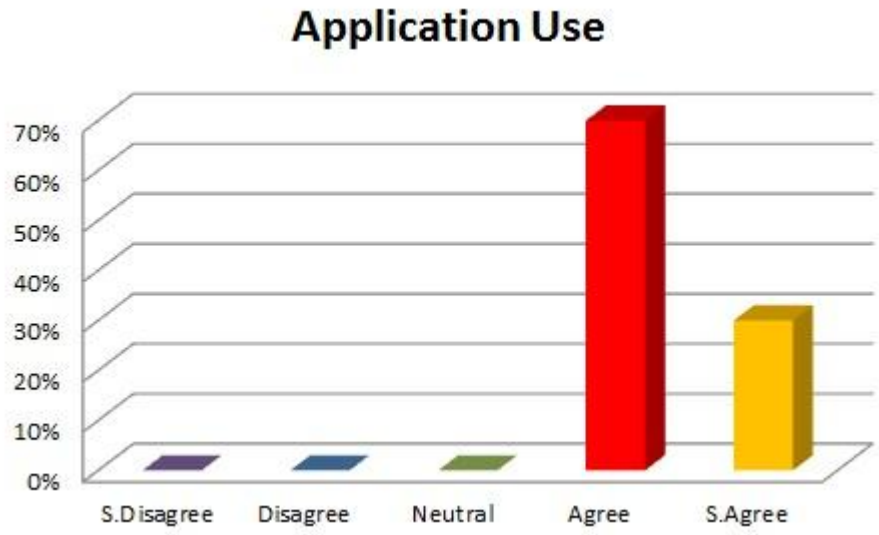


Figure 5.2: Application Use

Also the people found quite *easy* and accessible the use of the application without any difficulties in every step:

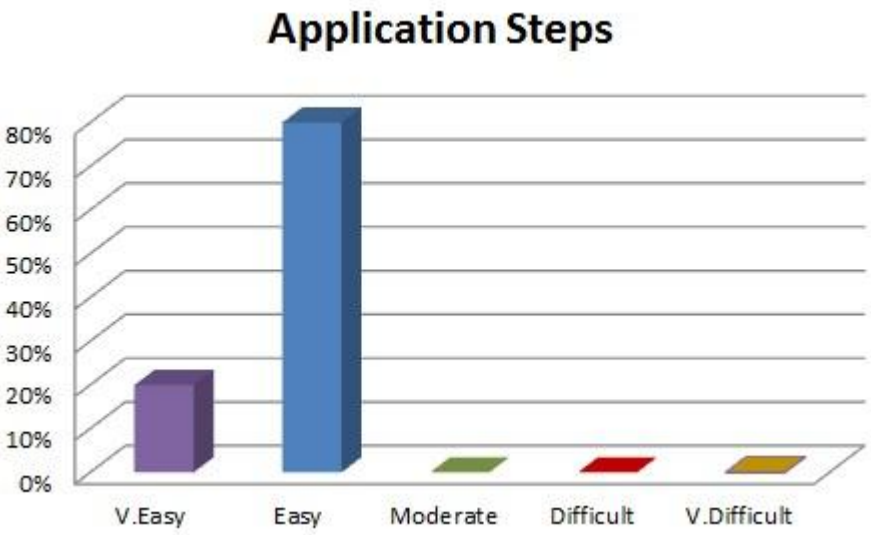


Figure 5.3: Application Steps

Furthermore, the functionality of Vigenere was *understandable* though there were also some minor *complications*:

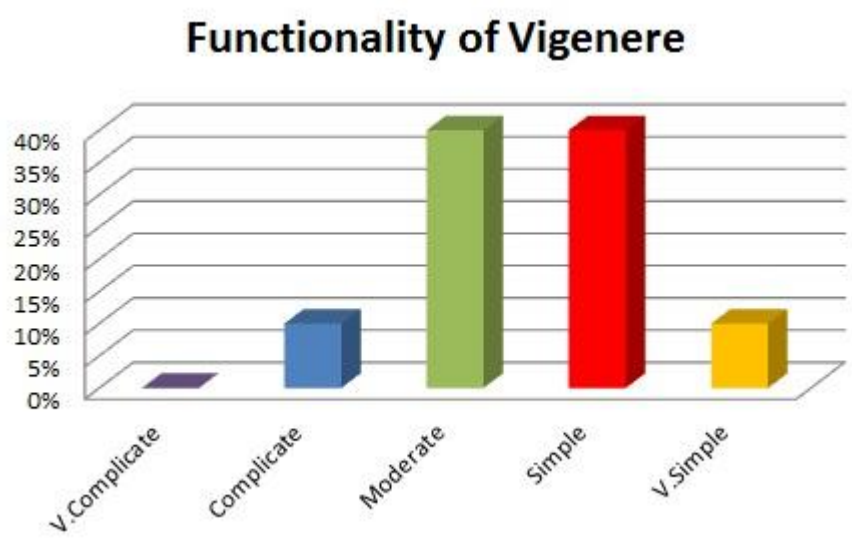


Figure 5.4: Functionality of Vigenere

The objective of the dissertation was to develop an e-learning tool for education in cryptography on a chat application, in order to understand the operation of the symmetric encryption method Vigenere. Taking into account the comments of participants in the questionnaire (e.g. "What would you add or change in the application to make it more interactive?", Figure 5.1), with necessary modifications, the application could be improved by adding new functionalities such as:

