Integrating Security Services Into Computer Supported Cooperative Work

MOHD AIZAINI BIN MAAROF

Doctor of Philosophy

ASTON UNIVERSITY

February 2000

ASTON UNIVERSITY

Integrating Security Services Into Computer Supported Cooperative Work

MOHD AIZAINI BIN MAAROF

Doctor of Philosophy

2000

THESIS SUMMARY

This research describes the development of a groupware system which adds security services to a Computer Supported Cooperative Work system operating over the Internet. The security services use cryptographic techniques to provide a *secure access control service* and an *information protection service*. These security services are implemented as a protection layer for the groupware system. These layers are called *External Security Layer* (ESL) and *Internal Security Layer* (ISL) respectively. The security services are sufficiently flexible to allow the groupware system to operate in both synchronous and asynchronous modes.

The groupware system developed - known as Secure Software Inspection Groupware (SecureSIG) - provides security for a distributed group performing software inspection. SecureSIG extends previous work on developing flexible software inspection groupware (FlexSIG) (Sahibuddin, 1999). The SecureSIG model extends the FlexSIG model, and the prototype system was added to the FlexSIG prototype. The prototype was built by integrating existing software, communication and cryptography tools and technology. Java Cryptography Extension (JCE) and Internet technology were used to build the prototype. To test the suitability and transparency of the system, an evaluation was conducted. A questionnaire was used to assess user acceptability.

Keywords : Computer Supported Cooperative Work, Security Services, Cryptography, Java, Internet.

To my Parent,

Hajjah Mariam Abdul Rahman &

My late father... Haji Maarof Haji Hussein

To my Wife,

Siti Salwa Abdul Samad

To my Children,

Raihan,

Abdullah Zahid,

Abdullah Munir,

Abdullah Zubayr,

Syifa' &

Abdullah Mus'ab

ACKNOWLEDGEMENTS

Alhamdulil-laahirab-bil 'alamiin.

I would like to thank my supervisor Mr Bernard S. Doherty for his invaluable advice and assistance throughout the course of this research.

My thanks also go to the folks in Room 268, Shamsul, Vasilas, and Zul, and to the support staff, Dr. Tony, Neil, and Nic, for their support.

I would like to acknowledge the University Technology of Malaysia for providing the financial support for this research.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND SYMBOLS

ANSI	American National Standard Institute
C	Ciphertext
CA	Certificate Authority
CBC	Cipher Block Chaining mode of operation
CFB	Cipher Feedback mode of operation
CSCW	Computer Supported Cooperative Work
DES	Data Encryption Standard
$\mathbf{D_K(X)}$	Decryption transformation of X with key K
ECB	Electronic Codebook mode of operation
$\mathbf{E_K(X)}$	Encryption transformation of X with key K
IAIK	Institute for Applied Information Processing and Communications
IBM	International Business Machines Corporation
IDEA	International Data Encryption Algorithm

IP	Internet Protocol
ISO	International Organisation for Standardisation
ITU-T	International Telecommunication Union – Telecommunication Standardisation Sector
IV	Initialisation Vector
JCE	Java Cryptography Extension
MAC	Message Authentication Code
MD	Message Digest
NBS	National Bureau of Standard
OFB	Output Freedback mode of operation
PBE	Passphrase-Based Encryption
PKCS	Public Key Cryptography Standards
RSA	Rivest Shamir Adelman public key cryptosystem
TCP	Transport Control Protocol
$s_1 \| \| s_2$	Concatenation of string s_1 and s_2
XOR, $\oplus$	Exclusive-OR boolean operation
$\| \| \mathrm{M} \| \|$	Length in octets of M

Chapter 1

INTRODUCTION

1.0 INTRODUCTION

This thesis present the results of research carried out by the author at Aston University, Birmingham. It describes the development and evaluation of a model and prototype system that adds security services to a Computer Supported Cooperative Work (CSCW) system operating over the Internet. The lack of security in CSCW system in general and software inspection groupware specifically has been raised by Teufel *et al.* (1995) and Sahibuddin (1999), respectively. This research adresses security deficiencies in CSCW systems. The security deficiencies in CSCW systems can be handled by incorporating security services, based on the following security requirements:

- Preventing unauthorised users accessing or participating in the CSCW system.
- Ensuring against disclosure of information flow in the CSCW system.
- Ensuring against disclosure of information stored in the CSCW system.

The security services use cryptographic techniques to provide a *secure access control service* and an *information protection service*. The CSCW system is able to operate in both synchronous and asynchronous modes, and the security services are sufficiently flexible to handle both synchronous and asynchronous distributed environments.

The secure access control service has been implemented as a protection layer for the CSCW system. It provides secure access control by encrypting all access information to make sure that the system is accessible only to authorised members of the group. This layer is called the *External Security Layer* (ESL).

The information protection service provides for secure transaction of information within the CSCW system. This information protection service will also provide protection to all the information stored in the system. This layer is called the *Internal Security Layer* (ISL).

The system developed has been given the name Secure Software Inspection Groupware (SecureSIG) system. SecureSIG is an extension of previous work on developing flexible software inspection groupware (FlexSIG) by Sahibuddin (Sahibuddin, 1999). The security model extends the FlexSIG model, and the prototype security system was added to the FlexSIG prototype.

The prototype was built by integrating existing software, communication and cryptography tools and technology. Both the model and the prototype are evaluated and the results analysed.

This research has demonstrated that it is possible to create a security system for groupware that is suitable, acceptable and transparent to the user, based on the SecureSIG model developed in this research. The prototype based on this model has been developed and evaluated by users.

As a whole this research has extended current technology of groupware, particularly software inspection groupware, by providing a secure environment to the software inspection process. The main contributions of this research are as follows:

- A secure software inspection groupware system that is not limited to any one of the four categories of the time and space taxonomy.
- A model of a secure software inspection groupware.
- A secure software inspection groupware system that is provided with a secure access control mechanism to protect the system from unauthorised user gain access to the system and resources.
- A secure software inspection groupware that is provided with an encryption mechanism to provide a safeguard to the information stored.
- A secure software inspection groupware that is provided with an encryption mechanism to provide protection to the information flow to and from the system.
- A secure software inspection groupware that is provided with two layers of protection - the internal and external protection.
- A secure software inspection groupware that is provided with a security shell architecture for protection of CSCW environments.

The results and the findings obtained have indicated that the objectives outlined have all been met.

1.1 ORGANISATION OF THE THESIS

The thesis is organised as follows:

Chapter 1: This chapter gives an overall introduction to the research. The aims and objectives of the research are also drawn out.

Chapter 2: This chapter presents the area Computer Supported Cooperative Work. The issue of software inspection, and in particular the FlexSIG system is discussed.

Chapter 3: This chapter presents the enabling technology used in this research. The topics of security services, security mechanisms, and cryptographic key management are also presented in this chapter.

Chapter 4: This chapter focuses on the identification of the research problem, the purpose and the importance of this research. Detailed research design and procedure are presented.

Chapter 5: This chapter discusses the proposed SecureSIG model.

Chapter 6: This chapter mainly describes the development and implementation of the prototype of the model. The enabling technology used in the development of the prototype is presented.

Chapter 7: This chapter discusses the evaluation of the prototype based on user evaluation.

Chapter 8: This chapter discusses the evaluation of the research as a whole. The areas that have been opened up for future extensions are also presented.

Chapter 9: This chapter gives a conclusion to the work in this research.

1.2 STRUCTURE OF THIS CHAPTER

The rest of the chapter will briefly introduce the overall background and areas related to the research. The chapter is structured as follows:

- **Section 1.3** gives the background and motivations of the research and the need for a secure environment.
- **Section 1.4** is an introduction to Computer Supported Cooperative Work (CSCW).
- **Section 1.5** is an introduction to the area of Cryptography, which is an important technology used of this research. These include the introduction to secret key cryptography, public key cryptography, hybrid systems, and cryptographic key management.
- **Section 1.6** gives an introduction to the technologies used in implementing the prototypes for this research: these include the Internet and Java technology (Java).
- **Section 1.7** introduces the aims and objectives of this research.

1.3 BACKGROUND AND MOTIVATIONS OF THE RESEARCH

Modern Civilisations are becoming increasingly dependent on computers in daily life. As computers have become smaller, cheaper, and more numerous, people have become more interested in connecting them together to form networks and distributed systems. The emergence of the personal computer as a major presence in the 1970's and 1980's and the introduction of local- and wide area networks into the personal computer environment, resulted in a trend to network machines together (Engelbert & Lehtman, 1988). Networking allows systems to

be designed to serve geographically dispersed groups or users or organisations. Recent technological innovations in portable computing, user interfaces, and computer networking make it feasible to explore and develop new facilities that will help people work together more efficiently and conveniently. This has opened possibilities for individuals working together as a group over networks, and has created the need for specialised software to support group activities. To support group activities, computer networks should make all programs, data, and other resources available to everyone on the networks without regard to the physical location of the resource and the user (Tanenbaum, 1981).

Modern Civilisation is entering the new phase, shifting from the paradigm of an industrial society to the paradigm of an information society. Due to the tremendous impact of computers and computer networks on society during the past decade, this period in history has come to called the "information age" (Black, 1993). In this new phase, Eltoweissy (1993) said that:

> *"The axiom that "information is power" and therefore should be doled out with extreme caution is replaced with the new axiom "information sharing is power" and everyone should therefore have access to the information they need to perform their jobs. This emanates from simple reality that, to be competitive in today's global economy, it will take the co-operative efforts of people with different skills to create innovative solutions and innovative products."*

Much of today's work is done not individually, but rather in a group (Olson *et al.*, 1993). Groups are here defined as sets of people who are knowingly collaborating on a common goal, and who thus require communication and co-ordination among group members. Much of today's work is collaborative in nature, due to the complexity of the task, the severe time constraints, or the requirement for broad expertise (Olson *et al.*, 1993). Today the success of most

projects relies on the co-operative activities of people and this requires that people communicate, jointly co-ordinate their activities, and share information and ideas more than ever (Eltoweissy, 1993).

The innovations of recent technology in computing, user interfaces and computer networking make it feasible to explore and develop new computer facilities that will help people work together more efficiently and conveniently (Eltoweissy, 1993). The field that deals with the development of such facilities and research is generally termed *Computer Supported Cooperative Work* (CSCW) (Greif, 1988). The purpose of CSCW is to provide computer support that facilitates co-operation between users (Foley & Jacob, 1995).

A great deal of work has been done on the technological aspects of CSCW: the problems of how one actually provides computer support for co-operation (Foley & Jacob, 1995). But the aspects of information security of CSCW technology have not received much attention (Teufel *et al.,* 1995), especially considering that shared information spaces and people working together in groups are the basis of most CSCW applications. Furthermore, computer technology, and especially in CSCW, enables for example the transmission of critical information not only within the boundaries on an organisation, but also around the world. This critical information is an asset, and it is vital that the confidentiality of such information can be guaranteed in the CSCW application. Teufel *et al.* (1995) stated that CSCW applications will be fully accepted only if the security mechanisms implemented in CSCW applications support the full complexity of interactions.

1.3.1 The Security of Information

Computers have brought about tremendous change in the way society lives and

operates, yet they have also brought additional threats and dangers. We rely so much on the information processed, stored and transmitted by computers. Information is the most valuable asset in many organisations and sharing, accessing and protecting it will become a priority in the future. The development of inexpensive computers and computer networks has brought with it the problem of unauthorised access and tampering with information. Increased connectivity not only provides access to a larger variety of information resources more quickly than ever before, it also provides an access path to the information from virtually anywhere on the network (Power, 1995).

Information that flows on the network is vulnerable to unauthorised access and tampering because it is transmitted through a communication link. Without much difficulty, someone could tap into the communications and monitor information being transmitted. Even more worrisome are scenarios that involve criminals who actively try to detect weaknesses in information systems and use them to their advantage.

The explosion in usage of the Internet has opened world-wide opprtunities. But along with these opportunities come a number of security concerns (IBM, 1997):

- Preventing eavesdropping on private or sensitive communications,
- ensuring that only an authorised user can access the system, and
- controlling access to confidential information.

For this reason there is a need for *security services* to provide information security. In general, the security services fall into five groups (Davies & Price, 1989; ISO, 1989; Menezes, van Oorschot & Vanstone, 1997): *authentication*, *confidentiality*, *integrity*, *non-repudiation* and *access control*. The International

Organisation for Standardisation (ISO) defines these basic security services (ISO, 1989) as follow:

- *Authentication*. To ensures that a principal (user, process, host) is really what it claims to be.
- *Data Confidentiality*. To ensures that only authorised principals can understand the protected information.
- *Data Integrity*. To ensure that no modifications of data has been performed by unauthorised principals.
- *Non-repudiation*. To ensures that a principal cannot deny performing some action on the data.
- *Access Control*. To ensures that only authorised principal can gain access to protected resources.

The security mechanisms that are used to provide the security services listed above are: *Encryption mechanisms, digital signature mechanisms, access control mechanism, data integrity mechanisms*, and *authentication mechanisms* (ISO, 1989) (Hassler, 1997). The security services and security mechanism are dealt in more detailed in Chapter Three. These security mechanisms can be achieved using Cryptography[1]. Cryptography is one of the main tools for information security. The original purpose of cryptography is to conceal information either while it is being transmitted or stored (Davies & Price, 1989). According to Davies and Price (1989) wherever a high level of security is needed a technique based on cryptography can be found. Cryptography will be introduced in Section 1.5 as one of the tools used in this research, and will be dealt with in more details in Chapter Three. Section 1.4 will briefly introduce

[1] Cryptography – means "hidden writing" from Greek *kryptos* meaning "hidden", and *graphia*, meaning "writing".

the area of Computer Supported Cooperative Work, where the security will be built on an application.

1.4 COMPUTER SUPPORTED COOPERATIVE WORK

Computer Supported Cooperative Work (or CSCW) has recently been established as the field that focuses on the role of computers to support co-operative work. *Computer Supported Cooperative Work* is a computing term coined by Irene Greif of Massachusetts Institute of Technology and Paul Cashman of Digital Equipment Corporation in 1984 (Bannon & Schmidt 1991; Grudin, 1993; Wilson 1990, 1991). The term was a shorthand way of referring to a set of concerns about supporting multiple individuals working together with computer system with no intention to any special emphasis to the meaning of the individual words within it (Bannon & Schmidt, 1991). Researchers and developers in this field make use of advances in enabling technologies, mainly portable computing, user-interfaces, and computer networking to connect disparate information systems, link products with one another, and promote inter-person communication.

1.4.1 Definition and Classification of CSCW

Wilson (1990, 1991) defines CSCW as a generic term that combines the understanding of the way people work in groups with the enabling technologies of computer networking and associated hardware, software, services and techniques. Wilson's definition embraced most of the other terms used to describe this field including Groupware and Workgroup Computing.

Wilson (1990, 1991) divides the field of CSCW into two distinct but interrelated fields: the *group working process* and the *enabling technology*

employ to support it. The group working process is further subdivided into four areas: *individual aspects*, *organisation aspects*, *group working design aspects* and *group dynamics aspects*. The enabling technology areas are subdivided into four areas: *communication systems*, *shared work space systems*, *shared information systems* and *group activity support systems*. For details Wilson's classification of CSCW see Appendix A.

1.5 CRYPTOGRAPHY

Cryptography is the science of mapping readable text into unreadable format and vice versa. The mapping process is a sequence of mathematical computations, which affect the appearance of the data without changing its meaning. For more details on the topic of cryptography see Appendix B.

Cryptography is used to provide the security services - *Confidentiality*, *Integrity*, *Authentication* and *Non-repudiation* defined in Section 1.3.1.

There are two basic types of cryptography: *secret key systems* (also called symmetric systems) and *public key systems* (also called asymmetric systems). Often, these two types of cryptography are combined to form a *hybrid system* to exploit the strength of each type. Secret key cryptography, public key cryptography, and hybrid cryptographic system are briefly described in section 1.5.1, 1.5.2, and 1.5.3 respectively.

1.5.1 Secret Key Cryptography

Secret key cryptography is characterised by the use of a single key to perform both the encrypting and decrypting of data. Since the same cryptographic key is

used for both encryption and decryption, it needs to be kept secret to protect the plaintext data from recovery by unauthorised parties.

Secret key cryptography can be divided into two categories (Schneier, 1996): *Stream ciphers* and *block ciphers*. *Stream ciphers* operate on a stream of bits or bytes. Stream ciphers are used either where a communications system requires a continuous (synchronous) data link to be maintained or where there is insufficient memory capacity to allow a block of data to be stored. *Block ciphers* encrypt and decrypt fixed size block of data, usually 64 bits long. The size of the blocks is dependent upon the particular algorithm, and each block is processed independently of other blocks.

The best-known secret key cryptography block cipher is the Data Encryption Standard (called DES), developed at International Business Machine (IBM) and adopted by the National Bureau of Standards (NBS) in the mid-1970s. It is the most widely accepted, publicly available cryptographic system today. The American National Standards Institute (ANSI) has adopted DES as the basis for encryption, integrity, access control, and key management standards (NIST, 1994). Another block cipher that seems to be more secure is the International Data Encryption Algorithm (called IDEA) (Schneier, 1996), was developed by Xuejia Lai and James Massey in 1990 (Lai & Massey, 1990). The DES and IDEA block cipher are used in this project and are discussed in Chapter Three.

1.5.2 Public Key Cryptography

Public key cryptography was introduced by Diffie and Hellman in 1976 (Diffie & Hellman, 1976). Public key cryptography differs from symmetric key cryptography in that key material is bound to a single user. The key material is

divided into two components:

- a *private key*, to which only the user has access and is never transmitted , and
- a *public key*, which may be published widely or distributed on request.

Each key generates a function used to transform text. The private key generates a private transformation function, and the public key generates a public transformation function. The functions are inversely related, i.e., if one function is used to encrypt a message, the other is used to decrypt the message. The order in which the transformations are invoked is irrelevant. These keys are always generated in matching pairs.

There are several public-key cryptographic systems. One of the first public key systems is RSA that was published in 1978 (Rivest, Shamir & Adleman, 1978). It is named after its inventors: Rivest, Shamir and Adleman. RSA is the most popular public key system in use today (Menezes, van Oorschot, Vanstone, 1997). The RSA public key cryptography is used in this project and will be discussed in Chapter Three.

1.5.3 Hybrid Systems

Public and secret key cryptography has relative advantages and disadvantages. Public key cryptography is much slower than secret key cryptography because the mathematical computations used to encrypt data in public key system require more time (Markovitz, 1994). To maximise the advantages of both secret and public key cryptography, hybrid system combines both types and used it in a complementary manner, with each performing different functions. In a hybrid system, public key cryptography is used to distribute cryptographic keys that are used to encrypt and decrypt data using secret key cryptography.

1.5.4 Cryptographic Key Management

Key management is the set of processes and mechanisms which support key establishment and the maintenance of ongoing keying relationships between parties, including replacing older keys with new keys as necessary (Menezes, van Oorschot, & Vanstone, 1997). Key management is the fundamental security requirement in all cryptosystems. The most difficult requirement is that a key must be chosen and made available at both ends of a communication path (Davies & Price, 1984). The proper management of cryptographic key is essential to the effective use of cryptography for security (NIST, 1994), because the security of information protected by cryptography directly depends upon the protected keys. The secret keys and private keys need to be protected against disclosure. In real world, key management is the hardest part of cryptography (Schneier, 1996). Key management issues will be discussed in detail in Chapter Three.

1.6 IMPLEMENTATION ISSUES

To implement and develop the prototype for this research, three major technologies were used: Cryptography, Java Technology and the Internet. The topic of cryptography has been discussed in Section 1.4. The other two technologies: Java Technology and the Internet are discussed below.

1.6.1 Java Technology

Java originated as part of a research project to developed advanced software for a wide variety of network devices and embedded systems (Sun, 1995). Java was developed in 1995. In 1990 Sun started to define Java based on C++, but without

its unclean and unsafe features (Ciancarini *et al.*, 1996). New principles and structure were inherited from a variety of languages such as Eiffel, SmallTalk, Objective C, and Cedar/Mesa. The result is a language environment that has proven ideal for developing secure, distributed, network-based end-user applications in environments ranging from networked-embedded devices to the World Wide Web and the desktop (Sun, 1995). The Java Security Application Program Interface (API) is a Java core API, built around the *java.security* package (and its sub-packages). This API is designed to allow developers to incorporate both low-level and high-level securities functionality into their Java applications. Java Cryptography Extension (JCE) is a set of APIs and implementations of cryptographic functionality, including symmetric, asymmetric, stream, and block encryption. The architecture of the JCE follows the same design principles found elsewhere in the Java Cryptography Architecture (JCA).

1.6.2 Internet

In 1973, the United State Defence Advanced Research Projects Agency (DARPA) initiated a research programs to investigate techniques and technologies for interlinking data packets of various kinds. The objective was to develop communication protocols that would allow networked computers to communicate transparently across multiple, linked packet networks. This was called the Internetting project and the system of the networks that emerged from the research was known as the "Internet" (Leiner *et al.,* 1998). The system of the protocols that was developed became known as the TCP/IP Protocol Suite, after the two initial protocols developed: Transmission Control Protocol (TCP) and Internet Protocol (IP).

The Internet is the global Internet Protocol based (IP-based) network over which the World Wide Web and other popular information systems operate (Hughes *et al.*, 1997).

1.7 THE AIM AND OBJECTIVES OF THE RESEARCH

Information is the most valuable asset in many organisations, and sharing, accessing and protecting it is a growing priority (Idris, 1995). The author of this thesis is interested in protecting the information in a CSCW system by providing two security layers, termed the External Security Layer (ESL) and Internal Security Layer (ISL), as described in Section 1.0. When the ELS and ILS are combined they provide an effective means of providing a secure CSCW environment.

The aim was to model, develop and evaluate a prototype implementation which was added to a distributed software inspection groupware system FlexSIG (Sahibuddin, 1999), which will be used as a test-bed. The secure software inspection process model is based on an extension of FlexSIG model. The prototype utilised existing tools and technology to achieve security outlined in the model developed. The system was evaluated and the results analysed. Specifically the aims of this research were:

- to develop a secure software inspection model.
- to develop a prototype based on the model.
- to evaluate the prototype to measure the suitability and transparency of the system.

The two securities layer, the ELS and ILS, were implemented on top of FlexSIG to demonstrate the security ideas developed in the research reported in

this thesis. The security model was sufficiently flexible to support the asynchronous and synchronous mode provided by FlexSIG.

The main objectives of the research were to,

- provide a mechanism to protect the CSCW system from an unauthorised user;
- provide a secure information flow for the CSCW system operating over the Internet;
- provide information/data authentication;
- provide information/data integrity;
- provide a mechanism to protect information stored in the system;
- provide a secure prototype system that is transparent to users;
- provide a prototype system to demonstrate, test and evaluate the proposed solutions.

Chapter 2

COMPUTER SUPPORTED COOPERATIVE WORK

2.0 INTRODUCTION

This chapter introduces the area of Computer Supported Cooperative Working, security related to it and the Flexible Software Inspection system which will used as a test-bed in this research. The chapter is structured as follows:

- **Section 2.1** describes the overview history, classification, development, design issues and examples of Computer Supported Cooperative Working.
- **Section 2.2** describes the security area related to area Computer Supported Cooperative Working. Existing information securities in CSCW are also discussed.
- **Section 2.3** highlights the security models that are used as a basis in this research.
- **Section 2.4** highlights the existing groupware software inspections tools. The FlexSIG system, used as a test-bed CSCW application in implementing the security services, is discussed in detail.

2.1 COMPUTER SUPPORTED COOPERATIVE WORK

Chapter One give an introduction to the area of Computer Supported Cooperative Work (CSCW). Computer Supported Cooperative Work (CSCW) is a research field that investigates, on an interdisciplinary basis, how people co-operate and how this co-operation is supported with modern computer technology (Foley & Jacob, 1995). Its focus is on both the nature of cooperative work forms and current work practice and on the way information technology can change, augment, and support co-operative work pattern.

CSCW involves contributions from a variety of disciplines. In the CSCW community (Eltoweissy, 1993):

- social scientists evaluate the impact of technology on group performance,
- computer scientists and electrical engineers explore new concepts and facilities for developing computer and communication applications,
- software developers aiming at creating useful tools for group work, and
- practitioners try to combine the diverse systems, applications, and knowledge about work groups to determine how changes can be made to the ways groups work so group work is more productive.

CSCW applications are commonly known as groupware (Johansen, 1988; Ellis *et al.*, 1991; Grudin, 1991). The term groupware was coined by Peter and Trudy Johnson-Lenz (1982) as follows:

> *"GROUPWARE = intentional GROUP processes and procedures to achieve specific purpose + softWARE applications designed to support and facilitate the group's work."*

Groupware is distinguished from formal software by the basic assumption it makes: groupware makes the user *aware* that he/she is part of a group, while

most other software seeks to hide and protect users from each other. Groupware is software that accentuates the multiple user environment, co-ordinating and orchestrating things so that users can "see" each other, yet do not conflict with each other (Lynch *et al.*, 1990). In the new wave of groupware applications there is more consideration of group processes and how individuals can interact with each other rather than just with their own computer system (Rogers, 1994).

2.1.1 Classification of CSCW

A wide variety of CSCW systems have been developed reflecting the many different views of co-operation. The potential benefits of CSCW systems are better understood in a framework for classifying these systems. The most widely used classification of CSCW systems distinguishes them in term of their abilities to bridge time and to bridge space (Ellis *et al.*, 1991).

Wilson's classification of CSCW has been presented in detail in the first chapter. According to Wilson (1991), as mention in the first chapter, the field of CSCW can be divided into two distinct but interrelated fields; the group working process and the enabling technology employ to support it. Wilson further subdivided these two areas. In the case of group process, individual, organisational, group work design, and group dynamics are all aspects that require addressing under the heading of CSCW. On the technical side, which enables CSCW, communication systems, shared work space facilities, shared information facilities, and group activity support facilities are identified as areas of CSCW interest.

Ellis *et al.* (1991) present two taxonomies for viewing groupware. The first taxonomy is based upon notions of time and space and the second taxonomy is based on application-level functionality. These time and space considerations

suggest four categories of groupware: face to face interaction at the same place and in the same time (e.g., meeting room technology), asynchronous interaction at the same place but in different time (e.g., physical bulletin board), synchronous distributed interaction at different places but at the same time (e.g., real-time document editor), and asynchronous distributed interaction at different places and in different times (e.g., electronic mail system). This view is similar to the classification suggested by Johansen (1988), which is summarised in Table 2.1. A comprehensive groupware system serves the needs of all the categories mentioned above.

	Same Time	**Different Time**
Same Place	*Face-to-face Interaction*	*Asynchronous Interaction*
Different Place	*Synchronous Distributed Interaction*	*Asynchronous Distributed Interaction*

Table 2.1: Johansen Space Time Matrix

The second taxonomy proposed by Ellis *et al.* is based on application-level functionality. This taxonomy is intended primarily to give a general idea of the breadth of the groupware domain. The second taxonomy can be classified into six major categories, i.e. message systems, multi-user editors, group decision support systems and electronic meeting rooms, computer conferencing, intelligent agents, and co-ordination systems. Many of the defined categories overlap and there is merging of these functionalities due to the increased demand for integrated systems.

Looking at the classification of CSCW defined by Ellis *et al.* (1991) and Wilson (1991) above, there is some similarity between their classification. This

similarity is between Ellis *et al.* application-level functionality taxonomy and Wilson's enabling technology classification. Both classifications divide and group CSCW system and research area based on the type of application. Also, both approaches consider applications and systems, even though they disagree on how many classes they should have.

According to Blair & Rodden (1994), there is some difficulty in classifying CSCW using the time and space taxonomy. The reason is that work often switches rapidly between synchronous and asynchronous interactions. Blair & Rodden (1994) mention that CSCW can be classified mainly by the form of co-operation: purely asynchronous systems, purely synchronous systems, and mixed systems.

Wilson's (1990) description of CSCW will be used as a main reference point because it embraces most of the other terms used to describe this field, including Groupware and Workgroup Computing. See Appendix A for other views on naming of CSCW. The term CSCW will be used throughout the thesis.

2.1.2 Development and Design Issues of CSCW

Among the first CSCW systems was the computer conferencing program (Opper & Fersko-Weiss, 1992). According to Opper & Fersko (1992), the first conferencing-like abilities were introduced on an existing email network in early 1970's. Around the same time, a few systems were developed. The Electronic Information Exchange System (EIES) was developed by Turoff and Hiltz, and Notepad was developed by Vallee in the 1970's (Opper & Fersko-Weiss, 1992). A number of major computer vendors were also interested in this area and were creating ad hoc internal CSCW systems to take advantage of their widespread telecommunications facilities.

