PERFORMANCE COMPARISON OF CRYPTOGRAPHIC FUNCTIONS IN SYMMETRIC KEY CRYPTOGRAPHY, ASYMMETRIC KEY CRYPTOGRAPHY AND HASH ALGORITHMS

BY

JAYAMALAR.T

Reg. No. 05MP40

A DISSERTATION SUBMITTED TO THE AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION FOR WOMEN (DEEMED UNIVERSITY), COIMBATORE-641 043 IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF PHILOSOPHY IN COMPUTER SCIENCE

SEPTEMBER 2006

Certificate

CERTIFICATE

This is to certify that the dissertation entitled "**Performance comparison of cryptographic functions in symmetric key cryptography, asymmetric key cryptography and hash algorithms"** submitted to the Avinashilingam Institute for Home Science and Higher Education for Women – Deemed University, Coimbatore -43, in partial fulfillment of the requirements for the award of the degree of **Master of Philosophy in Computer Science** is a record of original research work done by **T. Jayamalar (Reg. No. 05MP40),** during the period of her study in the Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women – Deemed University, Coimbatore -43, under my supervision and guidance and the dissertation has not formed the basis for the award of any Degree/ Diploma/Associate ship/ Fellowship or other similar title to any candidate of any other University.

Place:Coimbatore-43

Date: 15.09.06

Guide 15/9/06.

Forwarded

Head of the Department 15/9/06.

Declaration

DECLARATION

I here by declare that the dissertation entitled **"PERFORMANCE COMPARISON OF CRYPTOGRAPHIC FUNCTIONS IN SYMMETRIC KEY CRYPTOGRAPHY, ASYMMETRIC KEY CRYPTOGRAPHY AND HASH ALGORITHMS"** submitted to the Avinashilingam Institute for Home Science and Higher Education for Women – Deemed University, Coimbatore -43, in partial fulfillment of the requirements for the award of the degree of Master of Philosophy in Computer Science is a record of original research work done by me under the supervision and guidance of

Tmt. G. Padmavathi M.Sc.,M.Phil.,(P.hD)., Lecturer (Selection Grade), and Head of the Department, Department of Computer Science, Avinashilingam Deemed University and that it has not formed the basis for the award of any Degree/Diploma/Associate ship/Fellowship or similar title to any candidate of any other university.

T. Jayamalar

Place : Coimbatore – 43

Date : 15.09.06

Signature of the Candidate

(T.Jayamalar)

Countersigned

G. Padmavathi 15/9/06.

Signature of the Guide

(Tmt.G.Padmavathi)

Acknowledgement

ACKNOWLEDGEMENT

I would like to express my sincere thanks to God Almighty, for his constant love and grace that he has showered upon me.

I am very grateful to **Dr.K.Kulandaivel**, M.A., M.A., (Ohio State), Ph.D., (Madras), Chancellor, Avinashilingam Institute for Home Science and Higher Education for Women – Deemed University, Coimbatore, for his kind patronage and all facilities offered to do this research work.

My heartfelt thanks to **Dr.Saroja Prabhakaran**, M.A., Dip.Ed, (Madras), Ph.D., (Mother Theresa), Vice Chancellor for providing all facilities necessary for the study.

I express my gratitude to Registrar **Dr.Gowri Ramakrishnan**, M.Sc (Madras), M. Phil, Ph.D., (Avinashilingam) for providing all the facilities needed for the study.

I profusely thank the former Dean **Dr. SivagamaSundari**, and present Dean **Dr. Parvatham,** for providing all facilities needed for the study.

I wish to place my deep sense of gratitude to my supervisor **Tmt.G.Padmavathi M.Sc.,** M.Phil., P.hd, Lecturer(Selection Grade), and Head of the Department, Department of Computer Science, whose help, stimulating suggestions, extensive comments, strong support, meticulous care and encouragement helped me in all time of my research work.

I owe my special thanks to all staff members, Department of Computer Science for their constant support rendered throughout the study.

I feel a deep sense of gratitude for my parents who formed part of my vision and taught me the good things that really matter in life. Looking back, I am really grateful for all the bitter-sweet moments, the experience gained, and the insight obtained which leave me enriched as a person and as a student of Computer Science.

Table Of Contents

CONTENTS

Chapter No.	Title	Page No.
	ABSTRACT	
1.	INTRODUCTION	01
1.1	Motivation	02
1.2	Objectives	03
1.3	About Cryptography	03
1.4	Terminologies Used	16
1.5	Organization Of Thesis	18
2.	REVIEW OF LITERATURE	20
2.1	History of cryptography	20
2.2	Hardware accelerators for cryptography	24
2.3	Mathematical part of cryptography	27
2.4	Recent trends in cryptography	29
2.5	Ecc-accelerator cards	32
2.6	Hybrid cryptography	34
2.7	Conclusion	37
3.	METHOLDOLOGY	38
3.1.	Symmetric cryptography	38
3.1.1.	Blowfish	39
3.1.2.	International Data Encryption Algorithm (IDEA)	41
3.1.3.	Data Encryption Standard (DES)	43
3.1.4.	Triple DES	46
3.1.5.	GOST	47
3.1.6.	Serpent	48

3.1.7. Tiny Encryption Algorithm (TEA) 50
3.1.8. Skipjack 51
3.1.9. Twofish 52

3.2. Hash function cryptography 53
3.2.1. Message Digest Algorithm 2 56
3.2.2. Message Digest Algorithm 5 56
3.2.3. Race Integrity Primitives Evaluation Message Digest 57
3.2.4. Secure Hash Algorithm 59
3.3. Asymmetric cryptography 60
3.3.1. Problems with symmetric algorithms 60
3.3.2. General working of asymmetric cryptography 61
3.3.3. Rivest-Shamir-Adelman Algorithm 62
3.3.4. Elliptical Curve Cryptography 65

4. RESULTS AND DISCUSSION 69
4.1. Performance Evaluation of Symmetric Cryptography 69
4.2. Performance Evaluation of Hash Function Cryptography 71
4.3. Performance Evaluation of Asymmetric Cryptography 78
4.4. Conclusion 79

5. SUMMARY AND CONCLUSION 80

BIBLIOGRAPHY
APPENDIX

Abstract

ABSTRACT

The popularization of the Internet has enabled extensive communication between computers worldwide. There are enormous benefits due to this. However, it also raises many security considerations. Cryptography is a fundamental technology used to provide security of computer networks and there is currently a widespread engineering effort to incorporate cryptography into various aspects of Internet. Deciding and choosing a cryptographic algorithm for applications is a major decision making process and has to be done cautiously. Effective decision can be made only if the functioning, merits and demerits of the algorithms are known before. This dissertation attempts to throw light in this facet of decision.

Cryptography is probably best thought of as the art of secret writing that explains how a message can be written in such a way that only the intended receiver can actually read it. The cryptographic functions are categorized in to three main areas namely, symmetric, asymmetric and hash function cryptography. In this research work, implementation of nine symmetric, block encryption algorithms namely: Blowfish, Twofish, Data Encryption Standard (DES), Triple-Data Encryption Standard, Serpent, GOST, Tiny Encryption Algorithm (TEA), International Data Encryption Algorithm(IDEA) and Skipjack, hash function cryptography, namely, Message Digest Version 2(MD2), Message Digest Version 5 (MD5), Race Integrity Primitives Evaluation Message Digest (RIPEMD160) and Secure Hash Algorithm (SHA), asymmetric cryptographic algorithms Rivest – Shamir-Adelman (RSA) and Elliptic curve cryptography (ECC) are presented. All these algorithms are implemented under windows platform.

Execution speed is considered as a significant factor while evaluating the performance of any algorithm and therefore, in this research work, a comparison is performed between these algorithms based on the CPU execution time. The performance of the algorithms is evaluated in terms of the processing time

required in the kernel and user space for generating the secret key, encryption and decryption operations. Results show that, among the symmetric algorithms the Blowfish algorithm is the fastest, followed by the DES algorithm then the Triple-DES algorithm, when compared with the other algorithms. When hash function cryptography are concerned, RIPEMD gives better performance compared to SHA and MD families. When comparing the two famous, state-of-the-art public key cryptography, the results indicate that RSA can be replaced by ECC.

From the results obtained, one can either embed a security module within the processor or to dedicate a special server to be responsible for providing cryptographic services to highly secured environments. The following are the major conclusions derived:

- Asymmetric encryption provides more functionality than symmetric encryption, at the expense of speed and hardware cost.
- Symmetric encryption provides a cost-effective and efficient method of securing data without compromising security. Hence these methods should be considered as the correct and most appropriate security approach for applications like password protection, secure telnet connection etc.
- In some instances, the best possible solution may be the complementary use of both symmetric and asymmetric encryption.
- Cryptographic hash functions are suitable to use in various information security applications such as authentication, message integrity and so on.

Introduction

1. INTRODUCTION

Nowadays, nearly all companies, government agencies and home users depend on computer systems and communication systems such as the Internet and Intranet. The advent of computers and networks has completely changed the way in which people live and work. The expansion of the worldwide communication network such as the Internet and the increased dependency on digitized information in society makes information more vulnerable to abuse (Kim and Lee, 2004). If there are security problems in these information systems, users will fear that their sensitive information may be monitored and business secrets stolen. For these reasons, it is important to make information systems secure by protecting data and resources from malicious acts, Crypto (cryptography) algorithms are the core of such security systems (Paul *et al.*, 1996).

Different methods have been developed to assure that only the sender and the addressee would be able to read a message, while it would be illegible or without significant meaning to a third party. Today, this practice continues with more fervor. Wireless, wired and optical communication networks are able to transport unimaginable amounts of data and thus privacy of information and security of the network are of utmost concern mainly, because a good part of the transported information may be very sensitive and/or confidential. Confidentiality of information has been particularly popularized with the explosive growth of the Internet, which has touched most people's lives.

By encoding a message using crypto algorithms, users can make information transmitted over communication systems almost impossible to read, even if such information is intercepted for malicious purposes. It is fairly easy to implement crypto algorithms in software, but many different algorithms exist and each has its own merits and demerits. For this reason, it becomes necessary to study the various crypto algorithms available. This chapter introduces

cryptography as a technology for information security and also presents the main objective of the dissertation, studying various cryptographic models, understand its working and compare it with the latest elliptic curve cryptosystem.

1.1. MOTIVATION

The Internet is an open network architecture with computer-based nodes. Without network security, it is vulnerable to attackers and hackers. Wireless networks transmit messages in unguarded space and thus are also subject to malicious attacks. Therefore, the issues of information privacy and network security are equally important. As a result, the development of unbreakable cipher keys, cipherkey distribution, identification of malicious attacks, source authentication, physical-link signature identification and countermeasures have become the major thrust of research efforts with regard to cyber-security. This research work focuses on encryption and decryption of messages and/or files, or in other words, the field of cryptography.

Cryptography is considered as an important technology, a technique essential for securing and generating trust in digital transactions). During transmission no person wants to expose their message to others or to be forged by others. Therefore, the current market scenario requires security measures that can be used to protect data during transmissions.

For this purpose, extensive use of cryptography can be used to achieve security. Various algorithms are proposed and are being used in the field of cryptography and researches are still being widely carried out in this field. For this reason nine of symmetric key, four of hash functions and two of asymmetric algorithms are considered and analyzed in detail.

1.2. OBJECTIVES

This project is based on a study of the processes involved in cryptography. Cryptography is, put simply, the art of secret writing. In the dissertation, 9 popular encryption algorithms (Blowfish, IDEA, DES, Triple DES (3DES), Gost, Skipjack, TEA, Serpent and Twofish), six popular hash algorithms (SHA-1, SHA-256, MD2, MD4, MD5, and RipeMD) are studied and analyzed with the working of the public key encryption technique RSA and the most modern encryption algorithm elliptic curve cryptography. Based on the above, the dissertation is formulated with the following objectives. They are,

1. To examine the working of the symmetric algorithms like Blowfish, IDEA, DES, Triple DES (3DES), Gost, Skipjack, TEA, Serpent and Twofish.
2. To examine the working of hash algorithms SHA-1, SHA-256, MD2, MD4, MD5, and RipeMD.
3. To examine the working of the famous public key cryptography algorithm, RSA.
4. To examine the working of the Elliptic curve cryptography algorithm, that can be used when large amount of data has be crypted.
5. To analyze the performance of the algorithms mentioned in (1), (2), (3) and (4).
6. To use these techniques to keep data confidential, authenticate the sender of a message, and to ensure that the data has not been altered during transit.

1.3. ABOUT CRYPTOGRAPHY

1.3.1. Definition

Cryptography (or cryptology; derived from Greek work meaning "hidden") is a discipline of mathematics concerned with information security and related

issues, particularly encryption, authentication, and access control. Its purpose is to hide the meaning of a message rather than its existence. In modern times, it has also branched out into computer science. Cryptography is central to the techniques used in computer and network security for such things as access control and information confidentiality. Cryptography is used in many applications that touch everyday life; the security of ATM cards, computer passwords, and electronic commerce all depend on cryptography (Wikipedia, The Free Encyclopedia, 2006). The term encryption and decryption is often used in place of cryptography (Wikipedia, 2006). The following figure (Fig. 1.1) shows the process of cryptography.

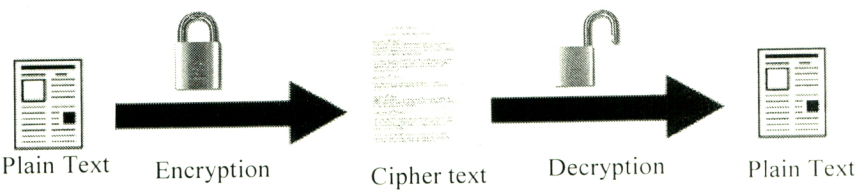


Fig. 1.1. : PROCESS OF CRYPTOGRAPHY

Data that can be read and understood without any special measures is called plaintext or cleartext. The method of disguising plaintext in such a way as to hide its substance is called encryption. Encrypting plaintext results in unreadable gibberish called ciphertext. One use encryption to make sure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. The process of reverting ciphertext to its original plaintext is called decryption.

1.3.2. Need for Cryptography

When designing a cryptographic model, the designer should be very clear regarding exactly what services the protocol intends to serve and should explicitly specify them as well. Within the context of any application-to-application

communication, there are some specific security requirements to satisfy (Eric *et al.*, 2003).

The explicit identification and specification will not only help the designer to choose correct cryptographic primitives or algorithms, but also help an implementer to correctly implement the model. Often, an identification of services to the refinement level of the general services given is not adequate, and further refinement of them is necessary.

Computer and network security research and development have focused on three or four general security services that encompass the various functions required of an information security facility. One useful classification of security services is the following :

- **Confidentiality :** Ensures that the information in a computer system and transmitted information are accessible only for reading by authorized parties. This type of access includes printing, displaying and other forms of disclosure, including simply revealing the existence of an object.

- **Authentication :** Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.

- **Integrity :** Ensures that only authorized parties are able to modify computer system assets and transmitted information. Modification includes writing, changing, changing status, deleting, creating and delaying or replaying of transmitted messages.

- **Nonrepudiation :** Requires that neither the sender nor the receiver of a message be able to deny the transmission.

- **Access control :** Requires that access to information resources may be controlled by or for the target system.

- **Availability :** Requires that computer system assets be available to authorized parties when needed.

Attacks on the security of a computer system or network are best characterized by viewing the function of the computer system as providing information. In general, there is a flow of information from a source to a destination (Fig. 1.2)

FIG. 1.2. : NORMAL FLOW

There are four categories of attacks, namely, Interruption, Interception, Modifications and Fabrication which are illustrated diagrammatically in Fig. 1.3a, 1.3b, 1.3c and 1.3d respectively and are explained below.

- **Interruption :** An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on availability. Examples include destruction of a piece of hardware, such as a hard disk, the cutting of a communication line, or the disabling of the file management system.

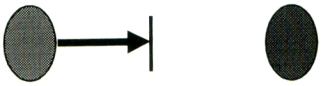


FIG. 1.3a : INTERRUPTION

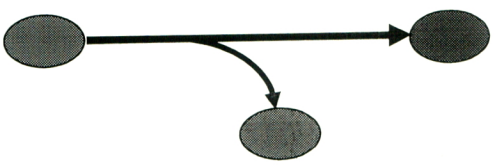


FIG. 1.3b : INTERCEPTION

- **Interception :** An unauthorized party gains access to an asset. This is an attack on confidentiality. The unauthorized party could be a person, a program, or a

computer. Examples include wiretapping to capture data in a network, and the illicit copying of files or programs.

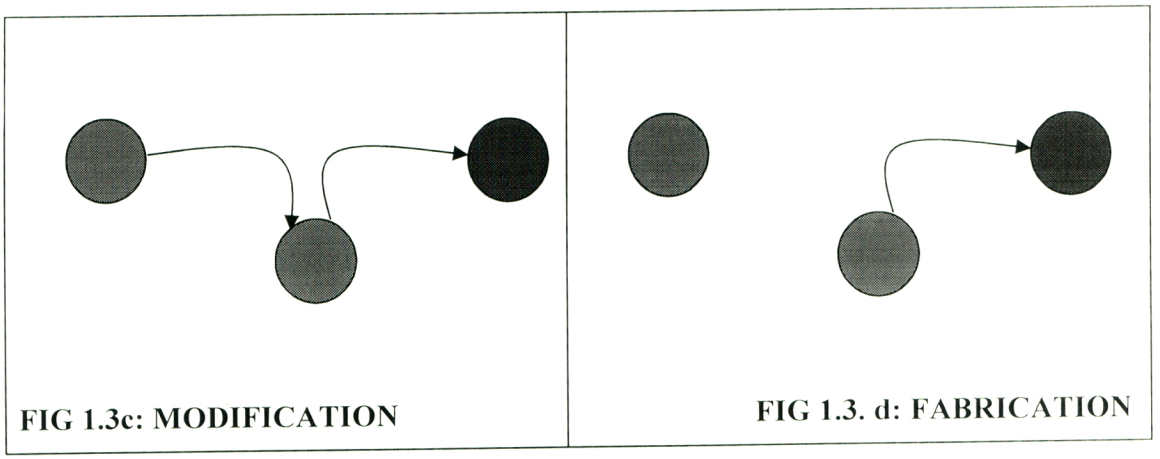


FIG 1.3c: MODIFICATION

FIG 1.3. d: FABRICATION

- **Modification :** An unauthorized party not only gains access to but tampers with an asset. This is an attack on integrity. Examples include changing values in a data file, altering a program so that it performs differently, and modifying the content of messages being transmitted in a network.

- **Fabrication :** An unauthorized party inserts counterfeit object into the system. This is an attack on authencity. Examples include the insertion of spurious messages in a network or the addition of records to a file.

A useful categorization of these attacks can be given in terms of passive attacks and active attacks (Fig. 1.4).

Passive attacks

Passive attacks are in the nature of eaves dropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are release of message contents and traffic analysis. In release of message contents attack, the aim is to prevent the

opponent from learning the contents of these transmissions. Traffic analysis is the concept of transmitting the message in such a way the intruder does not comprehend the content.

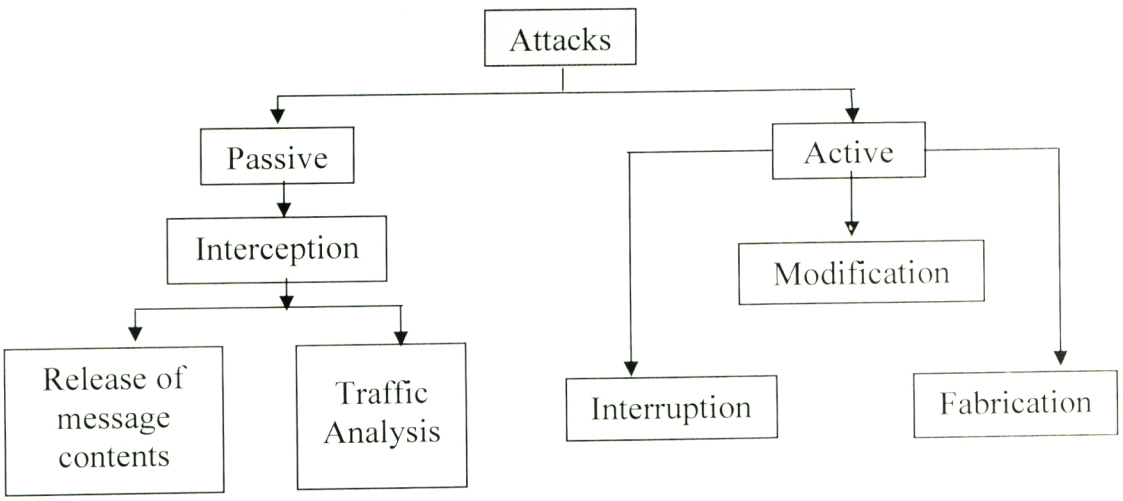


FIG. 1.4 : ACTIVE AND PASSIVE NETWORK SECURITY THREATS

Active attacks

The second category of attack is active attacks. These attacks involve some modification of data stream or the creation of a false stream and can be subdivided into three categories, namely, interruption, modification and fabrication.

Active attacks present the opposite characteristics of passive attacks. Passive attacks are difficult to detect. On the other hand, it is difficult to prevent active attacks absolutely. Here the goal is to detect and recover from any disruption or delays caused by them.

1.3.3. Components of a cryptography algorithm

A typical cipher consists of three components:

- the encryption algorithm

- the decryption algorithm and
- the key expansion algorithm (also known as key scheduling or key setup).

The key expansion algorithm expands the user key or cipher key to a larger intermediate key, to allow (ideally) all bits of the cipher key to influence every round of the encryption algorithm. For most ciphers, key expansion needs only be done once to cater for both encryption and decryption; for other ciphers however, key expansion has to be done separately for encryption and decryption. The most important parameters of a cipher are

- the key length (which determines the length) it supports,
- the word size and
- the nominal number of rounds.

1.3.4. Role of keys in cryptography

A key is a piece of information that controls the operation of a cryptography algorithm. In encryption, a key specifies the particular transformation of plaintext into ciphertext, or vice versa during decryption. Keys are also used in other cryptographic algorithms, such as digital signature schemes and keyed-hash functions (also known as MACs), often used for authentication.

For a well-designed algorithm, enciphering the same plaintext but with a different key should produce a totally different ciphertext. Similarly, decrypting ciphertext with the wrong key should produce random-looking gibberish. If the decryption key is lost, encrypted data should not in practice be recoverable (at least for high quality encryption algorithms and large enough key sizes).