- Add more users in a chat party

- Include numbers and special characters as well as more languages

- One important improvement would be to add more encryption methods and functions of cryptography (symmetric - asymmetric), e.g. sending/receiving keyword using the asymmetric encryption method Diffie-Hellman (Diffie – Hellman Key Exchange, Diffie–Hellman key exchange), with main objective to understand the functionality and advantages/disadvantages of each encryption method. One possible solution, for the construction of a class of strong encryption techniques, is delivering only the encrypted information and preventing the transmission of the key. Such systems can be created using synchronous generators keys.

Finally, with the help of simulations demonstrated that the application can operate reliably, fulfilling all the original specifications that were set. Cryptography nowadays has gained immense importance and is one of the major issues discussed and implemented globally.

Bibliography

[1] Ronald L Rivest, "The RC5 Encryption Algorithm", 1997

[2] Gary C. Kessler, "An Overview of Cryptography", 1998

[3] Kimberly C. Harper, Kuanchin Chen, David C. Yen, "Distance learning, virtual classrooms, and teaching pedagogy in the Internet environment"

[4] Peter J. Cameron, "Notes on cryptography"

[5] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, "Handbook of Applied Cryptography", 1996

[6] Shon Harris, "CISSP All-in-One Exam Guide"

[7] Floriane Anstett, Gilles Millerioux, Gerald Bloch, "Chaotic Cryptosystems: Cryptanalysis and Identifiability"

[8] Pascal Junod, "Statistical Cryptanalysis of Block Ciphers", 2005

[9] http://www.logicalsecurity.com/resources/whitepapers/Cryptography.pdf

[10] http://www.cypher.com.au/history.pdf

[11] http://www.cryptool.org/index.php/en/crypto-history-documentationmenu-54.html

[12] http://dsp-psd.pwgsc.gc.ca/collection_2007/nd-dn/D96-1-2004E.pdf

[13] http://jproc.ca/crypto/crypto_watch.html

[14] http://www.rsa.com/rsalabs/node.asp?id=2251

[15] http://searchsecurity.techtarget.com/definition/quantum-cryptography

[16] http://searchsecurity.techtarget.com/definition/elliptical-curve-cryptography