In the software universe, groupware is placed somewhere between single-user applications and information systems that support organisations (Grudin, 1994). Each software development area emerged independently. Systems designed to support organisations achieved prominence first. "Organisation goals" are major goals typically defined by upper management. These research activities have variously been labelled data processing (DP), information systems (IS), management information systems (MIS), and information technology (IT).

In early 1980s, the spread of interactive and personal computing created large markets for applications designed for individual users. Research and development activities drew on existing human factors (HF) approaches to design and evaluation prior to the emergence in the early 1980s of conferences and journals under such banners as Human and Computer Interaction (HCI).

In the mid-1980s, the terms groupware and CSCW were coined and conference series and literature appeared. According to Grudin (1994), conditions that encouraged the emergence of CSCW in the 80's included:

- computation inexpensive enough to be available to all members;
- technological infrastructure supporting communication and co-ordination, notably networks and associated software;
- widening familiarity with computers, yielding groups willing to try the software;
- maturing single-user application domains that pushed developers to seek new ways to enhance and differentiate products.

Grudin (1994) states that the emergence of CSCW in the 1980s in the United States included both government contract projects and small-group support projects, but is most strongly tied to the shift of attention to small

networked groups. Many United States researchers and developers focus on experimental, observational, and sociological data. However, European contributions to CSCW are often driven by philosophy or social, economic or political theory (Grudin, 1994).

Another study by Grudin (1991) said that European research in CSCW focused on internal development to address organisational needs, while in United States, the research is focused on off-the-shelf products, where it is a move by product developers to expand beyond single-user applications.

Groupware is largely a new market for product developers, along with telecommunications companies that have interest in multi-user applications. Attendance at the first three CSCW conferences was primarily from software product development companies (approximately 40%) and universities (30%) with steady telecommunications presence (5% to 10%) (Grudin, 1994). This confirms that early interest in groupware development is found largely among developers and users of single-user applications.

As developers shift from supporting individual users to supporting groups, many encounter for the first time the challenges described below.

Grudin (1994) listed eight major problems that stem from the social dynamics of groups, drawn from developer experiences, descriptions of short-lived products and research prototypes, and experimental and modelling studies in the literature.

1. *Disparity in work and benefit.* Groupware applications often require additional work from individuals who do not perceive a direct benefit from the use of application.

2. *Critical mass and Prisoner's dilemma problems.* Groupware may not enlist the "critical mass" of users required to be useful, or can fail because it is never to any one individual's advantage to use it.

3. *Disruption of social process.* Groupware can lead to activity that violates social taboos, threatens existing political structures, or otherwise demotivates users crucial to its success.

4. *Exception handling.* Groupware may not accommodate the wide range of exception handling and improvisation that characterises much group activity.

5. *Unobtrusive accessibility.* Features that support group process are used relatively infrequently, requiring unobstrusive accessibility and integration with more heavily used features.

6. *Difficulty of Evaluation.* The obstacles to meaningful, generalised analysis and evaluation of groupware prevent us from learning from experience.

7. *Failure of Intuition.* Intuitions in product development environments are especially poor for multi-user applications, resulting in bad management decisions and error-prone design process.

8. *The Adoption Process.* Groupware requires more careful implementation in the workplace than product developers have confronted.

Overall they call for better understanding of work environments and for corresponding adjustments by developers. Progress on the first five points requires better knowledge of the intended users' workplace. The final three require changes in the development process. The final challenge in particular, addressing the sensitivity of groupware to aspects of its introduction in workplace, demands that product developers expand their conception of

development process and product to include concerns that have been outside their sphere of activity.

Grudin (1994) wrote that computer support has focused on organisations and individuals. Groups are different. Repeated, expensive groupware failures (Grudin, 1994) result from not meeting the challenges in design and evaluation that arise from these differences. They result from not understanding the unique demands this class of software impose on developers and users.

Desktop conferencing, video conferencing, co-authoring features and applications, email and bulletin boards, meeting support systems, voice applications, work flow systems, and groupware calendars are common examples of accepted groupware.

2.2 CSCW AND SECURITY

Foley & Jacob (1995) stated that a great deal of work has been done on the technological aspects of CSCW that is the problem of how one actually provides computer support for co-operation but information security aspects of CSCW technology have not received as much attention (Teufel *et al.*, 1995). Teufel *et al.* (1995) stated that current approaches to information security are directed toward the protection of an individual object or an individual person. With CSCW technology the group aspect is introduced into the security discussion. CSCW applications are group aware and also support the natural working environment where people work together in groups, as opposed to information security models which have been modelled on individual interacting with a system.

Teufel *et al.* (1995) introduce information security concepts and especially address security requirements in the field CSCW. They first standardise CSCW

from a functional perspective. Table 2.2 depicts a functional view of CSCW technology.

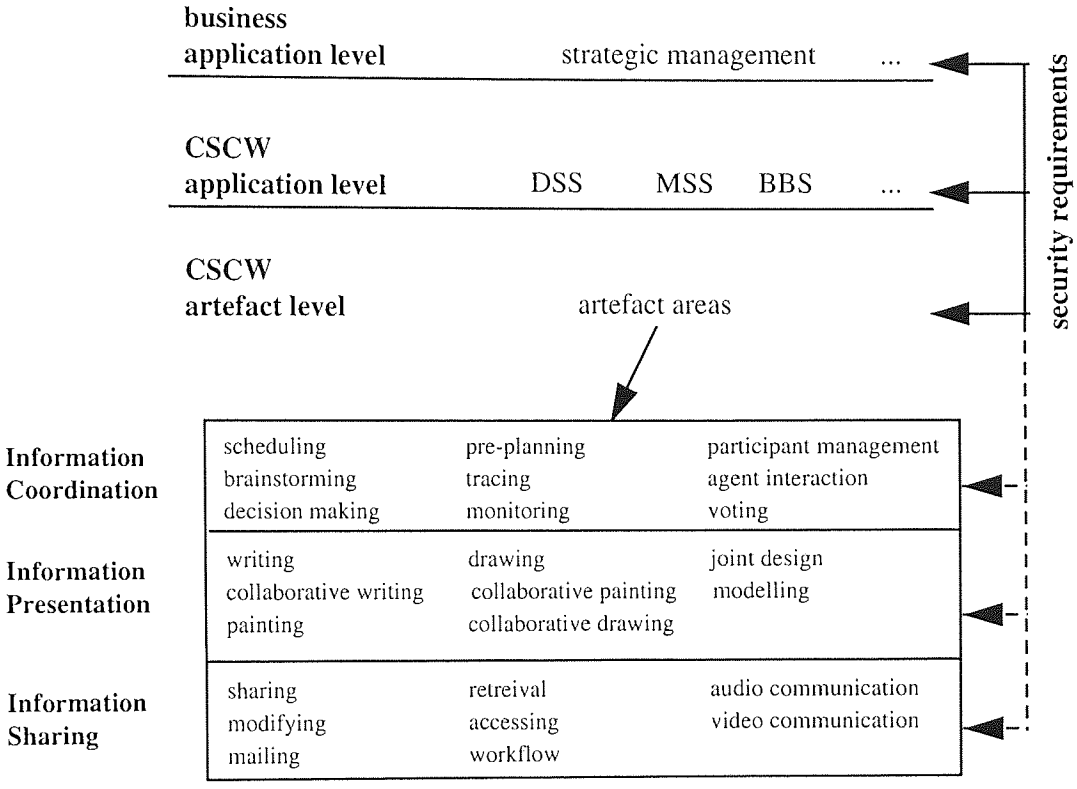

Table 2.2: A Functional View of CSCW Technology

In this functional view of CSCW technology as shown in Table 2.2, Teufel *et al.* (1995) categorise it into three levels: *business application level*, *CSCW applications level*, and *CSCW artefact level*.

- *Business application level*: On the business area level different business units within organisation will use different CSCW applications to fulfil their task, e.g. a strategic management unit might use decision support systems (DSS) and meeting support system (MSS) to improve the effectiveness of their decision making process.

- *CSCW applications level*: On the CSCW application level there exists different CSCW applications such as bulletin board systems (BBS) and personal conferencing systems.

- *CSCW artefact level*: The CSCW artefact level refers to artefacts usually employed in the construction of CSCW applications. These artefacts are grouped into three artefact areas and form a hierarchy: *Information Co-ordination*, *Information Presentation*, and *Information Sharing* (see Table 2.2).

  ◊ *Information Co-ordination Area*: Information co-ordination area refers to all the artefacts facilitating the co-ordination and the manipulation of information in the sense of evaluation, analysis, and agent interaction. In CSCW applications where different multiple artefacts are used, it is important to realise that the way agents interact also affects the security requirements. Some of these artefacts are scheduling, voting, agent interaction, and participant management.

  ◊ *Information Presentation Area*: Some of these artefacts are: writing, collaborative writing, drawing, modelling, and joint design.

  ◊ *Information Sharing Area*: Information sharing artefacts support the storage, retrieval, access to, sharing and mailing of information spaces.

  These three artefacts are not independent of each other. The artefacts on the higher levels are using the functionality of artefacts of the lower levels to support their own facilities, e.g. the scheduling facility (information co-ordination area) needs writing (information presentation area) and mailing (information sharing area) to function.

The security requirements on each level will be different. At the business application area level a security requirement from senior management might be

to authenticate the members participating in a strategic business meeting. The CSCW application level with applications such as meeting support systems might consider confidentiality as very important as compared to the lower confidentiality requirements for BBS. It is clear that classical information security criteria such as confidentiality have to be adapted and extended to fulfil the requirements on the different level (Teufel *et al.*, 1995).

2.2.1 The Security Areas in CSCW

Security areas that are relevant for CSCW applications are *Information Security*, *Group Security*, *Content Exchange Security*, and *Communication and Data Security* (Teufel *et al.*, 1995).

Information Security Area

Information security for the exchange of information takes a general view that not only includes but also encompasses information technology. Issues such as inter-personal relationships and the use of information by humans to support decisions are included. Criteria that are relevant to information security are *authenticity of content* and *goal conformity*.

- *Authenticity of Content.* Authenticity is a widely accepted security criterion for information systems. Authenticity requirements for CSCW environments encompass the authenticity of communication partners. Authenticity of content means that the representations can be validated according to the original source of the information. Possible security mechanisms to ensure authenticity of content could be digital signature, public key encryption, and backups.

- *Goal Conformity.* Goal conformity prevents the misuse of information, i.e. it ensures that the use of information is bound to the intended goals of the owner. Object classification, job description and Need-to-Know access control mechanism could be used to guarantee goal conformity.

Group Security Area

The group security approach views a group as a dynamic entity. Selected members of a group provide information on a dynamic basis (real time) about members belonging to the group. Security mechanisms for CSCW applications should be able to guarantee that anonymity is maintained in situation such as anonymous voting (Teufel *et al.*, 1995). Criteria that are relevant for group security are *group authentication*, *group identification*, *group accountability*, *group integrity*, and *anonymity*.

- *Group Authentication.* Group authentication encompasses the normal definition for authentication. The requirement is to authenticate a group and all the members belonging to this group. Group authentication is required in a group activities where the participants may be at different places at the same time or different time (asynchronous and synchronous distributed interaction). Groups authentication could be ensured by using passwords and mandatory access control (MAC).

- *Group Identification.* Group identification is a limited case of group authentication. Some CSCW applications require that a group and members belonging to a group have to be identified but not necessarily authenticated. The identification is used for communication purposes where activities, such as discussion of confidential information or decisions making, are not required. Identification is needed when the participants of the group are at

the same place at the same time (synchronous interaction). Location specific passwords and views are possibilities to fulfil the security requirement group identification.

- *Group Accountability.* Group accountability provides the service of maintaining a record of the activities not only of the members of the group but also for groups. Group accountability tracks all group activities done in the information co-ordination, information presentation, and information sharing areas. Security mechanisms in the majority of cases provide for accountability through the use of audit services.

- *Group Integrity.* CSCW applications should also preserve the integrity of objects stored in both the information sharing area or displayed in the information presentation area. Integrity for CSCW applications is much more complex when compared to classical definitions of integrity, where integrity is viewed as preserving the contents of objects. Possible security mechanisms to ensure group integrity are digital signatures and passwords.

- *Anonymity.* Anonymity guarantees that an individual belonging to a group may under certain circumstances not be identified. Anonymity has always been a preferred requirement of voting procedures and should therefore be adopted in the modelling of security requirements for CSCW applications. Privileged attribute certificates (PAC) are one possibility to guarantee anonymity.

Content Exchange Security Area

The pragmatic relevance of content exchange security is explained by the statement that information is only useful because someone can do something with it. CSCW applications require that content exchange additionally addresses

the semantics of inter-personal and inter-group communication. Criteria that are relevant for context exchange security are *acceptance of obligation* and *personal accountability*.

Communication and Data Security Area

Communication and data security refers to the fundamental building blocks of information security perceived by many international standards available today. CSCW applications are dependent on the secure functioning of the underlying levels and use supporting security mechanisms for communication and data security. Criteria that are relevant for communication and data security are *authenticity of communication partners*, *confidentially*, *availability*, *integrity*, and *identification*.

- *Authenticity of Communication Partners*. Authenticity of communication partners is necessary to trace the source of exchanged information. CSCW applications require a high degree of "trusted path" implementation. Possibilities to ensure authenticity of communication partners could be the use of inter-network rights or time validity mechanisms.

- *Confidentiality*. Confidentiality describes the state of being private or secret i.e. only accessible by authorised people. Confidentiality is implemented by using cryptography. The confidentiality requirements for CSCW require that various roles played by members in groups are reflected in the access control information. Access control information includes persons, processes, and protected resources. The interaction between members in a group is controlled by access rules which determines the flow of information. Access rules such as (subject, object, access right) need to be redesigned to include security.

- *Availability*. Availability is a state of existence or readiness of resources and services. Availability within CSCW applications requires that all objects belonging to the information co-ordination area, information presentation area and the information sharing area are always available within the protected environment. Possible security mechanisms to ensure availability could be the use of privileged certificates (PAC), distributed file services, mandatory access control (MAC), and backups.

- *Integrity*. Integrity relates to the internal state of objects in the information co-ordination area, the information sharing area, and the information presentation area. Digital signature, error detection mechanism, and object classification are appropriate to secure integrity.

- *Identification*. Users are identified but not authenticated. Identification is required for communication purposes where activity like discussion of confidential information is not required. Network access control mechanisms and access control lists are suitable possibilities to ensure identification.

2.2.2 Mapping Artefact Areas onto Security Areas

A generalised framework describing the major CSCW areas and security requirements for CSCW applications have been discussed in the previous section (Section 2.2 and 2.2.1 respectively). This section shows which security criteria can be utilised to address the security requirements of specific CSCW artefacts. For example, the integrity (security area: communication and data security) could be ensured by using public key encryption in the specific CSCW application. Table 2.3 shows the mapping of the various artefacts onto specific criteria. The artefacts list as used in Table 2.3 is not fully representative of the CSCW because CSCW is still a maturing technology (Teufel *et al.*, 1995).

Consequently, the implementation of the information security concepts with respects to a real CSCW application is the next step to be implemented (Teufel *et al.*, 1995).

		Information Security		**Group Security**					**Context Exchange Security**		**Communication and Data Security**				
Artefact Areas / **Security Areas**		Authenticity of content	Goal confirmity	Group Authentication	Group Identification	Group Accountability	Group Integrity	Anonymity	Acceptance of obligation	Personal Accountability	Authenticity of communication partners	Confidentiality	Availability	Integrity	Identification
Information coordination	scheduling				X	X	X		X				X		
	brainstorming				X			X	X					X	
	decision making	X	X	X	X	X	X	X	X	X	X	X		X	X
	pre-planning				X					X		X	X	X	
	tracing				X	X				X					X
	monitoring				X									X	X
	participant management				X						X	X		X	X
	agent interaction			X	X	X	X	X		X	X	X	X		X
	voting	X		X	X		X	X	X		X			X	X
Information presentation	writing	X	X				X							X	
	collaborative writing	X	X	X	X	X	X		X	X	X	X	X	X	
	painting/drawing	X	X				X								
	collaborative painting/ drawing	X		X	X	X	X		X	X	X	X	X	X	
	joint design	X		X	X	X	X		X	X	X	X	X	X	
	modelling	X	X				X						X	X	
Information sharing	sharing			X	X	X	X			X		X	X	X	
	modifying	X					X		X	X	X	X	X	X	
	mailing			X			X		X		X	X		X	
	retreival		X							X		X	X	X	
	accessing		X			X				X	X	X	X		
	workflow	X	X		X		X		X	X	X	X	X	X	
	audio communication	X		X	X								X	X	X
	video communication	X				X							X	X	X

Table 2.3: Mapping Artefact Areas onto Security Areas

Security areas that are relevant to CSCW have been laid out in previous section. The research described in this thesis concentrates on the security areas

of Group Security and Communication and Data Security, as proposed by Teufel *et al.* (1995). In a group security area, the area of interest is the issue of group authentication. In data communication and data security, confidentiality and integrity are the areas of interest. These areas correspond to the objectives of the research (Chapter One).

2.2.3 Existing Information Security in CSCW

Access Control

Shen & Dewan (1992), Kanawati & Riveill (1995), and Coulouris & Dollimore (1994) in their research focus on access control issues for collaborative environments. Shen & Dewan (1992) in their work stated that there has been much research done in computer applications for facilitating collaboration among multiple distributed users but there has been relatively little works done in controlling access to the collaboration. Almost all available collaborative systems or groupware applications provide all collaborators or users with the same rights (Shen & Dewan, 1992; Kanawati & Riveill, 1995).

Shen & Dewan (1992) in their research proposed a new access control model to meet the requirements of the collaborative environments. The model that they proposed is based on a generalised editing model of collaboration, which assumes that users interact with a collaborative application by concurrently editing its data structures. It associates fine-grained data displayed by a collaborative application with a set of collaborative rights, and provides programmers and users with a multi-dimensional, inheritance-based scheme for specifying these rights. They identified several requirements that a generic access control model for collaborative environments should support. Due to the limitations of the conventional model (Shen & Dewan, 1992) in supporting

the collaborative environments, they have extended the conventional model in several ways to overcome these limitations. They are: *Collaborative rights*, by defining a new set of access rights for generic collaborative model, *negative rights*, by supporting the notion of negative right to allow explicit denial of access, *inheritance-based specification*, by supporting an extended access matrix that supports not only individual subjects, objects, and rights but also groups of these entities, and *automation*, by including mechanisms that relieve an application of the task of implementing the details of access control. Shen & Dewan (1992) implemented all the extensions mention above in their "Suite" multi-user framework.

Kanawati & Riveill (1995) in their work address the problem of specifying user roles over the production space (e.g. the set of shared documents in a co-authoring system). Kanawati & Riveill (1995) stated that almost all available groupware applications provide all users with the same access rights. According to Kanawati & Riveill (1995) it is obvious that collaborative tasks require different users have different access control rights and few works have addressed this question. They provide a flexible, dynamic fine-grained and easy to use role attribution mechanism by proposing a model that is based on defining hierarchical schemes over the three dimensions involved in access control: the subject, the object and the requested rights.

Coulouris & Dollimore (1994) proposed a security model to support co-operative work in which the security of the information used and produced is critical, and where the participants are not equally trusted. They developed so called "access control model" for security in which access to information objects is controlled in accordance with a scheme of access rights. The model they proposed has two levels at which access control is represented - *user level* and *programming level*.

Secure Group Communication

Sakakibara *et al.* (1994) and Takizawa & Mita (1993) in their research focus on secure group communication, and have published several papers on this.

Sakakibara *et al.* (1994) state that attention on CSCW has increased, and it has led to consideration of the needs of secure group communication. In order to realise secure group communication, important data which is sent from a member of a group to other members of the group should be encrypted by a common cryptographic key of the group (i.e. group communication key) (Sakakibara *et al.*, 1994). Non-members of the group can not understand the encrypted data under the group communication key of the group. In their research they proposed an identity-based non-interactive group communication key-sharing scheme using smart cards based on the modified copy key (MCK) method. In the MCK method users did not hold group communication keys used for enciphering and deciphering, but hold a common key and key generator "*Pieces*" which are positive integers instead of keys themselves. Each user gets a communication key generated from a common key and Pieces held by himself, when he makes cryptographic communications with other users. If more than two users show their *Pieces* with each other, they can make any group communication key. Users can share a same group communication key among more than two users non-interactively. This method is however vulnerable to conspiracy attack. In order to defend against this attack, *Pieces* that are distributed and held by users must not be exposed to them but hidden in smart cards, and handed to users by a trusted centre in an organisation.

Takizawa & Mita (1993) stated that in distributed applications like teleconferencing and cooperative work, there is a need of secure group communications among multiple communication entities. Takizawa & Mita highlight that in the open system environment, computer systems can be easily

interconnected by the communication network. With this open environment, one critical problem of how to protect the system from attacks by malicious entities arises. One solution is to encipher the data by using a secret key cryptography, and providing a secure communication channel among two entities using public key cryptography (Takizawa & Mita, 1993). Takizawa & Mita (1993) named a group of entities as a *cluster* and a subset of entities in the cluster as *subcluster*. In their paper, Takizawa & Mita (1993) discuss how to provide secure group communication for a cluster of multiple entities in the presence of attacks by malicious entities by using a less-secure broadcast network like Internet and how to established a secure sub-cluster communication in the cluster.

Others

Foley & Jacob (1995) show interest in what is meant by confidentiality security in CSCW applications. They are less concerned with the technological aspects of how security should be enforced and more concerned with how one might specify the security requirements for CSCW applications and the resulting properties that a CSCW system, providing support for the applications, should uphold. This contributed to the motivation and formulation of the research problem.

Hanka & Buchan (1996) stated that the security of communications over the Internet combined with the security of data servers is an importance issue to be considered. In their health-related application they highlight the security requirements which are access control, secure transmission of data, authentication and non-repudiation for data exchange, and an audit trail to provide reliable and unalterable audit records. Hanka & Buchan (1996) also highlighted the need of secure electronic mail because electronic mail is one of the most used services over the Internet and is regarded as one of the

fundamental requirements of Internet access. This contributed to the conceptual framework of the model discussed in Chapter Five (Section 5.2).

Idris (1995), addresses the problem of security interoperability in multi-databases environments. He has focussed on the secrecy attribute and provided interoperability by constructing two types of security mechanism in the integrated environments, which he has termed the *static mechanism* (Static Security Layer) and *dynamic mechanism* (Dynamic Security Layer). The static mechanism is the default security which is blended within the integrated database, while the dynamic mechanism is a refinement in access control management. According to Idris (1995) the combination of both mechanisms has successfully provided a degree of security between the distributed databases. This contributed to the development of the model discussed in Chapter Five (Section 5.1.2).

2.3 SECURITY MODEL

This section mainly discusses the distributed security model. The McGhie's (1994) and the SecureWay (IBM, 1997) security model are discussed. Both of these models are used as the basis for the developing SecureSIG model.

McGhie's distributed security model established security utilities and services at the network level, supported by the central security group that conforms to open system standards (McGhie, 1994). McGhie's security model consists of two main components, namely: *primary components* and *supporting components*. McGhie's primary components consist of four utilities and services, namely: authentication, authorisation, administration, and audit. The supporting components consist of three utilities and services, namely: encryption, external file transfer, and risk assessment and data classification.

The SecureWay (IBM, 1997) security model consists nine of components, namely: *Credential services*, *Authentication Services*, *Security Context Services*, *Access Control Services*, *Cryptographic Services*, *Key Recovery Services*, *Secure Content Distribution Services*, *SET Secure Electronic Transaction*, *Applet Security*. Credential services are responsible for the management and use of credentials. Authentication services are responsible for establishing and proving identities. Security context services make authentication more efficient across a network by "remembering" that a user has been authenticated to a particular system. Access control services check a user's credentials to verify that the individual is authorised to access or use a specific resource and if so, what kind of usage is allowed. Cryptographic services provide the ability to communicate between parties in such a way that prevents other parties from accessing and understanding the communication. Key recovery services provide a mechanism to reconstruct cryptographic keys in case of the loss of a key. Secure content distribution services enable copyrighted intellectual property to be distributed while protecting the rights of the owners. Secure Electronic Transaction provides a mechanism for securely and automatically routing payment information among users, merchants, and their banks. Applet security enables the end user to interface with the Java applet in the usual manner but be transparently protected by cryptographic functions provided within the Java environment.

2.4 SOFTWARE INSPECTION

As mentioned in the introduction section, in this research we are using Flexible Software Inspection Groupware (FlexSIG) proposed by Sahibuddin (Sahibuddin, 1999) as the test-bed as CSCW application in the implementation of security services.

The aim of inspection is to analyse a products or document to detect defects (Fagan, 1976). Software inspection is a manual process. In the standard software inspection procedure, participants attending the software inspection meeting are assigned one of several roles: *moderator*: administer the meeting, *reader*: reads the text of module being inspected, *scribe*: records any proposed comment that are agreed on by the committee, *author*: author of the code being inspected, who answers questions from the inspectors about the module, and an *inspector*. The benefit of using inspection is well documented (Ackerman, 1984; Kitchenham *et al.*, 1986).

2.4.1 Software Inspection Models

There are three major formal software inspection methods, namely: *Fagan's software inspection* (1976), *Humprey's software inspection* (1989), and *Gilb & Grahm's software inspection* (1993).

Fagan's (1976) inspection process consists of five steps. The steps are overview, preparation, inspection, follow-up, and rework. A planning step was later added to improved versions of the inspection process (Fagan, 1986). There are four people in Fagan's inspection team. The number can be changed if circumstances indicate otherwise (Fagan, 1976). The Fagan's teams are assigned one of several roles; moderator, author, reader, and tester (Fagan, 1986). In Fagan's (1976, 1986) software inspection process, one inspection session should not last more than two hours.

Gilb & Grahm's (1993) software inspection consists of ten inspection process phases. The inspection process phases are request, entry, planning, kick-off meeting, individual checking, logging meeting, edit, follow-up, exit, and release. Two to three people are recommended for maximum efficiency in Gilb &

Grahm's (1993) software inspection. Gilb & Graham's (1993) inspection teams consist of an inspection leader, author, and checker. A scribe is appointed during the meeting. In Gilb & Graham's (1993) software inspection process, one inspection session should not exceed two hours.

Humprey's (1989) software inspection consists of preparation (consists of entry criteria and opening meeting), preparation, and post-inspection activity. Humprey's (1989) inspection teams consist of moderator, producers, reviewer, and recorder. Number of participants in the inspection team should not exceed 5 or 6 persons (1989). Time of inspection meeting should not exceed two hours per session (Humprey, 1989).

2.4.2 Groupware Software Inspection Tools

The inspection process was first described by Fagan (1976). Since then there have been many other inspection processes proposed by other researchers and practitioners. Sahibuddin (1999) established that software inspection could be described as a group activity. Among the first tools that supported software inspection were ICICLE (Intelligent Code Inspection Environment in C Language Environment) (Brothers *et al.*, 1990) and InspeQ (Inspecting software in phases to ensure Quality) (Knight & Myers, 1991; 1993). Both ICICLE and InspeQ did not support distributed inspection, in fact, InspeQ is a software inspection tool for one user only. Other software inspection tools such as Collaborative Software Inspection (CSI) (Mashayekhi *et al.*, 1993), Scrutiny (Gintell *et al.*, 1993), Collaborative Software Review System (CSRS) (Johnson, 1994), Asynchronous Inspector of Software Artefact (AISA) (Stein *et al.*, 1997) and FlexSIG (Flexible Software Inspection Groupware) (Doherty & Sahibuddin, 1996; 1997), (Sahibuddin, 1999), support distributed inspection either as

distributed asynchronous or distributed synchronous processes. Our interest in this research is a groupware software inspection tool that supports distributed inspection (CSI, CSRS, Scrutiny, AISA, and FlexSIG). FlexSIG is the system of interest for the remainder of this research and it will be dealt in more detail. Table 2.4 shows the time-space comparison between groupware software inspection tools mentioned above.

FlexSIG

FlexSIG extends the code inspection groupware proposed by Brother *et al.* by not limiting the meeting to the first quadrant (i.e. face-to-face interaction), but allowing any quadrant of the Johansen space-time matrix as shown in Table 2.4.