1.3.4.1. Need for key security

In designing security systems, it is wise to assume that the details of the cryptographic algorithm are already available to the attacker. This principle is known as Kerckhoffs' principle, "only secrecy of the key provides security", or "the enemy knows the system". The history of cryptography provides evidence that it can be difficult to keep the details of a widely-used algorithm secret. A key is often easier to protect (it's typically a small piece of information) than an encryption algorithm, and easier to change if compromised. Thus, the security of an encryption system in most cases relies on some key being kept secret.

Keeping keys secret is one of the most difficult problems in practical cryptography. For example: An attacker who obtains the key (by, for example, theft, extortion, dumpster diving, social engineering or inspection of a Poster using a note stuck to the side of a terminal) can recover the original message from the encrypted data.

1.3.4.2. Key sizes

For the one-time pad system the key must be at least as long as the message. In encryption systems that use a cipher algorithm, messages can be much longer than the key. The key must, however, be long enough so that an attacker cannot try all possible combinations.

A key length of 80 bits is generally considered the minimum for strong security with symmetric encryption algorithms. 128-bit keys are commonly used and considered very strong against attacks.

1.3.4.3. Key choice

To prevent a key from being guessed, keys need to be generated truly randomly and contain sufficient entropy. The problem of how to generate safe true

random keys is difficult, and has been addressed in many ways by various cryptographic systems (Levine, 2000). There is a RFC on generating randomness (RFC 1750, Randomness Recommendations for Security). Some operating systems include tools for "collecting" entropy from the timing of unpredictable operations such as disk drive head movements. For the production of small amounts of keying material, ordinary dice provide a good source of high quality randomness.

When a password (or passphrase) is used as an encryption key, well-designed cryptosystems first run it through a key-derivation algorithm which adds salt and reduces or expands it to the key length desired, for example by reducing a long phrase into a 128-bit value suitable for use in a block cipher.

1.3.5. Properties of an efficient cipher

Two chief cryptographic properties of a good cipher are confusion and diffusion, which are commonly ensured by the balance and avalanche properties of the ciphertext in conventional cryptography (Schneier, 1996). Apart from the above, the following are always desirable in the field of cryptography.

- The security of a strong system resides with the secrecy of the key rather than with the supposed secrecy of the algorithm. A strong cryptosystem should have a large keyspace.

- A strong cryptosystem should produce ciphertext which is random to all standard statistical tests (see, for example, Gustafson *et al.*, 1990). Attackers cannot determine key even if given ability to encrypt plaintext of the attacker's choice. This implies resistance to known plaintext and known ciphertext attacks

- A strong cryptosystem should be able to resist all known previous attacks.

1.3.6. APPLICATION AREAS

Cryptography is becoming more and more important for larger parts of business and industry as well as the society at large. The areas where they can be applied are increasing proportionately with the growing technology.

1.3.6.1. General Applications

Cryptography can be applied in various areas for security and protection of data, including the following :

1. Transfer of sensitive data in online payment systems, like e-commerce, online trading, online banking, etc.
2. E-voting, e-government, e-health systems
3. Web applications which need to secure their data
4. Industries and business organizations to protect their confidential data
5. Educational and research institutions to protect their intellectual properties.

1.3.6.2. Software applications

The following describes encryption techniques used by well-known software / platform to protect user data.

- Application encryption extensions: Many programs such as the Microsoft Office suite includes weak encryption that serves to block file access. Microsoft Outlook can encrypt its data files with "compressible encryption" (probably XOR) and "strong encryption". The password protection features of earlier versions of Microsoft Office can be defeated with commercially

available tools (Seltzer, 2002). Office XP includes stronger file protection as an option, but the default appears to be compatibility with earlier versions. A rough test for the strength of encryption systems included with software is to attempt to compress an encrypted file with WinZip or StuffIt. Files encrypted with strong systems cannot be compressed because the encryption process hides the redundancies used to compress the file. In most cases the encryption bundled with office applications is intentionally weak to avoid problems with export/import restrictions.

- Transparent file system encryption - Microsoft Encrypted File System (EFS) : This comes standard with Windows 2000 and Windows XP Professional. The user specifies one or more encrypted folders and Microsoft Windows automatically encrypts files saved into those folders. This is useful for protecting files on multi-user computers and on laptops. EFS is a hybrid system that encrypts each file with an extension of DES called DESX. The file is conventionally encrypted with a unique random key, and then the key is encrypted with the secret key. One security hole is the administrator account for Windows 2000 and Windows XP also has the ability to decrypt the files. Third party file system encryption software tools such as Pretty Good Privacy (PGPdisk) are available with the same functionality (Rubin, 2001).

- File encryption applications - PGP, HandyBits EasyCrypt, ABI Coder : These are programs that encrypt individual files using a strong encryption algorithm. PGP (Pretty Good Privacy) is a public key encryption program that also offers conventional encryption. HandyBits EasyCrypt and ABI Coder just offer conventional encryption. As an added bonus, these utilities usually compress the file before encrypting it. All three programs also include the option of creating self-decrypting files for Windows. Just double-click on the file and enter the key and the file extracts itself.

- Hardware protection: toggles, fingerprint scanners, smartcards. These are physical devices that substitute for passwords on a computer. Toggles are physical devices with a long random key stored in flash memory that plug in to a USB or serial port. If the device is not attached to the computer, software prevents the user from using the computer. Fingerprint scanners use identifying marks from a person's fingerprint as their password. Smartcards store a person's password on a card about the same size and weight as a credit card. The card is scanned by special device when the user wants to use the computer (Burnett and Paine, 2001).

- Password managers : Password managers are basically a self-encrypting database for storing passwords. Many of them also include random password generators. The disadvantage of using a password manager is that if the thief gets the key for the password manager, they can look at all files.

1.3.7. TYPES OF CRYPTOGRAPHY ALGORITHMS

There are several ways of classifying cryptographic algorithms. The following categories are based on the number of keys employed for encryption and decryption. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below. In all cases, the initial unencrypted data is referred to as plaintext. It is encrypted into ciphertext, which will in turn (usually) be decrypted into usable plaintext.

- Symmetric Cryptography or Public Key Cryptography (PKC): Uses a single key for encryption and decryption (Fig. 1.5a).

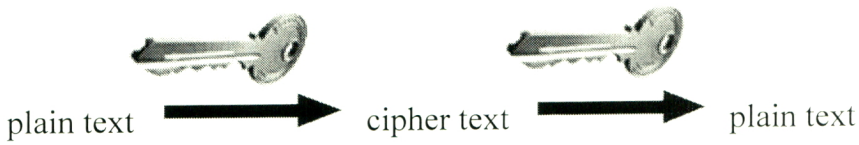


Fig. 1.5a : Symmetric Cryptography

- Asymmetric key-based algorithms or secret key cryptography (SKC) : This method uses one key to encrypt data and a different key to decrypt the same data. It is also sometimes called as public key/private key encryption (Fig. 1.5b).

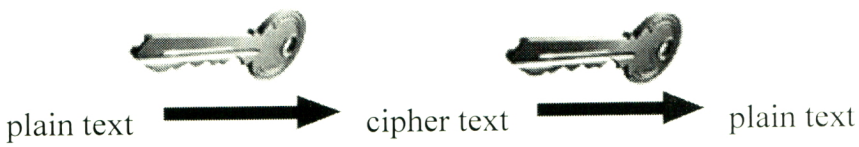


Fig. 1.5b : Asymmetric Cryptography

- Hashing, or creating a digital summary of a string or file. This is the most common way to store passwords on a system, as the passwords aren't really what's stored, just a hash that can't be decrypted. Uses a mathematical transformation to irreversibly "encrypt" information. Hash functions have no keys since the plain text is not recoverable from the ciphertext (Fig. 1.5c).

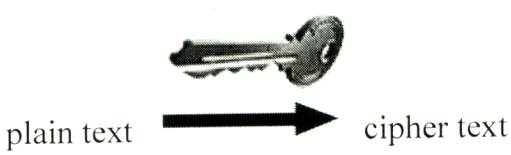


Fig. 1.5c : Hash Function

1.3.8. ISSUES IN USING CRYPTOGRAPHY TECHNIQUES

Cryptography can be used whenever there is a chance for interception or exposure by an individual who is not authorized to know the contents.

- There is no industry-wide encryption/decryption standard, that is, there is no "one-size-fits-all" solution for protecting data.
- There are technical and financial burdens for many algorithms
- Security awareness and cultural issues (behaviour change) supercede technical solutions for many entitites. Encryption is 'too technical' for decision makers.
- There is no single solution, interoperable solution for communication over open networks.
- It is hard to identify whether an unencrypted data as been viewed or altered.
- Loss of key generally indicates loss of data in cryptography.
- Encryption cannot stop an insider (employee, business partner, etc.) from abusing priviledges to access confidential information.

1.4. TERMINOLOGIES USED

Table 1.1. shows some of the terms and definitions (Answers.com, 2006; Wikipedia, 2006) used throughout the dissertation.

Table 1.1 : Terminologies

TERM	DEFINITION / MEANING
♦ Cipher	Pair of algorithms for encryption and decryption
♦ Ciphertext	Plaintext is information used as input to an encryption algorithm; the output is termed ciphertext.

TERM	DEFINITION / MEANING
♦ Code	Often used to mean any method of encryption or meaning concealment. In cryptography, *code* refers to a procedure which replaces a unit of plaintext (i.e. the meaningful words or phrases) with a code word.
♦ Cryptanalysis	The science of deducing the plaintext from a ciphertext, without knowledge of the key
♦ Cryptanalysts	Practitioners of cryptanalysis
♦ Cryptographers	People who do cryptography
♦ Cryptography	The art or science concerning the principles, means, and methods for rendering plaintext unintelligible and for converting encrypted messages into intelligible form
♦ Cryptosystem	A system that uses a set of programs/algorithms, protocols, operating procedures for encryption and decryption for implementing cryptography
♦ Cryptology	The study of cryptography and cryptanalysis
♦ Deciphering	The procedure of turning enciphered text into plain text with prior knowledge of the algorithms or keys involved. This is what the intended message receiver does.
♦ Decryption	Reverse process of encryption, which recovers the plain text from the cipher text
♦ Encryption	Process of obscuring information to make it unreadable without special knowledge.

TERM	DEFINITION / MEANING
♦ Encryption key	The message, during encryption, is encoded mathematically with a string of characters called a data encryption key.
♦ Inverse Cipher	Series of transformations that converts ciphertext to plaintext using the Cipher Key.
♦ Key	A key is a piece of information that controls the operation of a cryptography algorithm. In encryption, a key specifies the particular transformation of plaintext into ciphertext, or vice versa during decryption.
♦ Plain text	In cryptography, plaintext is information used as input to an encryption algorithm; the output is termed ciphertext.
♦ Public key cryptosystem	A system where a pair of keys are used, one freely distributed and the other known only to the recipient
♦ Steganography	The art of concealing a message's existence. One example would be through the use of photographic microdots
♦ Symmetric key cryptosystem	system where both sender and receiver use the same key for enciphering and deciphering
♦ Transmission security	The art of concealing an electrically transmitted message through burst encoding or spread spectrum methods.

1.5. ORGANIZATION OF THESIS

Chapter I provide a brief introduction to the field of cryptography, security and issues in encryption and decryption.

Cryptography is a demanding field and lot of research work has been conducted in the same. A brief review of previous work related to this dissertation is presented in Chapter II.

Chapter III discusses the various techniques used to achieve symmetric cryptography, asymmetric cryptography and hash function cryptography. The functioning of the nine symmetric algorithms selected, their advantages and disadvantages, etc., are discussed in detail in this chapter. In a similar fashion, the various cryptographic hashing techniques, its functioning along with its strength and weakness, are discussed in detail. This chapter also discusses in detail the working of the popular RSA and elliptic cryptographic models (asymmetric techniques).

Chapter IV presents the performance comparison of algorithms in terms of time discussed in Chapter III and discusses them.

Chapter V presents the general conclusions of the thesis and proposes possible improvements and directions of future research work.

Everyone needs references to build their knowledge from and the more sources the better. In case of cryptography, some references are more historical in nature, tracing the development of the science while others are routed in the mathematical basis of creating or analyzing an algorithm. The references used in the formation of this thesis is presented in the reference section. The Appendix section shows various screen shots of the developed cryptographic prototype model for various algorithms.

A brief review of literature pertaining to cryptography, encryption and decryption is presented in Chapter II.

2. REVIEW OF LITERATURE

Cryptology covers both cryptography and cryptanalysis. Although cryptology literatures are not as abundant compared to those of other fields, many of the literatures in existence are well-written. It is intriguing to see how the science of keeping secrets has evolved, especially in the information age. In this section, a brief review of literature pertaining to the study is given.

2.1. HISTORY OF CRYPTOGRAPHY

Before the modern era, cryptography was concerned solely with message confidentiality, that is, conversion of messages from a comprehensible form into an incomprehensible one and back again at the other end, rendering it unreadable without secret knowledge (namely, the key). In recent decades, the field has expanded beyond confidentiality concerns to include techniques for authentication, digital signatures, interactive proofs, and secure computation (Crowe *et al.*, 2005).

The earliest forms of secret writing required little more than pen and paper. The main classical cipher types are transposition ciphers, which rearrange the order of letters in a message (e.g. 'help me' becomes 'ehpl em'); and substitution ciphers, which systematically replace letters or groups of letters with other letters or groups of letters (e.g. 'fly at once' becomes 'gmz bu podf' by replacing each letter with the one following it in the alphabet). Simple versions of either offered little confidentiality. An early - and one of the simplest - substitution ciphers was the Caesar cipher, which was used by Julius Caesar during his military campaigns (Reference.com, 2006).

Encryption attempted to ensure secrecy in important communications, such as those of spies, military leaders, and diplomats, but it also had religious applications. For instance, early Christians used cryptography to obfuscate parts of their religious writings to avoid near certain persecution they would have faced had they been less obscured. Famously, 666, the Number of the Beast, is thought to be a ciphertext referring to the Roman Emperor Nero, one of whose policies was persecution of Christians (Dunn and Rogerson, 2003).

There is record of several, even earlier, Hebrew ciphers as well. Steganography (which is hiding a message so as to make its existence undetectable) was also first developed in ancient times. An early example, from Herodotus, concealed a message - a tattoo on a slave's head - by regrown hair (Kahn, 1967). More modern examples of steganography include the use of invisible ink, microdots, and digital watermarks to conceal information .

Ciphertexts produced by classical ciphers reveal statistical information about the plaintext, which can be used to break them. After the Arab discovery of frequency analysis (around the year 1000), nearly all such ciphers became more or less breakable by an informed attacker. Essentially all ciphers remained vulnerable to cryptanalysis using this technique until the invention of the polyalphabetic cipher by Leon Battista Alberti around the year 1467, in which different parts of the message (often each successive plaintext letter) are enciphered using a different key. In the polyalphabetic Vigenère cipher, for instance, encryption uses a key word, which controls letter enciphering depending on which letter of the key word is used. Despite this improvement, polyalphabetic ciphers of this type remained partially vulnerable to frequency analysis techniques (Kahn, 1967).

Although frequency analysis is a powerful and general technique, encryption was still often effective in practice: many a would-be cryptanalyst was unaware of the technique. Breaking a message without frequency analysis essentially required knowledge of the cipher used, thus encouraging espionage, bribery, burglary, defection, etc. to discover it. It was finally recognized in the 19th century that secrecy of a cipher's algorithm is not a sensible, nor practical, safeguard: in fact, any adequate cryptographic scheme (including ciphers) should still be secure even if the adversary knows the cipher itself. Secrecy of the key should be alone sufficient for confidentiality when it is attacked. This fundamental principle was first explicitly stated in 1883 by Auguste Kerckhoffs and is called Kerckhoffs' principle; alternatively and more bluntly, it was restated by Claude Shannon as Shannon's Maxim.

Various physical devices and aids have been used to assist with ciphers. One of the earliest may have been the scytale of ancient Greece, a rod supposedly used by the Spartans as an aid for a transposition cipher. In medieval times, other aids were invented such as the cipher grille, also used for a kind of steganography. With the invention of polyalphabetic ciphers came more sophisticated aids such as Alberti's own cipher disk, Johannes Trithemius' tabula recta and Thomas Jefferson's cylinder (reinvented by Bazeries around 1900). Early in the 20th century, several mechanical encryption/decryption devices were invented, and many patented, including rotor machines, most famously the Enigma machine used by Germany in World War II. The ciphers implemented by the better of these designs brought about a substantial increase in cryptanalytic difficulty (Gannon, 2001).

The development of digital computers and electronics after WWII made possible much more complex ciphers. Many computer ciphers can be

characterised by their operation on binary bits (sometimes in groups or blocks), unlike classical and mechanical schemes, which generally manipulate traditional characters (i.e. letters and digits). However, computers have also assisted cryptanalysis, which has compensated to some extent for increased cipher complexity. Nonetheless, good modern ciphers have stayed ahead of cryptanalysis: it is usually the case that use of a quality cipher is very efficient, while breaking it requires an effort many orders of magnitude larger, making cryptanalysis so inefficient and impractical as to be effectively impossible.

Extensive open academic research into cryptography is relatively recent, it began only in the mid-1970s with the public specification of DES (the Data Encryption Standard), the Diffie-Hellman paper, (Diffie and Hellman, 1976) and the public release of the RSA algorithm. Since then, cryptography has become a widely used tool in communications, computer networks, and computer security generally.

The security of many modern cryptographic techniques is based on the difficulty of certain computational problems, such as the integer factorisation problem or the discrete logarithm problem. In many cases, there are proofs that cryptographic techniques are secure if a certain computational problem cannot be solved efficiently (Goldreich, 2001). With one notable exception - the one-time pad - these contingent proofs are the best available for cryptographic algorithms and protocols.

As well as being aware of cryptographic history, cryptographic algorithm and system designers must also carefully consider probable future developments in their designs. For instance, the continued improvements in computer processing power in increasing the scope of brute-force attacks must be taken into account

when specifying key lengths, and the potential effects of quantum computing are already being considered by good cryptographic system designers (Menezes *et al.*, 1996).

Essentially, prior to the early 20th century, cryptography was chiefly concerned with linguistic patterns. Since then the emphasis has shifted, and cryptography now makes extensive use of mathematics, including aspects of information theory, computational complexity, statistics, combinatorics, abstract algebra, and number theory. Cryptography is also a branch of engineering, but an unusual one as it deals with active, intelligent, and malevolent opposition. There is also active research examining the relationship between cryptographic problems and quantum physics.

2.2. HARDWARE ACCELERATORS FOR CRYPTOGRAPHY

All modern security protocols use symmetric-key as well as public-key cryptographic algorithms. In order to be able to provide highly arithmetic intensive public-key cryptographic primitives, hardware accelerators are often used. An example is a high-end smart card, where a cryptographic coprocessor takes over all the expensive (area and time) computations. In practical applications the most used public-key algorithms are RSA and Elliptic Curve Cryptosystems (ECC).

Elliptic-curve cryptography is a technology that solves security concerns of electronic communication. Its ability to generate digital signatures is deemed as a backbone of e-commerce and e-government systems. Elliptic-curve cryptography is considered to be the most efficient technology to achieve this. Hardware implementations can provide strong cryptographic functions on smallest silicon area and they can process operations faster. Hardware implementations of elliptic-curve cryptography can be optimized in many ways.

The work proposed by Wolkerstorfer (2005) deals with aspects of implementing elliptic-curve cryptography in hardware. They gave a structured approach towards implementing elliptic-curve cryptography in hardware. They discussed the application scenarios, giving orientation in the mathematical background, presented efficient algorithms, and guided through the hardware design flow to realize elliptic-curve hardware on reconfigurable devices as well as standard-cell circuits. The work aspired to create optimized hardware architectures that bring out best performance on smallest silicon area. Architectures for low power consumption were another focus. Design space exploration on various abstraction levels was necessary to achieve the ambitious goals. The accelerator card was integrated into the Java cryptographic architecture. While testing on a 0.35 μm CMOS process, it used an area of only 0.46 mm^2. It was able to compute a 192-bit elliptic-curve signature in less than 7 ms.

One emerging and very promising public-key scheme is the HyperElliptic Curve Cryptosystem (HECC). HECC has been analyzed and implemented only recently both in software (Kuroki *et al.*, 2002; Lange, 2002a; Matsuo *et al.*, 2001; Miyamoto *et al.*, 2001; Pelzl *et al.*, 2003a; Pelzl *et al.*, 2003b; Sakai and Sakurai, 2000; Sakai *et al.*, 1998; Smart, 1999; Wollinger *et al.*, 2003) and in more hardware-oriented platforms such as FPGAs (Boston *et al.*, 2002; Wollinger, 2001; Wollinger and Paar, 2002). This section gives a short overview of the hardware implementations targeting HECC and of the previous research work to parallelize hardware ECC.

The first work discussing hardware architectures for the implementation of HECC appeared in (Wollinger, 2001; Wollinger and Paar, 2002). The authors described efficient architectures to implement the necessary field operations and polynomial arithmetic in hardware. In Wollinger (2001) they also estimated that

for a hypothetical clock frequency of 20 MHz, the scalar multiplication of HECC would take 21:4 ms using the window NAF method.

In Boston *et al.* (2002) the authors presented the first complete hardware implementation of a hyperelliptic curve coprocessor. This implementation targets a genus-2 HEC over F2113. The target platform is a Xilinx II FPGA. Point addition and point doubling with a clock frequency of 4 5MHz take 105's and 90's, respectively. The scalar multiplication could be computed in 10:1 ms.

The publications (Boston *et al.*, 2002; Wollinger, 2001; Wollinger and Paar, 2002) adopt the Cantor algorithm to compute group operations. Today, there exist more efficient algorithms to compute group addition and group doubling, the so-called explicit formulae.

In Mishra and Sarkar (2003) the authors proposed a parallelization of the explicit group operation of HECC. They developed a general methodology for obtaining parallel algorithms. The methodology guarantees that the obtained parallel version requires a minimum number of rounds. They show that for the inversion free arithmetic (Lange, 2002b) using 4, 8 and 12 multipliers in parallel, scalar multiplication can be carried out in 27, 14 and 10 parallel rounds, respectively. When using affine coordinates (Lange, 2002b) and 8 multipliers it can be performed in 11 rounds, including an inversion round.

A similar work as that presented here for HECC can be found in Bednara (2002) for ECC, where a study of the trade-off between the number of operators and different coordinate systems is presented. In Antola *et al.* (2003) two scalar multiplications are scheduled in parallel on the same architecture: the two operations are executed in different coordinate systems to improve the use of the operators. Note that the group operations of elliptic curves are much less complex

than those of the hyperelliptic ones. In ECC the silicon area of the possible architecture is easily bounded since the critical path can be computed by hand, while in the case of HECC it is much more complex.