Appendix

ChatClient

ChatClient.java

```
package com.jeeva.chatclient;
……
/**********Chat Client**********/
public class ChatClient extends Frame implements Serializable,Runnable,KeyListener,ActionListener,CommonSettings {
      String alpha="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
      String newline = System.getProperty("line.separator");

    ChatClient()
    {

          // Create an arrayList for storing all encrypted messages
          mHistory = new ArrayList<EncryptedMessageData>();

          // Create hashmap for storing user keywords
          userkeys = new HashMap<String, String>();

          // Create a list for all users have connected to server
          UserHistory = new ArrayList<String>();

        toolkit = Toolkit.getDefaultToolkit();
        if(toolkit.getScreenSize().getWidth() > 778)
            setSize(778, 675);
        else
            setSize((int)toolkit.getScreenSize().getWidth(),(int)toolkit.getScreenSize().getHeight() - 20);
        setResizable(true);
        dimension = getSize();
        setLayout(new BorderLayout());

        setTitle(PRODUCT_NAME);
        addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent evt) { DisconnectChat();System.exit(0);}});

        /*****Loading menubar ***********/
        ……
```

```
  Menu toolsmenu = new Menu("Tools ");
  toolsitem = new MenuItem("Vigenere ");
  toolsitem.addActionListener(this);
  toolsmenu.add(toolsitem);

  menubar.add(loginmenu);
  menubar.add(aboutmenu);
  menubar.add(toolsmenu);
  setMenuBar(menubar);
  ……………

/*********Extraction User Interface Starts..*********/
  Panel ExtractPanel = new Panel(new BorderLayout());
  Panel ExtractUserPanel = new Panel(new BorderLayout());
  Panel ExtractUserLabelPanel = new Panel(new BorderLayout());
     ExtractMessages = new CustomButton(this,"Extract Messages");
     ExtractMessages.addActionListener(this);
     FromUser = new JComboBox();
     ToUser = new JComboBox();
     Label FromUserLabel = new Label("From User:");
     Label ToUserLabel = new Label("To User:");
     ExtractUserLabelPanel.add("North",FromUserLabel);
     ExtractUserLabelPanel.add("South",ToUserLabel);
     ExtractUserPanel.add("North",FromUser);
     ExtractUserPanel.add("South",ToUser);
     ExtractPanel.add("West",ExtractUserLabelPanel);
     ExtractPanel.add("Center",ExtractUserPanel);
     ExtractPanel.add("East",ExtractMessages);
     MessagePanel.add("North",ExtractPanel);

     Panel exPanel = new Panel(new BorderLayout());
     Panel p1= new Panel(new BorderLayout());
     textArea = new JTextArea(5, 20);
     textArea.setEditable(true);
     JScrollPane scrollPane = new JScrollPane(textArea);
     p1.add("Center",scrollPane);
     Panel p11= new Panel(new BorderLayout());
     KeywordPhraze = new TextField();
     p11.add("Center", KeywordPhraze);
     p11.add("West", new Label("Keyword:"));
     CmdDecrypt = new CustomButton(this, "Decrypt conversation");
     CmdDecrypt.addActionListener(this);
     p11.add("East", CmdDecrypt);
     p1.add("South",p11);
     Panel p2= new Panel(new BorderLayout());
     decryptedTextArea = new JTextArea(10, 20);
     decryptedTextArea.setEditable(true);
     JScrollPane scrollPane2 = new JScrollPane(decryptedTextArea);
     p2.add("Center",scrollPane2);
     exPanel.add("Center",p1);
```

```
            exPanel.add("South",p2);

            TabbedMessagePanel.addTab("Extracted Messages", exPanel);
            CenterPanel.add("Center",TabbedMessagePanel);

          tappanel = new TapPanel(this);
        MessagePanel.add("East",tappanel);

        /*********Input Panel Coding Starts..*********/
        Panel InputPanel = new Panel(new BorderLayout());
        Panel TextBoxPanel = new Panel(new BorderLayout());
        Label LblGeneral = new Label("General Message!");
        TxtMessage = new TextField();
        TxtMessage.addKeyListener(this);
        TxtMessage.setFont(TextFont);
        CmdSend = new CustomButton(this,"Send Message!");
        CmdSend.addActionListener(this);
        TextBoxPanel.add("West",LblGeneral);
        TextBoxPanel.add("Center",TxtMessage);
        TextBoxPanel.add("East",CmdSend);

//        Added EncryptedTextBoxPanel to TextBoxPanel
          Panel EncryptedTextBoxPanel = new Panel(new BorderLayout());
          CmdEncrypted = new CustomButton(this,"Send Encrypted Message!");
          CmdEncrypted.addActionListener(this);
          EncryptedTextBoxPanel.add("East",CmdEncrypted);
          TextBoxPanel.add("South",EncryptedTextBoxPanel);

          InputPanel.add("Center",TextBoxPanel);

        Panel InputButtonPanel =new Panel(new BorderLayout());
        CmdExit = new CustomButton(this,"Exit Chat");
        CmdExit.addActionListener(this);

          InputButtonPanel.add("Center",CmdExit);
          InputPanel.add("East",InputButtonPanel);

          Panel EmptyPanel = new Panel();

          InputPanel.add("South",EmptyPanel);

          CenterPanel.add("South",InputPanel);


          add("Center",CenterPanel);

          DisableAll();

          LoginToChat();
      }
```

```
……..
    /**********Button Events *****/
    public void actionPerformed(ActionEvent evt)
    {
        if(evt.getSource().equals(CmdSend))
        {
            if (!(TxtMessage.getText().trim().equals("")))
                SendMessage();
        }

           if(evt.getSource().equals(ExtractMessages)){
              textArea.setText("");