	Same Time	Different Time
Different Place	Scrutiny CSI FlexSIG	AISA CSRS FlexSIG
Same Place	Scrutiny FlexSIG	FlexSIG

Table 2.4: Time-Space Comparison between Groupware Inspection Tools

FlexSIG enhances the system supporting the informal code review session. In their system, the enabling technology and the integration platform are handled by the World Wide Web (WWW). In the system they proposed, the role of moderator is different from the formal software inspection meeting set up. It is not necessary for the moderator to be on-line at the same time as the inspector or the author because of the integration of electronic mail (e-mail) into

their system. With the e-mail facilities provided, an asynchronous communication line can be established between moderator, author, and the inspector.

The FlexSIG system takes little account of security aspects. However, in an open distributed environment such as the Internet, systems connected to the network can easily be accessed and information can easily be tapped. FlexSIG provides only system access control, and it is handled by authorisation of the client during login into the system. The system access control provided is not secure because the password is passed back to the server program in a plaintext form. The information can easily be tapped if the communication line is being monitored. Furthermore, the authorisation file used is unprotected plaintext and this results in the file being easily accessible by unauthorised parties. This weakness also arises when determining the level of access for users. Another weakness is that the system did not provide a way to ensure information transferred on the network is secure. There is no protection scheme provided to ensure this. Issues of integrity and authentication of information transferred were not considered. Finally, in software inspection, source code is the most valuable asset. With no protection of the stored source code, it is easily accessible to unauthorised parties. If this issue is not taken into account, the often highly valuable source code is at risk of loss. All these weaknesses can be solved using Cryptography techniques, and will be discussed in Chapter Three, Four, and Five.

2.5 SUMMARY

In summary, section 2.1 discussed and reviewed papers on CSCW related to groupware, classification of CSCW, and development and design issues in CSCW. Many classifications have been reviewed and discussed but the classification of

CSCW that has been given by Wilson will be used here.

Section 2.2, discussed and reviewed papers related to the information security concepts, particularly to acquire security requirements in the field of CSCW, computer security and security related to CSCW. An access control and secure group communication related to CSCW were also considered. The research described here will focus on the security aspects of CSCW, noting the lack of consideration of security issues in reported work on CSCW.

Section 2.3 discussed and review papers of the existing distributed security model. McGhie's and the SecureWay security models are presented. These two security models are the basis for the development of the Secure Software Inspection Groupware discussed in Chapter Five.

Section 2.4, discussed and reviewed papers on the existing software inspection tools and concentrated on the issue of code inspection. Existing groupware software inspection tools were also highlighted. Flexible Software Inspection Groupware (FlexSIG) will be used as the test-bed, and has been discussed in detail.

Chapter 3

ENABLING TECHNOLOGY AND SECURITY SERVICES

3.0 INTRODUCTION

The chapter discusses the issues of the enabling technology and the security services. The topic of cryptographic key management is also discussed. The chapter is structured as follows:

- **Section 3.1** discusses the enabling technology used in the implementation. These include Cryptography, Java technology (Java), and the Internet.
- **Section 3.2** presents the security services and their mechanisms. The security services from the security architecture proposed by International Standards Organisation (ISO) are presented.
- **Section 3.3** describes the cryptographic key management. The secret and public key distributions are highlighted. Public key certificates are presented, pointing out their advantages in public key distribution. The standard certificate, the X.509 certificate, is also presented.

3.1 THE ENABLING TECHNOLOGY

The section focuses on the issues of the enabling technology. Three types of technologies that are of interest to this research are Cryptography, Java Technology (Java), and the Internet. All three areas provide the technology to enable this research to accomplish its goal.

3.1.1 Cryptography

Cryptography is the major enabling technology used in the implementation to provide the tool for security. A brief introduction of cryptography has been given in Chapter One. Cryptography is described in more detail in Appendix B.

Cryptographic systems fall into two general categories (identified by the types of keys they use): *secret key* and *public key* systems (Russell & Gangemi, 1991).

Secret Key Cryptography

Secret key cryptographic systems are very widely used for the protection of information (Davies & Price, 1989). One reasons for their widespread use is the fact that secret key cryptographic systems are substantially faster to encrypt and decrypt than public key cryptographic systems, and are considered harder to break given equivalent key lengths (Schneier, 1996).

Secret key cryptography is principally applied in *stream ciphers or block ciphers*. A stream cipher encrypts one bit of a plaintext message one at a time, using an encryption transformation that varies with time (Menezes, van Oorschot & Vanstone, 1997). A block cipher will encrypt a block of plaintext typically 64 or 128 bits at a time. Block ciphers operate by taking a fixed length

of plaintext as one block and generating the same number of bits of ciphertext. The advantages and disadvantages of these stream and block cipher are discussed in Appendix B. The secret key block cipher is used in this research and the DES (Data Encryption Standard) and the IDEA (International Data Encryption Algorithm) block cipher will be discussed in more detail due to their adoption in the implementation of this research.

In secret key block ciphers there are four common modes of operation, mainly: *Electronic Codebook* (ECB), *Cipher Block Chaining* (CBC), *Cipher Feedback Mode* (CFB), *and Output Feedback Mode* (OFB) (Menezes, van Oorschot & Vanstone, 1997). All of them can be used with any block cipher. The CBC mode is used in this research and it will be used in the development of the prototype. Block ciphers can be either symmetric-key or public-key. Details of these four common modes of block cipher operation are given in Appendix B.

The DES and IDEA Block Ciphers

This section focuses on the DES (Data Encryption Standard) and the IDEA (International Data Encryption Algorithm) block ciphers. These two block ciphers were adopted as the main block ciphers in the development of the prototype in this project. Full details of the DES and IDEA block cipher can be found in Appendix C. The DES block cipher was chosen because it has been the most popular algorithm in the cryptanalytic research for the last 20 years. Since it was introduced in 1977, the DES has been extensively analysed for its cryptographic strength. Kerchoff's (Schneier, 1996) principle stated that a cipher is cryptographically strong only if it is publicised and still not broken. The DES is probably the most widely accepted, publicly available, cryptoalgorithm today due to two main reasons (Smid & Branstad, 1992): first, no one has demonstrated

a fundamental weakness of the DES algorithm; second, its endorsement by the U.S. federal government – the only publicly available algorithm to have ever been endorsed by the U.S. government. DES is a commonly accepted standard encryption scheme, well known and well established (Karila, 1991). The IDEA block cipher is an effective instance of theoretic background applied to cipher design. Schneier (1996) stated that the IDEA block cipher is the most secure block algorithm available to the public at this time. The IDEA is included in Pretty Good Privacy[2] (PGP), which alone ensures wide spread use of the algorithm (Stallings, 1999).

There are many other block ciphers to be found in the literature (Schneier, 1996) (Menezes, van Oorschot & Vanstone, 1997), some of the examples are the NewDES (Scott, 1985), FEAL (Shimizu & Miyaguchi, 1988), and SAFER (Massey, 1994). See Appendix D for description of these block ciphers.

Security of the DES

There has been a fair amount of controversy about DES security. There has been much speculation on the key length, number of iterations, and design of the S-boxes (Schneier, 1996). Some have charged that the design was deliberately sabotaged by the National Security Agency (NSA), or that the key size is just small enough that a major government or large corporation could afford to build a machine that tries all 2^{56} possible keys for a given ciphertext (Cheswick & Bellovin, 1994). It was argued that, since the design criteria of the substitution tables (S-boxes), and indeed for the entire algorithm, were not made public , the entries could have been selected in such a manner as to hide a "trapdoor" (Stallings, 1999). According to Stallings (1999) a number of regularities and

[2] PGP is developed by Phil Zimmermann. It provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.

unexpected behaviours of the S-boxes have been discovered but despite this, no one has so far suceeded in discovering the supposed fatal weaknesses in the S-boxes. Despite the controversy over the security of DES, it is today the most widely accepted, publicly available, cryptoalgorithm (Schneier, 1996).

A lot of research has been carried out concerning the security of the DES. Research results by Biham & Shamir (1991) indicates that the basic design of the DES is actually quite strong, and was certainly not sabotaged.

There are known weaknesses of the DES, but these do not limit the effectiveness of the algorithm (Pfleeger, 1989). The first known weakness concerns *complement keys*. In the round function of DES, the sub-keys are XORed with the expanded right sub-block in every round and this configuration result to a *complementation property* of the cipher. If E denote DES, and p' the bitwise complement of p, then $c = E_K(p)$ implies $c' = E_K(p')$, that is, bit-wise complementing (replace all the 0s with 1s and the 1s with 0s) both the key K and the plaintext p results in complemented DES ciphertext. With this it means that a chosen-plaintext attack against DES succeeds with only half the possible keys: $2^{56}/2 = 2^{55}$ keys instead of 2^{56} (Pfleeger, 1989). It is still questionable whether the complementation property is a weakness (Schneier, 1996).

The second known weakness concerns choice of key. This weakness occurs because of the way the initial key is modified to get sub-key for each round of the algorithm (Schneier, 1996). The initial value is split into two halves, and each half is shifted independently. If all the bits in each half are either 0 or 1, the key used for any cycle of the algorithm is the same for all the cycles of the algorithm. This can occur if the key is entirely 1s, entirely 0s, or if one half of the key is entirely 1s and the other half is entirely 0s. If this scenario happened, the keys produced are *weak keys*. There are also six pairs of *semi-weak keys* that select the same DES permutation. Semi-weak key occur due to the way in which DES

generates sub-keys; instead of generating 16 different sub-keys, these keys generate only two different sub-keys and each key is used 8 times in the algorithm (Schneier, 1996). If this scenario happened some pairs of keys encrypt plaintext to the identical ciphertext, this allows one key in the pair to decrypt messages encrypted with the other key in the pair. However, the number of these weak keys and semi-weak keys is very small and does not significantly impact on the cryptosystem's security.

The success of Biham & Shamir (1991) differential cryptanalysis was found to be related to the number of rounds of the DES. Biham & Shamir found that DES with any number of rounds fewer than 16 could be broken more efficiently than by a brute-force attack. Before this result, variants of DES with a reduced number of rounds had been cryptanalysed. DES variants with three or four rounds were easily broken by Andelman & Reeds (1982). DES with six rounds was broken some year later by Chaum & Evertse (1986).

According to Menezes, van Oorschot & Vanstone (1997), linear cryptanalysis provides the most powerful attack on DES to date. Linear cryptanalysis is a cryptanalytic attack developed by Matsui (1994). This attack linearly approximates the S-boxes. Matsui (1994) illustrates that up to 8 DES S-boxes, while strong against differential cryptanalysis, prove relatively weak against linear cryptanalysis. Matsui (1994) has recovered a DES key for full 16-round DES in 50 days using twelve HP9735 workstations.

The Security of IDEA

Lai (1992) has argued, but has not proven, that the standard IDEA cipher is secure against differential cryptanalysis attack after only 4 of its 8 rounds. According to Biham (1993), his related-key cryptanalytic attack does not work

against IDEA. Meier (1994) implemented a cryptanalysis attack on IDEA, his attack is more efficient than brute-force for 2-round (2^{42} operations), but less efficient for 3-round IDEA or higher (normal IDEA with 8 rounds is safe). According to Meier (1994) the impressive theoretical foundation behind the design of IDEA provides a formal indication of security and any known attempt to cryptanalyse the IDEA has failed.

IDEA's key length is 128 bit, which is over twice as long as DES. By using brute-force attack it would require 2^{128} (10^{38}) encryptions to recover the key (Schneier, 1996). However, double-IDEA implementation would be susceptible to the same meet-in-the-middle attack[3] as is the DES. However, because IDEA's key length is more than double DES's, the attack is impractical because it requires a storage space of $64*2^{128}$ bits, or 10^{39} bytes.

Daeman, Govaerts & Vandewalle (1994) found a class of IDEA weak keys, but these keys are a small defect, because the chance of generating one of these keys is very small, one in 2^{96}. According to Schneier (1996) several academic and military groups have cryptanalysed IDEA, but none of them have published information about any successes they might have had.

Menezes, van Oorschot & Vanstone (1997) stated that for full 8-round IDEA, other than attack on weak keys, no published attack is better than exhaustive search on the 128 bit key space. They also stated that the security of IDEA currently appears bounded only by the weaknesses arising from relatively small (compared to its key length) block length of 64 bits.

[3] The *man-in-the-middle attack* occurs when an adversary acts as a third party in a two party conversation. Both legitimate parties assume that they are talking securely with each other; in fact the adversary is intercepting the entire conversation, decrypting it, re-encrypting it and sending it on to the intended recipient.

Public Key Cryptography

Public key systems differ from secret key systems in that there is no longer a single secret key shared by a pair of users. In the public key systems, each user has his owns key material. Furthermore, the key material of each user is divided into two portions, a private component known as *private key* and a public component known as *public key*. The public key and private key is used for encryption and decryption respectively and the decryption cannot be derived from the encryption key. Secret key cryptography permits the public key to be public. A cryptosystems that employ public key cryptography is known *as public key cryptosystem* (PKCS). The main characteristic of PKCS is that the knowledge of the public key does not reveal any significant knowledge about the private key, which is why the public key may be published freely and be available to communicating parties. On the other hand, only the intended recipient knows the private key and thus, the decryption key (private key) is kept secret. There are two major application areas for public key cryptosystems (Nechvatal, 1992*): distribution of secret keys* and *digital signatures* (See Appendix B). The first involves using PKCS for secure and authenticated exchange of data-encrypting key between two parties. Second, PCKS providing authentication, non-repudiation, and integrity checks.

Public key cryptosystems are slower to encrypt and decrypt than secret key cryptosystems. For this reason, public key cryptosystems are usually limited to the set-up of a communications session, for key distribution and key exchange.

There are many types of public key cryptography such as RSA cryptosystems, Knapsack cryptosystems, ElGamal cryptosystems, Rabin cryptosystems and others. In this research the RSA cryptosystem was used in the development of the prototype because it was the most popular, widely accepted and easiest to understand and implement (Schneier, 1996; Odlyzko,

1994). RSA public key is a full-fledged public key algorithm that supports encryption and digital signatures. Other than that, since it was proposed RSA has withstood extensive cryptanalysis (Schneier, 1996). Several public key cryptosystems other than RSA have been proposed (Merkle & Hellman, 1978; ElGamal, 1985; Rabin, 1979). See Appendix E for the description of RSA and other public key cryptography systems.

3.1.2 Java Technology

Java was introduced in 1995. The Java programming language was designed to meet the challenges of application development in the context of heterogeneous network-wide distributed environments. It started out as a programming language called Oak in 1991 (Linden, 1996). Oak was part of a research project to develop advanced software for a wide variety of network devices and embedded systems (Sun, 1995; Linden, 1996). Java was based on C++, but without its unclean and unsafe features (Ciancarini *et al.*, 1996). New principles and structure were inherited from a variety of languages such as Eiffel, SmallTalk, Objective C, and Cedar/Mesa (Sun, 1995; Linden, 1996). The result is a language environment that has proven ideal for developing secure, distributed, network-based end-user applications in environment ranging from networked-embedded devices to the World Wide Web and the desktop (Sun, 1995). According to Linden (1996), Java contains libraries highly tuned to the Internet environment.

Java is designed to enable the development of secure, high performance, and highly robust applications on multiple platforms in heterogeneous, distributed networks (Sun, 1995). The Java language is object oriented language

(Sun, 1995). The development of the prototype in this research is Java applications based.

Java is suited for Internet related tasks, it is also a solid general purpose language to be used in a variety of applications (Roberts 1996; Flynn & Clarke 1995). According to Flynn & Clarke (1995), Java standard classes also provide all the basics blocks necessary for client-server implementation.

The interest in this research regarding Java is the security aspect. Security is of paramount interest in a distributed environment. According to Sun (1995) "the security features designed into java allow applications to be constructed that are secure from intrusion by unauthorised code attempting to get behind the scenes".

Java Security API

The Java Security API (Application Program Interface) is a Java core API, built around the *java.security* package (and its sub-packages) (Sun, 1997). The first release of Java Security in JDK 1.1 contains a subset of this functionality, including APIs for *digital signatures* and *message digests*. In addition, there are abstract interfaces for key management, certificate management and access control.

The "Java Cryptography Architecture" (JCA) refers to the framework for accessing and developing cryptographic functionality for the Java Platform. It encompasses the parts of the JDK 1.1 Java Security API related to cryptography, as well as a set of conventions and specifications provided in this document. It

introduces a " *provider*[4] " architecture that allows for multiple and interoperable cryptography implementations.

The JCA was designed around the principles: *implementation independence and interoperability*, and *algorithm independence and extensibility* so that a new algorithm can be added later without much difficulty and can be utilised in the same way as existing algorithms. In other words, the aim is to let users of the API utilise cryptographic concepts, such as digital signatures and message digests, without concern for the implementations or even the algorithms being used to implement these concepts.

Implementation independence is achieved using a "provider"-based architecture. A Cryptography Package Provider ("provider" for short) is a package or set of packages that implement specific algorithms, such as the Digital Signature Algorithm (DSA) or the RSA Cryptosystem (RSA). Applications may simply request a particular type of object, such as a DSA object, and get an implementation from an installed provider. If desired, an application may instead request an implementation from a specific provider.

Algorithm independence is achieved by defining types of cryptographic "engines" (algorithms), and defining classes that provide the functionality of these cryptographic engines. These classes are referred to as engine classes, and examples include the Message Digest and Signature classes.

Implementation interoperability means that various implementations can work with each other, use each other's keys, or verify each other's signatures. This would mean, for example, that for the same algorithms, a key generated by one provider would be usable by another, and a signature generated by one

[4] Provider is short for Cryptography Package Provider which refer to a package (or a set of packages) that supply a concrete implementation of a subset of the cryptography aspects of the Java.

provider would be verifiable by another. Algorithm extensibility means that new algorithms that fit in one of the supported engine classes can be added easily.

Java Cryptography Extension

The Java Cryptography Extension (JCE) extends the JCA API with additional features for supporting encryption and key exchange. Together, JCE and the JCA provide a complete, platform-independent cryptography API (Sun, 1997).

There are few organisations that produce Java security software that produced and used Java cryptography toolkits, namely: SUN Microsystems (SunJCE), Institute for Applied Information Processing and Communications (IAIK), Systemics Ltd. (Cryptix), and JCP Computer Services (JCP) (Knudsen, 1998). The SunJCE produced by SUN is hampered by U.S. export controls and it is not available outside U.S. The other three security software sources, Cryptix, IAIK, and JCP are a re-implementation of the original JCE (SunJCE) and not hampered by U.S export law. They are available to be used outside U.S because of their development outside the United States.

IAIK Java Security Software produced by Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology (IAIK, 1998) was selected as a Java Cryptography Toolkit in developing the prototype. IAIK Java Security Software offers IAIK Java Cryptography Extension (IAIK-JCE). The IAIK Java Cryptography Extension (IAIK-JCE) is a set of APIs and implementations of cryptographic functionality, including symmetric, asymmetric, stream, and block encryption. The architecture of the IAIK-JCE follows the same design principles found elsewhere in the Java Cryptography Architecture (JCA). For detail regarding JCA refer to (Sun, 1997).

IAIK-JCE API includes all the functionality of Sun's original Java Cryptography Extension and is fully compatible for use within any Java environment (IAIK, 1998).

3.1.3 The Internet

The term Internet refers to a global public networking utility which consists of thousands of networks and hundred of thousands of computer (Hassler, 1997). According to Leiner *et al.* (1998), the Internet has revolutionised the computer and communication world. The Internet is at once a world-wide broadcasting capability, mechanism for information dissemination, and a medium for collaboration and interaction between individuals and their computers without regard for geographic location (Leiner *et al.*, 1998).

The internet model consists of four layers (Oppliger, 1998), namely *the network* (or *network access*) *layer*, *the Internet layer*, *the transport layer*, and *the application layer*. According to Oppliger (1998), the popularity of the TCP/IP (Transmission Control Protocol/Internet Protocol) communications protocol suite is due to its ability to be implemented on top of various technologies and corresponding network access layer protocols. TCP/IP networking is an internetworking technology. Internet protocol (IP) is the key for the TCP/IP communications protocol suite.

The Internet Protocol (IP), the backbone of the Internet, is not a secure protocol. To used an IP network securely, the cryptography needs to be applied on top of the IP network (Knudsen, 1998).

3.2 SECURITY SERVICES AND MECHANISMS

In this research the security services from the security architecture proposed by International Standards Organisation (ISO, 1989) were used. The security services proposed by the ISO security architecture are protocol and system neutral, they can be applied equally to TCP/IP and other network architectures.

3.2.1 Security Services

ISO (1989) defines security services in five major areas: *authentication*, *access control*, *data confidentiality*, *data integrity*, and *non-repudiation*. Each of these security services is discussed below.

Authentication

Authentication services deal with proof of identity. The ISO security architecture defines two forms of authentication: *peer entity authentication and data origin authentication*.

- *Peer entity authentication* is used between peer entities in a system, communication, or transaction. Peer entity authentication establishes, to some acceptable degree of certainty that an entity is in fact who it claims to be. Various mechanisms exist to provide this security service; the degree of certainty here ranges from marginal (simple password authentication) to high (cryptographic means).

- *Data origin authentication* is used to prove the source of a given block of data. This service is intended to allow the recipient of data to determine, to some acceptable degree of certainty, the originator of that data. It is essential in

dealing with data forwarded by or stored on a system that did not originate the data in question.

Authentication services are important because they are pre-requisite for proper authorisation[5], access control[6], and accountability[7] (Oppliger, 1998).

Access Control

Access control services, as their name implies, offer a means of controlling access to a given system or system resource. This service generally makes use of peer entity authentication (to authenticate the entity requesting resource usage), then applies some rule-based mechanism to allow/deny access to the requested resource. It may also require the use of other security services (e.g., confidentiality, data integrity, or non-repudiation discussed below) when invoked remotely. In general, access control services are the most commonly thought of services in both computer and communication security (Oppliger, 1998).

Data Confidentiality

Data Confidentiality refers to the property that information is not made available or disclosed to unauthorised individual, entities, or processes (Oppliger, 1998). Thus, *data confidentiality services* are to provide for the protection of data from unauthorised disclosure:

- A *connection confidentiality service* is to provide confidentiality for all data transmitted in a connection. For connection confidentiality, the use of a connection-oriented protocol, such as TCP, is assumed.

[5] Authorisation refers to the process of granting rights, which includes the granting of access based on access right.
[6] Access control refers to the process of enforcing access rights.
[7] Accountability ensures that the actions of a principal may be traced uniquely to this particular principal.

- A *connectionless confidentiality service* is to provide confidentiality of single data units.
- A *selective field confidentiality service* is to provide confidentiality of only certain fields within the data during a connection or a single data unit.
- A *traffic flow confidentiality service* is to provide protection of information that may otherwise be compromised or indirectly derived from a traffic analysis.

Confidentiality services typically require access control (to protect information in storage on a given system) coupled with either physical control of all communications media over which information is transmitted or some form of cryptography.

Data Integrity

Data integrity refers to the property that informations is not altered or destroyed in an unauthorised way (Oppliger, 1998). Thus, *data integrity services* are to provide for the protection of data from unauthorised modifications:

- A *connection integrity service with recovery* is to provide integrity of data in a connection. The loss of integrity is recovered, if possible.
- A *connection integrity service without recovery* is to provide integrity of data in a conection. However, the loss of integrity is not recovered.
- A *selective field connection integrity service* is to provide integrity of specific fields within the data during a connection.
- A *connectionless integrity service* is to provide integrity of single data units.

- A *selective field connectionless integrity service* is to provide integrity of specific fields within single data units.

Data integrity services typically rely on the use of some type of error detection code to provide the ability to detect alteration to data while in storage or transmission. Sequence numbers and time stamping are typically used to provide protection against insertion, deletion, or replay[8].

Non-Repudiation

Non-repudiation services are to prevent one of the entities involved in a communication later denying having participated in all or part of the communication. Consequently, they have to provide some sort of protection against the originator of a message or action denying that he has originated the message or the action, as well as against the recipient of a message denying that he has received the message. There are two non-repudiation services to be distinguished:

- A *non-repudiation service with proof of origin* is to provide the recipient of a message with a proof of origin. This service makes it impossible for the originator to later repudiate (claim not to have sent) the data to the recipient.
- A *non-repudiation service with proof of delivery* is to provide the sender of a message with a proof of delivery. This service makes it impossible for the recipient to later deny receipt of the data in question.

Non-repudiation services are becoming increasingly important in the context of electronic commerce on the Internet (Oppliger, 1998).

[8] Replay compromises the recording and replaying of previously sent messages or part thereof.

3.2.2 Security Mechanisms

The ISO (1989) security architecture also describes a number of specific security mechanisms that may be used to implement the security services listed in Section 3.2.1. These specific security mechanisms are shown in Table 3.1. The last three mechanisms are not considered in this research because they are not part of the requirements of this research.

- ***Encryption Mechanisms***. Encryption is the transformation of data to hide its information content, prevent undetected modification, and/or prevent its unauthorised used. It is used to protect the confidentiality of data units and traffic flow information, or to support or complement other security mechanisms (Oppliger, 1998). Encryption mechanisms can be implemented using secret key or public key cryptography (Hassler, 1997). Encryption mechanisms provide data confidentiality by protecting information from being accessible from unauthorised parties either during transmission or while it was stored. The secret key block cipher is used in this research. Most ciphers in use today employ secret key because the bulk data encryption of public key cryptography is slower than secret key cryptography (Schneier, 1996).

1	Encryption mechanisms
2	Digital Signature mechanisms
3	Access Control mechanisms
4	Data Integrity mechanisms
5	Authentication Exchange mechanisms
6	Traffic Padding mechanisms
7	Routing Control mechanisms
8	Notarisation mechanisms

Table 3.1: Specific Security Mechanisms

- ***Digital Signature Mechanisms.*** Digital signature mechanisms provide for proof of origin and provide protection against undetected modification of information. Digital signature mechanisms involve the use of information by the sender either to encrypt or to produce a message digest of data being signed. Verification uses publicly available information and procedures from which the signer's private information cannot be derived to determine whether the signature was in fact produced using the claimed signer's private information. Digital signature mechanisms can be implemented using public-key cryptography (Hassler, 1997). Digital signatures using a public key allow an authentic message to be broadcast to many destinations. Each of the destinations can obtain the public key of the sender and check the authenticity of the message, which would be unwise for authenticators using a secret key since a widespread knowledge of the key would weaken its security (Davies & Price, 1989).

- ***Access Control Mechanisms.*** Access control mechanisms use the authenticated identity of an entity, information about the entity (e.g., group memberships), or capabilities of the entity to enforce access rights (ISO, 1989). They are typically based on access control information bases, authentication information, entity capabilities (the possession and presentation of which is taken as right to access the requested resource), security labelling, time of day, routing, and duration. Access control mechanisms are tightly bound to authentication (Hassler, 1997).

- ***Data Integrity*** Mechanisms. Data integrity mechanisms are concerned with two forms of data integrity: the data integrity of a single data unit (field, packet, etc.) and the data integrity of a sequence of data units. Some form of message digest attached to the data unit and verified by the recipient typically provides the integrity of a single data unit. This provides protection against

alteration, but does not provide any form of protection against replay of valid data units (though time stamping can provide a limited form of protection against single-unit replay). The integrity of a stream of data units (e.g., that they arrive unaltered, in sequence, and without replay or the insertion of spurious data) is typically provided through the addition of sequence numbering, time-stamping, or cryptographic chaining. Data integrity mechanisms protect stored or transmitted documents and messages from unauthorised modification (Davies & Price, 1996). Data integrity mechanisms mostly use digital signatures of message digests computed by a cryptographic hash function (Hassler, 1997).

- ***Authentication Exchange Mechanisms.*** Authentication exchange mechanisms provide proof of identity. These mechanisms typically fall into one of four categories: authentication information exchange (e.g., a user id/password system), entity characteristics or possessions (biometrics and physical tokens are examples), and cryptographic techniques (e.g., digital signatures), or systems using a combination of these other three techniques. Authentication exchange mechanisms involving cryptographic techniques typically also use some form of handshaking protocol to avoid the possibility of replay.

The traffic padding mechanisms, routing control mechanisms, and notarisation mechanisms are not described because they are not part of this research.

3.2.3 Cryptography and Security Mechanisms

Schneier (1996) stated that the whole purpose of cryptography is to keep the plaintext (or the key, or both) secret from eavesdroppers or attackers. These

secrets (plaintext and key) are vulnerable to eavesdroppers who are assumed to have complete access to the communication between sender and receiver. Successful cryptanalysis may recover the plaintext or the key.

As mentioned in the first chapter, cryptography can support all the security mechanisms needed to provide security services to protect from eavesdroppers. In the previous section the topics of cryptography and security mechanisms have been discussed. The way in which they support the security services will be discussed in detail in Chapter Five.