2.3. MATHEMATICAL PART OF CRYPTOGRAPHY

In the last few years, a new approach of constructing cryptosystems based on application of the theory of both continuous and discrete chaotic dynamical systems has been developed. In frames of continuous theory the methods of synchronisation of chaotic systems Kapitaniak (1996) and the idea of controlling chaos Kapitaniak (1996), are applied. The discrete systems methods concentrate on iterations and inverse iterations of chaotic maps and possibilities of intelligent way of introducing keys.

The earliest applications of chaotic systems in cryptography were proposed by Pecora and Caroll (1990) as a possible application of the synchronisation of chaotic dynamical systems. This idea has been developed by Kocarev *et al.* (1992) and Parlitz *et al.* (1992), where they presented an experimental test system based on a chaotic electronic circuits. The first paper employed analog signals while the second one used binary information model. The overview of the methods connected with encrypting messages with the modulation of trajectories of continuous dynamical systems can be found in Koblitz (1994). Application of discrete chaotic dynamical systems to cryptography was first analysed by Habutsu *et al.* (1991) and then developed by Kotulski and Szczepañski (1997).

ECC is an attractive public-key cryptosystem recently endorsed by the US government for mobile/wireless environments which are limited in terms of their CPU, power, and network connectivity. Its efficiency enables constrained, mobile devices to establish secure end-to-end connections. Hence the server side has to be

enabled to perform ECC operations for a vast number of mobile devices that use variable parameters in an efficient way to reduce cost. We present algorithms that are especially suited to high-performance devices like large-scaled server computers.

Since the introduction of elliptic curves to cryptography, a vast amount of research has been done. When implementing ECC in software there are several choices to make. The parameter choices are given as below :

- Underlying field type, field representation, and field operation algorithms
- Elliptic curve point representation and point operation algorithms
- Protocol algorithms

The literature contains significant research on each of these issues.

According to Brown *et al.* (2000), the efficient use of prime fields requires platform dependent assembly code and thus the binary fields. A comprehensive survey of binary field and elliptic curve arithmetic for the NIST recommended elliptic curves was done by Hankerson (2000).

A specialized implementation was done for the field GF(2155) (Schroeppel *et al.*, 1995). There are also implementations available for constrained devices. PGP was ported to constrained devices in Brown *et al.* (2000) while Weimerskirch *et al.* (2001) optimizes an ECC implementation for a Palm PDA. L´opez and Dahab presented a Montgomery field multiplication for binary fields in L´opez and Dahab (1999a). They also did research on binary field multiplication (L´opez. and Dahab, 2000) and ECC over binary fields (L´opez and Dahab, 1999b).

More point multiplication algorithms can be found in Koyama and Tsuruoka (1993) and Lim and Lee (1994). Further work was done for special field

choices like composite fields, e.g., in De Win *et al.* (1996) and Guajardo and Paar (1997). There were, however, recent attacks Gaudry *et al.* (2002) on these fields. Finally, Solinas developed efficient algorithms for Koblitz curves over binary fields using complex multiplication (Solinas, 2000).

2.4. RECENT TRENDS IN CRYPTOGRAPHY

The expansion of the connectivity of computers makes ways of protecting data and messages from tampering or reading important. It is thus up to the user to ensure that communications which are expected to remain private actually do so. One of the techniques for ensuring privacy of files and communications is Cryptography. The following lists some starting place for the background and research work in cryptography.

Schneier (1996) wrote a book on Applied Cryptography and is considered as one of the most popular books on cryptography. It provides a good introduction to the various concepts involved in cryptography and some information on cryptanalysis. The book details how programmers and electronic communications professionals can use cryptography. The book discusses the techniques for enciphering and deciphering messages, maintaining privacy of computer data, etc. It described dozens of cryptography algorithms, gives practical advice on how to implement them in cryptographic software, and showed how they can be used to solve security problems.

Schneier (2000) charted an organized pathway towards learning how to cryptanalysis. He discussed various block cipher methods and discusses techniques for cryptanalysts. Several ways were suggested for breaking code and writing a secure cryptology program.

Graff (2001) provides a logical explanation of how cryptography works to solve real-world e-commerce problems, a tutorial on the underlying mathematics, and two case studies of Public Key Infrastructure (PKI) cryptographic architectures, showing how Kerberos and Public Key Cryptography (PKC) can be wedded to protect a company's Intranet and how a full-blown working PKI provides security to a company's Internet communications.

Schneier (2001) on the other hand provided a good explanation on why cryptography is hard and the issues which cryptographers have to consider in designing ciphers. In this essay, he discussed the various kinds of threats and risks faced by a computer system, what cryptography can and cannot do, different threat models, how cryptographic systems can be designed and implemented and the current state of security.

Schneier *et al.* (1999) proposed a new cipher as a candidate for the new AES. They discussed in detail the main concepts and issues involve in block cipher design and also suggested cryptanalysis as a measure of cipher strength. They developed a new technique called Twofish, which is a 128-bit block cipher submitted as an AES candidate. They developed the model in which no two distinct keys result in an identical sequence of round functions.

Yeun (2001) provided an encyclopedic look at the design, analysis and applications of cryptographic techniques. A new scheme was proposed for computationally secure online secret sharing, in which the shares of the participants can be reused. The security of the scheme was based on the intractability of factoring. This scheme had the advantage that it detected cheating and it enabled the identification of all cheaters by an arbitrator, regardless of their number.

Schaefer (1996) described a simplified version of DES. He proposed an architecture of DES that has less rounds and less bits. Mirzan (2000) presented a very good technical paper on block cipher and cryptanalysis.

Hayes (2002) has a somewhat similar notion as Mirzan (2000) but used a more descriptive approach. He focused on linear cryptanalysis and differential cryptanalysis of a given SPN cipher. The tutorial was based on the analysis of a simple, yet realistically structured, basic Substitution(S)-Permutation(P) Network cipher. Experimental data from the attacks was presented as confirmation of the applicability of the concepts as outlined.

Daemen and Rijmen (2000) described Rijndael, the new AES. They provided a good insight into many creative cryptographic techniques that increased cipher strength. Kilian and Rogaway (1996) investigated the security of Ron Rivest's DESX construction, a cheaper alternative to Triple DES. The model developed by them can also be combined with other block-cipher based constructions.

Aoki *et al.* (2000) proposed a new cipher called Camelia, a 128 block cipher, which supported 128-bit block size and 128, 192, and 256-bit keys, i.e. the same interface specifications as the Advanced Encryption Standard (AES). Efficiency on both software and hardware platforms was analysed and was found to be satisfactory. They proved that Camellia provided a strong security against differential and linear cryptanalysis. Compared to the AES finalists, i.e. MARS, RC6, Rijndael, Serpent, and Twofish, Camellia offered comparable encryption/decryption speed in software and hardware.

Garrett (2001) is a textbook in cryptology which uses a highly mathematical path to explain cryptologic concepts. Various algorithms starting

from the simplest ciphers like shift cipher, reduction/division algorithm etc. to modern symmetric ciphers like AES are discussed. The complexity of the algorithms and various types of attacks are also discussed in this book.

Singh (2000) is a good book that provides an introduction to cryptology and the current interests in this field. He also presented an entertaining yet informative look at the history of cryptography, classical ciphers and the effect of cryptography on society.

Landau (2000) is an article on DES which described the requirements of a good cryptosystem and cryptanalysis. The author has discussed about block cipher designs, differential cryptanalysis apart from DES. A brief discussion on attacks and breaking of DES was also analysed.

2.5. ECC-ACCELERATOR CARDS

Information security is one of the main aspects of e-Commerce and e-Government. In this fast-growing area new services only find acceptance when they provide a sufficient level of security in terms of authentication, confidentiality, data integrity, and non-repudiation. Elliptic Curve Cryptography (ECC) is a technology that fulfills all the demands and requires only moderate resources. ECC can implement asymmetric cryptography very efficiently because it is based on the Elliptic-Curve-Discrete-Logarithm-Problem (ECDLP).

The main part of accelerating PCI-card is to accelerate ECC. Such a dedicated hardware solution offers the advantage to process data at the full wordsize of m bits. This section provides a brief review of previous work done in this area.

Hankerson *et al.* (2000) presented the fastest known software implementation of EC-operations. Their assembler optimized implementation on a 400 MHz Pentium-II processor reached 594 EC-operations per second for a 163-bit curve and 252 operations for 233-bit. This compares roughly to 414 EC-operations for a 191-bit curve and it should be considered that the CPU load for these values is 100%. Nothing else can be computed without deteriorating the throughput.

One of the first ECC hardware implementations was reported by Agnew *et el.* (1993). Their implementation is a mere coprocessor for operations in the finite field GF(2155) and EC-operations are controlled by a multi-purpose processor. This implementation is basically a bit-serial multiplier with a clock frequency of 40 MHz.

Goodman and Chandrakasan (2001) published their so-called Domain-Specific Reconfigurable Cryptographic Processor. Their versatile design can operate with bitlengths from 4-bit up to 1024-bit and can calculate modular integer-arithmetic besides operations in the finite field GF(2m). This circuit can be clocked with 50 MHz and calculates approximately 125 191-bit EC-operations per second.

Orlando and Paar (2000) reported the fastest known EC-processor. Their implementation is also optimized for a particular class of elliptic curves (fixed bit length), has a digit-serial multiplier, and an extra square unit to exploit a shortcut offered by GF(2m)-arithmetic. A large internal memory allows the use of algorithms, which rely on pre-computations. This implementation should have a throughput of about 2000 EC-operations per second when it is clocked with 70 MHz and when a radix-256 multiplier is used.

Ernst *et al.* (2001) presented an FPGA-based PCI-card for ECC-acceleration. Contrary to the widely used polynomial-basis representation of GF(2m)-elements, they use an optimal-normal basis representation that cannot be directly compared with a polynomial representation. A 191-bit version of their PCI-card with a radix-32 multiplier can be clocked with 36 MHz and has a throughput of 431 EC-operations. The card can be accessed from a personal computer running under Windows NT via a C++-interface.

2.6. HYBRID CRYPTOGRAPHY

Symmetric and asymmetric ciphers each have their own advantages and disadvantages. Symmetric ciphers are significantly faster than asymmetric ciphers, but require all parties to somehow share a secret (the key). The asymmetric algorithms allow public key infrastructures and key exchange systems, but at the cost of speed (Cramer and Shoup, 2004).

A hybrid cryptosystem is a protocol using multiple ciphers of different types together, each to its best advantage. One common approach is to generate a random secret key for a symmetric cipher, and then encrypt this key via an asymmentric cipher using the recipient's public key. The message itself is then encrypted using the symmetric cipher and the secret key. Both the encrypted secret key and the encrypted message are then sent to the recipient. The recipient decrypts the secret key first, using his/her own private key, and then uses that key to decrypt the message (Abdalla *et al.*, 2000).

Almost all the of hybrid encryption schemes (Lucks, 2002; Okamoto *et al.*, 1999) are based on a simple idea:

1. asymmetric section generates a suitable symmetric key and encrypts that key using an asymmetric encryption scheme, and

2. symmetric section encrypts the message using the randomly generated symmetric key.

The asymmetric section is known as the key encapsulation mechanism or KEM. The symmetric section is known as the data encapsulation mechanism or DEM. The hybrid property is known as the KEM-DEM hybrid encryption scheme.

Dent (2005) in his work examined the methods in which the ideas behind a KEM-DEM hybrid encryption scheme can be extended to other types of asymmetric primitives. The central principle is a keyed symmetric algorithm can be used to provide a security service for in an asymmetric algorithm provided that that symmetric primitive is under the control of the asymmetric part of the cipher. This theory is applied to signcryption schemes with outsider security and an efficient, provably secure scheme, termed ECISS-KEM, was proposed. The theory was also applied to signature schemes, where it was shown that efficient hybrid signature schemes can never exist, and to signcryption schemes with insider security, where it was shown that several existing schemes can be considered hybrid signcryption schemes.

A number of reconfigurable architectures have been proposed to accelerate the performance of symmetric-key cryptography. Hybrid architectures composed of a microprocessor core combined with reconfigurable function blocks are typically used to accelerate the performance of a general purpose processor. Hybrid architectures targeted at accelerating symmetric-key cryptography include ConCISe, Garp, and MorphoSys (Hauser and Wawrzynek, 1997; Kastrup *et al.*, 1999; Singh *et al.*, 2000). Generalized reconfigurable architectures consist of an

interconnected network of configurable logic and storage elements where the granularity of the architecture is dependent upon the target application.

Chawla (2005) in his work proposed a formal hybrid approach to encrypt, decrypt, transmit and receive information using colors. A piece of information consists of set of symbols with a definite property imposed on the generating set. The symbols are usually encoded using ASCII scheme. A linear to 3D transformation was presented. The change of axis from traditional xyz to RGB is highlighted and their effects were studied. A formal notion on hybrid cryptography was introduced as the algorithm lies on the boundary of symmetric and asymmetric cryptography.

PGP (Pretty Good Privacy) was released in 1991 by Phil Zimmermann. PGP is not really an encryption algorithm in itself; rather it is a computer program that uses both public key encryption (RSA) and private key encryption (IDEA) combining them in a way that is both secure and efficient. The reason why Zimmermann chose to combine the two algorithms is the huge difference in speed between them. Private key encryption is much faster than public key cryptosystems. In the 1980s, when Zimmermann started his work, only government and large companies had computers powerful enough to make RSA implementation practical. PGP works by using the RSA algorithm to encrypt a private key that is used by the IDEA algorithm to encrypt the message itself. PGP can also be used for file encryption; in this case only the IDEA algorithm is used. Currently, PGP is sold as a commercial product by Network Associates, Inc.. It is available as a freeware for individual non-commercial use and can be downloaded from their Web site.

2.7. CONCLUSION

The various works done in cryptography show that there is no uniform standard algorithm that is available for all applications. There are certain limitations on the hardware requirements and they also service very strong theoretical support from mathematical theory. Numerous hand held devices are coming with enormous advancements in Internet. They basically exploit the cryptographic algorithms to maintain secrecy. So far there has been an attempt made to compare the functions of cryptographic algorithms. Hence this thesis aims at the study and performance comparison of cryptographic functions. The next chapter discusses them in details.

Methodology

3. METHODOLOGY

The methodology pertaining to the research work entitled "Performance comparison of cryptographic functions in secret key cryptography, public key cryptography and hash algorithms" is given under the following headings.

3.1. Symmetric cryptography

3.2. Hash function cryptography

3.3. Asymmetric cryptography

3.1. SYMMETRIC CRYPTOGRAPHY

Symmetric algorithms are designed in a way such that any two parties interested in encrypting/decrypting data have to use the same (secret) key generated for both encryption and decryption. Symmetric algorithms can be divided into two categories: block ciphers and stream ciphers. The block ciphers operate on data in groups or blocks. On the other hand, stream ciphers only operate on a single bit at a time, which makes them more suitable for real time applications such as multimedia. The following sections explain the working of nine conventional symmetric cryptology algorithms. The following symmetric cryptographic algorithms are implemented:

- Blowfish
- International Data Encryption Algorithm (IDEA)
- Data Encryption Standard (DES)
- Triple DES (TDES or 3DES)
- GOST
- Serpent
- Tiny Encryption Algorithm (TEA)

- Skipjack

3.1.1. Blowfish

In cryptography, Blowfish is a keyed, symmetric block cipher, designed in 1993 by Bruce Schneier (Schneier, 2004) and included in a large number of cipher suites and encryption products.

3.1.1.1. Working of Blowfish

Blowfish has 16 rounds. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

Blowfish uses a large number of subkeys. These keys must be precomputed before any data encryption or decryption. The P-array consists of 18 32-bit subkeys: P1, P2,..., P18. There are also four 32-bit S-boxes with 256 entries each: S1,0, S1,1,..., S1,255; S2,0, S2,1,..., S2,255; S3,0, S3,1,..., S3,255; S4,0, S4,1,..., S4,255.

3.1.1.2. Encryption and Decryption Process

Blowfish has 16 rounds. The input is a 64-bit data element, x.

1. *Divide x into two 32-bit halves: xL, xR.*
2. *For i = 1 to 16*

   xL = xL XOR Pi

   xR = F(xL) XOR xR

   Swap xL and xR

3. *After the sixteenth round, swap xL and xR again to undo the last swap. Then, xR = xR XOR P17 and xL = xL XOR P18. Finally, recombine xL and xR to get the ciphertext.*

Function F(xL)

1. Divide xL into four eight-bit quarters: a, b, c, and d.
2. Then, F(xL) = ((S1,a + S2,b mod 232) XOR S3,c) + S4,d mod 232.

Decryption is exactly the same as encryption, except that P1, P2,...., P18 are used in the reverse order.

3.1.1.3. Generating the Subkeys

The subkeys are calculated using the Blowfish algorithm:

1. *Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi: P1 = 0x243f6a88, P2 = 0x85a308d3, P3 = 0x13198a2e, P4 = 0x03707344, etc.*
2. *XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)*
3. *Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).*
4. *Replace P1 and P2 with the output of step (3).*
5. *Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.*
6. *Replace P3 and P4 with the output of step (5).*
7. *Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.*

In total, 521 iterations are required to generate all required subkeys. Applications can store the subkeys rather than execute this derivation process multiple times.

3.1.1.4. Advantages and Disadvantages

The advantages of blowfish are that the algorithm is very fast and it is most importantly are cryptanalysis-resistant. The significant drawback of blowfish is that it requires repeated encryptions to get the subkeys and S-Boxes, which is considered as repetitive task. The algorithm is key-dependent and the time required to initialize the algorithm with the key is high and a linear transform is needed in each round.

When choosing Blowfish as an encryption technology, various factors like speed and key strength are to be considered. The performance of Blowfish is more than satisfactory with both the parameters and is sure to set the standard for years to come and will continue to be the preferred encryption algorithm for corporations worldwide.

3.1.2. International Data Encryption Algorithm (IDEA)

In cryptography, the International Data Encryption Algorithm (IDEA) is a block cipher designed by Xuejia Lai and James L. Massey of ETH Zurich and was first described in 1991. The algorithm was intended as a replacement for the Data Encryption Standard. IDEA is a minor revision of an earlier cipher, PES (Proposed Encryption Standard); IDEA was originally called IPES (Improved PES).

3.1.2.1. Working of IDEA

IDEA operates on 64-bit blocks using a 128-bit key. IDEA consists of 8 rounds, followed by a final output transformation. All groups operations work on blocks of 16 bits. The 64-bit input (plain text) is broken into four 16 subblocks X1, X2, X3, X4. The 64 bit output is represented as R1, R2, R3, R4. Six 16-bit round keys are used at each round represented as K1, K2, K3, K4, K5, K6. It uses the same algorithm for encryption and decryption. The detailed step by step analysis of the algorithm is given below.

- *The 56 (16 bit each) sub keys are generated from the input 128 bit key.*
- *The bits are divided into blocks of 16.*
- *Consider the first 6 blocks (Keys 1 through 6).*
- *The 128 bit key is then shifted 25 bits to the left and reblocked.*
- *The second set of keys is taken. This process is repeated until all 56 keys are generated. (The reason for the shift of the bits being 25 is to simply ensure that there is no repetition occurring. It would take 128 * 25 shifts before the key is repeated in its entirety).*
- *The 64 bit blocks (input plaintext) is broken into four 16 bit sub-blocks (X1, X2, X3, and X4) and is taken as input to the algorithm.*
- *The bits are then manipulated (XOR, addition, multiplication) and between rounds the second and third (X2 and X3) are swapped. This process is repeated 8 times, each time incorporating 6 of the 56 keys.*
- *Finally the four sub-blocks are combined with the four final keys in an output transformation.*

The following shows the cipher in 15 steps.

1. *Multiply X1 and the first sub key*
2. *Add X2 and the second sub key*
3. *Add X3 and the third sub key*
4. *Multiply X4 and the fourth sub key*
5. *XOR the results of steps 1 and 3*
6. *XOR the results of steps 2 and 4*
7. *Multiply the result of step 5 with the fifth sub key*
8. *Add the results of steps 6 and 7*
9. *Multiply the results of step 8 with the sixth sub key*
10. *Add the results of steps 7 and 9*
11. *XOR the results of steps 1 and 9*
12. *XOR the results of steps 3 and 9*
13. *XOR the results of steps 2 and 10*
14. *XOR the results of steps 4 and 10*
15. *The results of the algorithm will be in order, 11, 12, 13, 14. Swap the 2 inner blocks so that it would read 11, 13, 12, 14*

In the next iteration of the algorithm, the keys are changed, that is instead of using keys 1 through 6, the next set of keys, that is keys from 7 to 12, will be used. The swapping step (step 15) happens on all rounds except for the eighth round, where the inputs are used for the final transformation. The final transformation is done as follows:

1. *Multiply X1 and the first sub key*
2. *Add X2 and the second sub key*
3. *Add X3 and the third sub key*
4. *Multiply X4 and the fourth sub key*

Here the 'first' sub key (mentioned above) is actually the 53rd subkey and the 4th sub key is actually the 56th sub key.

3.1.2.2. Advantages and Disadvantages

The 128 bit key length used by IDEA makes it impossible to break by simply trying every key, and so far no cryptanalyst has found any significant weaknesses in the algorithm, and therefore has become increasingly popular. It is a fast algorithm, and has also been implemented in hardware chipsets, making it even faster. IDEA has not been formally adopted by governments, and is patented, restricting its commercial use and cannot be used without a license.

3.1.3. Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a cipher (a method for encrypting information) selected as an official Federal Information Processing Standard (FIPS) for the United States in 1976, and which has subsequently enjoyed widespread use internationally

3.1.3.1. Working of DES

The steps given below explain the working of DES.