              String mMessages = "";
              for (int i = 0; i < mHistory.size(); i++) {
                 EncryptedMessageData mData= mHistory.get(i);
                 if ((mDa-
ta.getFromUser().equals(FromUser.getItemAt(FromUser.getSelectedIndex())) &&
                       mDa-
ta.getToUser().equals(ToUser.getItemAt(ToUser.getSelectedIndex()))) ||
                    (mDa-
ta.getFromUser().equals(ToUser.getItemAt(ToUser.getSelectedIndex())) &&
                       mDa-
ta.getToUser().equals(FromUser.getItemAt(FromUser.getSelectedIndex())))){
                       mMessages = myString.Doctor(mData.getMessage().trim());
                       mMessages = myString.prettyPrint(mMessages);
                       textArea.append(mMessages + newline);
                 }
              }
           }

           if(evt.getSource().equals(CmdDecrypt)){

              decryptedTextArea.setText("");
              StringTokenizer Tokenizer2 = new StringTokeniz-
er(textArea.getText(),newline);
              while(Tokenizer2.hasMoreTokens())
              {
                    String ptoken = Tokenizer2.nextToken();
                    String encryptedText = myString.Doctor(ptoken);
                    Keyword-
Phraze.setText(KeywordPhraze.getText().trim().toUpperCase());
                    String nonEncryptedText = en-
crypt(encryptedText,KeywordPhraze.getText(),-1);
                    decryptedTextArea.append(nonEncryptedText + newline);
              }

           }

        if(evt.getSource().equals(CmdEncrypted))
        {
```

```
            String tuser = tappanel.UserCanvas.SelectedUser;

                if (!(tuser==null || tuser.equals(""))){
                    String pkey = userkeys.get(tuser);
                    if (pkey == null){
                        pkey = "";
                    }
                    // Also check whether I have set a private key
                    if          ((!(TxtMessage.getText().trim().equals("")))          &&
(!(pkey.trim().equals(""))))){
                            SendEncrMessage(tuser,pkey);
                    }
                }
        }
        …….
        if(evt.getSource().equals(toolsitem))
        {
              vigForm = new VigenereFr();
              vigForm.setVisible(true);
              vigForm.setPreferredSize(new Dimension(100,100));
              vigForm.setSize(new Dimension(500,500));
              vigForm.setTitle("Vigenere");
              vigForm.setAlwaysOnTop(true);
        }


    }

   //text and keyword should be Doctored beforehand
   public String encrypt(String text, String keyword, int sign){
      String ctext="";
      int n = text.length(),p = keyword.length(),l;

      if (p > 0){
         for (int i=0; i<n; i++){
            l=(text.charAt(i)-'A'+sign*(keyword.charAt(i%p)-'A'))%26;
            if (l<0) l+=26;
            ctext+=alpha.substring(l,l+1);
         }
         return myString.prettyPrint(ctext);
      }
      return myString.prettyPrint(text);
   }

    /********** Key Listener Event **********/
    public void keyPressed(KeyEvent evt)
    {
        if((evt.getKeyCode() == 10) && (!(TxtMessage.getText().trim().equals(""))))

        {
```

```
            String tuser = tappanel.UserCanvas.SelectedUser;

                if (!(tuser==null || tuser.equals(""))){
                   String pkey = userkeys.get(tuser);
                   if (pkey == null){
                      pkey = "";
                   }
                   // Also check whether I have set a private key
                   if ((!(TxtMessage.getText().trim().equals(""))) &&
(!(pkey.trim().equals("")))){
                        SendEncrMessage(tuser,pkey);
                   }
                   else{
                      SendMessage();
                   }
                }else{
                   SendMessage();
                }



        }
    }

    public void keyTyped(KeyEvent e){}
    public void keyReleased(KeyEvent e){}

    /******** Function To Send MESS Rfc to Server **********/
    private void SendMessage()
    {
        /********Sending a Message To Server *********/
        SendMessageToServer("MESS "+UserRoom+"~"+UserName+":
"+TxtMessage.getText());
        messagecanvas.AddMessageToMessageObject(UserName+":
"+TxtMessage.getText(),MESSAGE_TYPE_DEFAULT);
        TxtMessage.setText("");
        TxtMessage.requestFocus();
    }

    private void SendEncrMessage(String tuser, String pkey)
    {
          String nonEncryptedText = myString.Doctor(TxtMessage.getText().trim());
          String encryptedText = en-
crypt(myString.Doctor(nonEncryptedText),pkey.trim().toUpperCase(),1);

        /********Sending a Message To Server *********/
          EncryptedMessageData mData= new EncryptedMessageDa-
ta(UserName,tuser,myString.Doctor(TxtMessage.getText().trim()));
          mHistory.add(mData);
```

```
        SendMessageToServer("PMESS      "+UserRoom+"~"+UserName+"¬"+tuser+":
"+encryptedText);
        messagecan-
vas.AddMessageToMessageObject(UserName+"["+pkey+"@"+tuser+"]:
"+TxtMessage.getText(),MESSAGE_TYPE_DEFAULT);
        TxtMessage.setText("");
        TxtMessage.requestFocus();
    }

     /*********Sending a Keyword To UserName **********/
    public void SendKeyword(String ToUserName, String pkey)
    {
          SendMessageToServer("PKEY "+ToUserName+"~"+UserName+": "+pkey);

    }

     /*********Clearing a Keyword To UserName **********/