3.3 CRYPTOGRAPHIC KEY MANAGEMENT

Key management is the set of techniques and procedures supporting the establishment and maintenance of keying relationship between authorised parties (Menezes, van Oorschot & Vanstone, 1997). Regardless of whether a secret or public key cryptosystem is used, it is necessary for a user to obtain the other user's key. Key management plays a fundamental role in cryptography as the basis for securing cryptographic techniques providing confidentiality, authentication, and digital signatures.

3.3.1 Secret Key Distribution

In secret key system, security is dependent on the secrecy of the key that is shared between two users. These two users who wish to communicate securely must first securely establish a common key. One possibility is to employ a third party such as courier but there are several disadvantages to this implementation (Smid & Branstad, 1992). An alternative is using a central issuing authority to obtain a common key but due to the concentration of trust, a single security

breach would compromise the entire system. Also it would probably need to be online, which in large networks might introduces a bottleneck, since each pair of users needing a key must access a central node. If the number of users is n then the number of pairs of users wishing to communicate privately must share a key, so the number of keys needed could theoretically be as high as $n(n - 1)/2$, which increases in proportion to the square of the number of people present (Hughes *et al.*, 1997). Furthermore, failure of the central authority disrupts the key distribution system.

Another possibility is to employ public key cryptography. First, the secret key and the public key are generated. Then, the sender encrypts the secret key using the recipient's public key. The encrypted secret key is sent to the recipient. Upon receipt, the secret key is decrypted using the recipient's private key. In this project, distribution of secret keys using public key cryptography is employed.

3.3.2 Public Key Distribution

The advantage of a public key is that two users can communicate securely without exchanging a secret key. One means to distribute public keys is a certificate. A certificate is a digital public document containing information identifying a user, the user's public key, the time period that the certificate is valid, and other information. Certificates are typically issued, managed, and signed by a central issuing authority called a CA (Certification Authority[9]).

[9] A Certificate Authority (CA) is a trusted entity whose central responsibility is certifying the authenticity of users. In essence, the function of a CA is analogous to that of the passport issuing office in the Government.

Public Key Certificates

In order for a public key scheme to be successful, a user must guarantee that the public key of another user truly belongs to that user. Public key certificate are a vehicle by which public keys may be stored, distributed or forwarded over unsecured media without danger of undetectable manipulation. The objective is to make one entity's public key available to others such that its authenticity and validity are verifiable (Menezes, van Oorschot & Vanstone, 1997). Both authentication and integrity in distribution of public components can be solved by using the certificates (Kohnfelder, 1978). X.509 certificates are commonly used in practice (Markovitz, 1994).

One method by which certificates can be distributed is described in the following example. *User A* and *User B* register with a CA. During the registration process, the users provide their public key information to the CA. The CA, in turn, provides each user with a signed certificate containing the user's public key, and the public key information of the CA. The users store their certificates in a public directory. To start off, *User A* (the originator) sends a signed message to *User B* (the recipient) using the originator's private key. Upon receipt the recipient queries the public key directory to obtain the originator's public key certificate. The recipient first uses the CA's public key to validate the certificate's signature, then verifies the originator's message signature using the public key contained in the certificate.

In the above example, the two users were registered with the same CA. In practice, users may be certified by different CAs or self-certified. Our concern is the self-certified certificate, because this scheme was employed in the public key distribution scheme used in this project. In the self-certified scheme each user himself computes their private key and corresponding public key.

Certificate Contents

A certificate associates a public key with the real identity of an individual, server, or other entity, known as the *subject*. Certificate information consists of information about:

- *Subject*. Includes identifying information (the distinguishing name), and the public key. The distinguishing name is used to provide an identity in a specific context, for instance, an individual might have a personal certificate as well as one for their identity as an employee.
- *Issuer*. Includes the identification and signature of the Certificate Authority that issued the certificate.
- *Period of Validity*. The period of time during which the certificate is valid.
- *Extensions*. It may have additional information as well as administrative information for the Certificate Authority's use, such as serial number.

The most widely accepted format for certificates is defined by the ITU-T X.509 international standard; thus, certificates can be read or written by any application complying with X.509.

X.509 Certificate

An important part of X.509 is its structure for public key certificates. A trusted Certificate Authority (CA) assigns a unique name to each user and issues a signed certificate containing the name and the user's public key. Figure 3.1 shows an X.509 certificate (CCITT, 1989). The certificate consists of the following:

- *Version* - identifies the certificate formats.

- *Serial Number* - used to uniquely identify the certificate from among those generated by a given CA;
- *Algorithm* - used to sign the certificate, together with any necessary parameters.
- *Issuer* - the name of CA.
- *Period of Validity* - pairs of date indicating the certificate is valid during the time period between the two.
- *Subject* - public key information includes the algorithm name, any necessary parameters, and the public key, and
- *Signature* - CA's signature.

Version
Serial Number
Algorithm Identifier: - *Algorithm* - *Parameters*
Issuer
Period of Validity: - *Not Before Date* - *Not After Date*
Subject
Subject's Public Key: - *Algorithm* - *Parameters* - *Public Key*
Signature

Figure 3.1: X.509 Certificate

3.4 SUMMARY

As a summary, Section 3.1 discussed the enabling technology used in the implementation in this research. These enabling technologies are Cryptography, Java Technology and the Internet.

In Section 3.1.1, Cryptography which is the major enabling technology used in the implementation, is discussed. Two main categories of cryptography: secret key cryptography and public key cryptography are presented. Stream and block ciphers secret key cryptography is highlighted. Greater detail is given on the secret key cryptography block cipher due to it adoption in the implementation of the prototype. The modes of block cipher operation mainly the ECB mode, the CBC mode, the CFD mode, and the OFB mode are mentioned. The CBC mode is highlighted because of it adoption in the research implementation. The security of the two main secret keys block ciphers, the DES and the IDEA secret key block cipher are discussed in detail. The DES was chosen because it is the most extensively studied algorithm and has motivated a lot of research both in cryptography and cryptanalysis. As for the IDEA it is a representative algorithm of impressive theoretic foundations. Public key cryptography is also presented. RSA public key cryptography was chosen in the implementation because it has withstood extensive cryptanalysis, is the most popular, and is easiest to understand and implement (Schneier, 1996).

In Section 3.1.2, Java, another enabling technology used for the development of the prototype in this research is discussed. Features of Java are highlighted. The Java Security API which is a Java core API is presented. The JCE that extends JCA API for supporting encryption and key exchange is discussed. The IAIK-JCE produced by Institute for Applied Information Processing and Communications, Graz University of Technology was adopted

because there is no restriction on non United States version of JCE APIs. This Java security APIs can provided security functionality for Java applications.

In Section 3.1.3, the Internet use as a platform for the prototype is discussed. It adoption because of its ability to be implemented on top of various technologies and corresponding network access layer protocol.

Section 3.2 discussed the security services and it mechanisms. The security services from the security architecture proposed by International Standards Organisation (ISO) were adopted in this research. Authentication, access control, confidentiality, data integrity, and non-repudiation are presented. The security mechanisms that are used to support the security services, mainly; encryption, digital signature, access control, data integrity, and authentication exchange are also presented.

Section 3.3 presented the importance of key management in cryptography. Secret key and public key distribution are also presented. Public key certificates are highlighted, pointing out its advantages in public key distribution. The common certificates, X.509 certificates, are also presented.

Chapter 4

RESEARCH PROBLEM, DESIGN AND PROCEDURE ISSUES

4.0 INTRODUCTION

This chapter describes the formulation of the research problem, research design, and research procedure issues. This chapter is structured as follows:

- **Section 4.1** reviews the literature review of the previous chapters and highlights their relation with the area interest in this research. Contributions from the literature review are identified and listed.
- **Section 4.2** describes the formulation of the research problem. The objectives, the purpose and the importance of this research are also presented.
- **Section 4.3** discusses the research design and procedures. The research methodology, research design, assumptions and the anticipated outcomes of this research are also discussed.

4.1 SUMMARY OF LITERATURE

This section highlights the relation between the areas of CSCW, software inspection groupware, the enabling technologies and security services found in the literature review discussed in the previous chapters.

According to Olson *et al.* (1993) much of today's work is done not individually, but rather in a group. Furthermore, Olson *et al.* (1993) also mentioned that most real work is collaborative in nature. Eltoweissy (1993) mentioned that the success of most projects today relies on the co-operative activities of people and this requires that people communicate, jointly co-ordinate their activities, and share information and ideas more than ever (Eltoweissy, 1993). The advent of networked technologies and groupware applications make possible the forming of work teams that are not physically collocated worked together more efficiently and conveniently (Eltoweissy, 1993). Grief (1988) mentioned that *Computer Supported Cooperative Work* (CSCW) is the field that deals with the development of such facilities.

According to Teufel *et al.* (1995) information security aspects of CSCW technology have not received as much attention compare to the work that has been done on the technological aspects of CSCW (Foley & Jacob, 1995). Teufel *et al.* (1995) introduced the information security concepts to the field of CSCW but implementations of the information security concepts with respects to a real CSCW applications are not described.

In this research, software inspection groupware used as a CSCW application is taken as an example. Software inspection is a groupware activity which aims to analyse a software product or document to detect defects (Fagan, 1976). There exist a few software inspection groupware systems but FlexSIG (Sahibuddin, 1999) is used as a software inspection groupware test-bed in this

research because of it flexibility and it availability. FlexSIG provides flexibility for the software inspection process but the issue of information security is not considered. The enabling technology and the security services discussed can overcome the information security aspects of CSCW in general and FlexSIG specifically. By using cryptography as a main enabling technology supported by Java, issues of information security in CSCW can be solved.

4.1.1 Summary

As a summary, the issue of information security has been raised. There is a need of information security in CSCW in general and specifically to the software inspection groupware process. The issue of information security in the group working process of CSCW can be supported by the enabling technology and security mechanisms discussed in the previous chapter.

Overall the literature has led to the identification of:

- the lack of research done in the security aspects of CSCW in general.
- existing software inspection groupware systems that have put no significant security consideration in the systems.
- the information security concepts and especially acquisition of security requirements in the field of CSCW.
- the enabling technology, security services, and security mechanisms that can be used to solve the problem of information security in CSCW in general, and specifically, software inspection groupware systems.

4.2 RESEARCH PROBLEM

This research is concerned with the issue of securing the groupware software inspection process. The issues of security and transparency of the secure CSCW system are also of interest to this research. This section identifies the research problem based on the literature discussed in the previous chapters. The objectives, the purpose and the importance of this research are also discussed in this section.

4.2.1 Formulation of Research Problem

This section will discuss the design of the model and the tool for secure groupware software inspection process. As mentioned before, FlexSIG (Sahibuddin, 1999) is the system of interest in this research. In order to enhance Sahibuddin's model, and his tool for a software inspection system, the model is first outlined, drawing from other models where needed.

The Drawbacks of Existing Software Inspection Groupware Systems

As mentioned in Chapter Two there exist a small number of software inspection groupware systems ranging from limited synchronous distributed system such as CSI (Mashayekhi *et al.*, 1993) to the most flexible systems such as FlexSIG (Sahibuddin, 1999). All the existing groupware software inspection tools have their limitations and their strengths (Sahibuddin, 1999), but one aspect that has been identified as lacking by Sahibuddin and from literature is that all the available software inspection groupware systems are deficient in security. The lack of security in groupware in general has been raised by Teufel *et al.* (1995).

Security tool such as cryptography is available to handle the deficiency in security of the software inspection groupware system. It can be tackled by providing security services (ISO, 1989) to the system using cryptography as discussed in Chapter Three as a tool.

The FlexSIG system (Sahibuddin, 1999) is the interest of this research and was used as a test-bed in implementing the security services to provide security through the External Security Layer (ESL) and Internal Security Layer (ISL) of the system discussed in Chapter One. The security services are built on the FlexSIG system.

FlexSIG Model

The FlexSIG model proposed by Sahibuddin (1999) provides an extension of existing software inspection models that can resolve the limitations of the existing inspection models while maintaining their strengths. The strengths and limitations of these three major inspection models have been presented by Sahibuddin (1999). The FlexSIG model considers processes, team membership, roles of member, and propose four different operating modes.

Sahibuddin's software inspection model consists of eight inspection processes. These inspection processes are:

- *Initiation.* This process directs team members to a kick-off meeting or to access on-line briefing meeting. Its goal is to start the inspection process. This phase is initiates by the moderator.
- *Kick-off Meeting.* The aim of this phase is to inform the team members on aspects of the document being inspect.

- *Briefing.* The phase is initiated instead of kick-off meeting if the document being inspected is to be distributed electronically.
- *Individual Inspection.* In this phase, each inspector is instructed to study the document being reviewed and log any potential defect or query.
- *Synchronous Group Inspection.* This phase is to be conducted differently depending on the mode of the software process either face-to-face mode or distributed synchronous mode. In this phase the discussion is through the synchronous communication component of the system.
- *Asynchronous Group Inspection.* This phase is to be conducted in the distributed asynchronous mode, team members access the system from different places at different time. Discussion is through the asynchronous communication component.
- *Consolidation.* In this phase, the moderator merge all the defects, queries and suggestions raised by the inspector during the inspection phase. The overall metrics of the inspection process are also collected here.
- *Follow-up.* In this phase, the moderator forwards the report on the consolidation phase to the document author.

The roles in FlexSIG are based on existing software inspection models, but with reduction of some roles that can be handled automatically by the system.

Sahibuddin's (1999) FlexSIG team members consists of:

- Moderator,
- Inspectors and
- Author.

The moderator is responsible for planning the whole inspection process and leads the inspection team. The role of author is to answer any query with regard to a document being inspected. The role of inspectors is to maximise the number of defects found in the document.

Sahibuddin (1999) extends the existing inspection process model by offering flexibility while maintaining the strengths of existing models.

Extension of FlexSIG Model

The research aims to provide security extension to FlexSIG model while maintaining its flexibility in terms of allowing distributed synchronous or distributed synchronous working mode. The process model is retained. This extension maintains the roles of the participants involved in FlexSIG model.

The extension aims to improve on the security aspects of the FlexSIG software inspection model, with the addition of security based on the security model discussed in Chapter Two to the model of the software inspection process proposed by FlexSIG. The extension of the model seeks to provide security in terms of providing a secure data transmission, secure data storage and secure access control.

FlexSIG Tool

FlexSIG (Sahibuddin, 1999) contains six key components and services, namely:

- *Access Control.* This component used to handle team member authorisation.
- *Briefing.* This component describes the set-up information of the software inspection process.

- *Browsing.* This component consists of two-sub-component, which are the facility to browse inspected document and to browse remotely document inspected by other team members.
- *Communication.* This component consists of three sub-components, namely, electronic mail, chat, and pop-up message. The aim of this component is to support both the synchronous and asynchronous mode of communication.
- *Data Logging.* This component is used for collecting team member comments.
- *Supporting Document.* This component contains information with regard to the document being inspected and also information about the FlexSIG system.

Extension of FlexSIG Tool

The FlexSIG tool lacks security in terms of providing protection to the resources stored and flow to and from the groupware software inspection system. In FlexSIG, all data transmitted and stored are in the form of plaintext. FlexSIG provides access control, but the password transmission and all the authorisation databases are in plaintext.

The security extension to FlexSIG can be achieved using cryptography tools as the main enabling technology together with standard security mechanisms (ISO, 1989), as discussed in Chapter Three. How these securities mechanism can provide security to FlexSIG and groupware applications in general will be discussed in detail in the next chapter (Chapter Five).

4.2.2 Statement of the Problem

The problem identified is that the FlexSIG model and tools are not sufficiently secure in terms of protecting their resources and the security of the information flow on the network. Therefore, the objective of this research is to design and build a groupware system that provides security services to secure its resources from unauthorised outsiders. These services allow members of a distributed group performing software inspection to be confident with the security of the information stored and flowing on the network. Another objective that this research intends to achieve is to build a secure groupware system such that its securities mechanisms are transparent to the user(s).

The groupware system aims to provide a system that will improve acceptability of groupware and provide a software inspection tool that is secure and transparent.

4.2.3 Purpose of the Study

The purpose of this study is to develop a Secure Software Inspection Groupware (SecureSIG) system that includes:

- the development of a secure software inspection model based on FlexSIG software inspection groupware model
- an implementation of a prototype based on the model
- the evaluation of the prototype by users in order to measure the suitability and transparency of the system

What it meant by secure in this research is that the system:

- Provides a secure access control mechanism to protect against an unauthorised outsider gaining access to or participating in a groupware inspection system.
- Provides communication security by securing the information flow between the user and the system or vice versa.
- Provides protection to the information (databases and documents) stored that are used in the groupware inspection system.

The security services used to provide security can be achieved using the security mechanisms which include encryption mechanisms, authentication mechanisms, data integrity mechanisms, data confidentiality mechanisms, and access control mechanisms (ISO, 1989). Recommended security mechanisms and algorithm to be implemented to FlexSIG will be discussed in detailed in Chapter Five.

Furthermore, the security services developed are intended to be transparent to the user(s). Transparency for the user means they do not have any distraction in doing their task because of the security mechanism provided. For example, the user does not have to repeatedly enter the cryptographic key to perform encryption processes which would interrupt the user and feel inconvenient to them in doing their work.

A model of SecureSIG is constructed based on FlexSIG (Sahibuddin, 1999) and security models (McGhie, 1994; IBM, 1997) discussed in Chapter Two with a combination of security services defines by ISO (ISO, 1989) discussed in Chapter Three. A prototype based on the model was built, using Internet and Java technology. In order to make sure of the effectiveness of the security services and the acceptability of the prototype, an evaluation is done on the prototype, using information gathered through a questionnaire completed by users.

4.2.4 Importance of the Study

Anyone working in an environment where computers have assumed an important role should be interested in computer security (Baskerville, 1988). Protection of the important resources that are controlled by a computer system is a concern of individuals, groups, or an organisation, and is an important issue because resources are very valuable. By accessing, tapping, or altering the data, an attacker can steal this valuable asset which can cause harm to individuals, groups, or an organisation.

Interest in CSCW has been increasing recently, and wider application has brought attention to the needs of security for CSCW applications. Several areas related to the security of CSCW has been done by researchers such as access control issues (Shen & Dewan, 1992; Kanawati & Riveill, 1995; Coulouris & Dollimore, 1994), secure group communication (Sakakibara *et al.*, 1994; Takizawa & Mita, 1993), and confidentiality security in CSCW applications (Foley & Jacob, 1995).

The survey done by Sahibuddin (1999) showed that security is needed in CSCW generally and specifically to the software inspection process, and as described earlier (Section 4.1) there has been little work on implementation of secure CSCW system.

This problem is addressed by this research, which seeks to design a CSCW system model and a prototype CSCW system that is secure.

This research will extend the current technology of software inspection groupware by making it secure. Someone who is using the system will be confident that unauthorised outsiders cannot gain access to the system and that information that will flow on the network will not be tapped.

4.3 RESEARCH DESIGN AND PROCEDURE

In this section, the research design and procedure are discussed in more detailed. The research methodology, research design, assumptions of the study, and the anticipated outcomes from the study are set out.

4.3.1 Research Methodology

The approach for this research involved the following activities:

- Carrying out a study on security services based on the cryptography techniques to be implemented on the model. This study was based on the literature review.
- Implementing the security services to the proposed model by using distributed software inspection process as a groupware application.
- Observing the way users use the system and evaluate the system using a questionnaire given to the test users.

4.3.2 Research Design

The research was conducted in three stages. These stages are:

- Developing the model for secure software inspection process.
- Developing the prototype based on the model.
- Testing the system to evaluate the acceptability and transparency among the user.

The first stage was the review of literature of on an existing software inspection groupware tools, especially the FlexSIG. The FlexSIG model was

extended to provide security to the model. The model produced was used to design the prototype. Chapter Five discusses in detailed the development of this model.

In the second stage, the prototype was built to test the functionality of the proposed security services model. The implementation of the prototype was carried out using the enabling technology mention earlier. The development of the prototype is discussed in detailed in Chapter Six.

In the third stage, a group of people was chosen to test the suitability, transparency and security of the system. Data was gathered after collecting responses from a questionnaire given to the test users. This is discussed in Chapter Seven. After using the system, a questionnaire was given to the members of the group in order to measure the suitability and transparency of the system. To measure the usability of the prototype, the test users were requested to execute a sequence of tasks given to them, and then answer question based on their experience using the prototype. The results were analysed based on the opinion rating given by the users in the questionnaire, giving information on the suitability and transparency of the prototype, and thus the model. A basic statistical tool was used to analyse the result. Developing an experiment for the evaluation of the prototype is discussed in detailed in Chapter Seven.

4.3.3 Assumptions

There are a few assumptions that were made. In the development of the model and prototype phase:

- The secure process extended in this research used some of the FlexSIG model process, namely, the initiation, briefing, individual inspection, synchronous, and asynchronous processes.

- It was assumed that weak and semi-weak keys can be ignored in the DES and IDEA block ciphers in the implementation. This was to avoid the weakness of these block ciphers as mention in Chapter Three.

In the testing of the prototype phase:

- It was assumed that the evaluation in an educational setting reflects the "real world" situation.
- It was assumed that the evaluation by a group of test users was sufficient in determining the effectiveness and transparency of the system.

4.3.4 Outcomes from the Study

Prototype system: The prototype system provides a security service to protect unauthorised outsiders gaining access to a groupware application, and also to secure information flow among group members and to secure the information kept in the system.

Subject and Data source: The subjects for this study are the test users who used the prototype system. The data sources for the study come from a questionnaire given to the test users after they have tested the prototype.

Evaluation: The success of an evaluation depends on the feedback of the test users. The data collected from the test users was analysed. The data indicate the suitability and transparency of the prototype.

Overall, the outcomes from this research provide the following:

- A model of a secure software inspection groupware.

- A secure software inspection groupware system that is provided with a secure access control mechanism to protect the system from unauthorised user gain access to the system and resources.
- A secure software inspection groupware that is provided with an encryption mechanism to provide a safeguard to the information stored.
- A secure software inspection groupware that is provided with an encryption mechanism to provide protection to the information flow to and from the system.
- A secure software inspection groupware that is provided with a data integrity mechanism to provide protection against undetected modification or alteration of information.
- A secure software inspection groupware that is provided with a digital signature mechanism to provide proof of origin.

4.4 SUMMARY

As a summary, Section 4.1 presented the relation between the area of interest to this research discussed in the previous chapters. The contributions from the literature review are highlighted.

Section 4.2 discussed the formulation of the research problem. The drawback of an existing software inspection groupware is pointed out. FlexSIG groupware software inspection mode and tools are discussed in more detailed because this research is based on it model and tools. Extension planned for security to FlexSIG model and tools are also presented. The aim, the purpose, and the importance of this research are also discussed in this section.

Section 4.3 details the issues of the design and procedure of this research

that includes the research methodology, assumptions that was made for this research, the outcomes from the study and the research procedures were discussed. This section also discussed the three stages involved in this research. The first stage is to develop a model for secure flexible software inspection process. The second stage is to develop the secure software inspection prototype, and the third stage is to evaluate the prototype. A basic statistical tool was used to analyse the result gathered.

The discussion of the development of the model is presented in the next chapter (Chapter Five).

Chapter 5

SECURESIG MODEL

5.0 INTRODUCTION

This chapter discusses the model of Secure Software Inspection Groupware (SecureSIG). The aim of the SecureSIG model is to provide security in terms of information flow, information stored and system access. This chapter is structured as follows:

- **Section 5.1** discusses the model proposed for Secure Software Inspection Groupware (SecureSIG).
- **Section 5.2** gives the conceptual framework for SecureSIG. The elements and security requirements considered in the development of the SecureSIG model are pointed out.
- **Section 5.3** discusses the SecureSIG functional model which include the components and services offered by SecureSIG functional model.
- **Section 5.4** discusses the recommended security mechanisms and algorithms to be provided to the SecureSIG model and components.

5.1 PROPOSED MODEL DESIGN

This section discusses the model proposed for SecureSIG. The model proposed is based on an extension of the FlexSIG model and the security model in the literature review. The model aims to meet the goals and objectives outlined in Chapter One and Chapter Four.

5.1.1 The FlexSIG Software Inspection Model

The FlexSIG software inspection model proposed by Sahibuddin (1999) is used for the reasons set out in Chapter Four (Section 4.1). The FlexSIG will be the basis of the development of this secure software inspection groupware (SecureSIG). The FlexSIG model is shown in Figure 5.1.

Only part of the process of the FlexSIG model process are used (shown as shadowed rectangles in Figure 5.1):

- Initiation process,
- Briefing process,
- Individual Inspection process,
- Synchronous process, and
- Asynchronous process.

In this research the process stops at the consolidation process because it is a single user process and was therefore considered less critical. Incorporating the consolidation process in the secured area is proposed for future work. The processes listed above form a complete path, and encompass the four quadrants of the Johansen (1988) space-time matrix. The processes form a usable core of activities in the inspection process. The choosing of briefing as the alternative

between kick-off and briefing is based on the survey done by Sahibuddin (1999), which stated that on-line briefing can replace the kick-off process.

Figure 5.1: FlexSIG Model (Sahibuddin, 1999)

5.1.2 The Proposed SecureSIG Model

The proposed model in this research which has been described by Doherty & Maarof (1997) is an extension of FlexSIG model proposed by Sahibuddin (1999) combined with the security model and security services defined by ISO (ISO,

1989) found in the literature. Most parts of FlexSIG model, the process, the roles, and techniques described are retained.

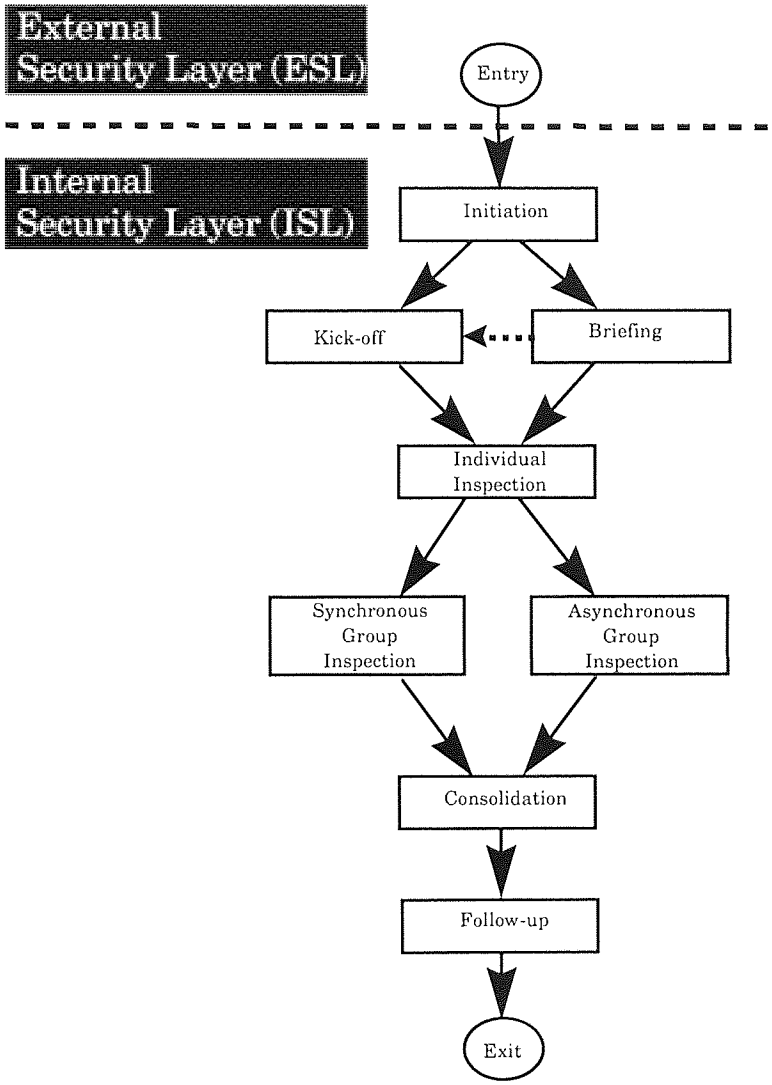

Figure 5.2: SecureSIG Model - External and Internal Security Layers

Two securities layer, the External Security Layer (ESL) and Internal Security Layer (ISL), was implemented on top of FlexSIG model to achieve the purpose of the proposed model (see Figure 5.2). The ESL provides a secure access control process, while the ISL provides secure information flow and information stored processes of the model. A new set-up process was added to the SecureSIG

model for generating public key pairs used for access control and encryption process related to the model.