- Process the key.
  - *Get a 64-bit key from the user. (Every 8th bit (the least significant bit of each byte) is considered a parity bit. For a key to have correct parity, each byte should contain an odd number of "1" bits.) This key can be entered directly, or it can be the result of hashing something else. There is no standard hashing algorithm for this purpose.*

  - Calculate the key schedule.
    - *Perform the following permutation on the 64-bit key. (The parity bits are discarded, reducing the key to 56 bits. Bit 1 (the most significant bit) of the permuted block is bit 57 of the original key, bit 2 is bit 49, and so on with bit 56 being bit 4 of the original key.)*
    - *Split the permuted key into two halves. The first 28 bits are called C[0] and the last 28 bits are called D[0].*
    - *Calculate the 16 sub keys. Start with i = 1.*
      - *Perform one or two circular left shifts on both C[i-1] and D[i-1] to get C[i] and D[i], respectively.*
      - *Permute the concatenation C[i]D[i] as indicated below. This will yield K[i], which is 48 bits long.*
      - *Loop until K[16] has been calculated.*

- Process a 64-bit data block.
  - *Get a 64-bit data block. If the block is shorter than 64 bits, it should be padded as appropriate for the application.*
  - *Perform the following permutation on the data block.*
  - *Split the block into two halves. The first 32 bits are called L[0], and the last 32 bits are called R[0].*
  - *Apply the 16 sub keys to the data block. Start with i = 1.*
    - *Expand the 32-bit R[i-1] into 48 bits according to the bit-selection function.*
    - *Exclusive-or E(R[i-1]) with K[i].*
    - *Break E(R[i-1]) XOR K[i] into eight 6-bit blocks. Bits 1-6 are B[1], bits 7-12 are B[2], and so on with bits 43-48 being B[8].*
    - *Substitute the values found in the S-boxes for all B[j]. Start with j=1. All values in the S-boxes should be considered 4 bits wide.*
      - *Take the 1st and 6th bits of B[j] together as a 2-bit value (call it m) indicating the row in S[j] to look in for the substitution.*

- *Take the 2nd through 5th bits of B[j] together as a 4-bit value (call it n) indicating the column in S[j] to find the substitution.*
- *Replace B[j] with S[j][m][n].*
- *Loop until all 8 blocks have been replaced.*

- *Permute the concatenation of B[1] through B[8].*
- *Exclusive-or the resulting value with L[i-1]. Thus, all together, your R[i] = L[i-1] XOR P(S[1](B[1])...S[8](B[8])), where B[j] is a 6-bit block of E(R[i-1]) XOR K[i]. (The function for R[i] is more concisely written as, R[i] = L[i-1] XOR f(R[i-1], K[i]).)*
- *L[i] = R[i-1].*
- *Loop until K[16] has been applied.*
- *Perform the permutation on the block R[16]L[16].*

This has been a description of how to use the DES algorithm to encrypt one 64-bit block. To decrypt, use the same process, but just use the keys K[i] in reverse order. That is, instead of applying K[1] for the first iteration, apply K[16], and then K[15] for the second, on down to K[1].

3.1.3.2. Advantages and Disadvantages

The algorithm is very fast which implies that large amounts of data can be encrypted and decrypted very quickly. The advancements in computing speed and power since 1977 have made DES more vulnerable to code breakers.

The data is divided into 64-bit blocks and each block is encrypted one at a time. If data is transmitted over a network or phone line and there is a transmission error, then the error will be carried forward to all subsequent blocks since each block is dependent upon the last. Because of this reason, data that is considered sensitive by the responsible authority or data that has a high value should be cryptographically protected from unauthorized disclosure or undetected modification during transmission or while in storage.

3.1.4. Triple DES

In cryptography, Triple DES is a block cipher formed from the Data Encryption Standard (DES) cipher by using it three times. Triple DES is also known as TDES or, more standard, TDEA (Triple Data Encryption Algorithm) (Barker, 2004).

The Triple DES algorithm is a simple variant on the DES-CBC algorithm. The DES function is replaced by three rounds of that function, an encryption followed by a decryption followed by an encryption, each with independent keys, k1, k2 and k3. The following gives a simple step by step implementation of Triple DES.

- DES(k3;DES(k2;DES(k1;M))), where M is the message block to be encrypted and k1, k2, and k3 are DES keys (This variant is known as EEE because all three DES operations are encryptions).

- In order to simplify interoperability between DES and TDES the middle step is usually replaced with decryption (EDE mode)

  DES(k3;DES − 1(k2;DES(k1;M)))

  and so a single DES encryption with key 'k' can be represented as TDES-EDE with k1 = k2 = k3 = k.

3.1.4.1. Advantages and Disadvantages

By design, DES and therefore TDES, suffer from slow performance in software. TDES is better suited to hardware implementations (e.g., VPN appliances and the cellular and data network), but even there AES outperforms it. Finally, AES offers markedly higher security margins: a larger block size, potentially longer keys, and (as of 2005) freedom from cryptanalytic attacks.

3.1.5. GOST

GOST is a Soviet and Russian government standard symmetric key block cipher. Developed in the 1970s, the standard had been marked "Top Secret" and then downgraded to "Secret" in 1990. Shortly after the dissolution of the USSR, it has been declassified and released to the public. GOST was a Soviet alternative to the United States standard algorithm, DES. Thus, the two are very similar in structure.

3.1.5.1. Working of GOST

GOST has a 64-bit block size and a key length of 256 bits. The S-boxes can be kept secret, and they contain about 512 bits of secret information, so the effective key size can be increased to 768 bits; however, a chosen-key attack can recover the contents of the S-Boxes in approximately 232 encryptions (Saarinen, 1998). GOST is a Feistel network of 32 rounds. Its round function is very simple and is given below :

- add a 32-bit subkey modulo 232
- put the result through a layer of S-boxes and
- rotate that result left by 11 bits.

The result of that is the output of the round function. The subkeys are chosen in a pre-specified order.

The key schedule is given below:

- break the 256-bit key into eight 32-bit subkeys, and
- each subkey is used four times in the algorithm; the first 24 rounds use the key words in order, the last 8 rounds use them in reverse order.

The S-boxes accept a four-bit input and produce a four-bit output. The S-box substitution in the round function consists of eight 4 × 4 S-boxes. The S-boxes are implementation-dependent - parties that want to secure their communications using GOST must be using the same S-boxes. For extra security, the S-boxes can be kept secret. In the original standard where GOST was specified, no S-boxes were given, but they were to be supplied somehow. GOST is an algorithm provides flexible security and can be implemented in both hardware and software.

3.1.6. Serpent

Serpent is a symmetric key block cipher, which was a finalist in the Advanced Encryption Standard contest, where it came second to Rijndael. Serpent was designed by Ross Anderson, Eli Biham, and Lars Knudsen.

Serpent has a block size of 128 bits and supports a key size of 128, 192 or 256 bits. The cipher is a 32-round substitution-permutation network operating on a block of four 32-bit words. Each round applies one of eight 4-bit to 4-bit S-boxes 32 times in parallel. Serpent was designed so that all operations can be executed in parallel, using 32 1-bit slices. This maximises parallelism, but also allows use of the extensive cryptanalysis work performed on DES.

Serpent was widely viewed as taking a more conservative approach to security than the other AES finalists, opting for a larger security margin: the designers deemed 16 rounds to be sufficient against known types of attack, but specified 32 rounds as insurance against future discoveries in cryptanalysis.

3.1.6.1. Working of Serpent

Serpent uses a block size of 128 bits encrypting a 128-bit plaintext block P to a 128-bit ciphertext C in 32 rounds under the control of 33 128-bit subkeys, K0, ...K32. The key length varies from 128 to 256 bits long. If the key is shorter than 256 bits, a "1" is appended to the end of most significant bit, followed by as many

"0" bits as required to make up 256 bits. The algorithm consists of three phases. First phase is the initial phase where eight S-boxes are generated. The second phase is where the plain text is transformed to semi-finish ciphertext, which is then transformed to ciphertext in the third phase.

The S-box is generated using a 32 x 16 matrix. The matrix was initialized with the 32 rows of the DES S-boxes and transformed by wrapping the entries in the rth array depending on the value of the entries in the $(r+1)^{st}$ array and on an initial string representing a key. If the resulting array has the desired (differential and linear) properties, save the array as a Serpent S-box. This procedure is repeated until eight S-boxes have been generated. Then, the algorithm runs three operations thirty-two rounds: Bit-wise XOR with the 128-bit Round Key Kr, Substitution via thirty-two copies of one of eight S-boxes, Data mixing via a Linear Transformation. These operations are performed in each of the thirty-two rounds with the exception of the last round. In the last round, the Linear Transformation is replaced with a bit-wise XOR with a final 128-bit key. This algorithm can be described in equation form:

- B0 = IP(P); B0 is the input to the first round, IP is initial permutation
- Bi+1 = Ri(Bi); Bi+1 is the out put of round Bi

Ri(Y) = L(Si(Y XOR Ki)), where i = 0,..,30 and

Ri(Y) = L(Si(Y XOR K32)), i = 31

Sj is the application of S-box, Sj mod 8, 32 times in parallel, j = 0,…,7

L is linear transformation.

- C = FP(B32); C is the ciphertext, FP is the final permutation, which is the inverse of the initial permutation.

3.1.6.2. Advantages and Disadvantages

The advantages include no weak keys or no semi-weak keys. The different rounds in the algorithm uses different S-boxes thus reducing the possibility of

attacks. The weakness lies mainly in the fixed substitution table, key distribution and key management. Serpent achieves its high performance by a design that makes very efficient use of parallelism, and this extends beyond the level of the algorithm itself.

3.1.7. Tiny Encryption Algorithm (TEA)

In cryptography, the Tiny Encryption Algorithm (TEA) is a block cipher notable for its simplicity of description and implementation. It was designed by David Wheeler and Roger Needham of the Cambridge Computer Laboratory, and first presented at the Fast Software Encryption workshop in 1994 (Wheeler and Needham, 1994). It is not subject to any patents. Andem (2003) presented a comprehensive study of TEA and cryptanalysis of the TEA.

3.1.7.1. TEA Encryption

The following shows the TEA implementation using C syntax.

```
void code(long* v, long* k) {
unsigned long y=v[0],z=v[1], sum=0,   /* set up */
delta=0x9e3779b9, n=32 ;          /* a key schedule constant */
while (n-->0) {                   /* basic cycle start */
 sum += delta ;
   y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
   z += (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;   /* end cycle */
         }
v[0]=y ; v[1]=z ; }
```

3.1.7.2. TEA Decryption

Given below is the decryption C code for TEA algorithm.

```
void decode(long* v,long* k) {
 unsigned long n=32, sum, y=v[0], z=v[1],
 delta=0x9e3779b9 ;
sum=delta<<5 ;
                  /* start cycle */
while (n-->0) {
```

```
    z-= (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;
    y-= (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
   sum-=delta ;  }
                /* end cycle */
v[0]=y ; v[1]=z ;  }
```

3.1.7.3. Advantages and Disadvantages

The code is lightweight and portable enough to be used just about anywhere. It even makes a great random number generator for Monte Carlo simulations and the like. The minor weaknesses identified by David Wagner at Berkeley are unlikely to have any impact in the real world, and one can always implement the new variant TEA which addresses them. If one wants a low-overhead end-to-end cipher (for real-time data, for example), then TEA fits the bill.

3.1.8. Skipjack

In cryptography, Skipjack is a block cipher, developed by the U.S. National Security Agency (NSA). Initially classified, it was originally intended for use in the controversial Clipper chip. Subsequently, the algorithm was declassified and now provides a unique insight into the cipher designs of a government intelligence agency.

3.1.8.1. Working of Skipjack

Skipjack encrypts and decrypts data in 64-bit blocks, using an 80-bit key. It takes a 64-bit block of plaintext as input and outputs a 64-bit block of ciphertext. Skipjack has 32 rounds, meaning the main algorithm is repeated 32 times to produce the ciphertext. It has been found that the number of rounds is exponentially proportional to the amount of time required to find a key using a brute-force attack. As the number of rounds increases, the security of the algorithm increases exponentially.

3.1.9. Twofish

In cryptography, Twofish is a symmetric key block cipher with a block size of 128 bits and key sizes up to 256 bits. It was one of the five finalists of the Advanced Encryption Standard contest, but was not selected for standardisation. Twofish is related to the earlier block cipher Blowfish. Twofish's distinctive features are the use of pre-computed key-dependent S-boxes, and a relatively complex key schedule. Twofish borrows some elements from other designs; for example, the Pseudo-Hadamard Transform (PHT) from the SAFER family of ciphers. Twofish uses the same Feistel structure as DES. Twofish was designed by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson (1999); the "extended Twofish team" who met to perform further cryptanalysis of Twofish and other AES contest entrants included Stefan Lucks, Tadayoshi Kohno, and Mike Stay. Twofish can:

- Encrypt data at 285 clock cycles per block on a Pentium Pro, after a 12700 clock-cycle key setup.
- Encrypt data at 860 clock cycles per block on a Pentium Pro, after a 1250 clock-cycle key setup.
- Encrypt data at 26500 clock cycles per block on a 6805 smart card, after a 1750 clock-cycle key setup.

3.1.9.1. Working of Twofish

Twofish uses a 16-round Feistel-like structure with additional whitening of the input and output. The only non-Feistel elements are the 1-bit rotates. The rotations can be moved into the F function to create a pure Feistel structure, but this requires an additional rotation of the words just before the output whitening step. The following steps explains the Twofish algorithm

- *Split the plaintext into four 32-bit words.*
- *XOR these four key words, followed by sixteen rounds.*

- *In each round, the two words on the left are used as input to the g functions.*
- *The g function consists of four byte-wide key-dependent S-boxes, followed by a linear mixing step based on an MDS matrix. The results of the two g functions are combined using a Pseudo-Hadamard Transform (PHT), and two keywords are added.*
- *These two results are then XORed into the words on the right (one of which is rotated left by 1 bit first, the other is rotated right afterwards).*
- *The left and right halves are then swapped for the next round.*
- *After all the rounds, the swap of the last round is reversed, and the four words are XORed with four more key words to produce the ciphertext.*

3.1.9.2. Advantages and Disadvantages

Twofish algorithm works in all standard models and is highly efficient in key setup on large microprocessors, in smart cards and in hardware. It is unpatented and uncopyrighted and therefore can be used freely. It has been effectively cryptanalysed and therefore is very secure. It is vulnerable to divide-and-conquer attack of the key space. It lacks simplicity and the weakness of Twofish mainly arises because of key distribution and key management.

The working of various cryptographic algorithms, in particular, the symmetric algorithm are discussed in this section. The performance evaluation in terms of time is discussed in Chapter IV.

3.2. HASH FUNCTIONS

In cryptography, a cryptographic hash function is a hash function with certain additional security properties to make it suitable for use as a primitive in various information security applications, such as authentication and message integrity. A hash function takes a long string (or message) of any length as input and produces a fixed length string as output, sometimes termed a message digest or a digital fingerprint. The length of the output being dependent on the particular

function or algorithm being used. In various standards and applications, the most-commonly used hash functions are MD2, MD5, RIPEMD-160 and SHA.

Broadly speaking, a cryptographic hash function should behave as much as possible like a random function while still being deterministic and efficiently computable. A cryptographic hash function is considered insecure if either of the following is computationally feasible:

- finding a (previously unseen) message that matches a given digest
- finding "collisions", wherein two different messages have the same message digest.

An attacker who can do either of these things might, for example, use them to substitute an unauthorized message for an authorized one. Ideally, it should not even be feasible to find two messages whose digests are substantially similar; nor would one want an attacker to be able to learn anything useful about a message given only its digest besides the digest itself.

Applications of hash functions

- Commitment Scheme

A typical use of a cryptographic hash would be as follows: Alice poses to Bob a tough math problem and claims she has solved it. Bob would like to try it himself, but would yet like to be sure that Alice is not bluffing. Therefore, Alice writes down her solution, appends a random nonce, computes its hash and tells Bob the hash value (whilst keeping the solution secret). This way, when Bob comes up with the solution himself a few days later, Alice can verify his solution but still be able to prove that she had the solution earlier. In actual practice, Alice and Bob will often be computer programs, and the secret would be something less easily spoofed than a claimed puzzle solution. The above application is called a commitment scheme.

- Verification of message integrity

Another important application of secure hashes is verification of message integrity. Determination of whether or not any changes have been made to a message (or a file), for example, can be accomplished by comparing message digests calculated before, and after, transmission (or any other event.

- Password Verification

A message digest can also serve as a means of reliably identifying a file. A related application is password verification. Passwords are usually not stored in clear text, for obvious reasons, but instead in digest form. To authenticate a user, the password presented by the user is hashed and compared with the stored hash.

Classification of hash functions

At the highest level, hash functions may be split into two classes:

- unkeyed hash functions : Functions whose specification dictates a single input parameter (a message); and
- keyed hash functions : Functions whose specification dictates two distinct inputs, a message and a secret key.

Non-keyed hash functions may be calculated by anyone, whereas keyed hash functions require the correct key value be supplied to reproduce a hash value. Common non-keyed hash functions include MD family (MD2, MD4, MD5) and the SHA; keyed hash functions include the DAC and RIPE-MAC. The hash functions MD2, MD5, SHA and RIPEMD 160 are implemented and explained in detail in the following sections.

3.2.1. MESSAGE DIGEST ALGORITHM 2 (MD2)

Message Digest Algorithm 2 (MD2) is a cryptographic hash function developed by Ronald Rivest in 1989. The algorithm is optimized for 8-bit computers. MD2 is specified in RFC 1319. Although other algorithms have been proposed since, such as MD4, MD5 and SHA, even as of 2004 MD2 remains in use in public key infrastructures as part of certificates generated with MD2 and RSA.

3.2.1.1. Description

The 128-bit hash value of any message is formed by padding it to a multiple of the block length on the computer (128 bits or 16 bytes) and adding a 16-byte checksum to it. For the actual calculation, a 48-byte auxiliary block and a 256-byte table generated indirectly from the digits of the fractional part of pi are used. Once all of the blocks of the (lengthened) message have been processed, the first partial block of the auxiliary block becomes the hash value of the message.

3.2.2. Message Digest Algorithm 5 (MD5)

In cryptography, MD5 (Message-Digest algorithm 5) is a widely-used cryptographic hash function with a 128-bit hash value. As an Internet standard (RFC 1321), MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. MD5 was designed by Ronald Rivest in 1991 to replace an earlier hash function, MD4. In 1996, a flaw was found with the design of MD5; while it was not a clearly fatal weakness, cryptographers began to recommend using other algorithms, such as SHA-1. In 2004, more serious flaws were discovered making further use of the algorithm for security purposes questionable.

3.2.2.1. Working of MD5

MD5 processes a variable length message into a fixed-length output of 128 bits. The pseudocode for MD5 is given as below :

- *The input message is broken up into chunks of 512-bit blocks; the message is padded so that its length is divisible by 512.*

  The padding works as follows:

  - *first a single bit, 1, is appended to the end of the message.*
  - *This is followed by as many zeros as are required to bring the length of the message up to 64 bits fewer than a multiple of 512.*
  - *The remaining bits are filled up with a 64-bit integer representing the length of the original message.*
- *The 128-bit state is divided into four 32-bit words, denoted A, B, C and D.*
- *Initialize them to fixed constants.*
- *Process each 512-bit message block, which consist of four similar stages, termed rounds.*
- *Each round is composed of 16 similar operations based on a non-linear function F, modular addition, and left rotation. There are four possible functions F; a different one is used in each round:*

  F(X, Y, Z) = (X AND Y) OR (NOT(X) AND Z)
  G(X,Y,Z) = (X AND Y) OR (NOT(X) AND Z)
  H(X, Y, Z) = X XOR Y XOR Z
  I(X, Y, Z) = Y XOR (X OR (NOT(Z)))
- *The output of the fourst round is added to the input to the first round to produce the result.*
- *After all 512 blocks have been processed, the outut from the the last stage is the 128-bit message digest.*

3.2.3. Race Integrity Primitives Evaluation Message Digest (RIPEMD 160)

RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digest) is a 160-bit message digest algorithm developed in Europe by Hans Dobbertin, Antoon Bosselaers and Bart Preneel, and first published in 1996. It is an improved version of RIPEMD, which in turn was based upon the design principles used in MD4, and is similar in performance to the more popular SHA. There also exist 128, 256 and

320-bit versions of this algorithm, called RIPEMD-128, RIPEMD-256, and RIPEMD-320, respectively. RIPEMD-160 was designed in the open academic community, in contrast to the NSA-designed algorithm, SHA. On the other hand, RIPEMD-160 is a less popular and correspondingly less well-studied design. RIPEMD-160 is not constrained by any patents. In August 2004, a collision was reported for the original RIPEMD.

3.2.3.1. RIPEMD-160 Algorithm

The algorithm takes as input a message of arbitrary length and produces as output a 160-bit message digest. The input is processed in 512-bit blocks. The processing consist of the following steps.

- *The message is padded so that its length is congruent to 448 modulo 512.*
- *A block of 64 bits is appended to the message, which is treated as an unsigned 64-bit integer and contains the length of the original message.*
- *To hold the intermediate and final results, a 160-bit buffer is used, which is represented as five 32-bit registers (A, B, C, D, E). These registers are initialized to the following hexadecimal values :*

  A=67452301,B=EFCDAB89, C=98BADCFE, D=10325476, E=C3D2E1F0

- *Process message in 512-bit (16 word) blocks which are processed in 10 rounds through 16 steps each. This is explained as below.*
  - *Arrange the 10 rounds as two parallel lines of five rounds (f1, f2, f3, f4, f5).*
  - *For each round*
    - *Take as input the current 512-bit block being processed and the 160-bit buffer values ABCDE and update the contents of the buffer.*
    - *The output of the fifth round (18^{th} round) is added to the chaining variable input to the first round.*
- *After all 512-bit blocks are processed, the output from the last stage is the 160-bit message digest.*

3.2.4. Secure Hash Algorithm (SHA) Hash function

The SHA (Secure Hash Algorithm) family is a set of related cryptographic hash functions. The first member of the family, published in 1993, is officially called SHA; however, it is often called SHA-0 to avoid confusion with its successors. Two years later, SHA-1, the first successor to SHA, was published. Four more variants have since been issued with increased output ranges and a slightly different design: SHA-224, SHA-256, SHA-384, and SHA-512, collectively referred to as SHA-2.

3.2.4.1. SHA ALGORITHM

The algorithm takes as input a message with a maximum length of less than 2^{64} bits and produces as output a 160-bit message digest. The input is processed in 512 bit blocks. The process of SHA can be summarized as below :

- *The message is padded so that its length is congruent to 488 modulo 512.*
- *Append a block of 64 bits to the message. This block is treated as an unsigned 64-bit integer and contains the length of the original message.*
- *To hold the intermediate and final result initialize a 160-bit buffer, which is represented as five 32 bit registers (A, B, C, D, E), initialized to the 32 bit integers in hexa form (67452301, EFCDAB89, 98BADCFE, 10325476, C3D2E1F0).*
- *Process message in 512-bit (16-word) blocks, consisting of four rounds each of which is processed in 20 steps. All the four rounds have similar structures and are refered as f1, f2, f3 and f4. The process is explained below :*
    - *For each round, take as input the curret 512-bit block and 160-bit buffer value ABCDE and update the content of the buffer.*
    - *The output of the fourth round is added to the input of the first round.*
- *Aftr all 512-bit blocks have been processed, the output from the last stage is the 160-bit message digest.*

The performance evaluation of the hash function cryptographic techniques discussed here are dealt in Chapter IV.