    public void ClearKeyword(String ToUserName)
    {
        SendMessageToServer("RKEY    "+ToUserName+"~"+UserName+":    "+"Key
cleared");
    }
……….
    /*********Implements the Thread ******************/
    public void run()
    {
        ……
                    /******** MESS RFC Coding Starts ***********/
                    if( ServerData.startsWith("MESS"))
                    {
                        /**** Chk whether ignored user **********/

    if(!(tappanel.UserCanvas.IsIgnoredUser(ServerData.substring(5,ServerData.indexOf
(":")))))){
                            messagecan-
vas.AddMessageToMessageObject(ServerData.substring(5),MESSAGE_TYPE_DEFA
ULT);
                                }

                    }

                    if( ServerData.startsWith("PMESS"))
                    {
                        /**** Chk whether ignored user **********/

    if(!(tappanel.UserCanvas.IsIgnoredUser(ServerData.substring(6,ServerData.indexOf
("->")))))){
                                SplitString      =      Serv-
erData.substring(6,ServerData.indexOf("->"));
```

```
                                    String toUser = ServerData.substring(ServerData.indexOf("->")+2,ServerData.indexOf(":"));
                                    String keyword = userkeys.get(SplitString);

                                    if ((keyword!=null) && (toUser.equals(UserName))){
                                        String encryptedText = myString.Doctor(ServerData.substring(ServerData.indexOf(":")+2, ServerData.length()));
                                        String nonEncryptedText = encrypt(myString.Doctor(encryptedText),keyword,-1);
                                        messagecanvas.AddMessageToMessageObject(ServerData.substring(6,ServerData.indexOf("->")) + "[" + keyword +"]: " +nonEncryptedText,MESSAGE_TYPE_DEFAULT);
                                        EncryptedMessageData mData= new EncryptedMessageData(SplitString,toUser,myString.Doctor(nonEncryptedText));
                                        mHistory.add(mData);
                                    }else{
                                        messagecanvas.AddMessageToMessageObject(ServerData.substring(6,ServerData.indexOf("->")) + "[@" + toUser +"]: " + ServerData.substring(ServerData.indexOf(":")+1),MESSAGE_TYPE_DEFAULT);
                                        EncryptedMessageData mData= new EncryptedMessageData(SplitString,toUser,myString.Doctor(ServerData.substring(ServerData.indexOf(":")+1)));
                                        mHistory.add(mData);
                                    }


                                }

                            }

                            if( ServerData.startsWith("RKEY"))
                            {
                                SplitString = ServerData.substring(5,ServerData.indexOf(":"));
                                userkeys.remove(SplitString);
                                if (tappanel.UserCanvas.SelectedUser.equals(SplitString)){
                                    tappanel.CmdSendKey.setEnabled(true);
                                    tappanel.CmdClearKey.setEnabled(false);
                                }
                            }
……………
                            /******* Private Keyword RFC **********/
                            if( ServerData.startsWith("PKEY"))
                            {
                                SplitString = ServerData.substring(5,ServerData.indexOf(":"));
```

```
                                    userkeys.put(SplitString, ServerData.substring(ServerData.indexOf(":")+2, ServerData.length()));
                                    if (tappanel.UserCanvas.SelectedUser.equals(SplitString)){
                                        tappanel.CmdSendKey.setEnabled(false);
                                        tappanel.CmdClearKey.setEnabled(true);
                                    }

                            }
                        }catch(Exception _Exc) {
                                messagecanvas.AddMessageToMessageObject(_Exc.getMessage(),MESSAGE_TYPE_ADMIN);
                                QuitConnection(QUIT_TYPE_DEFAULT);
                            }
                }
            }

    // Adds User to UserHistory List
    private void AddListItemToUserHistory(String ListItem) {
        boolean exist = false;
        for (int i = 0; i < UserHistory.size(); i++) {
            String user = UserHistory.get(i);
            if (user.equals(ListItem)){
                exist = true;
            }
        }

        if (!exist){
            UserHistory.add(ListItem);
        }
    }

    // Refresh extraction GUI of User List
    private void RefreshGUIUserList() {
        FromUser.removeAllItems();
        ToUser.removeAllItems();

        for (int i = 0; i < UserHistory.size(); i++) {
            FromUser.addItem(UserHistory.get(i));
            ToUser.addItem(UserHistory.get(i));
        }

    }
……
     /***************Global Variable Declarations*****************/
     …….
     MenuItem quititem,aboutitem, toolsitem;
      JTextArea textArea;
      JTextArea decryptedTextArea;
      TextField KeywordPhraze;
```

```
    VigenereFr vigForm;
    HashMap<String, String> userkeys;
   Button ExtractMessages;
    JComboBox FromUser, ToUser;
    List<String> UserHistory;
    List<EncryptedMessageData> mHistory;
}
```