5.2 CONCEPTUAL FRAMEWORK OF SECURESIG

This section outlines the framework for SecureSIG which consists of the elements needed to provided a secure environment for SecureSIG, security requirements for SecureSIG and the existing security models on which SecureSIG was based.

5.2.1 Considerations

Among the elements considered when constructing the SecureSIG model are: *system access control*, *communication security*, and *stored information confidentiality*. These elements are taken from Teufel *et al.* (1995), McGhie (1994) and security services defined by ISO (1989). These are summarised in the literature review.

System Access Control

A secure software inspection model should prevent unauthorised users from accessing or participating in the system. A model that includes system access from remote locations should have some form of access control (Sahibuddin, 1999). Access control should involve a combination of some form of entity identification with a rules-based permission system. An entity requesting access to a system or system resources should first be authenticated (e.g., its identity is verified) (McGhie, 1994). After authentication, a rule-based process determines whether or not this entity is allowed access to the requested resource. Thus, access control of the system must use different roles as one of it criteria (NIST,

1994). Shen & Dewan (1992), Kanawati & Riveill (1995), and Coulouris & Dollimore (1994) also consider the access control based on role. Team members with different roles should have different levels of access to the information stored and to the functions they can perform.

Authentication, data integrity, and confidentiality were necessary in system access control (IBM, 1997). Authentication verified the validity of the information being transferred between client and server. Data integrity protects the integrity of authentication information while being transferred. Confidentiality is required during the transfer of access control requests and responses and to protect the information in the database relating to system access control.

Communication Security

A secure software inspection model should ensure that the information flow is secure against being tapped or disclosed during the transmission. The need of secured (encrypted) information transmitted across public and broadcast networks has been raised by McGhie (1994), IBM (1997), and Hanka & Buchan (1996). The communications security concerns will be limited to the consideration of how to protect information when being transferred over communications networks.

Other than the confidentiality of information, the integrity and authentication of the information transmitted are communication security concerns. A model that is implemented in a distributed mode or environment must have some form of protection of the information flow. The information flow should be in scrambled form rather than in a plaintext form to avoid the disclosure of the information if it has been intercepted. Any information that is

transmitted from any team members in the software inspection process must be kept secret, and should be disclosed only to intended recipients.

Stored Information Confidentiality

A secure software inspection model must ensure that the information stored (databases or documents) is safe from being disclosed. The model must ensure that the information stored is in unreadable form and may only be accessed by authorised user. The information should not be kept in a plaintext form. This follows the suggestion by McGhie (1994) that there is a need to encrypt information stored on computer system.

5.2.2 Security Requirements for the Model

The security requirements are based on the secure distributed computing model by McGhie (1994), and SecureWay security model (IBM, 1997). These security requirements listed are considered to achieve the objectives mentioned in Chapter One (Section 1.5). The security requirements considered for the SecureSIG are as follows:

- Access Control.
- Data flow confidentiality.
- Data flow integrity.
- Data flow authentication.
- Database and dacument protection.
- All information is encrypted prior to transmission.
- Adaptable to a new encryption algorithm.

5.2.3 Existing Security Models

In Chapter Two, the security model proposed by McGhie (1994), and SecureWay (IBM, 1997) and their components for providing secure computing environments were discussed.

For access control both McGhie and SecureWay proposed an access control verifying that the individual is authorised to access or use specific resources and setting what kind of usage is allowed, based on roles and responsibilities. Access control in SecureSIG took McGhie's idea of primary and suporting components as a model for access control. One of McGhie's primary components was that password should not be transmitted in plaintext across unprotected or unencrypted networks.

In term of data flow protection both McGhie and SecureWay highlight the need for protecting the data. SecureWay proposed cryptographic services supporting security functions that provide data confidentiality, data integrity, and data authentication, while McGhie highlighted the need to encrypt information transmitted across public networks. Both these approaches were adopted for SecureSIG data flow protection.

For database and document protection, only McGhie highlights the need to encrypt information stored in the computer system, and her approach is adopted for this purpose.

In summary, both papers pointed out the need for providing access control services and secure information transfer services in their model. The need for securing the data stored has been raised by McGhie.

5.3 FUNCTIONAL MODEL OF SECURESIG

The conceptual framework discussed in the previous section serves as the basis for the functional model. As mentioned in Section 5.1.2, the SecureSIG functional model retained the processes of the FlexSIG model with addition of security based on the elements and security requirements considered in the previous section. As stated by Sahibuddin (1999) the functional model is described by examining its key components and services. The component and services offered by the SecureSIG functional model are discussed.

In both the FlexSIG and the SecureSIG model, there are three different user roles: the *moderator* who is responsible for planning the whole inspection process and leads the inspection team, the *author* of the inspected document who answers any query regarding the inspected document, and the *inspector/s* who point out any defect found in the document.

In SecureSIG, the moderator controls all aspects of security. The author and inspector/s can choose to generate or regenerate a key pair, but can do nothing else. Only the moderator is able to setup the group involved in the inspection process. In the *Setup* process, the moderator allocates access rights and can perform the key pair generation process as well as the encrypt data file process. Only the moderator is able to perform the encrypt data file process (database encryption) which encrypts all the databases used for user authentication in the SecureSIG system. In the *normal running* process, the moderator is able to perform regeneration of the key pair and the document encryption process which allows the moderator to encrypt all the briefing files and the source code used in the software inspection process.

The author and the inspector/s are permitted only to perform key generation during the *setup* process and regeneration of the key pair during

normal running process. Other capabilities of the moderator, author and the inspector are similar to FlexSIG (Sahibuddin, 1999) as discussed in Section 5.3.1.

5.3.1 Components and Services for SecureSIG

There are seven components and services of SecureSIG. These components are based on the FlexSIG (Sahibuddin, 1999) components with an extension of providing protection to all of these components and the database related to it. The components developed corresponds to the elements and security requirements considered for the model discussed in Section 5.2.1 and 5.2.2. Figure 5.1 shows the process model of FlexSIG. Extra components are provided for the SecureSIG model: the *set-up* component to generate participant's key pair and encryption all the documents and files used in the prototype. Other SecureSIG components are *secure access control*, *key pair generation*, *document encryption, database encryption, secure briefing*, *secure browsing*, and *secure communication*.

Secure Access Control

Access control involves the granting of access to system resources. Shen & Dewan (1992) discussed the importance of access control in collaborative work. The access rights presented by Shen & Dewan (1992) provides the basis for SecureSIG access control.

Access control into the system is managed by the access control component by requiring team members to enter valid username and password. To avoid passing plaintext password from client to server, a message digest is send instead of the password (Knudsen, 1988). The information entered with

addition of time-stamp and random number is digested using a one-way hash function. The digested information is then embedded with username, encrypted time-stamp, and encrypted random number before being transmitted to the authorisation server. Upon receipt by the server, this information is broken into its component parts and all the encrypted information decrypted. The server uses the given username to look up the password from the encrypted access control database, then uses the given username, random number, time-stamp and the password it has just retrieved to calculate a message digest. This digest is compared to the digested information received from the client and thus determines the access level of the user. Different roles have different access levels. Menus presented to the user depend on their role. This means that certain functions and services are not available to certain users.

Key Pair Generation

The key pair generation component allows participants in the team to generate the public key pair required for them to access and use other components in the SecureSIG system. There are two situations where key pair generation is available. The first situation must be done before the first access to the SecureSIG system. The second situation is after a successful first access to the system, when a new public key pair is generated, which replaces the public key pair generated before. Periodically regenerating the public key pair during the software inspection process reduces the risk of information being tapped and disclosured by a masquerade attack[10]. In other words, the exposure of these keys to potential compromise is reduced. For both situations username and passphrase are requested from the user. This passphrase is used to encrypt the

[10] A masquerade attack is one in which an attacker poses as a legitimate entity, in order to either gain access to the particular data belonging to the user or to the system in general.

private key generated to protect the private key from being disclosed to an attacker. Two distinct public key pairs are generated in both situations mentioned above when this component is activated.

Document Encryption

The document encryption component is used to encrypt all the documents used in SecureSIG. In SecureSIG documents are an asset, and it must be protected from disclosure. The documents should not be kept in plaintext. The function of this component is similar to McGhie's (1994) encryption and utilities and service component. The document encryption consists of two sub-components: encrypt briefing file and encrypt program file. Both of the components are only available to the moderator. The encrypt briefing file allows the moderator to encrypt the briefing file (see Secure Briefing below). The encrypted program file allows the moderator to encrypt all the source code to be used in SecureSIG system.

Database Encryption

The database encryption component is used to encrypt all the critical databases used in the SecureSIG system. This database encryption component is similar to McGhie's (1994) encryption supporting components in terms of the requirement to encrypt information stored in the computer system. Critical databases used for user authentication in access control are encrypted using this component, which should be available to the system administrator. The encryption of all databases used in the SecureSIG system should be done before the start of the software inspection meeting.

Secure Briefing

The secure briefing component is based on the briefing component of FlexSIG (Sahibuddin, 1999) with addition of security. Sahibuddin (1999) has presented the function of the briefing component. The briefing material is prepared by the moderator and is presented to the team members. Before it is presented to the team members, all the briefing materials are encrypted by a moderator to provide confidentiality to the materials. The need to encrypt these briefing materials stored on computer system has been pointed out by McGhie (1994). Users can select a briefing file from a list of files. A file needs to be decrypted before it can be browsed. A key is fetched from the server to decrypt the file. The information provided in the briefing components is necessary before the actual inspection phase can start.

Secure Browsing

The secure document browsing component is based on FlexSIG (Sahibuddin, 1999) with addition of security. This component contains two sub-components, which are the secure document viewer and the secure remote viewer. The secure document viewer provides the facility to browse the encrypted inspected document (source code) while the secure remote viewer provides the facility to browse remotely document inspected by other team members.

All the documents in these two sub-components are prepared by the moderator and presented to the team members, after having been encrypted by the moderator to ensure confidentiality of the document.

The first sub-component allows team members to choose and browse the inspected document. Users can select a document to be inspected from a list of files. By performing the selection of a document from a list, a secret key used to

decrypt the document is automatically fetched from the server. In the second sub-component, users can browse the same document that other team members are currently looking at. A list of other team members is given by names. Thus, by choosing any name given from the list, a secret key is fetched from the server to decrypt the corresponding document which is currently being browsed by other users. Only the username of the user who is logged into the system is displayed in the list.

Secure Communication

The secure communication component is similar to Sahibuddin's proposal. The communication component contains two sub-components, namely, secure electronic mail and secure group chat.

The first sub-component, secure e-mail, allows users to send e-mail securely from the sub-component to other team members. To send an encrypted e-mail, a recipient public key should be selected from a key listing. The need for security of electronic mail has been raised by Hanka & Buchan (1996) and McGhie (1994). The second sub-component, secure group chat provides a secure chatting facility to the team members. A secure chatting facility allows team members of the group to chat between themselves in a secure environment by encrypting all the message flows to and from the chat component. The message in the secure group chat is encrypted and decrypted using the public key generated during the process.

All the message types in the secure e-mail or secure group chat are encrypted to provide confidentiality to the messages between the communicants. Integrity and authentication of the messages are also provided.

5.3.2 Functionality Not Provided in the Model

The following functionalities that are not provided in the SecureSIG system:

- Excluding members of the group from participating in the system. For example, if there is a moderator, 4 inspectors and the author involved in the software inspection group meeting, the model is not able to exclude any member(s) from the group during the software inspection meeting.
- Broadcasting information only to a certain user or certain number of users, rather than the full group. Any information broadcast will be displayed to all the members involved in the software inspection meeting.
- Detecting two users with the same user name attempting to access the SecureSIG system.
- Recovering and decrypting old version of a file. Only the current key is retained. Previous versions of documents are discarded along with their key and they are thus not available.
- Decrypting the file (documents and source code) outside the SecureSIG system.

5.4 RECOMMENDED SECURITY MECHANISMS AND ALGORITHMS

In this section, the recommended security mechanisms and security algorithms used to provide the security services to SecureSIG are discussed. These recommended security mechanisms and algorithms are the basis for the development of the SecureSIG prototype covered in this chapter.

5.4.1 Recommended Security Mechanisms

This section provides recommendations for specific security mechanisms to be used to provide security in the SecureSIG model and to be implemented into SecureSIG components. All the security requirements describe in the previous section (Section 5.2.1) can be grouped into three main areas, namely: *communications security*, *access control*, and *information stored confidentiality*, according to the elements considered in the development of SecureSIG model. The security mechanisms recommended for these three main areas are:

- Encryption is required for information transmitted or stored.
- A combination of secret key and public key algorithms is recommended.
- Two distinct public key pairs recommended; one pair for key transfer and the pair for digital signatures.
- The X.509 certification process is recommended for public key management.
- Confidentiality, data integrity, and digital signatures should be in access control and communication security.
- A combination of authentication system (password and cryptography) is recommended.

Recommended Communications Security Mechanisms

The use of Internet to transfer SecureSIG information indicates that encryption is required. A secret key encryption algorithm is recommended for the encryption of this data instead of public key encryption, as secret key encryption algorithms are typically faster in encryption and decryption execution than public key algorithms (Schneier, 1996).

Secret keys used for the encryption of SecureSIG information should be locally generated on a per-session basis because per-session use of these keys reduces the potential for compromise by exposure of the keys. This will ease the problem posed by key distribution of secret keying material.

A public key cryptographic algorithm is recommended for the transfer of per-session keys (symmetric key) between communicants. A combination of secret key and public key cryptography are used to provide optimum security (Knudsen, 1998): the relatively slow public key only suitable for encrypting small amount of information while the faster secret key cryptography is suitable for encrypting large amount of information. The public key cryptographic algorithm can also be used in creating digital signatures (via signature of a message digest from the entire message) to provide non-repudiation services and strong authentication. Cryptographic hash function and digital signatures are discussed in Appendix B.

All information transferred in a SecureSIG communications session should be digitally signed by the sender and this signature verified by the receiver. This provides data integrity for each SecureSIG data transfer.

Two different public key pairs are recommended in SecureSIG communication security, one public key pair is used for key transfer (transferring symmetric key) and the other public key pair is used for digital signature. Using two distinct public key pair enhances the security of the information (Schneier, 1996).

Recommended Access Control Mechanisms

The critical importance of authentication to access control dictates that a strong authentication system must be used. This system should be equally capable of

effecting strong authentication both locally and remotely. Additionally, it should take in to account the possibility that intruders may examine its components.

A combination authentication system is recommended for SecureSIG systems. This authentication should combine personal knowledge and cryptography. This combination avoids transferring of password from client to the server, also it provides authentication, integrity, confidentiality of the data transferred during the access control process. A password coupled with symmetric and asymmetric cryptography fulfils this requirement.

Recommended Information Stored Confidentiality Mechanisms

The critical nature of SecureSIG indicates that confidentiality of information stored is an absolute requirement. It is recommended that all documents and databases be encrypted to prevent information disclosure. A secret key (symmetric key) encryption algorithm is recommended for the encryption of these documents and databases.

Other Security Recommendations

Other general security measures are also appropriate for SecureSIG system is the use of a firewall. A firewall provides a measure of protection by providing a first level screening on traffic attempting to enter SecureSIG system.

5.4.2 Recommended Security Algorithms and Mechanisms

A SecureSIG system should not be tied to any specific security mechanism or algorithm; its design should be flexible enough to support multiple mechanisms

and algorithms to perform security functions. This allows a system to specify the algorithm it wishes to use for a given digital signature or encryption session. The implementation of a SecureSIG system must select algorithms for use from among the secret key algorithms, public key algorithms, cryptographic hash function, and digital signatures available (see Appendix B). The following are the recommended algorithms and mechanisms for use within SecureSIG systems:

- Symmetric Encryption Algorithm: DES and IDEA block cipher.
- Asymmetric Encryption: RSA.
- Session Key Exchange: RSA.
- Digital Signature: RSA using MD5.
- Message Digest: MD5.
- Data Origin Authentication Mechanism: Digital Signature.
- Key Management: X.509 Certification Hierarchy (asymmetric keys); Local generation and asymmetric encryption (symmetric keys); Passphrase-based encryption used for private key encryption; Self-certified public keys.

The rationale of the recommendations of the above security algorithms are given below (Schneier, 1996; Smid & Branstad, 1992; Stallings, 1995, Girault, 1991; Odlyzko, 1994):

Symmetric Encryption Algorithms

The Data Encryption Standard, or DES (NBS, 1977) and International Data Encryption Algorithm, or IDEA (Lai, 1992), are the recommended symmetric encryption algorithm for use in SecureSIG system.

DES Advantages and Disadvantages

DES has several advantages. Among the advantages is that a DES is mature and extensively analysed block encryption algorithm (Schneier, 1996). With this features DES is cryptographically strong cipher because it has been publicised for a long time and still not broken (Schneier, 1996). Furthermore, other features such as no one has demonstrated a fundamental weakness DES and DES has been endorsed by the U.S government while no other publicly available algorithm has ever been endorsed (Smid & Branstad, 1992) makes DES widely accepted and publicly available cryptoalgorithm today as discussed in Chapter Three (Section 3.1.1). DES also has the advantage, with a 56-bit key, of potentially being exportable under proposed new US Commerce Department export guidelines (algorithms with longer keys are not) (Stallings, 1995).

There is a known weakness of the DES algorithm. Among the weaknesses of DES as mentioned in Section 3.1.1 is the existing of the weak key, semi-weak keys and complement keys in DES but these known weaknesses of the DES does not limit the effectiveness of the algorithm (Pfleeger, 1989).

IDEA Advantages and Disadvantages

IDEA is relatively a new cryptographic algorithm, but it is seem to be the best and most secure block cipher algorithm available to the public at this time (Schneier, 1996). The advantages of IDEA as mentioned in Section 3.1.1 is that it design is based on the impressive theoretical foundation, it has longer key length (128 bits) than the DES, 128 bits, and it has withstood brute force attack[11] (Schneier, 1996).

[11] A brute force attack is one in which an attacker searches the entire cryptographic key space until the correct key is found.

Regarding the disadvantages of IDEA as mentioned in Section 3.1.1, there is a class of IDEA weak keys found by Daeman, Govaerts & Vandewalle (1994), but these keys are a small defect to IDEA due to the probability of generating one of these keys being very small (one in 2^{96}). Other than that, IDEA algorithm is a new and not extensively analysed (Schneier, 1996).

Asymmetric Encryption and Algorithms

The algorithm that recommended for general asymmetric encryption is the RSA cryptosystem (Rivest, Shamir & Adleman, 1978). Among the advantages of the RSA algorithm is that it has a long (20 year) history of scrutiny. Other advantage is that since it is proposed RSA has withstood extensive cryptanalysis (Schneier, 1996). Finally, the RSA algorithm is a complete cryptographic system, supporting both digital signature and confidentiality (Schneier, 1996). This means that it can serve multiple functions within a SecureSIG system, thus reducing overall system complexity.

There are some limitations of public key cryptography. Among the limitations is the computational burden they impose (Odlyzko, 1994). RSA is only used for special tasks where its unique capabilities are needed, such as key exchange, authentication, and digital signatures (Odlyzko, 1994). Another reason why RSA public key cryptography is not used more widely is because of patent licensing issues.

Message Digest, Digital Signature and Session Key Exchange Algorithms

The MD5 (Rivest, 1992) algorithm is the recommended message digest algorithm for producing digital signatures (Hash function and digital signatures are presented in Appendix B). Like RSA, it widely accepted and used. MD5 is specified as the message digest for use in RSA digital signatures by RSA Data Security, Inc., in its public key cryptography standards (PKCS) (RSA, 1993). As such, digital signatures based on MD5 and RSA are widely used, and many products exist that can verify them. This combination is the recommended digital signature combination for SecureSIG systems.

The MD5 hashing algorithm (Rivest, 1992) likewise is widely accepted and used. While its digest length is somewhat smaller than other available algorithms, the digest length (128-bits) is long enough for practical security. The algorithm itself does not appear to have serious weaknesses.

Various alternatives to the MD5 message digest exist; however, among these the Secure Hash Algorithm (SHA) (NIST, 1994) appears the best. The major drawback to the use of the SHA message digest is the fact that it is not widely used. Even though SHA dose not requires license (RSA does for commercial use), the fact that it is not widely used is sufficient to make MD5 and RSA the recommended algorithms for message digest and digital signature.

Data Origin Authentication Mechanisms

Data origin authentication (or message authentication) provides to the party which receives a message assurance of the identity of the party which originated the message. This counters the threat of masquerade attack (i.e., impersonation of the message originator) (Markovitz, 1994). The mechanism proposed for data

origin authentication mechanisms is RSA digital signature using MD5 message digests. A signature consists of a MD5 hash of the message which is encrypted with the RSA private key of the originator. Data origin authentication implicitly provides data integrity (Menezes, van Oorschot & Vanstone, 1997). The advantages and disadvantages of RSA and MD5 have been discussed in the previous sub-section.

Key Management

The mechanism recommended for asymmetric key management is the X.509 certification. Certification Authorities (CAs) certify identification and public keying information concerning system users. This is done by having the CA digitally sign an information structure consisting of user identification, public keying, and administrative information regarding a system user; the digital signature process irrevocably binds this information together in a structure called a certificate.

The use of the X.509 certification hierarchy with public key simplifies the management of symmetric keys as well. Simply put symmetric keys need not be centrally managed. They may be locally generated on an as-needed basis, encrypted with the public key of the intended recipient, and transferred along with the data protected by the key.

Self-certified public keys are used for public key cryptography management. Self-certified public keys do not require that the public keys are accompanied by a separate certificate to be authenticated by other users (Girault, 1991). Both the authority and user compute the public key, so that the certificate is embedded in the public key itself, and therefore does not take the form of a separate value. Self-certified public keys contribute to reduce the amount of

storage and computations in public key schemes, while private key are still chosen by the user himself and remain unknown to the authority (Girault, 1991). The private key is kept secret using the passphrase-based encryption (PBE) as a cipher key to encrypt and decrypt the private key. PBE is used because it is easier to manage the passphrase than a cryptographic key (Knudsen, 1998). The PBE with MD5 and DES (RSA, 1993) is recommended due to the MD5 and DES features given in the previous section. In this particular variant of PBE (*PBEWithMD5AndDES*), an MD5 digest is used to digest the passphrase, the digest value is then used as a DES key. Detail of this PBE approach is given in Appendix F.

5.5 SUMMARY

As summary, Section 5.1 presented the proposed model and elements considered for the constructions of the SecureSIG model.

Section 5.2 discusses the components and services offered by the functional model of SecureSIG. The functional model presented in this chapter is the basis of the functional architecture of SecureSIG, which is discuss in the next chapter.

Section 5.3 dicusses the security mechanisms and algorithms related to the SecureSIG model and it components and services. Recommendations of security mechanisms and algorithms to provide security to SecureSIG model and components are discussed in detail. The advantages and disadvantages of the recommended security algorithms have been pointed out. These recommended security mechanisms and algorithms will be the basis for the implementation of the components of SecureSIG prototype covered in the next chapter.

Chapter 6

PROTOTYPE OF SECURESIG

6.0 INTRODUCTION

The SecureSIG prototype was developed based on the services and components provide by the functional model discussed in Chapter Five. This chapter is structured as follows:

- **Section 6.1** presents the functional architecture of a Secure Internet-Based Groupware System based on the Internet and client-server architecture.
- **Section 6.2** discusses the general client-server architecture and the Internet-based client-server groupware system. It also discusses the architecture of the SecureSIG which includes the data transmission, data storage, key pair generation, access control, and user interface architectures. This includes the discussion of how the security mechanisms provide security to the SecureSIG system.
- **Section 6.3** describes the components of the SecureSIG prototype in detail, which includes examples of screen shots from the prototype.

6.1 THE SECURESIG PROTOTYPE

The SecureSIG prototype is based on the functional model developed earlier in Chapter Five. The purpose of the prototype is to implement the model discussed in Chapter Five in order to provide facilities to meet all the overall purposes of this research. The prototype aims to address the security issue in software inspection groupware. The development of the security services that provide the security to the SecureSIG is based on the security mechanisms describes in Chapter Three. The aspect of enabling technology and the security mechanisms are re-examined from the perspective of the SecureSIG prototype in this chapter.

6.1.1 Secure Groupware Architecture

To understand the overall architecture of the system, first, the client-server architecture has to be understood. The client-server system consists of three components: the *client*, the *server*, and *network connection* that links the client and the server as shown in Figure 6.1. A server is a process that is waiting to be contacted by a client process so that the server can do something for the client. A typical scenario is as follows (Stevens, 1990):

- The server process is started on some computer system. It initialises itself then goes to sleep waiting for a client process to contact it requesting some service.
- A client process is started, either on the same system or on another system that is connected to the server's system network. The client process sends a request across the network to the server requesting service of some form.

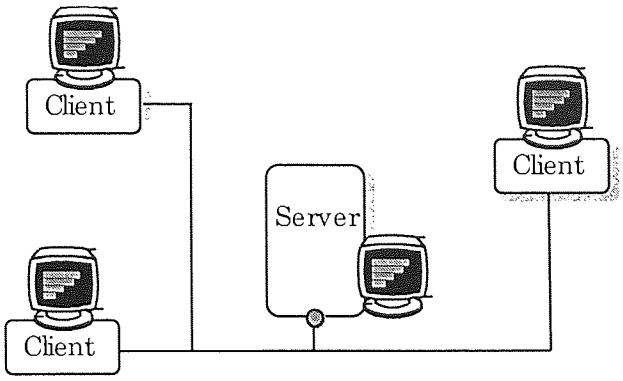

Figure 6.1: General Client-Server Architecture

The Internet-based client-server groupware system can be derived by expanding the general client-server architecture to operate in the Internet environment. In this architecture the connection between the client and the server is handled by the Internet instead of by a direct connection. Figure 6.2 shows the general Internet-based client-server groupware system.

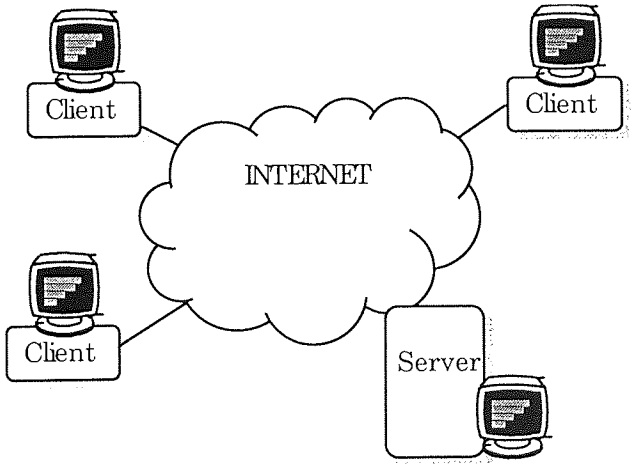

Figure 6.2: General Internet-Based Client-Server Architecture

Finally a secure Internet-based client-server groupware system can be derived by integrating the security services defined in Chapter 1 (Section 1.1)

with the system as shown in Figure 6.3. This architecture is based on the generic TCP-based Client-Server Security Model (LANL, 1996). This security for Internet-based client-server communications is based on the principles that unencrypted sensitive information should not be transmitted on the network.

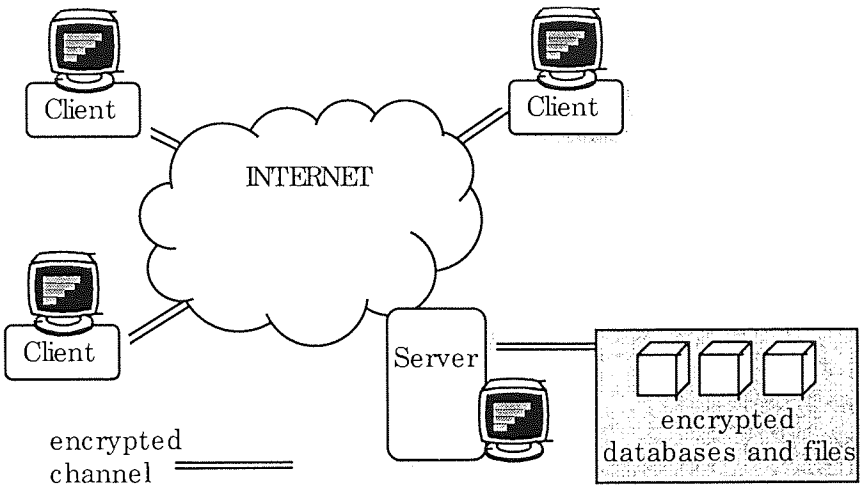

Figure 6.3: Secure Internet-Based Client-Server Groupware System

6.2 SECURESIG ARCHITECTURE

In this section, the SecureSIG architecture will be outlined. Based on the model described in Chapter 5 and the general architecture of the secure Internet-based groupware system in the previous section, the SecureSIG architecture can be illustrated. The members of SecureSIG, namely: the moderator, the inspectors, and the author(s), each connect to a server as a client. Figure 6.4 shows the overall architecture of SecureSIG.