3.3. ASYMMETRIC ALGORITHM

Symmetric algorithms, as discussed above, provide efficient and powerful cryptographic solutions, especially for encrypting bulk data. However, under certain circumstances, symmetric algorithms can come up short in two important respects: key exchange and trust. In this section, these two shortcomings are considered and suggestions to solve those using asymmetric algorithms are presented. The working of asymmetric algorithms, in particular, the RSA algorithm and ECC algorithm, which are currently the most popular asymmetric algorithms, are also dealt with.

3.3.1. Problems with symmetric algorithms

One big issue with using symmetric algorithms is the key exchange problem, referred as catch-22 problem. The other main issue is the problem of trust between two parties that share a secret symmetric key. Problems of trust may be encountered when encryption is used for authentication and integrity checking. A symmetric key can be used to verify the identity of the other communicating party, but requires that one party trust the other.

3.3.1.1. The Key Exchange Problem

The key exchange problem arises from the fact that communicating parties must somehow share a secret key before any secure communication can be initiated, and both parties must then ensure that the key remains secret. This issue can be dealt with effectively by using asymmetric algorithms.

3.3.1.2. The Trust Problem

Ensuring the integrity of received data and verifying the identity of the source of that data is very important. A symmetric key can be used to check the identity of the individual who originated a particular set of data, but this authentication scheme can encounter some thorny problems involving trust. This technique will work only if the other party with whom the data is send can be trusted. This scheme cannot discriminate between the two individuals who know the shared key.

Other problems could arise if sender's partner shared the secret key with others without telling the sender about it. The sender has no stand on if certain disputes were to arise. This problem is known as repudiation, and often needs a way to enforce nonrepudiation between untrusting parties.

Asymmetric algorithms can be used to solve these problems by performing the same basic operations but encrypting the hash using a private key (belonging to an asymmetric key pair) that one individual and only one individual knows. Then anyone can use the associated public key to verify the hash. This effectively eliminates the problems of trust and repudiation. This technique is called a digital signature.

3.3.2. General working of asymmetric cryptography

The following figure (Fig. 3.11) shows the working and usage of asymmetric cryptography.

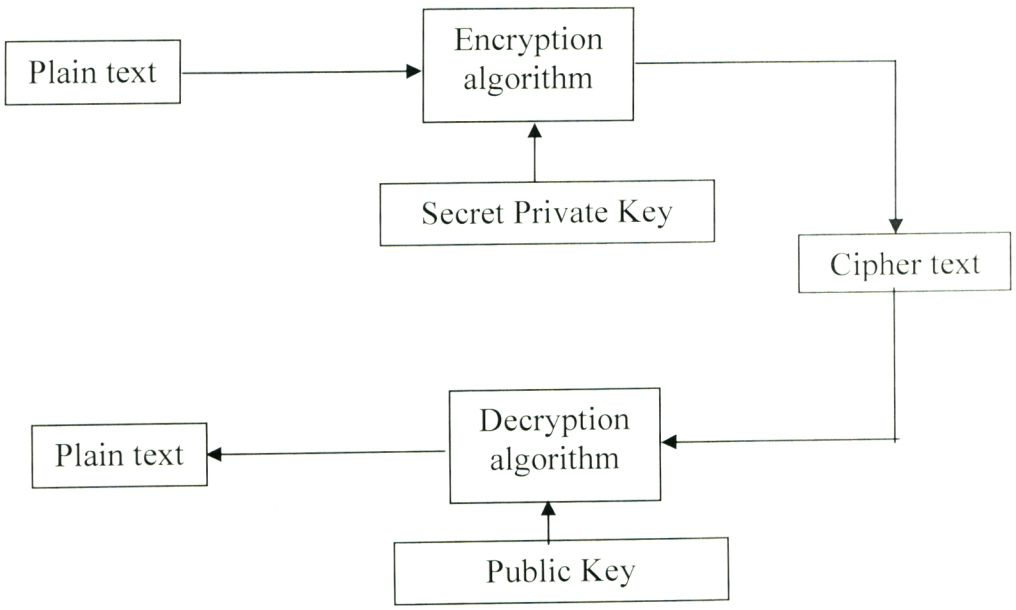


Fig 3.1. Working of Asymmetric Cryptography

To use asymmetric cryptography, Bob randomly generates a public/private key pair. He allows everyone access to the public key, including Alice. Then, when Alice has some secret information that she would like to send to Bob, she encrypts the data using an appropriate asymmetric algorithm and the public key generated by Bob. She then sends the resulting ciphertext to Bob. Anyone who does not know the matching secret key will have an enormously difficult time retrieving the plaintext from this ciphertext, but since Bob has the matching secret key (i.e., the trapdoor information), Bob can very easily discover the original plaintext.

3.3.3. RSA Algorithm

RSA is a highly secure, public key encryption algorithm which uses a public key and private key to encrypt and decrypt a message. "Public key" means that there are two different keys: a public key that is released publically so anyone can find it, and a private key that is kept secret. The public key is used to encrypt a message, and the private key is used to decrypt it. The security that RSA provides

comes from the fact that it is very difficult to find out what the private key is based on the public key.

3.3.1. Working of RSA

The working of RSA cipher is explained in this section. First, a public and private key pair is randomly generated. As is always the case in cryptography, it is very important to generate keys in the most random and unpredictable manner possible. Then, the data is encrypted with the public key, using the RSA algorithm. Finally, the encrypted data is decrypted with the private key. Encryption is done using the public key and decryption using the private key. This achieves confidentiality. The following are the steps involved while generating the public and private keys.

1. *Randomly select two prime numbers p and q. For the algebra to work properly, these two primes must not be equal. To make the cipher strong, these prime numbers should be large, and they should be in the form of arbitrary precision integers with a size of at least 1024 bits.*
2. *Calculate the product:* $n = p \cdot q$.
3. *Calculate the Euler quotient for these two primes, which is represented by the Greek letter* φ. *This is easily computed with the formula*

$$\varphi = (p - 1) \cdot (q - 1).$$

The Euler quotient, symbolized with the Greek letter π, *represents the number of positive integers less than or equal to n that are relatively prime to 'n' (i.e., have no prime factors in common with n). One is considered to be relatively prime with all integers.*

4. *After calculating n and n and* φ, *the values p and q should be destroyed, since it is very easy for an attacker to reconstruct the key pair and decipher ciphertext if these values are known.*
5. *Select a number 'e' randomly that is greater than 1, less than* φ, *and relatively prime to* φ. *Two numbers are said to be relatively prime if they have no prime factors in common and 'e' does not necessarily have to be prime. The value of e is used along with the value n to represent the public key used for encryption.*
6. *Calculate the unique value d (to be used during decryption) that satisfies the requirement that, if* $d \cdot e$ *is divided by* φ, *then the remainder of the division is 1. The mathematical notation for this is* $d \cdot e = 1 \pmod{\varphi}$. *Mathematically 'd' is said to be the multiplicative inverse of e modulo* φ. *The value of d is to be kept secret.*

If the value of φ is known, the value of 'd' can be easily obtained from 'e' using a technique known as the Euclidean algorithm. If one has the 'n' value (which is public), but not 'p' or 'q', then the value of φ is very hard to determine. The secret value of d together with the value n represents the private key.

Once the public/private key pair is generated, the message can be encrypted with the public key using the following steps.

1. *Take a positive integer 'm' to represent a piece of plaintext message. In order for the algebra to work properly, the value of m must be less than the modulus 'n', which was originally computed as p · q. Long messages must therefore be broken into small enough pieces, so that each piece can be uniquely represented by an integer of this bit size, and each piece is then individually encrypted.*

2. *Calculate the ciphertext 'c' using the public key containing e and n. This is calculated using the equation c = me (mod n).*

Finally, the decryption procedure is performed with the private key using the following steps.

1. *Calculate the original plaintext message from the ciphertext using the private key containing d and n. This is calculated using the equation m = cd (mod n).*

2. *Compare this value of m with the original m, and make sure that they are equal, since decryption is the inverse operation to encryption.*

3.3.2. Key generation

- Finding the large primes p and q is usually done by testing random numbers of the right size with probabilistic primality tests, which quickly eliminate virtually all non-primes.

- p and q should not be 'too close', for n be successful. Furthermore, if either p-1 or q-1 has only small prime factors, n can be factored quickly and these values of p or q should therefore be discarded as well.

- One should not employ a prime search method which gives any information whatsoever about the primes to the attacker. In particular, a good random

number generator for the start value needs to be employed, which is both 'random' and 'unpredictable'. These are not the same criteria; a number may have been chosen by a random process (ie, no pattern in the results), but if it is predictable in any manner (or even partially predicatable), the method used will result in loss of security.

- Here, the secret key 'd' should be large

3.3.4. Elliptical Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is emerging as an attractive public-key cryptosystem for mobile and wireless environments. Compared to traditional cryptosystems like RSA, ECC offers equivalent security with smaller key sizes, which results in faster computations, lower power consumption, as well as memory and bandwidth savings. This is especially useful for mobile devices which are typically limited in terms of their CPU, power and network connectivity. Elliptic Curve Cryptography (ECC) has recently been endorsed by the US government. Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor S. Miller in 1985.

3.3.4.1 Need for ECC

The need for elliptic code cryptography is best defined using the following facts.

- First, the fact that the security and practicality of a given asymmetric cryptosystems relies upon the difference in difficulty between doing a given operation and its inverse.

- Second, the fact that the difference in difficulty between the forward and the inverse operation in a given system is a function of the key length in use, due to the fact that the difficulty of the forward and the inverse operations increase as very different functions of the key length; the inverse operations get harder faster.
- Third, the fact that one is forced to use longer key lengths to adjust to the greater processing power now available to attack the cryptosystem, even the 'legitimate' forward operations get harder, and require greater resources (chip space and/or processor time), though by a lesser degree than do the inverse operations.

Based on the above facts, implementation of ECC gives more advantages. As security requirements become more stringent, and as processing power gets cheaper and more available, ECC becomes the more practical system for use. As security requirements become more demanding, and processors become more powerful, considerably more modest increases in key length are necessary, ECC cryptosystem is used, to address the threat.

This keeps ECC implementations smaller and more efficient than other implementations. ECC can use a considerably shorter key and offer the same level of security as other asymmetric algorithms using much larger ones. Moreover, the gulf between ECC and its competitors in terms of key size required for a given level of security becomes dramatically more pronounced, at higher levels of security.

3.3.4.2. Working of ECC

An Elliptic Curve Cryptosystem (ECC) operates over points on an elliptic curve. The way that the elliptic curve operations are defined is what gives ECC its higher security at smaller key sizes. An elliptic curve is defined in a standard, two dimensional x,y Cartesian coordinate system by an equation of the form:

$$y^2 = x^3 + ax + b$$

The graph, when plotted with the above equation, turns out to be gently looping lines of various forms. The encryption and decryption algorithm is explained by using the following steps.

- *Let the finite field be GF(p) and the elliptic curve be E.*
- *Choose randomly a base point (x, y) lying on the elliptic curve E.*
- *Code the plaintext into an elliptic curve point (xm, ym)*
- *Each user selects a private key n and compute his/her public key P = n(x, y). For example, user A's private key is nA and the public key is PA = nA(x, y). For any one to encrypt and send the message point (xm, ym) to user A, he/she needs to choose a random integer k and generate the ciphertext Cm = {k(x, y), (xm, ym) + kPA}. The ciphertext pair of points uses A's public key, where only user A can decrypt the plain text using his/her private key.*
- *To decrypt the ciphertext Cm, the first point in the pair of Cm, k(x, y), is multiplied by A's private key to get the point : nA(k(x, y)). Then this point is subtracted from the second point of Cm, the result will be the plaintext point (xm, ym). The decryption operation is summarized as below :*

((xm, ym) + kPA – nA(k(x, y)) = (xm, ym) + k(nA(x, y))- nA(k(x, y)) = (xm, ym)

3.3.4.3. What makes ECC hard to crack?

The security of ECC relies on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP), i.e. finding k, given P and Q = kP. The problem is computationally intractable for large values of k. Among other things, this makes it possible for two entities to agree on a shared secret across an insecure communication channel without revealing that secret to an eavesdropper. This secret can then be used as a key to encrypt/decrypt sensitive information.

Since the best known algorithm to attack ECC runs more slowly than the best known algorithm to attack other cryptosystems, ECC can offer equivalent security with substantially smaller key sizes. For example, a 160-bit ECC key provides the same level of security as a 1024-bit RSA key and 224-bit ECC is

equivalent to 2048-bit RSA. Smaller keys result in faster computations, lower power consumption, as well as memory and bandwidth savings. While these characteristics make ECC especially appealing for small embedded devices, they can also alleviate the computational burden on secure web servers.

The results obtained while executing these lists of algorithms are discussed in detail in Chapter IV.

Results and Discussion

4. RESULTS AND DISCUSSION

The results related to the research work "Performance comparison of cryptographic functions in secret key cryptography, public key cryptography and hash algorithms" with respect to time factor is discussed under the following sections.

4.1. Performance Evaluation of Symmetric Cryptography

4.2. Performance Evaluation of Hash Function Cryptography

4.3. Performance Evaluation of Asymmetric Cryptography

Speed is considered as a significant factor while evaluating the execution of any algorithm and therefore, in this research work, a comparison is performed between these algorithms based on the CPU execution time. The performance of the algorithms is evaluated in terms of the processing time required in the kernel and user space for generating the secret key, encryption and decryption operations

4.1. PERFORMANCE EVALUATION OF SYMMETRIC CRYPTOGRAPHY

The nine symmetric cryptography algorithms are tested and the results of their performance are given in this section. Table 4.1 and Figure. 4.1 show the performance comparison of various ciphers based on time, when the file size is equivalent to 10MB.

From these results it is clear that Blowfish outperforms all other algorithms in terms of speed, closely followed by TDES and DES. Serpent algorithm takes the maximum time encrypt and decrypt the file. Even though GOST and Twofish algorithms work slower when compared to Blowfish, DES and TDES, still it can be substituted for this three in extreme cases, since the time increase is negligible.

TABLE 4.1 : PERFORMANCE COMPARISON OF SYMMETRIC CRYPTOGRAPHY ALGORITHMS

Algorithm	Encryption Time(ms)	Decryption Time(ms)
Blowfish	0.14	0.09
IDEA	0.3	0.2
DES	0.19	0.18
TDES	0.15	0.12
GOST	0.2	0.1
Skipjack	0.44	0.2
TEA	0.4	0.2
Serpent	0.5	0.3
Twofish	0.21	0.25

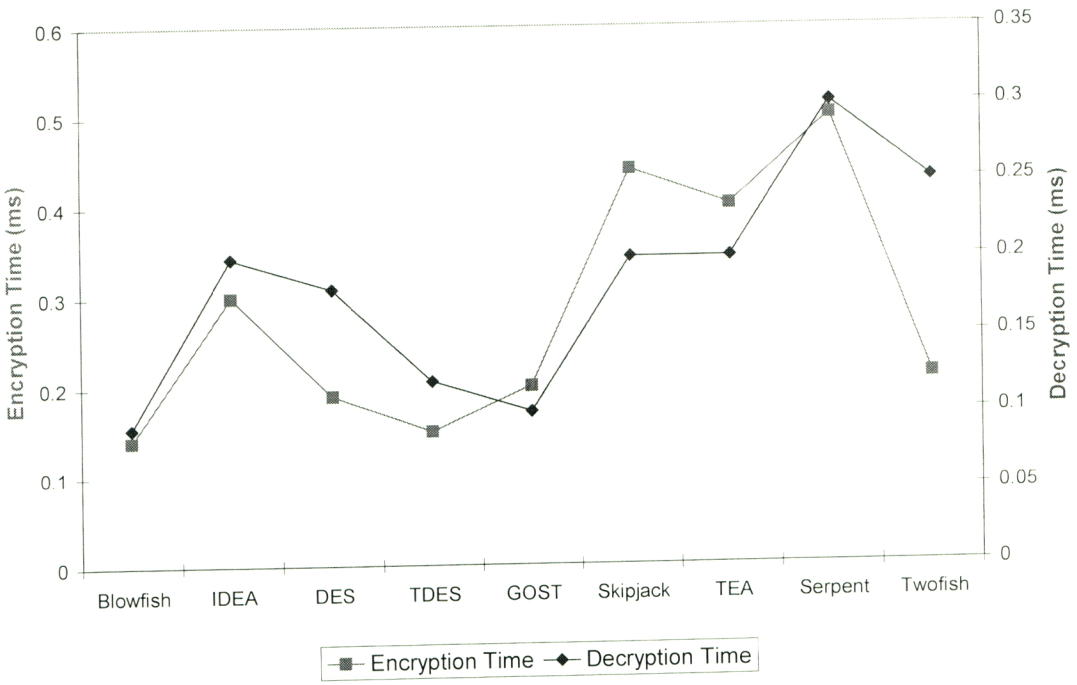


Fig. 4.1 : Time Comparison of Block Ciphers

4.2. PERFORMANCE EVALUATION OF HASH FUNCTION CRYPTOGRAPHY

This section analysis the performance of hash function crypotography techniques in terms of executive speed.

4.2.1. MD2

Table 4.2 shows the time analysis in milliseconds for MD2 hash function for various message length. Figure 4.2 illustrates the same in pictographic form. It is clearly seen from the figure that when the message size increases, the performance of the algorithm also increases, that is, the speed of the algorithm is inversely proportional to the message size.

TABLE 4.2: TIME ANALYSIS OF MD-2 HASH FUNCTION

Size(bytes)	Time (mS)
1	3.03
16	3.0561
32	2.8769
48	2.543
64	2.2565
80	2.1675
128	2.3212
256	1.4982
512	1.2253
1024	1.0321
2048	1.4561
16384	1.4882
65536	1.2301

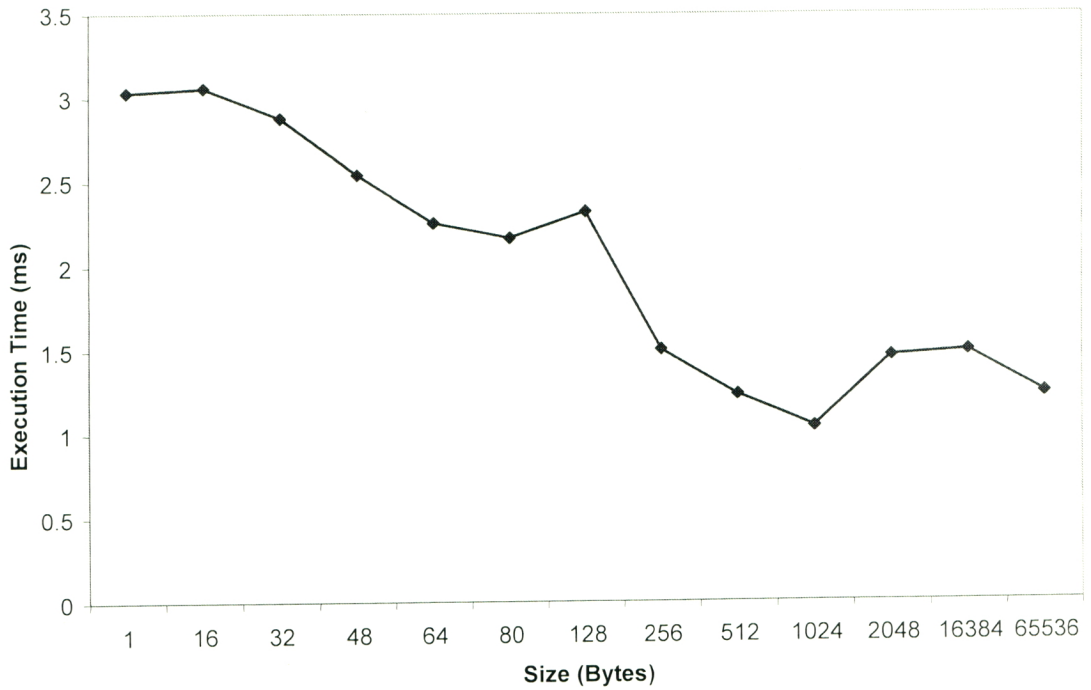


Fig. 4.2 : MD2 HASH FUNCTION CRYPTOGRAPHY

4.2.2. MD5 HASH FUNCTION

MD5 digests have been widely used in the software world to provide assurance that a downloaded file has not been altered. MD5 is widely used to store passwords. The performance of MD5 algorithm in terms of speed of execution is shown in Table 4.3. Figure 4.3 shows the time analysis in milliseconds for MD5 hash function for various message lengths.

From the values, it is evident that the speed of MD5 decreases as the message size increases. When compared with MD2, the performance of MD5 algorithm has increased with respect of execution time.

TABLE 4.3 : TIME ANALYSIS OF MD-5 HASH FUNCTION

Size(bytes)	Time (mS)
1	2.6525
16	2.6715
32	2.2122
48	1.8914
64	1.5223
80	1.3021
128	1.0002
256	1.2888
512	1.1032
1024	0.9889
2048	0.8776
16384	0.7654
65536	1.5234

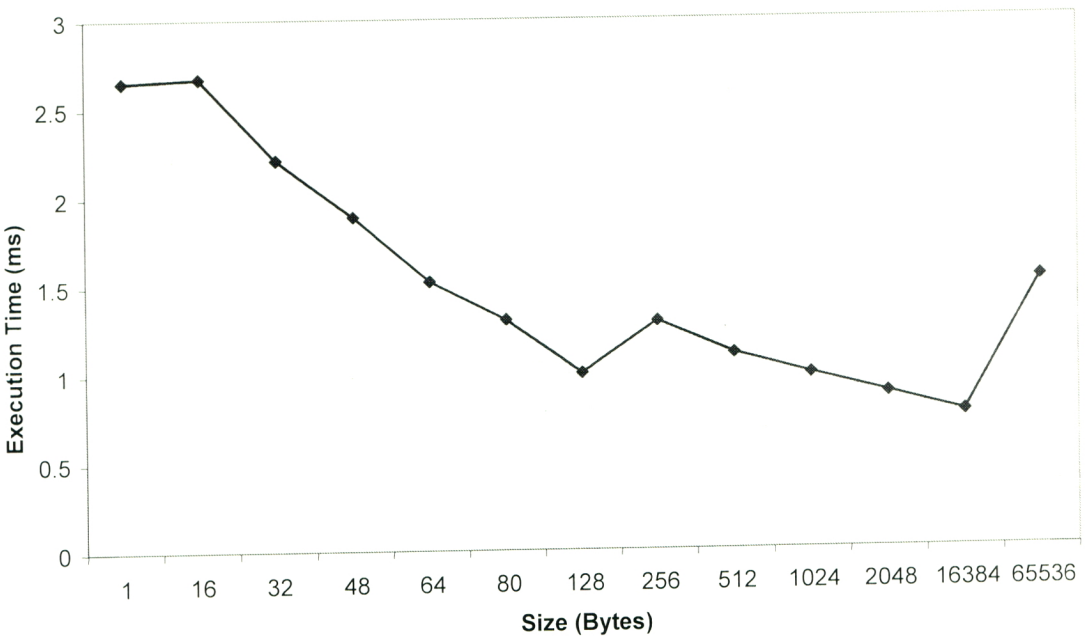


Fig. 4.3 : MD5 HASH FUNCTION CRYPTOGRAPHY

4.2.3. RIPEMD 160

Table 4.4 and Figure 4.4 shows the time analysis in milliseconds for RIPEMD 160 hash function for various message length.