EncryptedMessageData.java

```
package com.jeeva.chatclient;

public class EncryptedMessageData {

   private String fromUser;
   private String toUser;
   private String message;

   public EncryptedMessageData() {

   }

   EncryptedMessageData(String fuser, String tuser, String mes) {
      this.fromUser = fuser;
      this.toUser = tuser;
      this.message = mes;
   }

   public String getFromUser() {
      return fromUser;
   }

   public void setFromUser(String fromUser) {
      this.fromUser = fromUser;
   }

   public String getMessage() {
      return message;
   }

   public void setMessage(String message) {
      this.message = message;
   }

   public String getToUser() {
      return toUser;
   }

   public void setToUser(String toUser) {
```

```
        this.toUser = toUser;
    }
}
```

—●—

ListViewCanvas.java

```
package com.jeeva.chatclient;
……...

    public boolean   mouseDown(Event event, int i, int j)
      {
      ……………

                                String pkey = chatclient.userkeys.get(SelectedUser);
                                if (pkey==null || pkey.equals("")){
                                   chatclient.tappanel.CmdSendKey.setEnabled(true);
                                   chatclient.tappanel.CmdClearKey.setEnabled(false);
                                }
                                else{
                                   chatclient.tappanel.CmdSendKey.setEnabled(false);
                                   chatclient.tappanel.CmdClearKey.setEnabled(true);
                                }

                                if (SelectedUser.equals(chatclient.UserName)){
                                   chatclient.tappanel.CmdSendKey.setEnabled(false);
                                   chatclient.tappanel.CmdClearKey.setEnabled(false);
                                }
……………
```

—●—

TapPanel.java

```
package com.jeeva.chatclient;
…….
/**********Action Listener coding **********/
     public void actionPerformed(ActionEvent evt)
     {

     …….
          if(evt.getSource().equals(CmdSendDirect))
          {
               UserCanvas.SendDirectMessage();
          }

          if(evt.getSource().equals(CmdSendKey))
          {
                String tuser = UserCanvas.SelectedUser;

                if (!(tuser==null || tuser.equals(""))){
```

```
                String pkey = chatclient.userkeys.get(tuser);

            if (pkey==null || pkey.equals("")){
                PrivateKeyDialog pkdialog = new PrivateKeyDialog(chatclient, "",
null, tuser);
                pkdialog.setPreferredSize(new Dimension(100,100));
                pkdialog.setSize(new Dimension(400,150));
                pkdialog.setTitle("Private Keyword");
                pkdialog.setAlwaysOnTop(true);
                pkdialog.setVisible(true);
              }
//                UserCanvas.SendDirectMessage();

                pkey = chatclient.userkeys.get(tuser);

                if ((!(pkey==null)) && (!(pkey.equals("")))){
                  chatclient.SentPrivateMessageToServer("Private keyword Send: " +
pkey,UserCanvas.SelectedUser);
                  chatclient.SendKeyword(UserCanvas.SelectedUser,pkey);
                  CmdSendKey.setEnabled(false);
                  CmdClearKey.setEnabled(true);
                }

            }
        }

        if(evt.getSource().equals(CmdClearKey))
        {
            String tuser = UserCanvas.SelectedUser;

            if (!(tuser==null || tuser.equals(""))){
              chatclient.userkeys.remove(tuser);
              chatclient.ClearKeyword(tuser);
              CmdSendKey.setEnabled(true);
              CmdClearKey.setEnabled(false);

            }
        }
    }
…….
```

ChatServer

ChatCommunication.java

```
package com.jeeva.chatserver;

……
```

```
public class ChatCommunication implements Runnable,CommonSettings
{
    ……
/********Implement the Thread Interface*********/
    public void run()
    {
        while(thread != null)
        {
            try {
                RFC = inputstream.readLine();
                    System.out.println(RFC);
                /*******RFC Checking**************/
…….
                if(RFC.startsWith("MESS"))
                {
                    Par-
ent.SendGeneralMessage(socket,RFC.substring(RFC.indexOf(":")+1),RFC.substring(R
FC.indexOf("~")+1,RFC.indexOf(":")),RFC.substring(5,RFC.indexOf("~")));
                }

                if(RFC.startsWith("PMESS"))
                {
                    Par-
ent.SendGeneralEncryptedMessage(socket,RFC.substring(RFC.indexOf(":")+1),RFC.su
bstring(RFC.indexOf("~")+1,RFC.indexOf("¬")),RFC.substring(6,RFC.indexOf("~")),
RFC.substring(RFC.indexOf("¬")+1,RFC.indexOf(":")));
                }

                if(RFC.startsWith("RKEY"))
                {
```

```
                Par-
ent.SendKeyDelete(RFC.substring(RFC.indexOf("~")+1),RFC.substring(5,RFC.indexO
f("~")));
            }

            if(RFC.startsWith("PKEY"))
            {
                Par-
ent.SendPrivateKeyword(RFC.substring(RFC.indexOf("~")+1),RFC.substring(5,RFC.in
dexOf("~")));
            }
……..
}catch(Exception _Exc) { Par-
ent.RemoveUserWhenException(socket);QuitConnection();}
        }
    }
…….
}
```