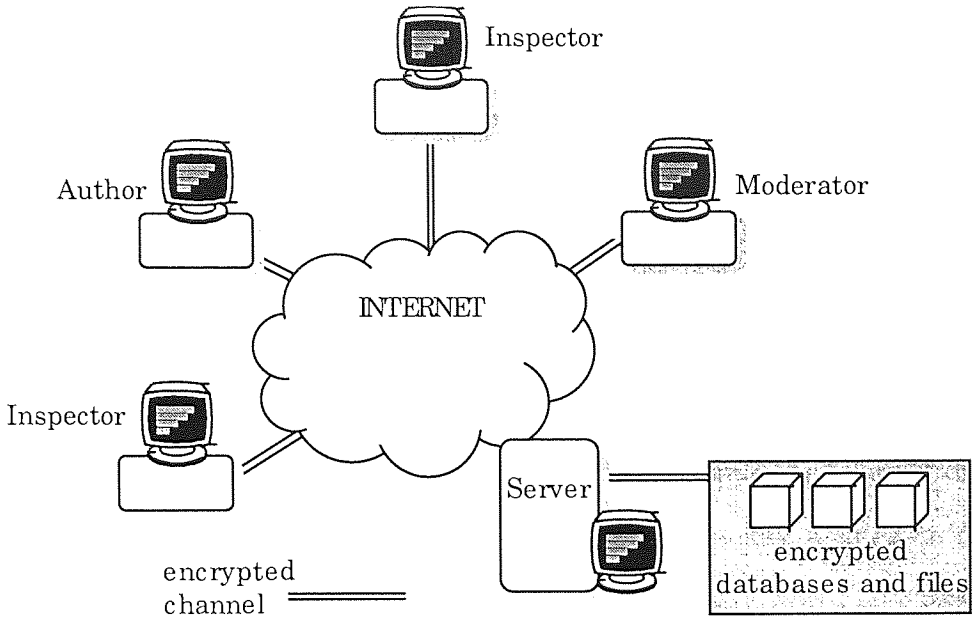

Figure 6.4: The SecureSIG Architecture

6.2.1 The Cryptographic Protocol

In this project, a public key protocol was required for the transmission and exchange of data (messages or keys) during a communication session between the client and the server. The RSA public key cryptosystem was used in this project. This protocol was invoked during the transmission of data and keys between the client and the server.

Public key cryptosystems are not efficient for encrypting large volumes of data, but combined with secret key cryptography they can be used to provide authentication, integrity and secrecy in an efficient manner (Markovitz, 1994). The following examples illustrate the idea.

A sender needs to send a signed, confidential message to a recipient. A sender first computes a digital signature as a function of the sender's private key and a digest of the plaintext message. Second, the sender generates a secret key, and uses this key to transform the plaintext to ciphertext. Third, the sender

encrypts the secret key using the recipient's public key. The sender finally appends the encrypted secret key and the digital signature to the ciphertext, and transmits the information to the recipient.

Upon receipt, the secret key is decrypted using the recipient's private key. The secret key is then used to decrypt the ciphertext. Once the plaintext is obtained, the recipients validate the message signature as a function of the signature and the sender's public key. Secrecy is guaranteed, because only the recipient's private key can be used to decrypt the secret key needed to decrypt the message. Integrity is guaranteed because a digital signature was generated using a digest of the original plaintext message. Finally, authentication is achieved, because the digital signature provides unforgeable evidence that the plaintext message was generated by the sender.

6.2.2 Data Transmission Architecture

The security of the information sent across the network during communication is a very big concern. The security of the data transmitted is ensured by encrypting the data at the client before it is transmitted and decrypting when the data reaches the server (or vice versa). A combination of symmetric and asymmetric cryptographic algorithm are used to provide this service. As recommended in Chapter Five, the IDEA block cipher and DES block cipher are adopted as symmetric cryptographic algorithm, while RSA is adopted as asymmetric cryptographic algorithm. This combination provides flexibility of the asymmetric cryptography with the speed of the symmetric cryptography (Hughes *et al.*, 1996). The IAIK-JCE provides the implementation of the symmetric and asymmetric cryptography mention above. To generate a key IAIK-JCE provides

the DES Key Generator and IDEA Key Generator to generate a DES key and the IDEA key respectively. To generate the DES and IDEA key, an application uses

```
KeyGenerator des_keygen = KeyGenerator.getInstance("DES");
SecretKey des_key = des_keygen.generateKey();
```

and

```
KeyGenerator idea_keygen = KeyGenerator.getInstance("IDEA");
SecretKey idea_key = idea_keygen.generateKey();
```

respectively.

Authentication and data integrity were handled by using digital signatures and message digest respectively. The recommended digital signature is the RSA using MD5 as mention in Chapter Five. The protocol to handled the authentication, integrity and secrecy in the data transmission is based on the protocol that was described in Section 6.2.1.

6.2.3 Database Storage Architecture

Sensitive documents, namely the program code, are an asset that should be protected. All sensitive databases are encrypted and securely stored. The keys used to decrypt the databases are only accessible through the server. Each server can access certain files only and different keys are used to encrypt and decrypt the files. Access to files also depends on the role of the client. For example, the moderator can access all the files while the author has no access to the file encryption component.

All the databases in the SecureSIG are encrypted either using DES or IDEA block cipher. All the keys generated are stored securely using the passphrase key, and can only accessible through the server as shown in Figure 6.5.

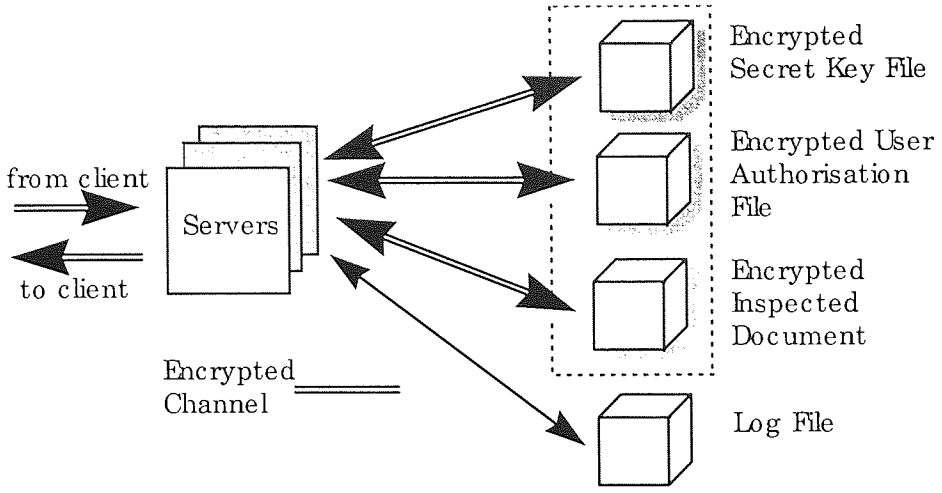

Figure 6.5: Database Storage Architecture

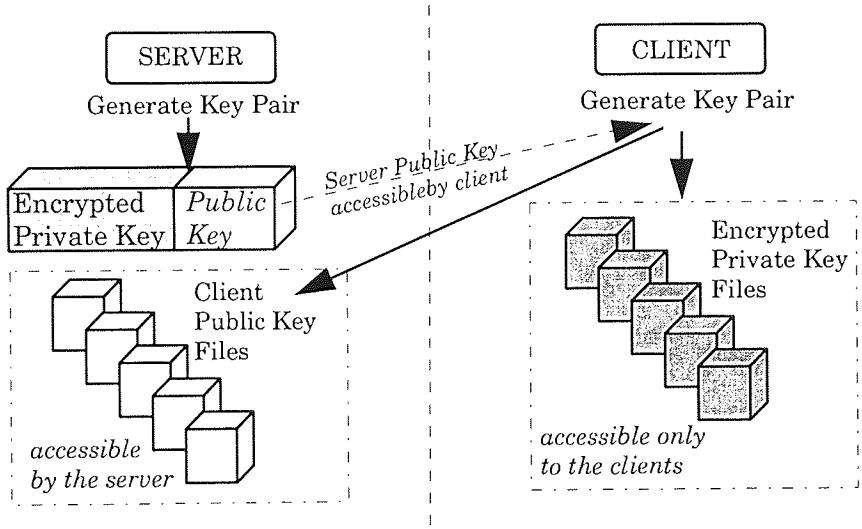

Figure 6.6: Key Pair Generation Architecture

6.2.4 Key Pair Generation Architecture

The server key pair is generated when the server providing user authorisation is activated. The client key pair is pre-generated before using the system. Both server and client use the self-certified certificates and passphrase key to store their public key and secret key. Two key pairs are generated for each user, one key pair is used to encrypt the secret key used to encrypt the message, and the

second key pair is used in digital signature and authentication of messages. A copy of both generated client public key pairs are kept at the server, and used for the encryption process when communicating with the client. This is to produce a client public key list at the server. The whole process is shown in Figure 6.6.

6.2.5 Access Control Architecture

The access control issue is handled by authorisation of the client during login into the system by the server.

The Client

In the login process, client is ask to enter a username and password (see Figure 6.7). This is follow by a request for the passphrase key. The plaintext password is not sent to the server. The username, password and a random number is digested then encrypted before being passed to the server program. The data is encrypted using the secret key generated by the client. The secret key is encrypted using the server public key fetched from the server before it is transmitted to the server (step 1). The client computes a digital signature as a function of the sender's private key and a digest of the plaintext message. The passphrase key is used to fetch and decrypt the pre-generated client private (step 2), the same passphrase key used to encrypt the private key when it was generated must be used. A wrong passphrase key entered results in the system displaying a warning message (step 3) and then exiting the user from the system. Otherwise, the message is sent to the server through the encrypted channel (step 4).

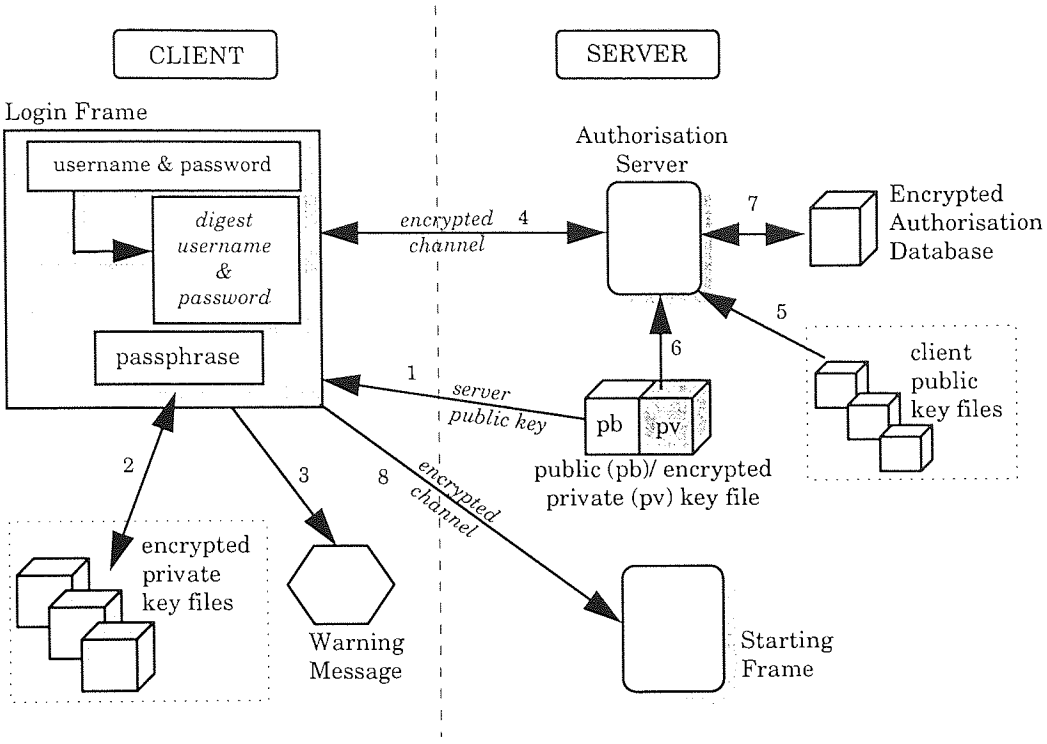

Figure 6.7: Access Control Architecture

The Server

Upon receipt (see Figure 6.7), the secret key is decrypted using the server's private key (step 6). The secret key is then used to decrypt the ciphertext. Once the plaintext messages is obtained, the server verifies the digital signature and validate the message digest using the client's private key (step 5). If either one of the checks fails, the server will send an error message through the encrypted channel to the client and display a warning message frame and ask the client to login again. A valid digested message received is compared to the digested message produced (calculated) from the authorisation file (step 7). The authorisation file is encrypted, only the server can access the file. When the comparison is valid, an encrypted success message is sent to the client and allows the client to activate starting frame (step 8). Otherwise, if the comparison is not valid a warning message frame is generated and displayed to the client. The

process of transmitting the messages to and from the client and the server is through an encrypted channel.

The access level of users with regards to the database files and actions that can be taken by the users can be dealt with by checking the roles of the client from the encrypted authorisation file.

In this process, the client used different key pairs as mention in Section 6.3.4). One key pair is used for message encryption and the other key pair is used to compute the digital signature and digest of the plaintext message.

6.2.6 User Interface Architecture

The SecureSIG user interface is based on the frame window interface. The interface architecture is based on the standard architecture of frame window interface. It is also based on the observation of the existing window interface architecture. The user interface (frame) displayed is based on the role of user and it is determined by the user authentication servers when the user logs in to the system. Further actions depend on the options chosen by the user. The options are presented in a button form. There are two possibilities that will be displayed when an options button is pressed: another frame is displayed or an information frame is displayed (see Figure 6.8). More than one option button which displays either a sub-menu or an information frame can be used at the same time, or closed without affecting the other component.

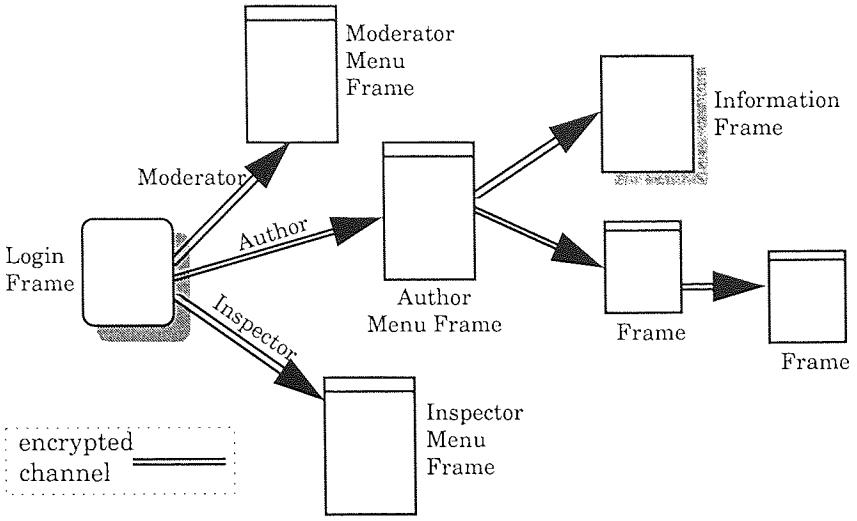

Figure 6.8: User Interface Architecture

6.3 COMPONENTS OF SECURESIG

This section describes the component of SecureSIG. Each component is describe according to it function. These components are set-up, access control, file encryption, key pair generation, briefing, document inspection, communication, and comment log. The briefing, document inspection, communication, and comment log components are taken from FlexSIG with an addition of security.

6.3.1 Set-up

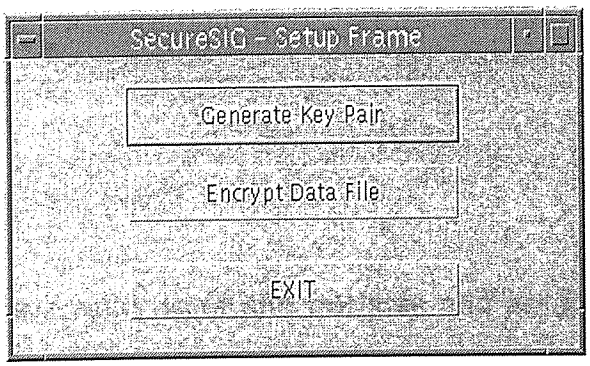

Figure 6.9: Set-up Frame

The set-up component consists of two options (see Figure 6.9). The first option, *Generate Key Pair*, is for all the team members or participants in the software inspection groupware and the second option, *Encrypt Data File*, is for the system supervisor.

The generate key pair option is used by all the participants in the software inspection process to generate their RSA key pair. The generate key pair frame is shown in Figure 6.10. The encrypt data file option is used by the system supervisor to encrypt authorisation file related to access control process in the system. The encrypt data file frame is shown in Figure 6.11.

Both of the options in the set-up component above must be generated (pre-generated) before accessing the SecureSIG. In a 'real world' implementation of the SecureSIG system, the set-up component would be implemented separately from the main SecureSIG system.

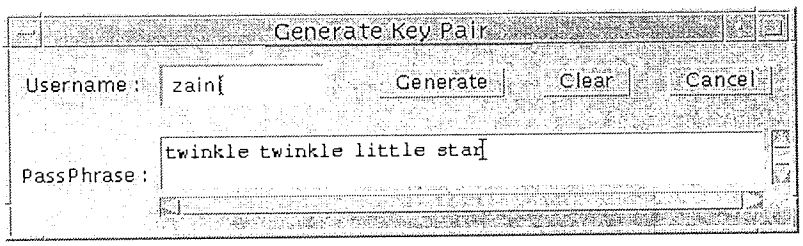

Figure 6.10: Generate Key Pair Frame

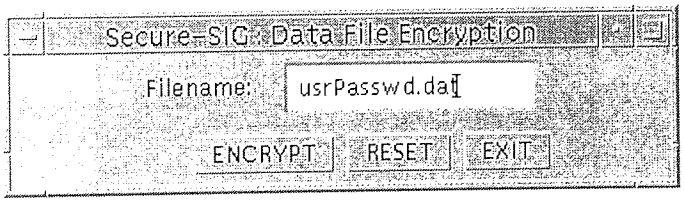

Figure 6.11: Encrypt Data File frame

6.3.2 Secure Access Control

The access control component authorises valid users and initiates and assigns access levels of the users according to their roles. Access to the system is handle by the login frame. The username, password and passphrase key are collected by the frame (see Figure 6.12 and Figure 6.13).

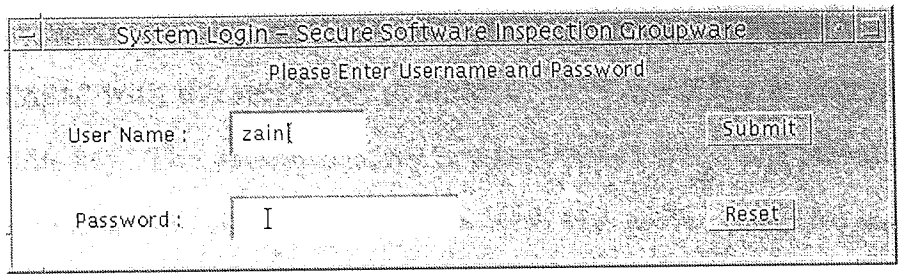

Figure 6.12: SecureSIG Login Frame

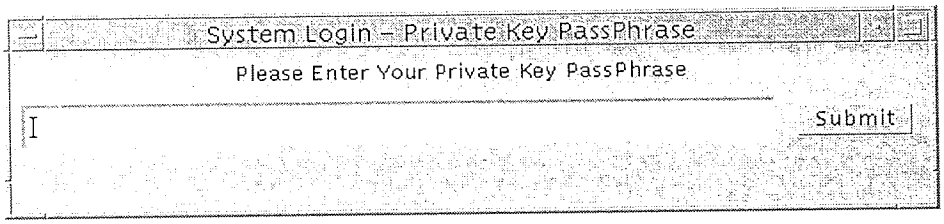

Figure 6.13: System Login Passphrase Frame

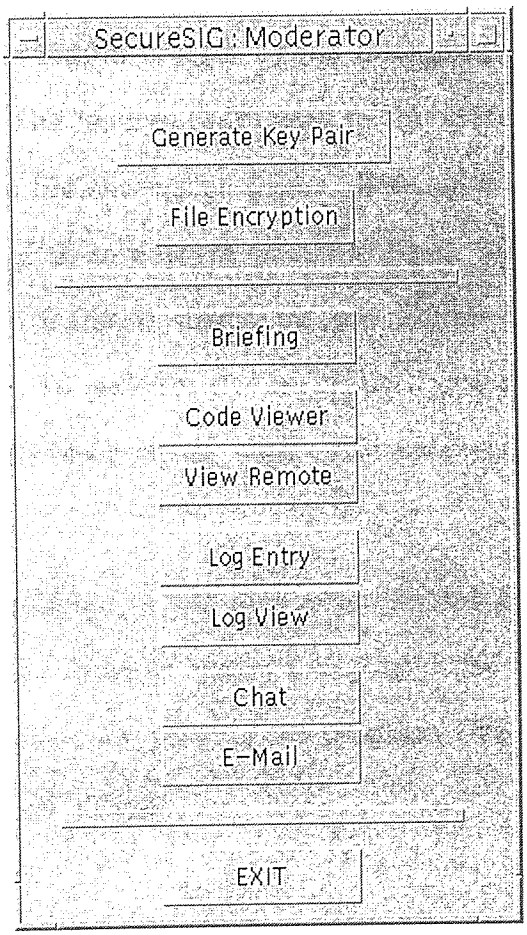

Figure 6.14: Moderator Main Menu Frame

The username and password are digested with a random number and then encrypted with the secret key generated. The secret key is encrypted using server public key. The passphrase key is used to fetch the client private key used in the encryption and decryption process with the server. Digital signature is used to sign the digested data. The server will verify the information from the authorisation file and send the confirmation back to the client. A frame based on the role of the user is displayed as shown in Figure 6.14. This frame is an extension of similar frame in FlexSIG, adding the top two buttons for the security functions.

6.3.3 File Encryption

The file encryption button appears only on the moderator frame, is only available to the moderator. This button is used by the moderator to encrypt the briefing files and the program files (source code) used in the system. The file encryption process is handled by a file encryption frame. Two options are given either 1) to encrypt briefing files or 2) to encrypt program files. If option 1 is selected, a briefing file encryption frame is displayed (see Figure 6.15). The filename is collected by the frame. To activate the encryption process the user presses the *Encrypt* button. A DES block cipher with CBC mode is used for the encryption process.

If option 2 is selected, a file encryption frame is displayed (see Figure 6.16). The process is the same as option 1 only here the IDEA block cipher with CBC mode is used for the encryption of the program. To activate the encryption process press the *Encrypt* button.

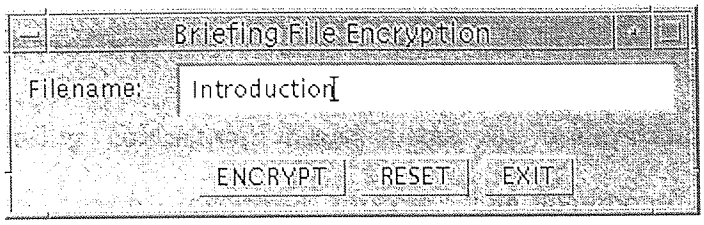

Figure 6.15: Briefing File Encryption Frame

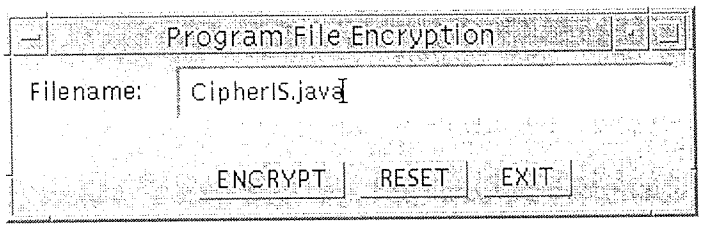

Figure 6.16: Program File Encryption Frame

Secret keys used to encrypt the briefing files and the program files above are protected by encrypting the key using the passphrase key by the server program.

6.3.4 Key Pair Generation

The key pair generation button generates a new key pair for the users. Key pair generation is handled by key pair generation frame. RSA is used for the key pair generation process.

The passphrase key frame is displayed to collect the passphrase key used to encrypt and decrypt the user private key generated. The process is the same as the key pair generation in Section 6.3.1.

6.3.5 Secure Briefing

The secure briefing component assist users in understanding and using SecureSIG in the software inspection process. The documents are prepared by the moderator and placed in the server. All the briefing documents were encrypted using the DES block cipher with CBC mode before being placed on the server. The secure briefing is similar to FlexSIG with an addition of providing secure fetching of a selected encrypted briefing file. The briefing document fetching involved the process of interaction with the server to fetch the secret key to decrypt the document. Using this key, the selected document is decrypted and displayed on the frame.

6.3.6 Secure Document Inspection

The secure document inspection produce two types of frames that allow browsing of the documents that need to be inspected: *Code Viewer* and *Remote Viewer*. The Code Viewer allows the participants in the SecureSIG system to select a document (program code) from a given list, while the Remote Viewer allows the participants to view the other users code that is currently being inspected. Both Code Viewer and Remote Viewer frames are similar to FlexSIG in terms of screen format and display, but the process of fetching the selected document is similar to secure briefing process described in Section 6.3.5. Using this key, the selected document is decrypted and displayed on the frame. The IDEA block cipher with CBC mode was used to encrypt and decrypt the document for both Code Viewer and Remote Viewer.

As mentioned above the code viewer frame has to interact with the server to fetch the secret key to decrypt the document. Using this key, the selected document is decrypted and displayed on the code viewer frame. The process is

also the same for the remote viewer process, the only different is that the document inspected is selected on the basis of username.

6.3.7 Secure Communication

The secure communication component can be divided into two sections; the synchronous and asynchronous modes of communication. The secure communication component is based on FlexSIG. The secure group chat is used in a distributed synchronous mode of communication and the secure e-mail is used in a distributed asynchronous mode of communication.

Secure Group Chat

Group chat is used to communicate in the synchronous distributed mode of communication. It provides the support for synchronous distributed group inspection. By using this group chat, the user can communicate immediately with the other team members, provided that the other team members activate the group chat frame on their screens as shown in Figure 6.17. This component is based on FlexSIG with the addition of a secure process of communication by encrypting the messages transmitted between the communicants.

Secure E-mail

Secure e-mail is used in the asynchronous distributed mode of communication. To send an e-mail, the user needs to press the *Compose* button. The recipient's e-mail address is entered followed by a subject and then the message. A recipient's public key needs to be selected from the *key* pull-down menu (see Figure 6.18). The *Send* button is pressed to activate the sending process. This public key is

used to encrypt the message before it is transmitted to the specific recipient. The encrypted mail is kept in the mail server program. In the read mail facility, the user can read their mail by pressing the Get button and the e-mail system decrypts the message using the recipients private key.