TABLE 4.4 : TIME ANALYSIS OF RIPEMD 160 HASH FUNCTION

Size(bytes)	Time (mS)
1	0.0122
16	0.0106
32	0.0094
48	0.0087
64	0.0084
80	0.0084
128	0.0085
256	0.0087
512	0.0088
1024	0.0087
2048	0.0088
16384	0.0089
65536	0.0093

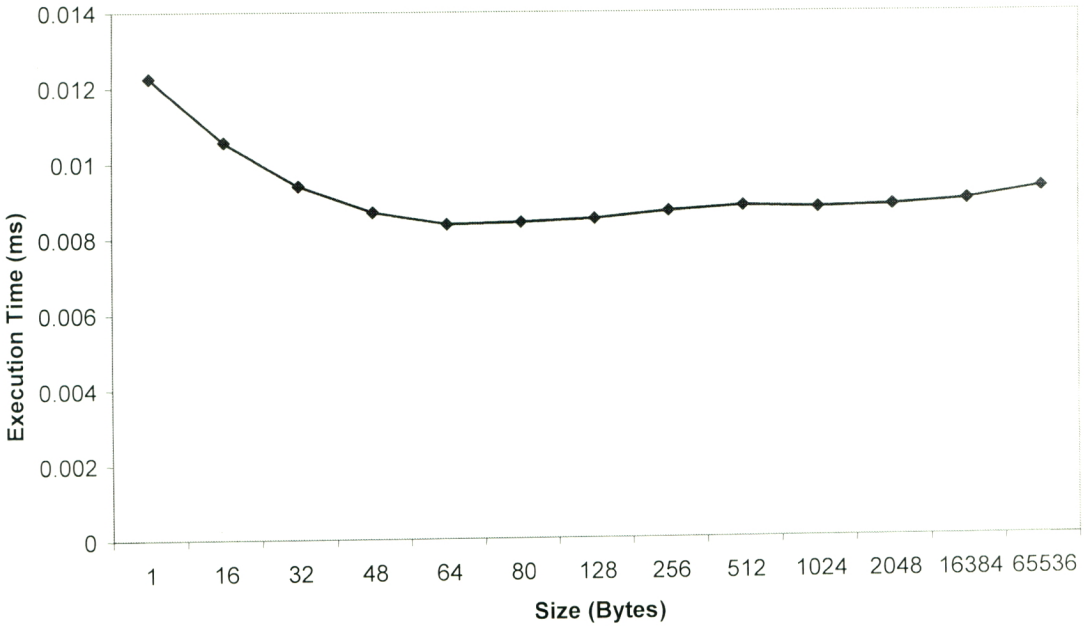


Fig. 4.4 : RIPEMD 160 HASH FUNCTION CRYPTOGRAPHY

Figure 4.4 clearly conveys that the performance of the algorithm even though initially starts in a decreasing trend, stabilizes with the increase in the message size. This is the reason behind using RIPEMD algorithm extensively with e-mail message protection.

4.2.4. SHA

Table 4.5 and Figure 4.5 shows the time in milliseconds needed to compute the SHA hash of messages of various lengths. Very short messages (1 byte) takes longer than 16 byte messages, because internally SHA pads messages to a whole number of blocks and this padding process takes a measurable amount of time. Apart from this effect, the time taken to compute a hash is approximately a constant and it is proportional to the size of the message.

TABLE 4.5 : TIME ANALYSIS OF SHA HASH FUNCTION

Size(bytes)	Time (mS)
1	0.0153
16	0.04
32	0.03
48	0.04
64	0.11
80	0.16
128	0.32
256	0.47
512	0.74
1024	1.25
2048	1.66
16384	2.14
65536	2.82

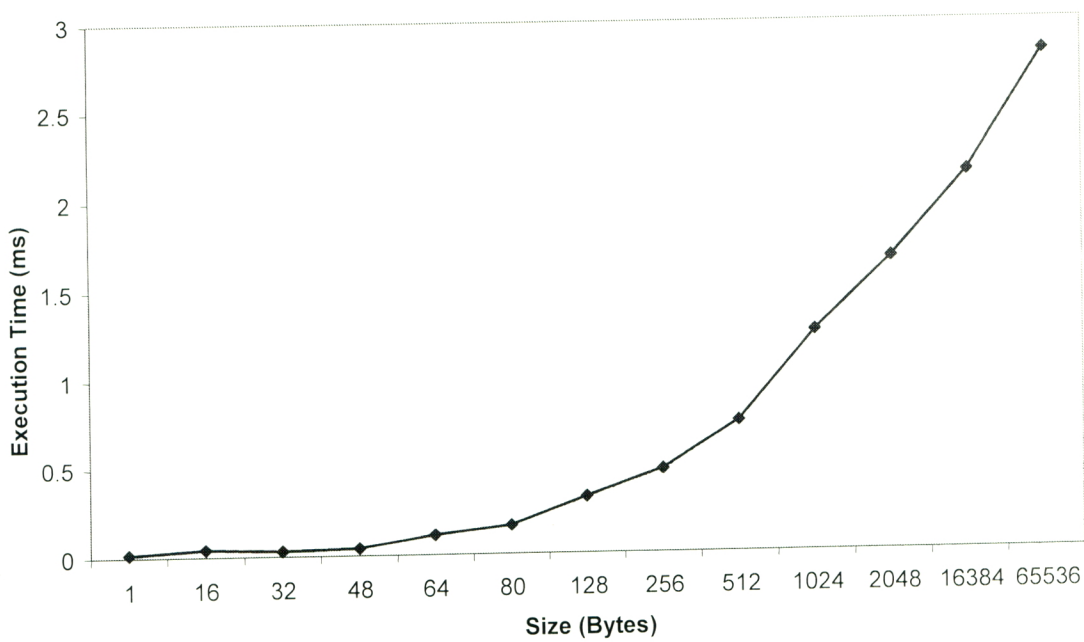


Fig. 4.5 : SHA HASH FUNCTION CRYPTOGRAPHY

Considering the overall performance of the four hash algorithms considered in this dissertation, it is observed that, within hash function cryptography, RIPEMD achieves better performance than SHA and MD families. This fact is supported with the Table 4.6 and Fig. 4.6. Implementations are tested with Pentium processor and the data size was 256 Megabytes.

TABLE 4.6 : PERFORMANCE COMPARISON OF HASH FUNCTION CRYPTOGRAPHY

Algorithm	Performance (in millisecs)
MD4	81.4
MD5	59.7
SHA	21.2
RIPEMD-160	19.3

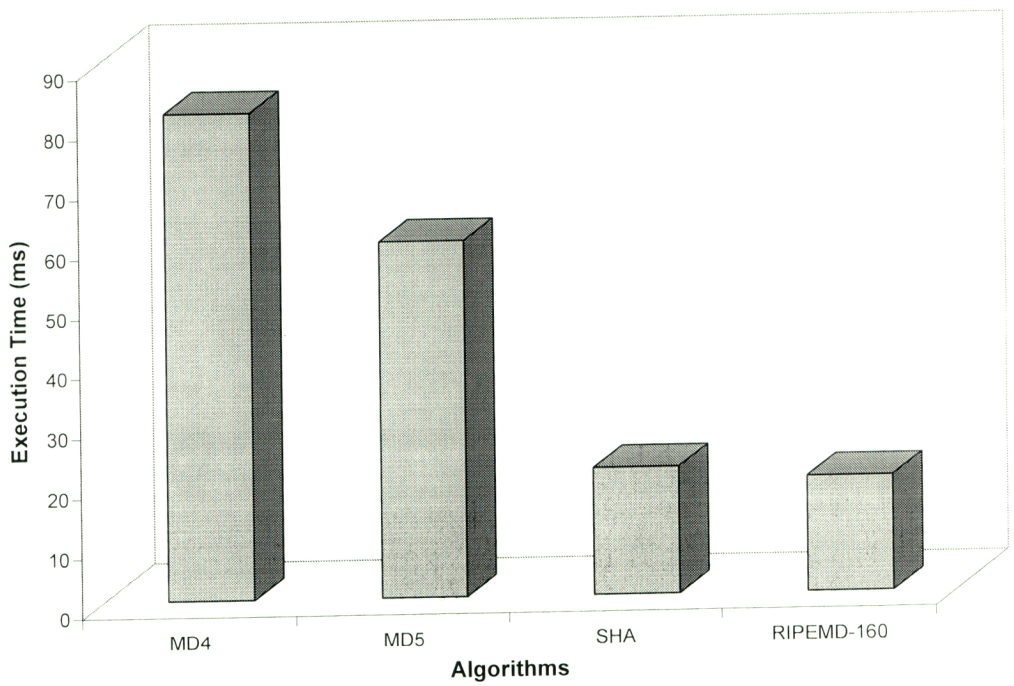


Fig. 4.6 : COMPARISON OF CRYPTOGRAPHIC HASH FUNCTIONS

From the above results, it is clear that RIPEMD-160 cryptographic hash function, which is an enhanced version of RIPEMD design, performances better

when compared to the other three algorithms. The security features of RIPEMD 160 are also comparable with the market requirements.

4.3. ASYMMETRIC CRYPTOGRAPHY

While comparing the performance of RSA algorithm and ECC algorithm in terms of execution speed, the following results were obtained (Table 4.7 and Fig. 4.7).

Table 4.7 : PERFORMANCE COMPARISON OF ASYMMETRIC CRYPTOGRAPHY ALGORITHMS

Algorithm	Time (ms)
ECC	3.69
RSA	8.75

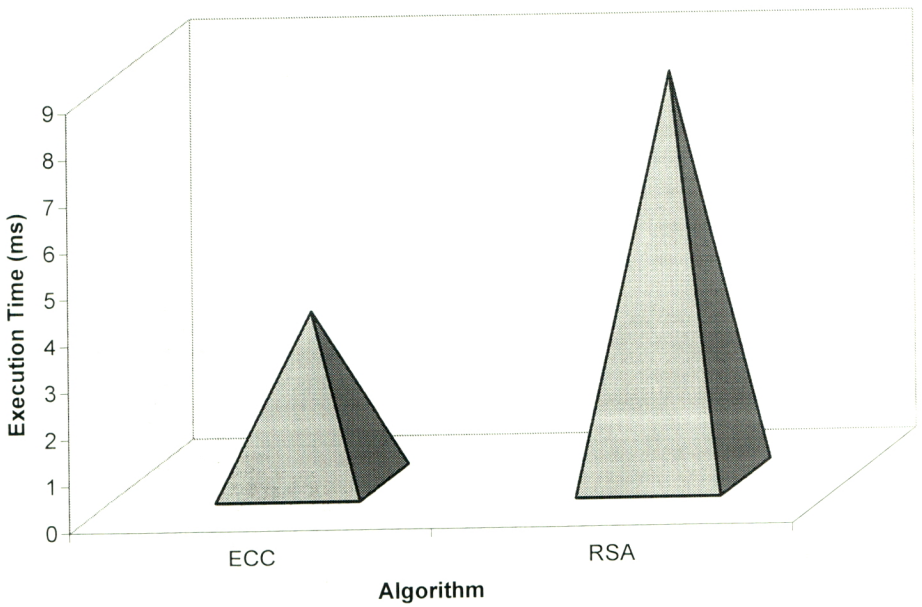


Fig. 4.7 : RSA AND ECC CRYPTOGRAPHY ALGORITHMS

Implementations are tested with Pentium processor and the data size was 256 Megabytes.From the table, it is evident that the performance of ECC outperforms the performance of RSA. Hence ECC is an excellent choice for doing asymmetric cryptography. The recommended RSA key size for most applications is 2,048 bits, whereas, for equivalent security using ECC, the key size needs to be only 224 bits. The difference becomes more and more pronounced as security levels increase. A sample of screen shots are given in Appendix .

4.4. CONCLUSION

Selecting and choosing a cryptographic technique for a particular environment often requires extensive research in the areas of technology, application, expenditure, etc. apart from speed. The following lists some other points can be taken into account while considering a cryptographic technique.

- Encryption is much more expensive than hashing
- Public key crypto is much more expensive than symmetric cryptosystems
- Symmetric techniques can work well if all parties are online simultaneously
- The choice is often difficult and frequently depends on estimates of likely scale and deployment patterns

In spite of all the security measures taken, hacking and information stealing is increasing steadfastly. The hackers are getting a lot better and therefore to secure information and message, techniques like cryptography has become an absolute necessity in today's modern world.

The next chapter summarizes the work and paves way for future work.

Summary and Conclusion

5. SUMMARY AND CONCLUSION

Cryptography is a particularly interesting field because of the amount of work that is done in secret. The irony is that today, secrecy is not the key to the goodness of a cryptographic algorithm. Regardless of the mathematical theory behind an algorithm, the best algorithms are those that are well-known and well-documented because they are also well-tested and well-studied! In fact, time is the only true test of good cryptography; any cryptographic scheme that stays in use year after year is most likely a good one. The strength of cryptography lies in the choice (and management) of the keys; longer keys will resist attack better than shorter keys.

The principal goal guiding the design of any cryptographic algorithm must be security. If an algorithm can be successfully cryptanalysed, then it is not worth using. In the real world, however, performance and efficiency are always of main concern. Efficiency means many different things, depending on context. Efficiency in bulk encryption means speed, where in hardware, it is key setup time. It is very difficult to compare cipher designs for efficiency and even more difficult to design ciphers that are efficient for all platforms.

In this thesis an attempt has been made to compare the performance efficiency of different cryptographic functions such as nine symmetric, block encryption algorithms namely: Blowfish, Twofish, Data Encryption Standard (DES), Triple-Data Encryption Standard, Serpent, GOST, Tiny Encryption Algorithm (TEA), International Data Encryption Algorithm(IDEA) and Skipjack, hash function cryptography, namely, Message Digest Version 2(MD2), Message Digest Version 5 (MD5), Race Integrity Primitives Evaluation Message Digest (RIPEMD160) and Secure Hash Algorithm (SHA), asymmetric cryptographic algorithms Rivest –Shamir-Adelman (RSA) and Elliptic curve cryptography (ECC) .

The following are the conclusions derived from this research work.

- Asymmetric encryption provides more functionality than symmetric encryption, at the expense of speed and hardware cost.
- Symmetric encryption provides a cost-effective and efficient method of securing data without compromising security, and should be considered as the correct and most appropriate security solution for applications like password protection, secure telnet connection etc.
- In some instances, the best possible solution may be the complementary use of both symmetric and asymmetric encryption.
- Cryptographic hash functions are suitable to use in various information security applications such as authentication, message integrity.

From the results it is evident that in the symmetric algorithms Blowfish is the fastest, followed by 3DES which is better than DES algorithm. Serpent algorithm takes the maximum time to encrypt and decrypt the file. GOST and Twofish algorithms eventhough works slower when compared to Blowfish, DES and TDES, still can be substituted for these three in extreme cases, since the time increase is negligible. In a similar fashion, with hash functions, RIPEMD is better than the MD2, MD5 and SHA algorithms. Finally in public key cryptography, the usage of ECC is recommended in place of RSA.

FUTURE WORK

There are various applications of cryptographic in the world of data security and secure transmission that are not yet probed. Some areas where cryptology is very much needed are listed below:

- Identification of new cryptography standards for wireless sensor networks targeting to the specific needs of wireless transmission should be developed.
- Formulation of New hybrid cryptography algorithms, which combines both public and private key primitives, that reduces the number of public-key operations to a minimum.
- New techniques that reduce the communication overhead during key establishment and/or authentication can be formulated.
- Cryptography can be applied to various areas like electronic voting, auctions and other forms of electronic trading, privacy preserving in data mining, etc and the performance of various algorithms with reference to these applications can be studied.

Bibliography

REFERENCES

Abdalla, M., Bellare, M. and Rogaway, P. (2000) DHAES: An encryption scheme based on the Di±e-Hellman problem. Submission to P1363a: Standard Specifications for Public-Key Cryptography, Additional Techniques, 2000.

Agnew, G. B., Mullin, R. C. and Vanstone, S. A. (1993) An Implementation of Elliptic Curve Cryptosystems over F2155, IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, 155-161..

Andem, V.R. (2003) A Cryptanalysis of the Tiny Encryption Algorithm, Masters thesis, The University of Alabama, Tuscaloosa.

Answers.com, The World's greatest encyclodictionalmanacapedia, 2006 (http://www.answers.com).

Antola, A., Bertoni, G., Breveglieri, L. and Maistri, P. (2003) Parallel Architectures for Elliptic Curve Crypto-Processors over Binary Extension Fields, IEEE Midwest symposium on circuit and system 03, MWSCS03, 23-28.

Aoki, K., Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J. and Tokita, T. (2000), Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms, NTT Corporation and Mitsubishi Electric 2000-2002, Workshop Record of SAC 2000, Seventh Annual Workshop on Selected Areas in Cryptography, 41-54, 14-15.

Barker, W.C. (2004), National Institute of Stanards and Technology (NIST), Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Information Security, 1-40.

Bednara, M., Daldrup, M., Shokrollahi, J., Teich, J. and von zur, J. (2002) Reconfigurable Implementation of Elliptic Curve Crypto Algorithms, The 9th Reconfigurable Architectures Workshop (RAW-02), 78-87.

Boston, N., Clancy, T., Liow, Y. and Webster, J. (2002) Genus Two Hyperelliptic Curve Coprocessor. In J. c. K. K. B. S. Kaliski and C. Paar, editors, Cryptographic Hardware and Embedded Systems, CHES 2002, Springer-Verlag, Vol LNCS 2523, 529–539.

Brown, M., Cheung, D., Hankerson, D., Hernandez, J. L., Kirkup, M. and Menezes, A. (2000) PGP in Constrained Wireless Devices. Proceedings of the 9th USENIX Security Symposium, 44-57.

Burnett, S., and Paine, S. (2001). RSA Security's official guide to cryptography, New York: Osborne/McGraw-Hill.

Chawla, K. (2005) A 3D RGB Axis-based Color-oriented Cryptography, Document signature, 07072005_1023:20914, http://www.citebase.org/fulltext?format=application%2Fpdf&identifier=oai%3AarXiv.org%3Acs%2F0508080.

Cramer, R. and Shoup, V. (2004) Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing, 33(1), 167-226.

Crowe, F., Kerins, T., Daly, A., Popovici, E. M., Marnane, W.P. (2005) Prototyping of Cryptographic Algorithms", Invited Chapter (Chapter 6) in Industry Days 2003-2004, edited by Aquilano D., Bezzi M., Capasso V., Micheletti A., Naldi G., Nieus T., Rizzo O., Sala M., Società Editrice Esculapio, ISBN 88-7488-109-6, 49-56.

Daemen,J. and Rijmen, V. (2000), AES Proposal:Rijndael, http://csrc.nist.gov/CryptoToolkit/aes/ rijndael/Rijndael.pdf .

De Win, E., Bosselaers, A., Vandenberghe, S., De Gersem, P. and Vandewalle, J. (1996) A fast software implementation for arithmetic operations in GF(2n). Advances in Cryptology – ASIACRYPT '96, LNCS 1163, Springer-Verlag, 65-76.

Dent, W.A. (2005) Hybrid Cryptography, Cryptology ePrint Archive: Report 2004/210

Diffie, W. and Hellman, M. (1976) New Directions in Cryptography", IEEE Transactions on Information Theory, vol. IT-22, 644-654.

Dobbertin, H. (1996) Cryptanalysis of MD5 compress. Announcement on Internet.

Dunn, J.D.G. and Rogerson, T.W. (2003) Eerdmans Commentary on the Bible, eds., Wm. B. Eerdmans Publishing, ISBN 0-802-83711-5.

Eric, C., Fossen, F., Northcutt, S., and Pomeranz, H. (2003) SANS Security Essentials with CISSP CBK, Version 2.1.USA: SANS Press.

Ernst, M., Klupsch, S., Hauck, O. and Huss, S. A. (2001) Rapid Prototyp-ing for Hardware Accelerated Elliptic Curve Public-Key Cryptosystems, 12th Workshop on Rapid System Prototyping, Monterey, CA, 56-67.

Gannon, J. (2001) Stealing Secrets, Telling Lies: How Spies and Codebreakers Helped Shape the Twentieth Century, Washington, D.C., Brassey's, 2001, ISBN 1-57488-367-4.

Garrett, P. (2001), Making, Breaking Codes, Prentice Hall, U.S.A, ISBN 0-13-030369-0.

Gaudry, P., Hess, F. and Smart, N. (2002) Constructive and Destructive Facets of Weil Descent on Elliptic Curves, Journal of Cryptology, 15, 19-46,.

Goldreich, O. (2001) Foundations of Cryptography, Volume 1: Basic Tools", Cambridge University Press, 2001, ISBN 0-521-79172-3.

Goodman, J. and Chandrakasan, A. P. (2001) An Energy-Efficient Reconfigurable Public-Key Cryptography Processor, IEEE Journal of Solid-State Circuits, Vol. 36, No. 11, 256-273.

Graff, J.C. (2001) Cryptography and E-Commerce: A Wiley Tech Brief, Frederick, M. (Ed.), Benchmark Productions Inc., John Wiley & Sons Inc., Canada.

Guajardo, J. and Paar, C. (1997) Efficient algorithms for elliptic curve cryptosystems. Advances in Cryptology – CRYPTO '97, LNCS 1294, Springer-Verlag, 342-356.

Gustafson, H., Dawson, E., Caelli, W. (1990) Comparison of block ciphers. In Proceedings of AUSCRYPT '90, J. Seberry and J. Piepryzk eds., 208--220.