Vigenere

VigenereFr.java

```
package vigenere;

import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.Color;
import java.awt.Event;
import java.awt.Panel;
import javax.swing.JDialog;
import javax.swing.JFrame;
```

```
import javax.swing.JInternalFrame;

/**
 *
 * @author Administrator
 */
public class VigenereFr extends JDialog {

    CryptVig cryptvig;
    BreakVig breakvig;
    Panel cards;
    String currentcard,CYPHERTEXT;

    /** Creates new form VigenereForm */
    public VigenereFr() {
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        cryptvig = new CryptVig(CryptVig.STANDARD);
        breakvig = new BreakVig();
            //breakvig.vigstatspanel.i = createImage(800,640);
            //breakvig.vigstatspanel.g = breakvig.vigstatspanel.i.getGraphics();

        setBackground(new Color(100,150,200));
        setLayout(new BorderLayout());
        cards = new Panel();
            cards.setLayout(new CardLayout(15,15));
            cards.add("VIGENERE",cryptvig);
            cards.add("BREAKVIG", breakvig);
        add("Center",cards);
```

```
        currentcard="VIGENERE";
        ((CardLayout) cards.getLayout()).show(cards,currentcard);

        CYPHERTEXT="";
        repaint();
    }

    public boolean action(Event evt, Object arg){
        if (currentcard=="VIGENERE"){
            if ("Break".equals(arg)) {
                currentcard="BREAKVIG";
                ((CardLayout) cards.getLayout()).show(cards,currentcard);

                String key=myString.Doctor(cryptvig.tf.getText());
                if (key.length()>breakvig.maxperiod) key = key.substring(0,breakvig.maxperiod);
                if (key.length()<=0) key="A";
                breakvig.vigstatspanel.position=1;
                breakvig.vigstatspanel.KEY = key;

                breakvig.up.setEnabled( false );
            breakvig.down.setEnabled( true );
                breakvig.left.setEnabled( false );
            breakvig.right.setEnabled( true );
                if (key.length()<breakvig.maxperiod) breakvig.up.setEnabled( true );
                if (key.length()==1){
                   breakvig.down.setEnabled( false );
                breakvig.right.setEnabled( false );
                }
```

```
            CYPHERTEXT=myString.Doctor(cryptvig.ta.getText());
            breakvig.cyphertext=CYPHERTEXT;
            break-
vig.vigstatspanel.SAMPLETEXT=CYPHERTEXT.substring(0,Math.min(25,CYPHER
TEXT.length()));
            break-
vig.vigstatspanel.cstats=breakvig.statistic(CYPHERTEXT,key.length(),1);
            breakvig.vigstatspanel.repaint();
            return true;
         } else {
            return cryptvig.action(evt,arg);
         }
      } else if (currentcard=="BREAKVIG"){
         if ("Decrypt".equals(arg)) {
            currentcard="VIGENERE";
            ((CardLayout) cards.getLayout()).show(cards,"VIGENERE");
            String guess=breakvig.vigstatspanel.KEY;
            cryptvig.tf.setText(guess);
            cryptvig.ta.setText("Computing plaintext...");
            cryptvig.ta.setText(cryptvig.encrypt(CYPHERTEXT,guess,-1));
            return true;
         } else {
            return breakvig.action(evt,arg);
         }
      }
      return super.handleEvent(evt);
   }

   public boolean keyDown(Event evt, int key){
      if (currentcard=="BREAKVIG"){
```

```
            return breakvig.vigstatspanel.keyDown(evt,key);
        } else {
            return super.keyDown(evt,key);
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new VigenereFr().setVisible(true);
            }
        });
    }
}
```