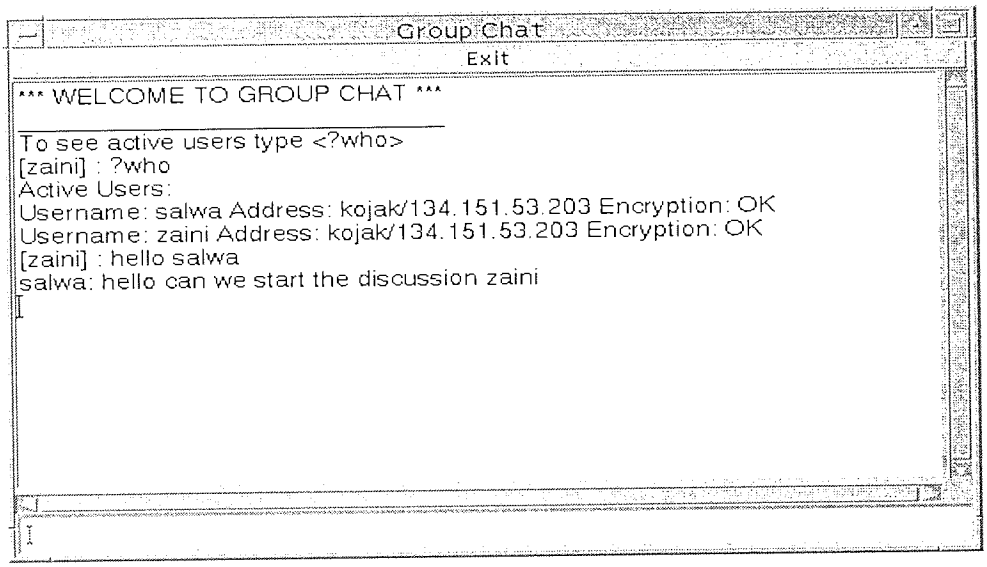

Figure 6.17: Secure Group Chat Frame

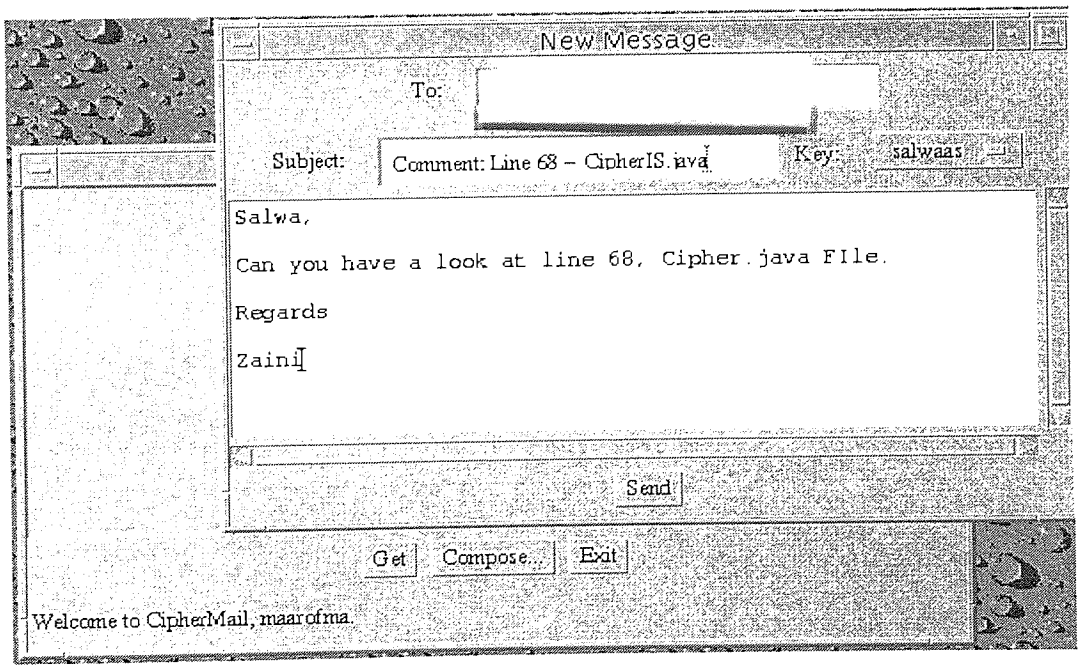

Figure 6.18: Secure E-mail Frame

6.3.7 Comment Log

The comment log is used for collecting team members comments, namely the defects, the queries, and suggestions. It is similar with FlexSIG comment log component with addition of encryption of process and project number stored.

6.4 SUMMARY

As a summary, Section 6.1 has presented the functional architecture of a Secure Internet-Based Groupware System based on the Internet and client-server architecture.

Section 6.2 presented the SecureSIG architecture, which includes the cryptographic protocol, the access control architecture, the key pair generation architecture, the file encryption architecture, and the database architecture.

Section 6.3 discussed the component of the SecureSIG prototype which includes explanation of the set-up, secure access control, file key pair generation, file encryption, secure briefing, secure document inspection, and secure communication component and comment log.

Chapter 7

USER EVALUATION OF THE PROTOTYPE

7.0 INTRODUCTION

This chapter presents the collected data analysis and results from user responses after using the SecureSIG prototype. This chapter is structured as follows:

- **Section 7.1** summarises the procedures of this research covered in chapter four.
- **Section 7.2** describes the procedures used to evaluate the prototype.
- **Section 7.3** discusses and analyses data collected from the questionnaire that formed part of the evaluation of the prototype.

7.1 REVIEW OF EVALUATION DESIGN AND PROCEDURES

The experimental design of this research was presented briefly in chapter four. The experimental design involved three stages. The first stage involved development of the model of the secure software inspection process. In the

second stage, based on the developed model, a prototype was developed using the enabling technologies described in chapter three. In the final stage, the prototype developed was evaluated, to find the suitability, transparency and security of the SecureSIG prototype, by a group of post-graduate students in computer science.

Data was collected from the users after using the prototype, using a questionnaire. Rating scales were used in the questionnaire and basic statistical analysis tools were used in analysing the gathered data to provide an evaluation of the system. Among the methods employed are calculation of the mean, median, standard deviation and percentage. The mean and the percentage are used to show the tendency or the inclination of the sample.

In the next section, the procedure for development of the user evaluation of the prototype is discussed.

7.2 USER EVALUATION PROCEDURE

As mentioned in the previous section, the purpose of evaluation is to measure the suitability, transparency and security of the prototype. To measure these features users were requested to execute a sequence of guided tasks presented to them and then answer a questionnaire based on their experience using the prototype. The reasons for providing the test users with guided tasks are as follows:

- The test users can be taken through all aspect of the SecureSIG system.
- There is a necessary sequence to activate SecureSIG, and the test users were required to follow this.

- The test users were fully informed of the operation of SecureSIG and this gave a more complete view on transparency and suitability of the system.
- Guided tasks are structured, making it quicker and easier for test users to complete the testing, giving them more time to concentrate on evaluation of the system.
- Unguided tasks test the robustness of the system more fully but this is not one of the purposes of testing the system.

To support this, an experiment was developed and is discussed in the following section.

7.2.1 Developing the Experiment

In this research, developing an experiment for the evaluation of the prototype involved four main activities (Abdullah, 1994). These activities were:

- Developing tasks for users to perform.
- Preparing questionnaires.
- Determining procedures for the evaluation session.
- Selecting test users.

Selecting the Tasks

The tasks were selected to represent functions supported by the system and to demonstrate the general capability of the system. The instructions to carry out these tasks were listed on paper in the order in which users were asked to

perform them. The instructions provided were brief and indicate what the users should do rather than how the users should do it.

Developing Questionnaire

Some guidelines regarding the questionnaire have been laid down by several researchers. Among the guidelines are (Nielsen, 1993):

- It is recommended to use a short questionnaire (single page or at least the two sides of a single sheet of paper). A short questionnaire stands a better chance of receiving attention from the users.
- Ask questions to which you really want to know the answer.
- The rating scale should be the same throughout the questionnaire.

Determine Procedures for Evaluation Sessions

The success of an evaluation depends on the feedback from the users. Users must be given a clear explanation of the system, the purpose of the evaluation, the importance of their participation and what is expected from them at the different stages of the evaluation.

The evaluation of the prototype can be divided into two main stages:

- *Executing tasks*: In this stage, users are required to execute the tasks that are assigned to them.
- *Answering the questionnaire*: Users are requested to answer the questionnaire based on their experience using the system.

In the first stage, the chosen group is requested to generate their RSA key

pair before accessing the system. They tested the system in two different environments.

- *Asynchronous distributed environment*: members of the group were physically and temporally distributed to test the system in different places and at different times.
- *Synchronous distributed environment*: members of the group were physically distributed to test the system in different places but at the same time.

Test users are given written instruction about the evaluation and its procedures. According to Hix & Hartson (1993) this approach will ensure consistency and remove some of the variances from test sessions.

Test users are requested to answer the questionnaire after they have executed all the test tasks, but they were allowed to access the system when answering the questionnaire.

Selecting Test Users

Test users are volunteers who help designers to evaluate the systems. It is important to indicate to representative users taking part in evaluating the systems that it is the system being tested and not the users. Since the prototype was developed in the educational environment, the assumption was made that test users in an educational environment act in a similar way to those in a non-educational environment.

7.3 DATA COLLECTION PHASE

Data collection is related to the evaluation of the prototype developed. Data collected from test users is used to contribute to the conclusion on the model and prototype of SecureSIG.

As mentioned in section 7.2, the evaluation of the prototype was divided into two main stages: *executing tasks* and *answering the questionnaire*. First test users were asked to evaluate the prototype based on the guidelines given. Then the test users were asked to complete a questionnaire to gather feedback from their experience of using the prototype. The participants involved as test users were a group of post-graduate students from School of Engineering and Applied Science at Aston University. The data gathered from test users was analysed based on the rating scaled used in the questionnaire.

7.3.1 Evaluation of the Prototype

In the evaluation of the prototype phase, the test users were asked to evaluate the prototype using the user task given. These consist of three main sections in the evaluation guidelines. The first section deals with the security related features, which include the key pair generation, accessing the prototype and file encryption. The second section deals with the asynchronous mode of communication of the prototype. The evaluations of the secure e-mail, secure document viewer and log entry component were carried out in the asynchronous mode. The third sections deals with the synchronous mode of communication of the prototype. The evaluation carried out in the synchronous mode was the secure remote document viewer component and the chat component.

The complete written instructions for evaluating the prototype are given in Appendix L.

7.3.2 Questionnaire

The questionnaire given is related to the evaluation of the prototype developed. As mentioned in Section 4.2.5 and Section 7.1, a rating scale was used in the questionnaire. The rating scale 1-5 used is based on the scale rating derived by Nielsen & Ley (1993) with 1 is taken as the best and 5 is taken as the worst. The complete test users detail questionnaire and questionnaire for evaluating the prototype is given in Appendix J and K respectively.

There are two categories of question given to the user: test user details and prototype related questions. In the user details category a few questions were asked. This included the personal background and experience of the participant in computer related fields and software inspection process. Examples of questions from this category are the participant's academic background, their gender, how many years of experience in computer related field, what areas they have experience in, and have they ever used any software inspection process tool.

In the second category, there are twenty one questions given to the participants. All the questions given deal with the evaluation of the SecureSIG prototype. Examples of questions from this category are:

- Do you think that the remote viewer that helps you in accessing the file viewed by other group member needs to secure it files by encrypting all the related files?
- When using all the components in the SecureSIG system are you aware the process of fetching the public key pair need for the encryption process?

Line spaces for writing any comment or suggestion for the betterment of the prototype were also given.

7.3.3 Analysis of Questionnaire on Evaluation

The analysis of the questionnaire is based on the categories mentioned in the previous section. Two categories of questionnaire were given to the test users: test user details and the prototype evaluation. Appendix M gives test users responses to these two categories of questionnaire.

Analysis of Test Users

Six test users participated in the evaluation of the prototype. The test users are the post-graduate students at the Department of Computer Science and Applied Mathematics, Aston University. All the test users have a degree in computer science or are attending a computer science program. All the test users are male. Almost all of them usually work on the UNIX platform.

In terms of experience working in information technology and computer related field, thirty percent of the test users have experience of more than seven years, fifty percent between four to five years, ten percent between three to four year, and ten percent less than two year. In the area experience by the test users, all of them have experience in programming. Twenty percent of the test users have experience in system analysis. Twenty percent of the test users have experience as network administrator. Only ten percent of the test users have experience in software project management. None of the test users have experience in data security. Ten percent of the test users have experience in computer security. Thirty percent of the test users have experience in network security and thirty percent of them have experience in cryptography. In the questionnaire, the test users could chose more than one category of experience. Figure 7.1 show the complete listing of the categories.

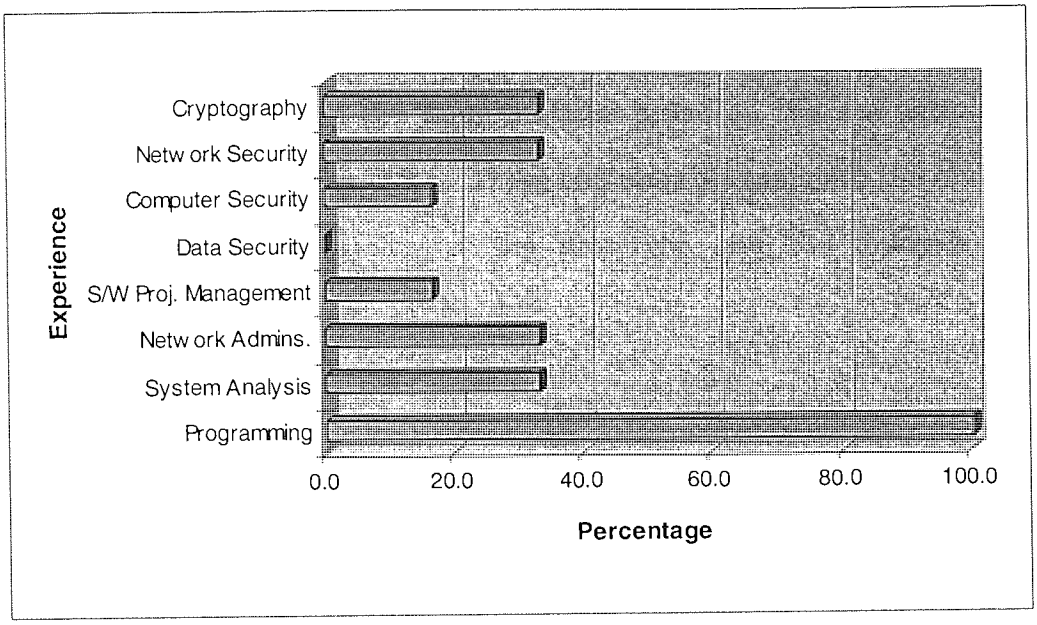

Figure 7.1: Respondent IT Experience

From the response gathered it shows that a larger number of the test users have experience in programming and some number of the test users have experience in areas related to network security and cryptography. From the response, only thirty three percent of the test users have experience in using any inspection or review techniques in software development. Thirty three percent of the test users said that they have used code inspection techniques. Only around seventeen percent of the test users said that they have used software audit techniques.

The information gathered has contributed to the evaluation of the SecureSIG. The information gathered was provided by test users with and without experience of information security. The suitability of the chosen test users is discussed in Chapter 8 (Section 8.2.5).

Analysis of the Prototype

The analysis is divided into two main sections based on the evaluation of the prototype discussed in Section 7.3.1. These two main sections are the analysis of the suitability and acceptability of the prototype and the analysis of the transparency of the prototype.

Suitability of the Prototype

As mentioned in Section 7.1, the suitability of the prototype are measured by analysis of the question asked of the test users concerning the degree of their acceptance that the security functionality provided to the SecureSIG prototype is appropriate to offer secure working environment. The analysis was done in both asynchronous and the synchronous modes. In the questions regarding the suitability of SecureSIG process all the question using the rating scale 1-5 with 1 is taken as *storngly agree*, 2 is taken as *agree*, 3 is taken as *not sure*, 4 is taken as *disagree* and 5 is taken *as strongly disagree*.

Asynchronous Mode of Inspection

In the analysis of the suitability of SecureSIG in asynchronous process there are seven questions asked. The summarised questions with the mean, median and standard deviation from the test users responses are shown in Table 7.1.

With regard to the need to protect all the briefing files stored and to secure the information flow related to the briefing process, around seventeen percent of the test users chose 1 (strongly agree) and around eighty three percent of the test users chose 2 (agree). The percentage and the mean (see Table 7.1) indicated that there is a need to protect all the briefing files store and to secure all the information flow related to the briefing process.

Questionnaire	Mean	Median	SD
The need to protect all the related briefing files stored.	1.83	2	0.41
The need to secure the information flow related to the briefing process.	1.83	2	0.41
The need to protect all the files view in the remote viewer process.	1.50	1.5	0.55
The need to protect all the information flow involved in the remote viewer process.	1.67	2	0.52
The need to protect information stored and flow in the electronic mail.	1.67	2	052
The need to protect information stored and flow in the log entry process.	2.17	2	0.17
Do SecureSIG components provided offer a secure working environment for asynchronous mode of process	1.83	2	0.41

SD – Standard Deviation

Table 7.1: Means for Questionnaire on Asynchronous Mode

The majority of the test users agreed there is a need to protect the files in the remote viewer process (see Table 7.1). Around eighty three percent of the test users chose 1 (strongly agree) and around seventeen percent chose 2 (agree). The mean from Table 7.1 and the percentage show that in asynchronous meeting, all the files related to the remote viewer need to be protected.

On the question of the requirement to protect the information flow involved in the remote viewer process, around seventeen percent chose 1 (strongly agree) and around eighty three percent chose 2 (agree). The mean from Table 7.1 and the percentage regarding this question shows that the need for protection of information flow in the remote viewer process is strongly supported by the test users.

Around seventeen percent of the test users chose 1 (strongly agree) and around eighty three percent chose 2 (agree) when they were asked the question

on the need to provide protection to the information stored and transmitted in the electronic mail process. The statistics gathered (see Table 7.1) supported that the test users agreed with the need for security of the electronic mail process in SecureSIG system.

With regard to the question on the need to protect the log entry process, all the test users chose 2 (agree). The percentage and the mean (see Table 7.1) gathered showed that it is required to protect the log entry process.

In summary, the analysis from all the questions asked above regarding the need of the secure components provided in the SecureSIG system indicated the suitability of the secure components provided for SecureSIG performing secure asynchronous mode of inspection (see Table 7.1). This analysis is supported by a asking the test users a specific question "Do you think that the SecureSIG components provided offered a secure working environment for asynchronous mode of inspection?. Analysis from this question show an agreement from the test users that SecureSIG offered a secure working environment for asynchronous mode of inspection (see Figure 7.2 and Table 7.1) with around seventeen percent of the test users chose 1 (strongly agree) and eighty three percent chose 2 (agree).

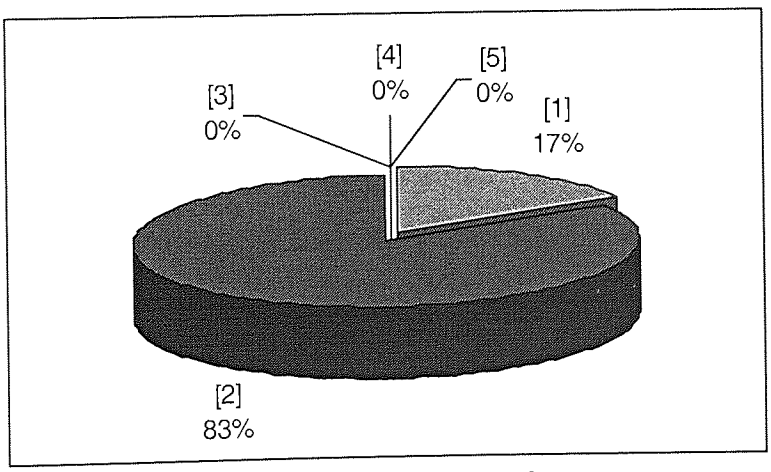

Figure 7.2: Test users respond regarding SecureSIG components providing secure environment to the asynchronous process

Synchronous Mode of Inspection

In the analysis of the suitability of SecureSIG in synchronous process there are six questions asked. The summarised questions and means for the questionnaire on the synchronous mode of inspection are shown in Table 7.2.

Questionnaire	Mean	Median	SD
The need to protect all the related briefing files in synchronous process	1.83	2	0.41
The need to secure the information flow related to the briefing process in synchronous mode.	1.83	2	0.41
The need to protect all the files view in the document viewer process.	1.33	1	0.52
The need to protect all the information flow involved in the document viewer process.	1.67	2	0.52
The need to protect information flow in the group chat process.	2.33	2	0.52
Do SecureSIG components provided offered a secure working environment for synchronous mode of process	2.00	2	0.63

Table 7.2: Means for Questionnaire on Synchronous Mode

On the questions regarding the need to protect all the briefing files stored and to secure the information flow related to the briefing process, around seventeen percent of the test users chose 1 (strongly agree) and around eighty three percent of the test users chose 2 (agree). The percentage and the mean from Table 7.2 shows that there is a need to protect all the briefing files stored and to secure all the information flow related to briefing process in the synchronous mode of inspection.

The majority of the test users agreed that there is a need to protect the files in the remote viewer process (see Table 7.2). Around sixty seven percent of

the test users chose 1 (strongly agree) and around thirty three percent chose 2 (agree). The statistics indicate that in synchronous meeting, all the files related to the remote viewer need to be protected (see Table 7.2).

With regard to the protection of the information flow involved in the document viewer process, around thirty three percent of the test users chose 1 (strongly agree) and sixty seven percent chose 2 (agree). The mean from Table 7.2 and the percentage gathered showed the need for protection of information flow in the document viewer process.

Around sixty seven percent of the test users chose 1 (strongly agree) and around thirty three percent chose 2 (agree) when they were asked on the question of the need to provide protection to the information stored and flow in the group chat process. Even though the perceived need regarding this question (see Table 7.2) is the lowest compared to other questions, it indicate that the test users still agreed with the need for security of the group chat process in SecureSIG system.

In summary, the analysis of the questions asked regarding the need indicated the suitability of the provisions in SecureSIG in the secure synchronous mode of inspection (see Table 7.2). As in the asynchronous mode this analysis is supported by a specific question asked to the test users "Do you think that the SecureSIG components provided offered a secure working environment for synchronous mode of inspection?". In responding to this question the majority of the test users agreed that SecureSIG provided the secure environment for the synchronous mode of inspection with seventeen percent chose 1 (strongly agree), sixty six percent chose 2 (agree), and seventeen percent chose 3 (not sure) (see Figure 7.3 and Table 7.2). This question is based on the need of the security functions rather than evaluating the security

mechanisms and algorithms used in each of the components in SecureSIG for the synchronous mode of inspection.

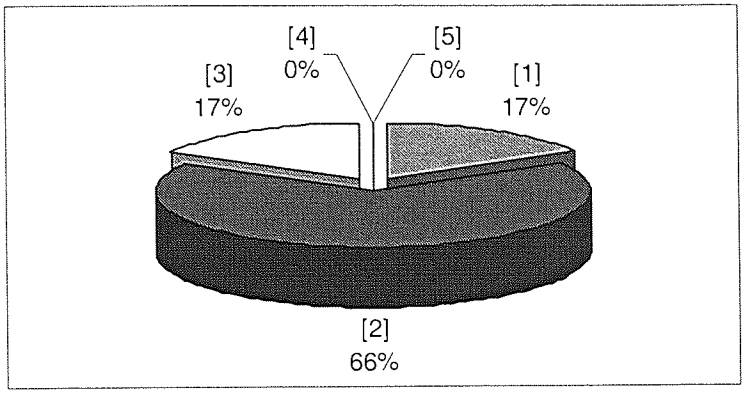

Figure 7.3: Test users respond regarding SecureSIG components providing secure environment to the synchronous process

With regard to a specific question asked of the user "Overall, do you think that with all the secure components provided the SecureSIG offered a secure working environment for an asynchronous and synchronous mode of inspection?, the result indicated that the majority of the test users agreed that all the secure components provided by SecureSIG offered a secure working environment for with asynchronous and synchronous modes of process (see Table 7.3 and Figure 7.4). Around seventeen percent of the test users chose 1 (strongly agree), around sixty six percent chose 2 (agree) and around seventeen percent chose 3 (not sure).

Questionnaire	Mean	Median	SD
Does SecureSIG system offered a secure working environment for asynchronous and synchronous mode of process.	2.00	2	0.63
The need of secure access control in SecureSIG.	1.17	1	0.41

Table 7.3: Means for Questionnaire on Specific Task

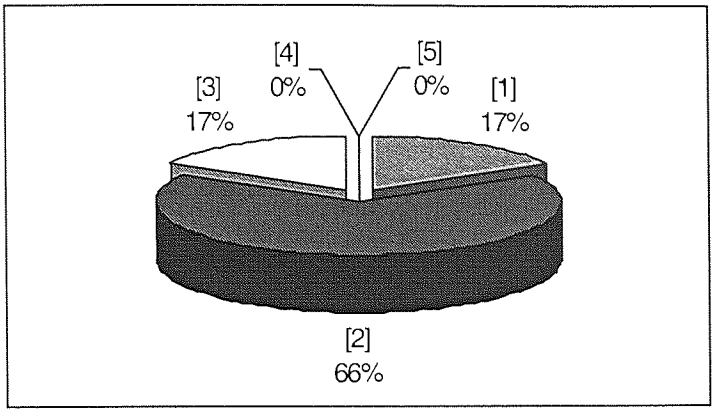

Figure 7.4: Acceptability of the SecureSIG providing secure working environement to asynchronous and synchronous inspection process

With regard to a specific question regarding the need for a secure access control to the inspection process system, all the test users agreed that the system must be provided with a secure access control (see Table 7.3 and Figure 7.5). Around eighty three percent of the chose 1 (strongly agree) and around seventeen percent chose (agree). This response indicated that the test users strongly supported the secure access control provided to the software inspection process.

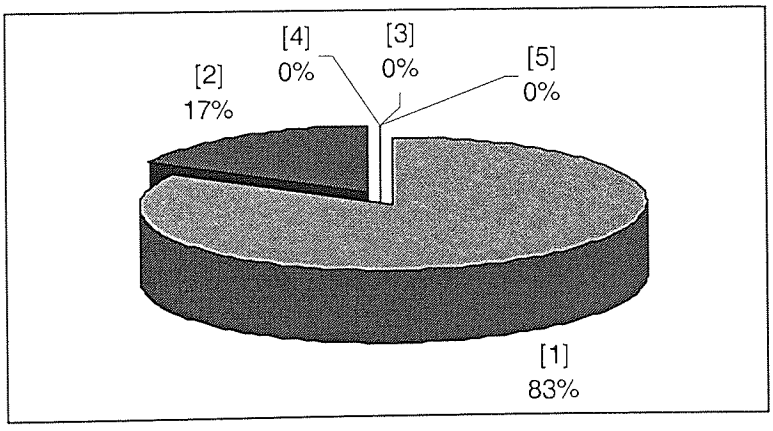

Figure 7.5: Respond regarding the needs of Access Control in SecureSIG System

Transparency of the Prototype

The transparency of the prototype is measured by analysis of the questions asked of the test users concerning the degree of their awareness that security mechanisms were operating and whether they were distracted from their work flow by the security functions in the SecureSIG system. In the questions regarding the transparency of the prototype almost all the question using the rating scale 1-5 with 1 is taken as *not aware*, 2 is taken as slightly aware, 3 is taken as *don't know*, 4 is taken as *aware*, and 5 is taken as *fully aware*. The summarised questions with it mean from the test users respond is shown in Table 7.4.

Questionnaire	Mean	Median	SD
Awareness of the encryption/decryption process during login process.	2.17	2	0.41
Awareness of the process of fetching the public key for the encryption/decryption process.	1.00	1	0.00
Awareness of the encryption/decryption process in each process of the SecureSIG component.	1.00	1	0.00
Awareness of the process of generating and updating the public key pair.	1.83	2	0.75
Accessiblity of files and databases during SecureSIG process.	1.00	1	0.55
The transparency of the encryption and decryption process in the SecureSIG system.	1.67	2	0.52

Table 7.4: Means for Questionnaire on Transparency of the Prototype

With regard to the question regarding the awareness of the encryption and decryption process during login to the SecureSIG system, around eighty three percent chose 2 (slightly aware) and around seventeen percent chose 3

(don't know). Even though the score is the lowest compared to the other questions (see Table 7.4), it indicated that test users agreed that they were not really aware of the encryption and decryption process. This result is expected because the login process involved many information exchanges as well as the encryption and decryption processes.

On the questions whether the test users were aware of the fetching of the public key during the encryption process and did they realise the encryption and decryption process were taking place in each process of the SecureSIG components, all the test users agreed that they are not aware the encryption and decryption process in both of the questions (see Table 7.4). All (100 percent) the test users chose 1 (not aware).

Around thirty three percent of the test users chose 1 (not aware), fifty percent chose 2 (slightly aware), and around seventeen percent chose 3 (don't know) when they were asked on their awareness of the process of generating and updating the key pair. The statistic confirmed that the test users were not aware of the encryption and decryption process (see Table 7.4).

The test users gave a positive answer to the question of the accessibility of the files and databases every time they needed it. In this question a different rating scale is used, 1 is refer to *highly accessible*, 2 is refer to *easily accessible*, 3 refer to *neither good or bad*, 4 refer to *not easily access*, and 5 is *difficult to access*. All the test users responded that they can access all the files when they needed, all (100 percent) test users chose 1 (highly accessible).

With regard to a specific question regarding the transparency of the encryption and decryption processing in SecureSIG system (see Figure 7.6). In this question a different rating scale is used, 1 is refer to *very transparent*, 2 is refer to *transparent*, 3 refer to *not sure*, 4 refer to *slightly transparent*, and 5 is *not transparent*. Around thirty three percent of the test users chose 1 (very

transparent), and around sixty seven percent of the test users chose 2 (transparent). This result indicated that the majority of the test users agreed that the SecureSIG system is transparent to them.