Habutsu, T., Nishio, Y., Sasase, I. and Mori S. (1991) A secret key cryptosystem by iterating a chaotic map, EUROCRYPT' 91, 127-136.

Hankerson, D., Hernandez, J. L., Menezes, A. (2000) Software Implementation of Elliptic Curve Cryptography Over Binary Fields, Proceedings of Cryptographic Hardware Embedded Systerms (CHES) 2000 Workshop, Springer-Verlag, LNCS 1965, 1-24.

Hauser, J. and Wawrzynek, J. (1997) A MIPS Processor With A Reconfigurable Coprocessor. In Fifth Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM '97, Napa Valley, California, USA, 16–18.

Hayes, H.M. (2002), A Tutorial on Linear and Differential Cryptanalysis, Cryptologia, XXVI(3), 189-221.

Kahn, D., (1967) The Codebreakers, 1967, ISBN 0-684-83130--9.

Kapitaniak, T. (1996) Controlling Chaos, Theoretical and Practical Methods in Non-linear Dynamics., Academic Press, London 1996.

Kastrup, B., Bink, A. and Hoggerbrugge, J. (1999) ConCISe: A Compiler-Driven CPLD-Based Instruction Set Accelerator. In Seventh Annual IEEE Symposium on Field- Programmable Custom Computing Machines, FCCM '99, Napa Valley, California, USA, 92–101.

Kilian, J. and Rogaway, P. (1996), How to Protect DES Against Exhaustive Key Search, Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes In Computer Science, Springer-Verlag, London, 1109, 252 – 267.

Kim, H. and Lee, S. (2004), Design and implementation of a private and public key crypto processor and its application to a security system, IEEE Transactions on Consumer Electronics, 50(1), 214-224.

Koblitz, N. (1994) A course in Number Theory and Cryptography, Springer-Verlag, Berlin.

Kocarev, L.J., Halle, K.S., Eckert, K., Chua, L.O., Parlitz, U. (1992) Experimental demonstration of secure communications via chaos synchronization, Int.J.Bifurc. & Chaos 2, 709-716.

Kotulski, Z. and Szczepański, J. (1997) Discrete chaotic cryptography, Annalen der Physik 6, 381-394.

Koyama, K. and Tsuruoka, Y. (1993) Speeding up elliptic curve cryptosystems by using a signed binary window method, Advances in Cryptology – Crypto '92, LNCS 740, Springer-Verlag, 345-357.

Kuroki, J., Gonda, M., Matsuo, K., Chao, J. and Tsujii, S. (2002) Fast Genus Three Hyperelliptic CurveCryptosystems, The 2002 Symposium on Cryptography and Information Security, Japan, SCIS 2002, 631- 629.

L´opez, J. and Dahab, R. (1999b). Improved Algorithms for Elliptic Curve Arithmetic in GF(2n). Selected Areas in Cryptography - SAC '98, LNCS 1556, Springer-Verlag, 201-212.

L´opez, J. and Dahab, R. (1999a) Fast multiplication on Elliptic Curves over GF(2n) without Precomputation, Cryptographic Hardware and Embedded Systems-CHES '99, LNCS 1717, Springer-Verlag, 316-327.

L´opez, J. and Dahab, R. (2000) High-speed Software Multiplication in F2m. Indocrypt 2000, LNCS 1977, Springer-Verlag, 203-212.

Lai, X., Massey, J. L. and Murphy, S. (1991) Markov Ciphers and Differential Cryptanalysis, Advances in Cryptology - EUROCRYPT '91, Lecture Notes in Computer Science, No. 547, New York, Springer-Verlag, 17-38.

Landau, S. (2000) Standing the Test of Time: The Data Encryption Standard, Notices of the AMS, Sun Microsystems, 47(3), 341-349.

Lange, T. (2002a) Efficient Arithmetic on Genus 2 Hyperelliptic Curves over Finite Fields via Explicit Formulae, Cryptology ePrint Archive, Report 2002/121, 2002. http://eprint.iacr.org/.

Lange, T. (2002b). Inversion-Free Arithmetic on Genus 2 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2002/147, 2002. http:eprint.iacr.org.

Levine, J.R. Ed., (2000) Internet Secrets, 2nd Edition, Chapter 37, IDG Books, 2000, ISBN 0764532391.

Lim, C. and Lee, P. (1994) More flexible exponentiation with precomputation. Advances in Cryptology - Crypto '94, LNCS 839, Springer-Verlag, 95-107.

Lucks, S. (2002) A variant of the Cramer-Shoup cryptosystem for groups of unknown order. In Y. Zheng, editor, Advances in Cryptology, Asiacrypt 2002, Vol. 2501 of Lecture Notes in Computer Science, Springer-Verlag, 27-45.

Matsuo, K., Chao, J. and Tsujii, S. (2001) Fast Genus Two Hyperelliptic Curve Cryptosystems, In ISEC2001-31, IEICE, 231-244

Menezes, A.J., van Oorschot, P.C. and Vanstone, S.A. (1996), Handbook of Applied Cryptography, ISBN 0-849-38523-7.

Mirzan, F. (2000) Block Ciphers and Cryptanalysis, Department of Mathematics, Royal Holloway University of London

Mishra, P.K. and Sarkar, P. (2003) Parallelizing Explicit Formula for Arithmetic in the Jacobian of Hyperelliptic Curves. In Advances in Cryptology — Asiacrypt 2003, volume LNCS. Springer-Verlag, 24-36.

Miyamoto, Y., Doi, H., Matsuo, K., Chao, J. and Tsuji, S. (2002) A Fast Addition Algorithm of Genus Two Hyperelliptic Curve. In The 2002 Symposium on Cryptography and Information Security, SCIS 2002, IEICE Japan, 497 – 502.

Okamoto, T., Uchiyama, S. and Fujisaki, E. (1999) EPOC: Efficient probabilistic public-key encryption, Submission to P1363a: Standard Specifications for Public-Key Cryptography, Additional Techniques, 1-15.

Orlando, G. and Paar, (2000) A High-Performance Reconfigurable Elliptic Curve Processor for GF(2m), Proceedings of CHES 2000 Workshop, Springer-Verlag, LNCS 1965, 899-894.

Parlitz, U., Chua, L.O., Kocarev L.J., Halle, K.S. and Shang, A. (1992) Transmission of digital signals by chaotic synchronization, Int.J.Bifurc. & Chaos 2, 973-977.

Paul C. van Oorschot, Alfred J. Menezes, and Scott A. Vanstone (1996) Handbook of applied cryptography, CRC press Inc., Florida, 1996.

Pecora L.M. and Caroll T.L. (1990) Synchronization in chaotic systems, Phys.Rev.Lett, 64, 821-824.

Pelzl, J., Wollinger, T. and Paar, C. (2003b) Low Cost Security: Explicit Formulae for Genus-4 Hyperelliptic Curves. In Tenth Annual Workshop on Selected Areas in Cryptography — SAC 2003, Springer-Verlag, 132-154.

Pelzl, J., Wollinger, T., Guajardo, J. and Paar, C. (2003a) Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves, K. Koc, and C. Paar, editors, Workshop on Cryptographic Hardware and Embedded Systems, CHES 2003, Springer-Verlag, 67-83.

Reference.com (2006) University of Phoenix, Lexico Publications Group, LLC, http://www.reference.com.

Sakai, Y. and Sakurai, K. (2000) On the Practical Performance of Hyperelliptic Curve Cryptosystems in Software Implementation, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E83-A, NO.4, 692 – 703,.

Sakai, Y., Sakurai, K. and Ishizuka, H. (1998) Secure Hyperelliptic Cryptosystems and their Performance. In Public Key Cryptography, volume 1431 of Lecture Notes in Computer Science, 164 – 181, Berlin, Springer-Verlag, 987-999.

Schaefer, E. (1996) A Simplified Data Encryption Standard Algorithm, Cryptologia 96, 45, 23-28.

Schneier, B (2001), Why Cryptography Is Harder Than It Looks, Counterpane Internet Security, CounterPane Systems, 1-8 (http://www.schneier.com/essay-037.pdf - Last Accessed 23-08-2006).

Schneier, B. (1996) Applied Cryptography – Protocols, algorithms, and source code in C, 2nd Edition, John Wiley & Sons, Inc., New York.

Schneier, B. (1996), Applied Cryptography, Second Edition, John Wiley and Sons, ISBN 0-471-11709-9, 1- 784.

Schneier, B. (2000), A Self-Study Course in Block-Cipher Cryptanalysis, Counterpane Internet Security, Cryptologia, 24(1), 18-34.

Schneier, B. (2004) The Blowfish Encryption Algorithm., http://www.schneier.com/blowfish.html

Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C. and Ferguson, N. (1999) The Twofish Encryption Algorithm, John Wiley and Sons,

Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C. and Ferguson, N. (1999) The Twofish Encryption Algorithm, John Wiley and Sons,

Schroeppel, R., Orman, H., O'Malley, S. and Spatscheck, O.(1995) Fast Key Exchange with Elliptic Curve Systems. Advances in Cryptology - Crypto '95, LNCS 963, Springer-Verlag, 43-56.

Seltzer, L. J. (2002). Password Crackers, PC Magazine, 68-71.

Singh, H., Lee, M., Lu, G. Kurdahi, F. J., Bagherzadeh, N. and Filho, E. M. C. (2000) MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Applications. IEEE Transactions on Computers, 49(5), 465–481.

Singh, S. (2000), The Science of Secrecy, Fourth Estate Limited, London, ISBN : 0-358-49532-3.

Smart, N. (1999) On the performance of hyperelliptic cryptosystems. In J. Stern, editor, Advances in Cryptology – EUROCRYPT '99, LNCS 1592, Springer-Verlag 165–175.

Solinas, J.A. (2000) Efficient Arithmetic on Koblitz Curves. Designs, Codes and Cryptography, 19(2/3), 195-249.

Weimerskirch, A., Paar, C. and Shantz, S.C. (2001) Elliptic Curve Cryptography on a Palm OS Device, The 6th Australasian Conference on Information Security and Privacy (ACISP 2001), LNCS 2119, Springer-Verlag, 502-513.

Wheeler, D.J. and Needham, R>M> (1994) TEA, a tiny encryption algorithm, In Bart Preneel, editor, Fast Software Encryption: Second International Workshop, Vol. 1008 of Lecture Notes in Computer Science, 363-366.

Wikipedia, The Free Encyclopedia, Cryptography, 2006, http://en.wikipedia.org/wiki/Cryptography.

Wolkerstorfer, J. (2005), Hardware Aspects of Elliptic Curve Cryptography, Hardware Aspects of Elliptic Curve Cryptography, IAIK Tugarz, publications@iaik, Abstract available online, last accessed 08-09-2006, (http://www.iaik.tugraz.at/research/publications/theses/wolkerstorfer.htm).

Wolkerstorfer, J. (2005), Hardware Aspects of Elliptic Curve Cryptography, Hardware Aspects of Elliptic Curve Cryptography, IAIK Tugarz,

publications@iaik, Abstract available online, last accessed 08-09-2006, (http://www.iaik.tugraz.at/research/publications/theses/wolkerstorfer.htm).

Wollinger, T. (2001) Computer Architectures for Cryptosystems Based on Hyperelliptic Curves. Master's thesis, ECE Department, Worcester Polytechnic Institute, Worcester, Massachusetts, USA.

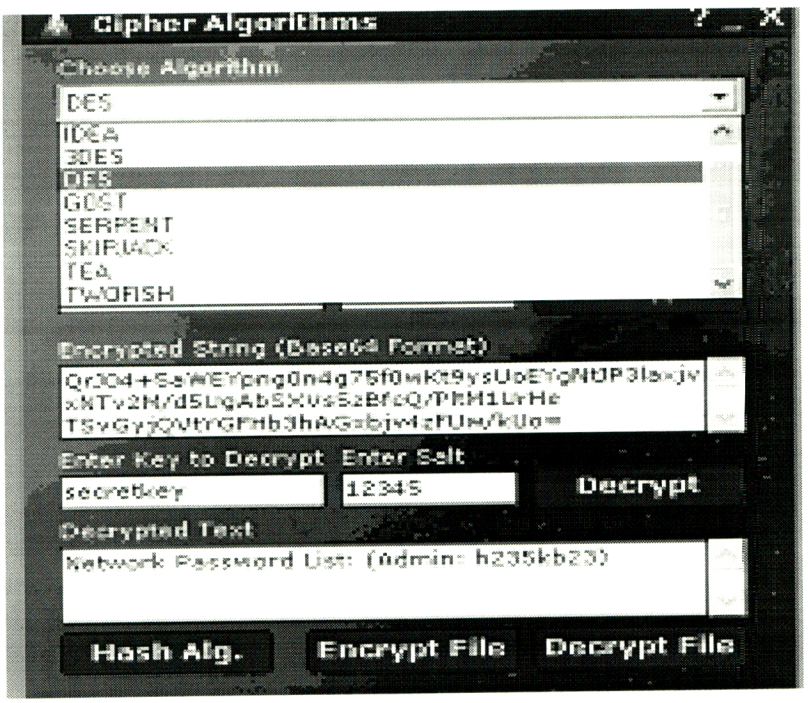
Wollinger, T. and Paar, C. (2002) Hardware Architectures proposed for Cryptosystems Based on Hyperelliptic Curves. In Proceedings of the 9th IEEE International Conference on Electronics, Circuits and Systems - ICECS 2002, Vol. III, 1159 – 1163..

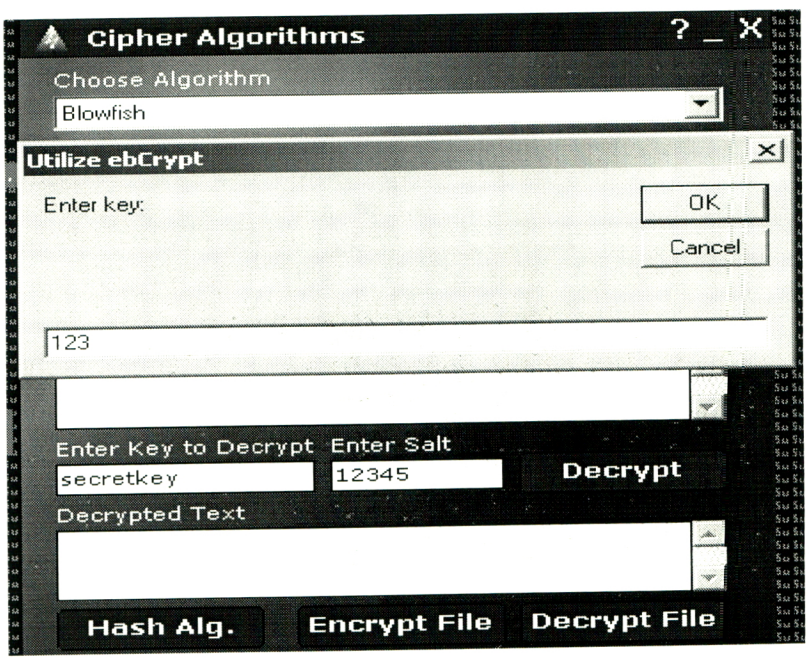
Wollinger, T., Pelzl, J., Wittelsberger, V., Paar, C., Saldamli, G. and Koc, C.K. (2003) Elliptic & hyperelliptic curves on embedded ^{1}p, ACM Transactions in Embedded Computing Systems (TECS), 2003, Special Issue on Embedded Systems and Security, 91-104.

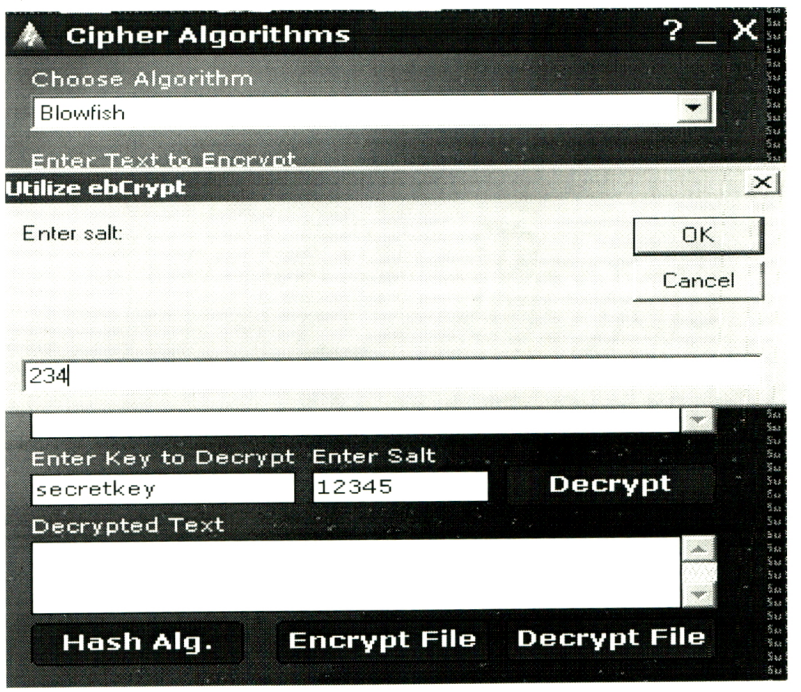
Yeun, C.Y. (2001), Design, Analysis and applications of cryptographic techniques, Ph.D. Thesis, Department of Mathematics, Royal Holloway and Bedford New College, University of London.

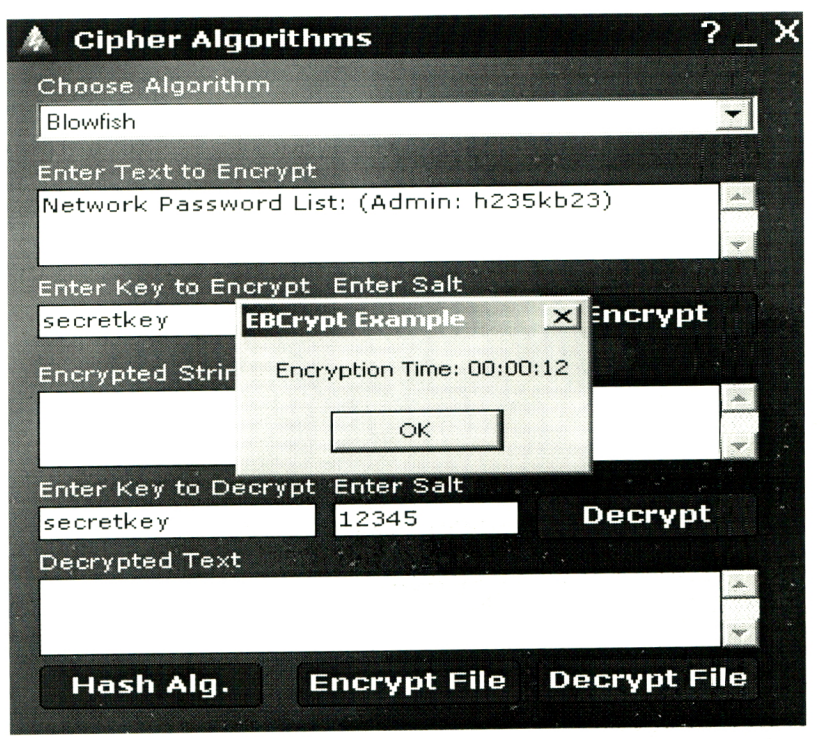
Appendix

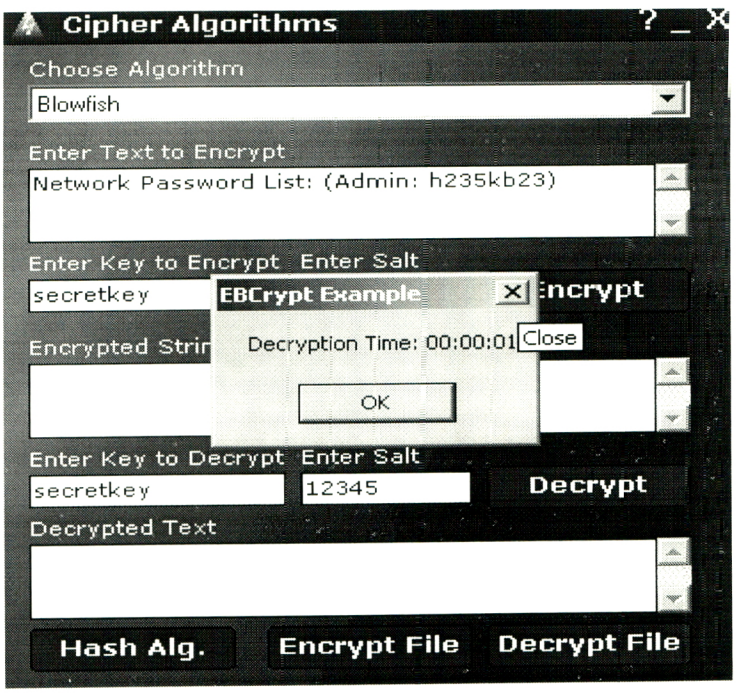
Symmetric Algorithms

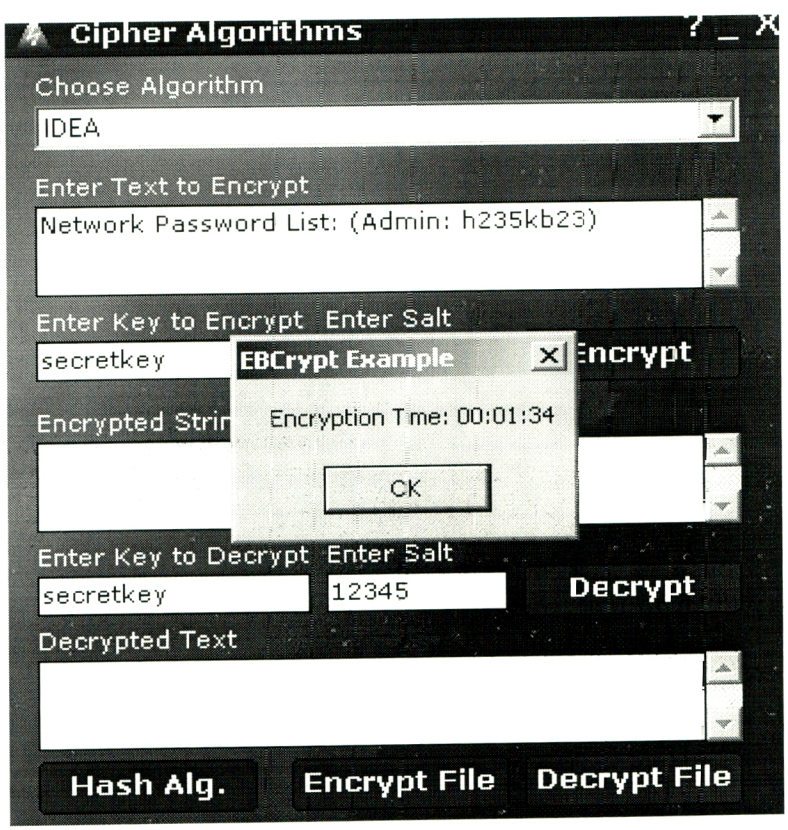


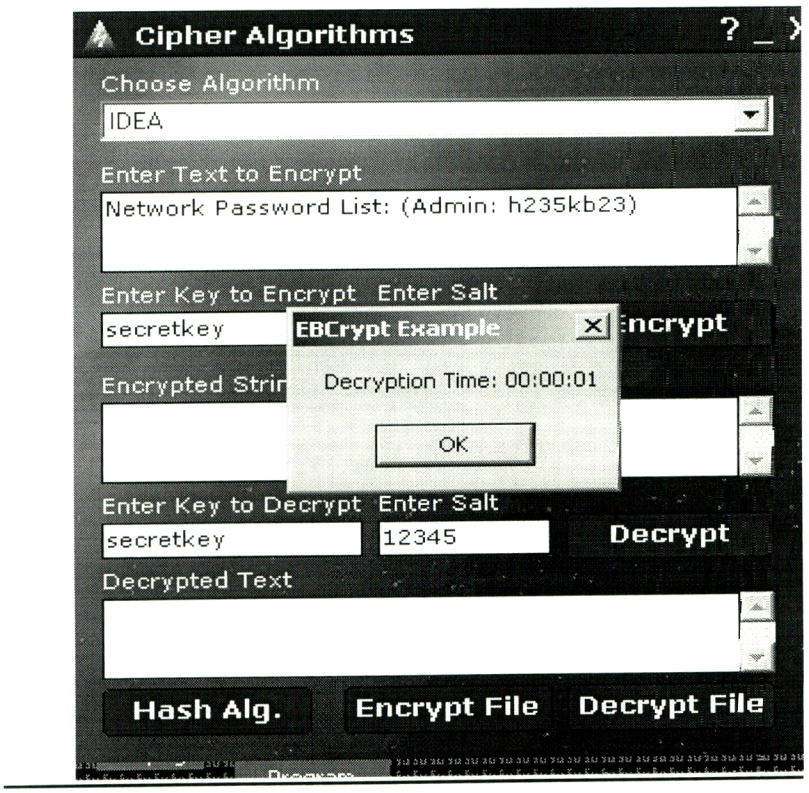



Cipher Algorithms
Choose Algorithm
Blowfish
Enter Text to Encrypt
Utilize ebCrypt
Enter salt:
OK
Cancel
234
Enter Key to Decrypt
Enter Salt
secretkey
12345
Decrypt
Decrypted Text
Hash Alg.
Encrypt File
Decrypt File