VigStatsPanel.java

```
package vigenere;

/**
 * @author David Little and Mike Zabrocki
 * @version VigStatsPanel1.0
 * Last updated Apr. 24, 1999
 * Compiled with JDK1.0.2
 */

import java.awt.*;
```

```
import java.lang.*;

public class VigStatsPanel extends Panel{

   Graphics g;
   Image i;
   String KEY,SAMPLETEXT;
   int[] cstats;
   String ALPHA = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   int position,xnot,ynot;
   float[] english =
{73,9,30,44,130,28,16,35,74,2,3,35,25,78,74,27,3,77,63,93,27,13,16,5,19,1};


   public VigStatsPanel(){
      KEY="test";
      SAMPLETEXT="test";

      setBackground(Color.white);

      position=1;

      cstats = new int[26];
      for (int i=0; i<26; i++) cstats[i]=0;
   }


   public void update(Graphics graph){
      paint(graph);
   }



   public void paint(Graphics graph) {
```

```
int n,w,shift;
int width = getSize().width;
int height = getSize().height;
int h = height-85;
double w1 = (double)(width)/27.0;
String letter,tmp;

i = createImage(width,height);
g = i.getGraphics();

g.setPaintMode();
g.setColor(Color.white);
g.fillRect(0,0,width,height);
g.setColor(Color.black);
g.drawRect(0,0,width-1,height-1);
g.drawRect(0,0,width-1,height-35);

for (int i=1; i<27; i++){
   n = (int)(i*w1);

   //draw english statistics & letters
   g.setColor(Color.blue);
   g.fillRect(n-(int)(w1/8),height-35-(int)(h*english[i-1]/130),
         (int)(w1/2),(int)(h*english[i-1]/130));
   letter=ALPHA.substring(i-1,i);
   g.drawString(letter,n-g.getFontMetrics().stringWidth(letter)/2,height-5);
   g.setColor(Color.black);
   g.drawRect(n-(int)(w1/8),height-35-(int)(h*english[i-1]/130),
         (int)(w1/2),(int)(h*english[i-1]/130));

   //draw cypher statistics & letters

   shift=(i-1+KEY.charAt(position-1)-'A')%26;
```

```
    g.setColor(Color.red);
    letter = ALPHA.substring(shift,shift+1);
    g.drawString(letter,n-g.getFontMetrics().stringWidth(letter)/2,height-21);
    if (cstats[shift]>130){
       g.fillRect(n-(int)(3*w1/8),height-35-h,(int)(w1/2),h);
       g.setColor(Color.black);
       g.drawRect(n-(int)(3*w1/8),height-35-h,(int)(w1/2),h);
       g.drawString("+",n-(int)(w1/4)-1,47);
    } else {
       g.fillRect(n-(int)(3*w1/8),height-35-(int)(h*(float)cstats[shift]/130),
             (int)(w1/2),(int)(h*(float)cstats[shift]/130));
       g.setColor(Color.black);
       g.drawRect(n-(int)(3*w1/8),height-35-(int)(h*(float)cstats[shift]/130),
             (int)(w1/2),(int)(h*(float)cstats[shift]/130));
    }
}

//draw keyword
g.setColor(Color.black);
g.setFont(new Font("Helvetica",Font.BOLD,12));
w = g.getFontMetrics().stringWidth("KEYWORD");
g.drawString("KEYWORD",width/4-w/2,15);
g.drawLine(width/4-w/2,16,width/4+w/2,16);

w=width/4-g.getFontMetrics().stringWidth(KEY)/2;
for (int i=0; i<KEY.length(); i++){
   g.setColor(Color.black);
   if (i==position-1) g.setColor(Color.red);
   g.drawString(KEY.substring(i,i+1),w,30);
   w+=g.getFontMetrics().stringWidth(KEY.substring(i,i+1));
}

//draw sampletext
g.setColor(Color.black);
```

```
    w = g.getFontMetrics().stringWidth("SAMPLETEXT");
    g.drawString("SAMPLETEXT",3*width/4-w/2,15);
    g.drawLine(3*width/4-w/2,16,3*width/4+w/2,16);
    //encrypt sampletext
    tmp="";
    for (int i=0; i<SAMPLETEXT.length(); i++){
      n = (SAMPLETEXT.charAt(i)-KEY.charAt(i%KEY.length()))%26;
      if (n<0) n+=26;
      tmp+=ALPHA.substring(n,n+1);
    }
    w=3*width/4-g.getFontMetrics().stringWidth(tmp)/2;
    for (int i=0; i<tmp.length(); i++){
      g.setColor(Color.black);
      if (i%KEY.length()==position-1) g.setColor(Color.red);
      g.drawString(tmp.substring(i,i+1),w,30);
      w+=g.getFontMetrics().stringWidth(tmp.substring(i,i+1));
    }

    graph.drawImage(i,0,0,this);
  }


  public void editKey(int n,String to){
    KEY = KEY.substring(0,n)+to.toUpperCase()+KEY.substring(n+1);
  }


  public boolean keyDown(Event evt, int key){
    int n;
    if (key>='A' && key<='Z'){
      n = key-'A'-4;
      if (n<0) n+=26;
      editKey(position-1,ALPHA.substring(n,n+1));
    } else if (key>='a' && key<='z'){
```

```
        n = key-'a'-4;
        if (n<0) n+=26;
        editKey(position-1,ALPHA.substring(n,n+1));
    } else if (key==' '){
        n = (KEY.charAt(position-1)-'A'+1)%26;
        editKey(position-1,ALPHA.substring(n,n+1));
    }
    repaint();
    return true;
}

public boolean mouseDown(Event event, int x, int y){
    xnot=x;
    ynot=y;
    return true;
}


public boolean mouseDrag(Event event, int x, int y){
    int n;
    int w=getSize().width/27;
    if (ynot < 50 || ynot > getSize().height-35) return true;
    else{
        //move cypher statistics right
        if (x-xnot>w){
            n = (KEY.charAt(position-1)-'A'-((x-xnot)/w))%26;
            if (n<0) n+=26;
            editKey(position-1,ALPHA.substring(n,n+1));
            xnot+=w*((x-xnot)/w);
            repaint();
            return true;
        }

        //move cypher statistics left
```

```
            if (xnot-x>w){
                n = (KEY.charAt(position-1)-'A'-((x-xnot)/w))%26;
                if (n<0) n+=26;
                editKey(position-1,ALPHA.substring(n,n+1));
                xnot-=w*((xnot-x)/w);
                repaint();
                return true;
            }
        }
        return true;
    }
}
```