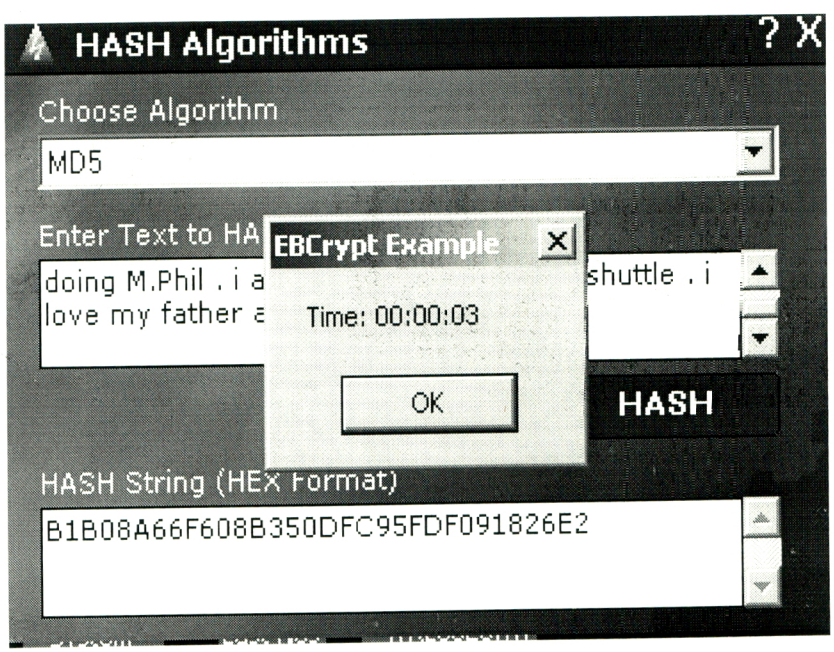

Cipher Algorithms
Choose Algorithm
Blowfish
Enter Text to Encrypt
Network Password List: (Admin: h235kb23)
Enter Key to Encrypt
Enter Salt
secretkey
EBCrypt Example
Encryption Time: 00:00:12
OK
Encrypted Strir
Enter Key to Decrypt
Enter Salt
secretkey
12345
Decrypt
Decrypted Text
Hash Alg.
Encrypt File
Decrypt File

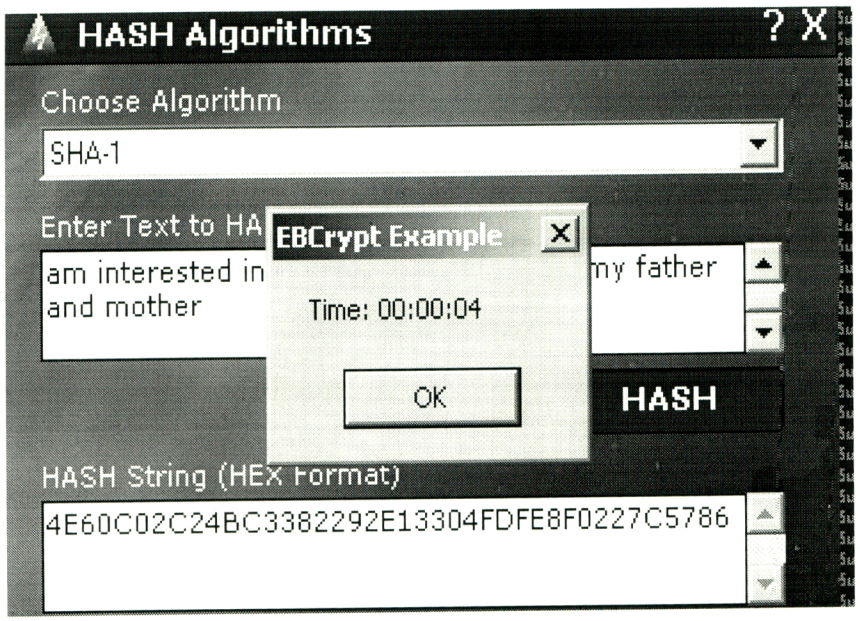

Cipher Algorithms
Choose Algorithm
Blowfish
Enter Text to Encrypt
Network Password List: (Admin: h235kb23)
Enter Key to Encrypt
Enter Salt
secretkey
EBCrypt Example
Decryption Time: 00:00:01
Close
OK
Encrypted Strir
Enter Key to Decrypt
Enter Salt
secretkey
12345
Decrypt
Decrypted Text
Hash Alg.
Encrypt File
Decrypt File

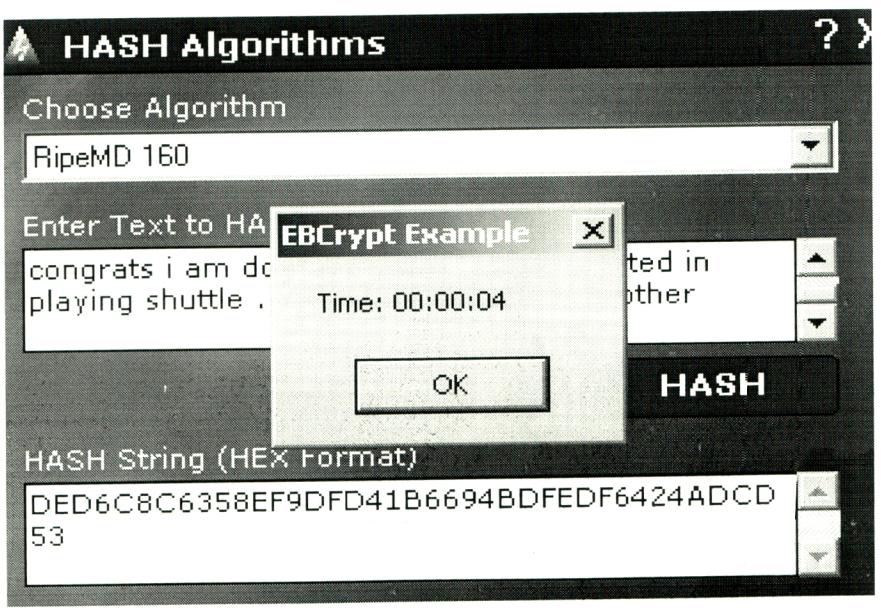

Cipher Algorithms
Choose Algorithm
IDEA
Enter Text to Encrypt
Network Password List: (Admin: h235kb23)
Enter Key to Encrypt
Enter Salt
secretkey
EBCrypt Example
Encryption Time: 00:01:34
OK
Encrypted Strir
Enter Key to Decrypt
Enter Salt
secretkey
12345
Decrypt
Decrypted Text
Hash Alg.
Encrypt File
Decrypt File

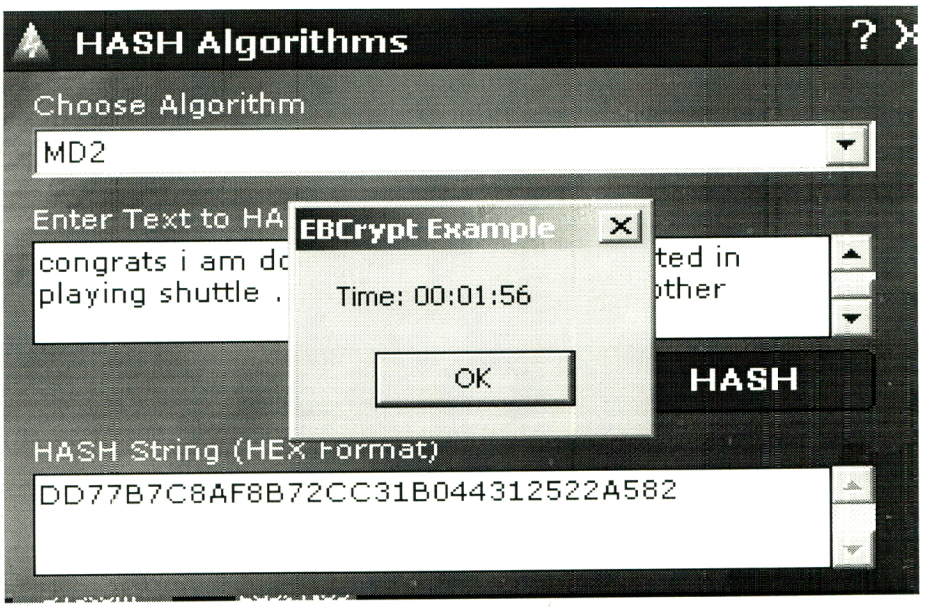


Hash Algorithms

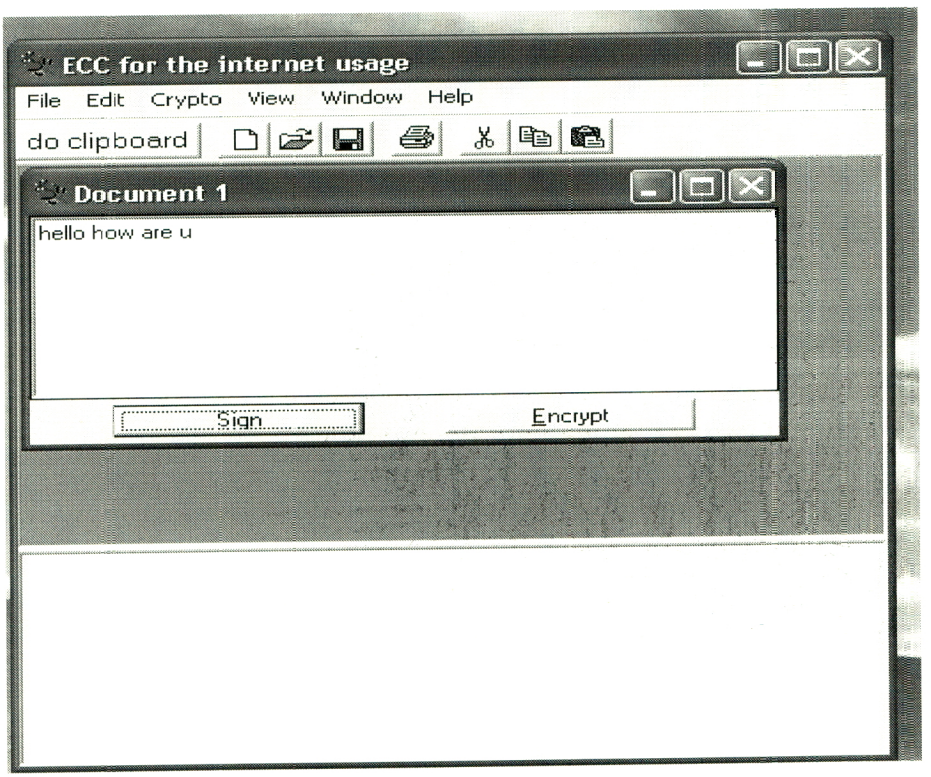


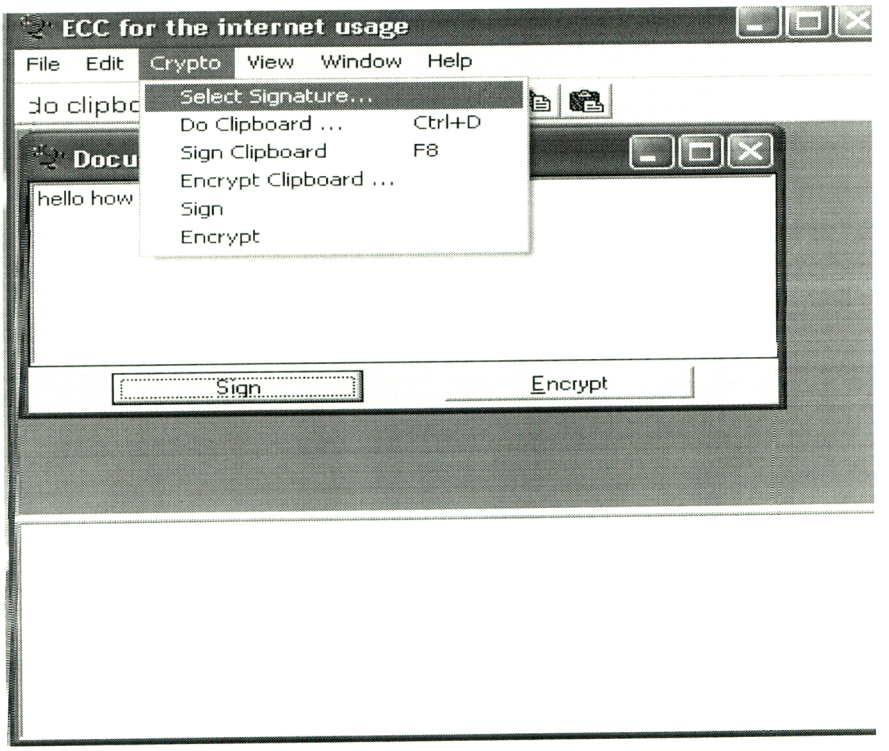

HASH Algorithms
Choose Algorithm
SHA-1
Enter Text to HA
am interested in
and mother
ny father
EBCrypt Example
Time: 00:00:04
OK
HASH
HASH String (HEX Format)
4E60C02C24BC3382292E13304FDFE8F0227C5786

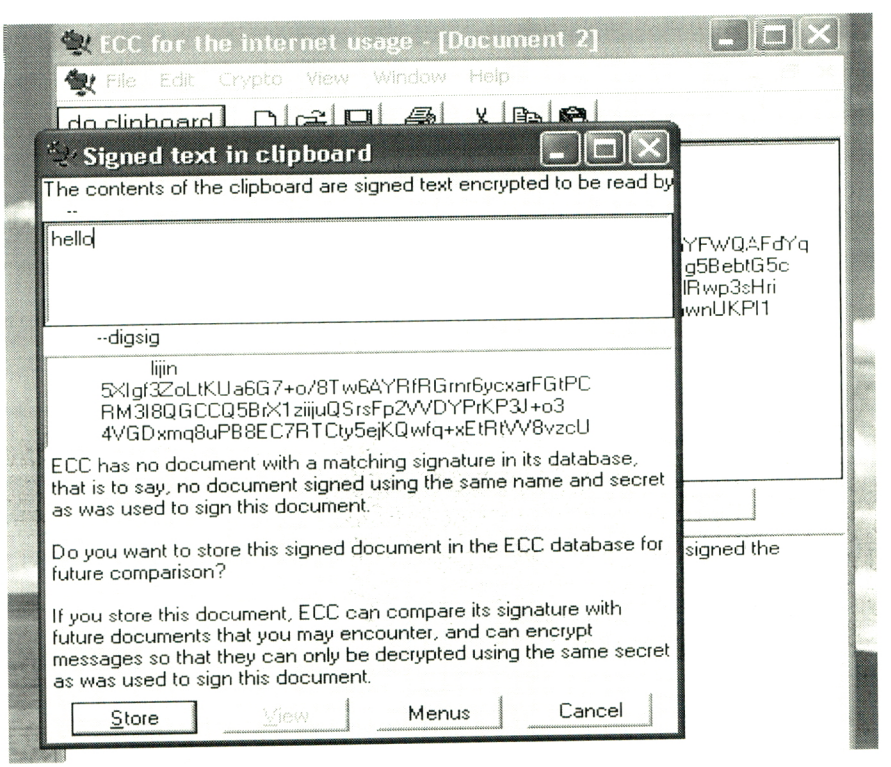

HASH Algorithms
Choose Algorithm
RipeMD 160
Enter Text to HA
congrats i am dc
playing shuttle .
ted in
other
EBCrypt Example
Time: 00:00:04
OK
HASH
HASH String (HEX Format)
DED6C8C6358EF9DFD41B6694BDFEDF6424ADCD
53

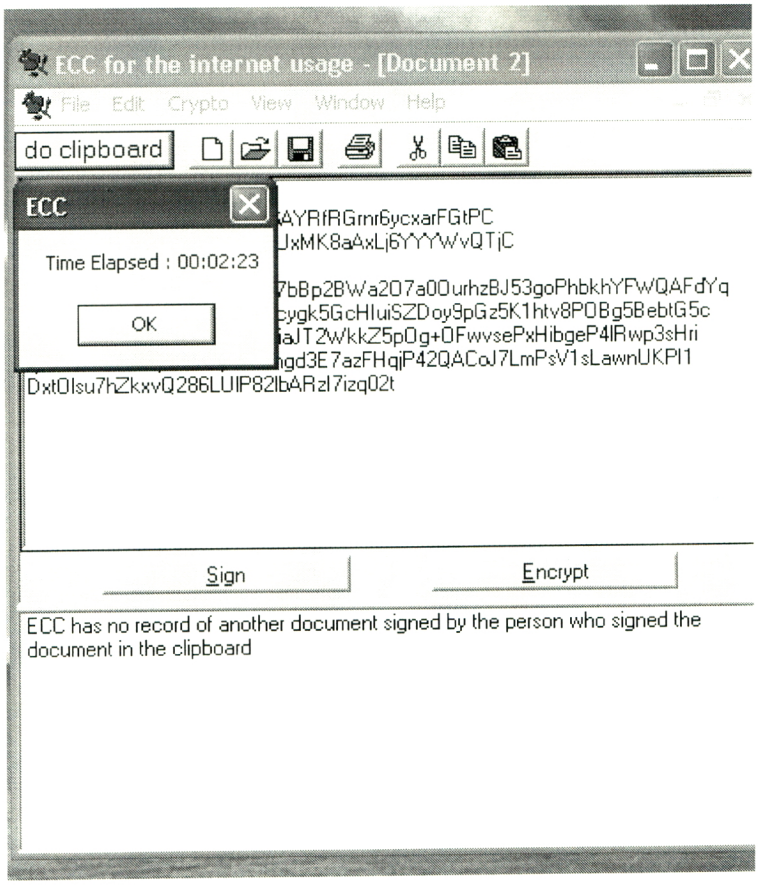


Asymmetric Algorithms




ECC for the internet usage
File Edit Crypto View Window Help
Select Signature...
Do Clipboard ... Ctrl+D
Sign Clipboard F8
Encrypt Clipboard ...
Sign
Encrypt
hello how
Sign
Encrypt


ECC for the internet usage - [Document 2]
File Edit Crypto View Window Help
Signed text in clipboard
The contents of the clipboard are signed text encrypted to be read by
hello
--digsig
lijin
5XIgf3ZoLtKUa6G7+o/8Tw6AYRfRGmr6ycxarFGtPC
RM3I8QGCCQ5BrX1ziijuQSrsFp2VVDYPrKP3J+o3
4VGDxmq8uPB8EC7RTCty5ejKQwfq+xEtRtVV8vzcU
ECC has no document with a matching signature in its database, that is to say, no document signed using the same name and secret as was used to sign this document.
Do you want to store this signed document in the ECC database for future comparison?
If you store this document, ECC can compare its signature with future documents that you may encounter, and can encrypt messages so that they can only be decrypted using the same secret as was used to sign this document.
Store
View
Menus
Cancel
signed the


ECC for the internet usage - [Document 2]
File Edit Crypto View Window Help
do clipboard
ECC
Time Elapsed : 00:02:23
OK
AYRfRGmr6ycxarFGtPC
UxMK8aAxLj6YYYWvQTjC
7bBp2BWa2O7a0OurhzBJ53goPhbkhYFWQAFdYq
cygk5GcHIuiSZDoy9pGz5K1htv8POBg5BebtG5c
aJT2WkkZ5pOg+OFwvsePxHibgeP4lRwp3sHri
gd3E7azFHqjP42QACoJ7LmPsV1sLawnUKPI1
DxtOIsu7hZkxvQ286LUIP82lbARzI7izq02t
Sign
Encrypt
ECC has no record of another document signed by the person who signed the document in the clipboard