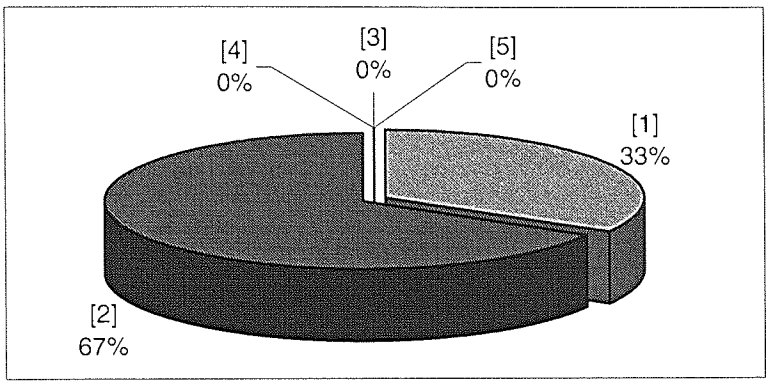

Figure 7.6: Transparency of the SecureSIG System

Analysis Summary

The analysis of test users indicated that test users with a background of network administrator and a security related background (cryptography, network security, and computer security) showed more concern toward security issues. For better analysis and result, test users should be taken from a wider range of backgrounds, particularly potential users and is discussed in Chapter 8 (Section 8.2.5).

The prototype analysis results indicated the suitability of the SecureSIG system to offer a secure working environment for asynchronous and synchronous mode of software inspection process. Based on the results gathered on the suitability of the prototype in synchronous (see Figure 7.2 and Table 7.1) and asynchronous modes of inspection (see Figure 7.3 and Table 7.2) the results show that the asynchronous mode of inspection is thought to be more suitable for software inspection process. The results from the analysis (see Table 7.4 and Figure 7.5) also indicated test users were satisfied with the transparency of the

SecureSIG system. The results also supported the need for and the suitability of the secure access control in SecureSIG system.

Overall, the result from the analysis of the prototype supported the suitability of the SecureSIG system for offering secure asynchronous and synchronous working environment for software inspection process and it also confirmed the transparency of the system.

In the suitability of the prototype questionnaire there is only one question asked regarding the acceptability issue in every category of the questionnaire (see Table 7.1, 7.2, and 7.3). This question (the "Do..." or "Does...") is a confirmation question asked to get confirmation from the test user regarding their degree of acceptance toward the SecureSIG system. It avoids overlapping questions regarding the suitability and transparency of the SecureSIG system. It is also a leader to provoke the test user to make their comments toward the end of the questionnaire.

Another issue that is not raised here is the testing of the robustness of the SecureSIG system. This is discussed in Chapter 8 (Section 8.2.5).

The evaluation supported the suitability and the transparency of the SecureSIG system. The test users were positive about the security, although the security was not fully tested. Deeper testing of the security was not considered necessary because:

- SecureSIG system using established secure methods, and
- most known cryptographic attack need processing power and time, due to the nature of the software inspection meeting which not exceeded two hours, SecureSIG has advantages on most of this known attack.

Resistance to known attacks is discussed in Chapter 8 (Section 8.2.6).

Comments

Among the comments provided by users are the following:

- Java is a developing programming language and Java is still not stable. Since the development of Java, many version of Java has been released and are inconsistent in the features supported.
- It is difficult to be confident of the security of the system by just using the components of the system.
- Some of the components in the SecureSIG are slow to activate.

Suggestions

Among the suggestions given are as follows:

- SecureSIG should provide a user with flexibility in choosing the encryption algorithm for different data in the system.
- Suggestion to rebuild the SecureSIG system using the World Wide Web as a platform.

These useful suggestions are the only suggestion given and are not incorporated in this research. They are proposed for future work.

The responses to this questionnaires address the need for and suitability of the security functions for all the components, but do not seek to evaluate the security mechanisms and algorithms used in each of the components in SecureSIG. The question of how well the security function provided met the needs is proposed for future work. Simply activating all the functional components does not test the security. The real test is to attempt to break into the system. This testing requires a different set of users performing different

tasks. The testing has addressed usability with security in place rather than strength of the security.

7.4 SUMMARY

As a summary, the evaluation design and procedures have been presented. Four activities involved in the development of an experiment for the evaluation of the prototype were laid out. These four activities are selecting test users, selecting the tasks, developing questionnaires, and determine a procedure for evaluation sessions.

Data collection phase which is related to the evaluation of the prototype has been discussed. In this section the analysis and the evaluation of model and prototype of the SecureSIG was presented. Based on the responses gathered from the test users, the results indicated the suitability of the SecureSIG system as a secure working environment for software inspection process. The result also indicated the users were content with the transparency of the SecureSIG system. Overall the analysis of the prototype evaluation indicated the suitability and the transparency of the SecureSIG system.

In the next chapter (Chapter Eight), the evaluation of this research and future work will be discussed.

Chapter 8

RESEARCH EVALUATION AND FUTURE WORK

8.0 INTRODUCTION

This chapter discusses the evaluation of the thesis as a whole and points out several suggestions for future work. This chapter is structured as follows:

- **Section 8.1** highlights the aims of the research.
- **Section 8.2** provides the evaluation of this research.
- **Section 8.3** gives suggestions for continuing the project in future work.

8.1 AIMS OF RESEARCH

This section reiterates the aims of this research. The aims of this research were:

- to develop a secure software inspection model.
- to implement a prototype based on the model.

- to evaluate the prototype in order to measure the suitability and transparency of the system.

8.2 EVALUATION OF THE RESEARCH

This section presents the evaluation of this research, which includes the literature review, the formulation of the research problem, the model, the prototype, the user evaluation, and the overall evaluation of this thesis.

8.2.1 Literature Survey

An extensive survey of the literature was successfully completed. A large number of references, more than one hundred, from various sources such as refereed journals, theses, books, conference publications, indexed search on electronic libraries, as well as sources on the Internet. The scope of literature found was spread evenly, covering the main issue of CSCW and it related security area and the issue of the enabling technology and security services (see Figure 8.1).

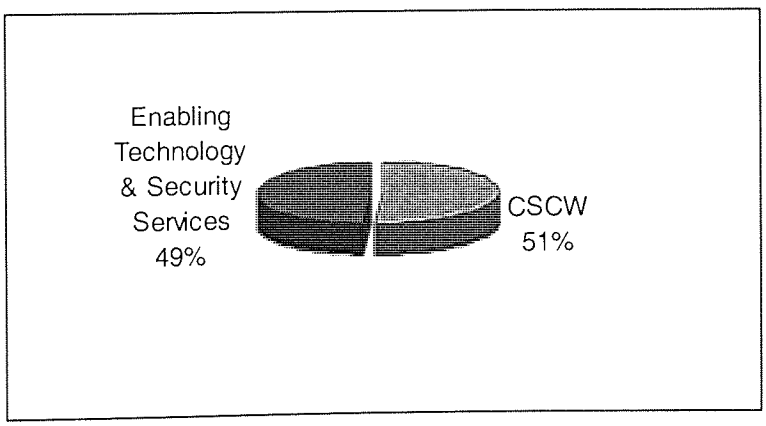

Figure 8.1: Literature Distribution

Many tools were used to gather all the literature, such as all the available searching tools and databases[1] in Aston University library, abstracts and index of theses, and Netscape Internet searching tools. The literature found was filtered according to the scope of this research. The literature in CSCW covered topics starting from general issues of CSCW and moving toward security areas related to CSCW before focusing on the existing software inspection groupware and the lack of security in this software inspection groupware process. The literature survey also covered security services and the enabling technologies, namely, cryptography, Java technology, and the Internet. Detail of the literature distribution using these areas is shown in Figure 8.2.

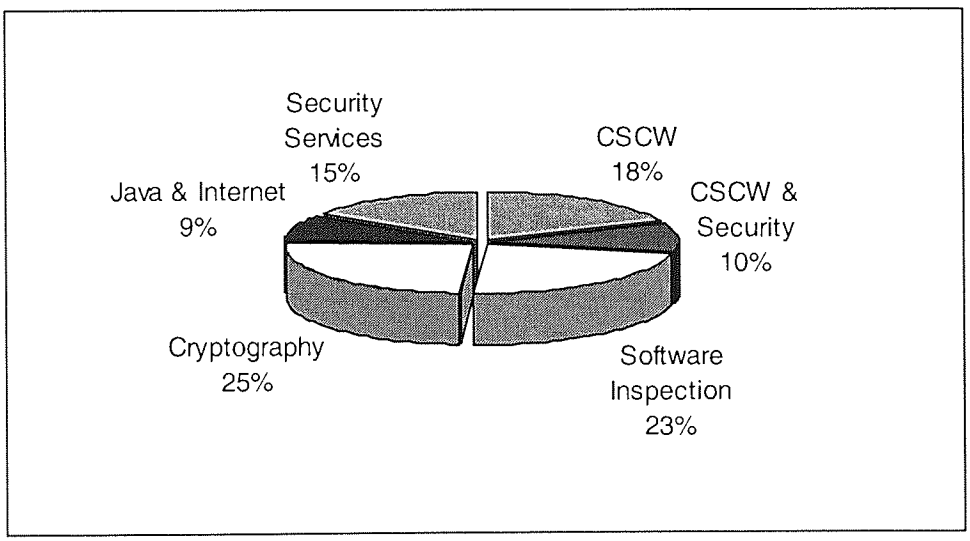

Figure 8.2: Detail Distribution of the Literature

The wide range of sources of information ensured the survey found most of the related literature. This was supported by checking with the index of theses, proceedings of the latest conferences related to the topic, and cross-

[1] BIDS (Bath Information and Data Services) is a collection of bibliographic databases.
INSPEC is a bibliographic database, providing references and summaries of mostly journal articles and conference papers.
IDEAL® (International Digital Electronic Access Library) is an online electronic library.

referencing from literature on hand. The literature survey revealed a large coverage of current literature in some areas and also showed that some areas covered in this thesis have not yet been much reported.

8.2.2 Research Problem, Design and Procedure

The literature survey showed little integration of CSCW aspects of Software Inspection Groupware with information security. Although around twenty three percent of papers found discussed the topic of the software inspection process, none addressed security in the software inspection process. The need to add security to the software inspection process has been raised based on the survey done by Sahibuddin (1999). This confirmed the work of Foley & Jacob (1995) and Teufel *et al.* (1995) who pointed out that much work in CSCW and it applications has been done on the technological aspects, but there is little focus on aspects of information security of CSCW and related applications. Only Teufel *et al.* (1995) covered, in general, information security concepts in CSCW, but they did not implement any solutions.

The research method was conducted in three stages, namely; to develop a model, followed by a prototype, and to perform user evaluation of the prototype. A one-shot case study was used for the user evaluation of the prototype. Even though the one-shot case study is not the best experimental method support for this, but it is chosen because of the limited availability of time and participants.

8.2.3 The Model

The purpose of model is to provide a secure software inspection process to use all the four quadrants of the Johansen (1988) space-time matrix and the time and

space taxonomy proposed by Ellis *et al.* (1991). A secure software inspection process model was constructed based on an extension of FlexSIG model (Sahibuddin, 1999), combined with a security model and security services based on information in the literature. Recognising the security limitations of FlexSIG and other existing inspection models, an extended security model was proposed. The model addressed the issue of lack of security in the software inspection groupware process and the evaluation of the model is based on this issue of providing secure data process to the model.

Two securities layer, the External Security Layer (ESL) and Internal Security Layer (ISL), was implemented on top of FlexSIG model to demonstrate the security ideas developed in this research. The ESL provides a secure access control process, while the ISL provides secure information flow and information storage processes of the model. Each security layer built with established secure technology and thus it is expected to be secure. The two security layers (ESL and ISL) are independent with no interaction between both of them, so if each of the layers is secure their combination is also expected to be secure.

To design the security model, the conceptual framework for the model was firstly identified, then the functional components identified, and finally the security mechanisms and algorithms proposed. The conceptual framework was the basis to the development of the functional components of SecureSIG. The security requirements identified for the SecureSIG model are mentioned in Chapter Five (Section 5.2.1), namely: secure access control, data flow confidentiality, data flow integrity, data flow authentication, database and file protection, encryption of all information prior to transmission, and adaptable to a new encryption algorithm. The security mechanism and algorithms adopted to implement the functional components were based on publicised and established

secure techniques which implies (Schneier, 1996) that they are cryptographically strong.

The model developed is considered secure because it provides secure access control process, secure software inspection process flow, and secure information stored which fulfil the security functional requirements mentioned above. The model developed is not tied to any specific security mechanism or algorithm, and the design is flexible enough to support multiple mechanisms and algorithms and to allow any new stronger algorithm to be adopted later.

The model could be extended with different views of users and key structure. The design of the model could possibly be improved further if the security requirements were enhanced by user feedback derived from the prototype and not merely based on the literature. Regarding the key structure a secret sharing schemes[2] could possibly be added to enhance the security of the access control process. Combining different secret key block cipher algorithms is another requirement that could increase the security of data in transmission and stored. These areas are put forward for future work.

Overall, the model is shown to provide a good basis for supporting secure software inspection processes by incorporating the two security layers (ESL and ISL layer) to enhance security to the process which is lacking in the existing software inspection model.

8.2.4 The Prototype

The prototype was developed to implement the model propose to prove its feasibility. What is meant by feasibility is that:

[2] In this scheme the secret key is broken up into small pieces and each piece is given to individuals from the selected trusted group so that a quorum of them need to contribute their shares to reconstruct the original secret or secret key. Without a quorum, the secret key would remain unknown.

- the prototype can be implemented and perform required tasks accordance to the security functional requirements mentioned in Section 8.2.3.
- the prototype components developed are fully functional.
- the prototype performance is satisfactory because all the component operate correctly together so it fulfils it tasks.
- the prototype is flexibly constructed to be easily adapted to new encryption algorithms.
- the prototype is flexible to work in the synchronous and asynchronous working modes.

The prototype was also developed to provide a test bed for user evaluation of the model in providing the security for the software inspection groupware process.

The prototype was based on the model described in Chapter Five and the general architecture of the secure Internet-based groupware system discussed in Chapter Six. The prototype developed is Java application based rather than WWW based because during the early development of the prototype Java applet security that support cryptography API for WWW which based on IAIK-JCE is not yet supported. Even though, Sun-JCE applet security is available during the early stage of the development but it is not available outside the United States because of the U.S export policy. Currently SecureSIG consists of about 15K lines of Java codes.

The prototype was developed using cryptography tools, Java technology and the Internet. Utilising Java programming language with the cryptography extension (JCE) handled the security functionality of the SecureSIG while maintaining the flexibility in terms of different working modes (distributed asynchronous and distributed synchronous modes).

The security mechanisms and algorithms provided in the SecureSIG tools met the aims of the design. The security provision in SecureSIG can be assumed secure because of the use of mechanisms and algorithms that are considered secure and their use in a security architecture which does not compromise their security. The algorithms used are:

- DES and IDEA block cipher for symmetric encryption algorithm.
- RSA for asymmetric Encryption algorithm.
- RSA for session key exchange.
- RSA using MD5 for digital signature.

The strength of the algorithms listed above has been discussed by Menezes, van Oorschot & Vanstone (1997), Schneier (1996), Stallings (1995), Meir (1994), RSA (1993), Smid & Branstad, (1992), Biham & Shamir (1991) as mentioned in Chapter Three (Section 3.1.1) and Chapter Five (Section 5.3.2).

The prototype developed is thus considered secure because it was based on de facto standard, known and established algorithms mentioned above that have been put together to construct SecureSIG prototype. Architecture combination (Idris, 1995) developed does nothing to weaken, because independence of layer.

The prototype was able to generate all the required components in SecureSIG. The secure tools provided to the SecureSIG prototype met the purpose of the development. The encryption algorithm that built in object in Java makes it adaptable to a new encryption algorithm. Different modules that implemented different encryption algorithm were tested in the prototype and proved successful. The development of the prototype could possibly be improved further if the key quorum idea mentioned in the previous section were

implemented in the prototype. Furthermore, developing the prototype on more a flexible platform such as WWW and providing flexible choice of algorithm by the test user could also improve the SecureSIG prototype.

Overall, the SecureSIG prototype developed is successful in providing the security missing from an existing software inspection tools because the development is based on established secure technology.

8.2.5 User Evaluation

The user evaluation on the prototype was conducted in order to measure the suitability and transparency of the prototype as mentioned in Chapter Seven. Two categories of questionnaire were developed, namely; test user details and the prototype evaluation. The information gathered from test users provides evaluation of the prototype from the viewpoint of two different groups of test users (with and without information security experience), and the test users evaluation of the prototype is discussed. As a whole, based on the mean and percentage of the tendency or the inclination of the test users, the result of the questionnaire's analysis indicated the suitability of the SecureSIG system in providing a secure environment for the software inspection process. The analysis also indicated the transparency of the SecureSIG system. The evaluation showed that the prototype met the user aims outlined in Chapter Four.

The design of the experiment for the evaluation of the prototype was based on the literature. The questionnaire developed was successful in extracting the information needed from the test users because it tested the suitability, transparency and security aspects, and because the comments from the test users gave useful suggestions that could have improved the design of the system. Test users selected were knowledgeable about computing and security, and made

no comment on the choice of question, implying the questions were appropriate. Nevertheless, the contents of the questionnaire could be improved further if a pilot study were conducted in order to check any bias in the design of the questions.

In the evaluation, six people were involved as test users. The user evaluation could have been done better with more people from a wider background involved in the evaluation. This would result in stronger statistical and experimental evaluation conclusions. Because of time and personnel constraints, a single experiment was conducted. A single experiment is limited because users evaluate, give comments and suggestions only once, and they don't see the prototype refinements generated from this input. A post testing was not done, but should be done because test users were naive when doing the first experiment, and with the experience that they have gained from the first testing, different and more concrete replies may be gathered.

The background of the test users affects their answers, so a range of user backgrounds gives a broader sample, and the result gained are more likely to be widely applicable.

In addition, the evaluation by the test users should include unguided task. Unguided task performed by the test users should test the transparency, suitability and the security of the SecureSIG system.

8.2.6 Security Against a Selective of Known Attacks

This section lists some known attacks from the literature and considers the ability of the SecureSIG system to stand against these attacks. Details of these known attacks are given in Appendix H. These attacks were selected because they are known to show a range of weaknesses in cryptographic system, and

were likely to be the most common in a distributed environment. The selected attacks also demonstrate two different categories of attack, the protocol-based attack (man-in-the-middle, replay attack, and ciphertext-only attack) and direct-attack (dictionary attack, brute force attack, and masquerade attack). SecureSIG contains known strong defences against each of the attacks. The ability of SecureSIG to withstand known attacks is summarised in Table 8.1.

Known Attacks	Ability
Man-in-the-middle	Yes
Dictionary Attack	Yes
Brute Force Attack	Yes
Replay Attack	No
Ciphertext-only Attack	Yes
Masquerade Attack	Yes

Yes - able to resist the attack.
No - Not able to resist the attack.

Table 8.1: SecureSIG – Security Against a Selective Known Attacks (See Appendix H)

Analysis Against a Selective Known Attacks

- *Man-in-the-middle.* The man-in-the-middle attack occurs when an adversary acts a third party in a two party conversation. The man-in-the-middle attack can be prevented using certificates. The SecureSIG system using public key certificates. By using the certified key pair (private key and public key) for security communication purpose, the man-in-the-middle attack can be handled by the SecureSIG system.

- *Dictionary Attack*. This attack is a general threat to all passwords. The hashed-password is not secure against the dictionary attack. By getting the hashed-password, an attacker can perform a dictionary attack by performing series of compution using every guest for the password. In the SecureSIG system, besides hashing the password, the password is also encrypted before being sent to the server for authorisation purpose. In other words, it is the encrypted hashed-password that was transmitted to the server. The encryption of the hashed-password secures the hashed-password from the dictionary attack because the attacker need to decrypt the hashed-password before the dictionary attack can be performed.

- *Brute Force Attack*. A brute force attack searches the entire cryptographic key space until the correct key is found. Brute force attack need processing power and time. In SecureSIG, block cipher DES and IDEA with 56 and 128 key size were used respectively. DES with 56 key size, needs a machine that tries all possible 2^{56} possible keys for a given ciphertext (Cheswick & Bellovin, 1994). For block cipher IDEA with 128 key size, the brute force attack will require 2^{128} tries to recover the key (Schneier, 1996). For the software inspection meeting which is suggested not to exceed two hours, it seems quite secure against the brute force attack. Other than that, the passwords are changed after each meeting.

- *Replay Attack*. SecureSIG cannot withstand the replay attack. SecureSIG has implemented only data origin authentication but not included the peer entity authentication. Because this function is excluded, an attacker (third party) can perform the replay attack by intercepting the message and replaying the message.

- *Ciphertext-only-attack*. A successful ciphertext-only attack is generally difficult. In the secureSIG system all the information transmitted and stored

is encrypted. Encrypted information is secure from the ciphertext-only attack. Other than that, time duriation of the meeting is short and the keys generated are changed after each meeting.

- *Masquerade Attack*. It is difficult for an attacker to perform the masquerade attack. The only legitimate user that the attacker can impersonate is through login access. The attacker has to provide the password and the passphrase used which provides two levels of security. Time duration of the meeting is short and the chance to test all possible combinations of the password and the passphrase is limited. The password and the passphrase are changed after each meeting.

8.2.7 Overall Evaluation

From the evaluation, it can be concluded that this research has met its aims. The evaluation has supported that this research has:

- developed a secure software inspection process model.
- implemented an effective prototype based on the model.
- implemented a suitable prototype system with good transparency.

The result also showed that the security and transparency of the system align with the definitions of 'secure' and 'transparent' in Chapter Four. The prototype was indicated to be secure and transparent in terms of the following:

- Allowed secured access to the system.
- Allowed the secured transmission of information in the system.
- Provided confidentiality to the documents, file, and databases.

- Allowed no awareness of the security mechanisms operating and distraction of workflow by the security function in the system.

Overall, it can be concluded that the system developed is secure, based on the implementation of the security mechanism and the use of algorithms that are considered secure, which have been put together in a way that does not introduce security 'holes'. The independence of the layers developed should avoid compromising the security of each layer when they are combined. This is supported by the attack analysis of Section 8.2.6.

Despite the successful achieving of the aims, there are problems and limitations inherent in this research. Some of the problems and limitation arose from the need to further research to make the research more manageable and some are caused by the limited availability of resources.

- In the development of the SecureSIG model, the basis of the development of the secure model of software inspections is based on the literature review. A 'real' case study should be used to refine security requirements of the model developed.

- The research would benefit from a more comprehensive procedure of evaluation, using a bigger group. In this research the evaluation is conducted using a small group of computer scientist with a little background of security. Futhermore, security benchmarking is needed for the prototype to validate it security performance. At present the system can only be considerable secure because of the use of established secure techniques in the implementation.

- In the implementation of the prototype the cryptographic key pair generated is stored in an individual file. In the ideal situation there should be one centre to generate the key pairs that produce the private and public keys.

The public keys could be kept in one database while the private keys could be kept in secure different place, or a portable place such as Smart Card.

- In the implementation of the prototype in this research, the encryption algorithm provided for the information flow is fixed by the system. Provision could be made to give users and the administrator flexibility in choosing their own encryption algorithm for transmitted and stored information according to their role.
- In the development of the prototype, the system only considers the confidentiality of the databases and document. In a real implemented system, the integrity of the databases and documents should be included.
- In the development of the model and prototype there are no audit services provided to keep a record of the system activity.
- The prototype only provides simple authentication protocols as the basis of access control. More secure authentication protocols are needed to provide more secure access control.
- In the development of the prototype, the system authentication process is based on personal knowledge (something we know). In a real environment a more secure authentication process is required. The combination of personal knowledge with something we have such as smart card, could provide a more secure system authentication process.
- The enabling technology used in this research is only limited to text. No audio or video was used to support the system. Also, no graphical format documents can be inspected.
- The different version of Java and JCE API release caused an inconsistency in the feature support.

8.3 FUTURE WORK

The research has opened up a number of possibilities for future work. The suggested list is provided below:

- There is much scope to include more advanced techniques in the secure access control system. An inclusion of a secret sharing schemes (Shamir, 1979; Blakely, 1979) for secure group access will open up a new paradigm for secure access control in distributed computing environment and CSCW.

- A combination authentication system that combines a physical token, personal knowledge and cryptography will provide more security to the system. Using a physical token such as smart card will provides a place to store and protected user private key data and provide portability of the key.

- Many other password authentication protocol such as SRP Protocol (Secure Remote Password Protocol) (Wu, 1998) and other available authentication methods can be explored with a view to their being implemented in secure system access for the CSCW application. SRP is a new mechanism for performing secure password-based authentication and key exchange over any type of network.

- The SecureSIG communication aspects are based on securing text. Expanding the system to support graphical documents will open up the area of image security. Furthermore, to improve the flexibility of the software inspection groupware process, the inclusion of audio and video in the communication suite should be provided. This inclusion will also open up to an area of audio and video security issues in the distributing computing environment in general and specifically to the area of CSCW.

- Currently the public key pairs for the users in the system are kept in an individual file. There should be one database at a central location that keeps

all the public key pairs for the users. This approach will make the handling of these public key pairs much more secure, easier to access and to manage.

- SecureSIG is limited to the confidentiality aspect of the databases and documents. Integrity, which another important aspect of security in databases and stored information, should be considered for future work. Furthermore, this work did not deal with random access databases. This can be further explored regarding their confidentiality and integrity.

- Currently there is no integration of any standard database management system with the developed system. The system utilises its own file system. The integration with a standard database management system allows easier access and timeliness in accessing documents. This also opens up the broader scope of database security related to CSCW applications.

- The system can be expanded to include audit utilities and services. Audit trails can keep a record of system activity both by system and application processes and by user activity of systems and applications. Audit trails can provide a means to accomplish several security related objectives, including individual accountability, reconstruction of events, intrusion detection, and problem analysis (NIST, 1994). In the distributed computing environment, there is an opportunity to consider exactly what audit and tracking information is required. Exploring this area will open up many areas of research related to security and CSCW.

- SecureSIG can be expanded by providing security to the consolidation and follow-up processes of the FlexSIG model.

- The prototype can be fine-tuned to include more flexibility in providing encryption services: flexibility in choosing encryption algorithm component is one example. This component should allow a user to chose their own

encryption algorithm based on the choice given to encrypt all the information transmitted in the prototype.

- The work can be expanded to others aspects of CSCW applications that require security support. The security requirements of the model in this research can be applied to other CSCW applications, using the same or different security classes, provided by other suppliers or by developing new security classes.

- Evaluation of the SecureSIG prototype is limited to the user evaluation. Security benchmarking for SecureSIG system to evaluate the security of the system should be extended. Further testing of the prototype in the real-life working environments is needed in order to examine the acceptability and to improve the model and the system.

- The current SecureSIG is Java application based. Further development could be implemented using the current available and implementable applet security (IAIK, 1998) to provide SecureSIG web-based platform. The development the web-based SecureSIG will provide easier implementation of the system.

8.4 SUMMARY

To summarise this chapter, the evaluation was done on every aspect of this research, comparing the research outcomes against the original aim. An extensive survey of the literature was successfully completed involved investigating various sources available. The SecureSIG model and prototype developed are successful in providing the security missing from the existing software inspection process and tools. The user evaluation indicated the suitability of the SecureSIG system for providing a secure software inspection

working environment. In addition, the transparency of the SecureSIG system was also proven.

The overall evaluation of this research concludes that this research has met its aims. Future work for further development of this research has been proposed. The proposals extend the level of security of the software inspection process. Furthermore, the proposals suggest new research areas in CSCW security.

The next chapter will give the conclusions of this research.

Chapter 9

CONCLUSION

9.0 INTRODUCTION

This chapter covers the summary of the previous chapter and presents the general conclusion.

9.1 LITERATURE

The literature was found by investigating various sources, such as journal, indexed search of electronic libraries, books, as well as sources on the Internet which provided an extensive source of relevant information. The literature survey revealed extensive coverage in CSCW, enabling technologies and security, but little on security applied to CSCW.

This research covered three main areas: Computer Supported Cooperative Work (CSCW), security services and the enabling technologies.

9.1.1 CSCW

The literature on CSCW concentrated on the technological aspects on CSCW rather than it information security aspects (Foley & Jacob, 1995; Teufel *et al.*, 1995). Lack of published research in the security aspects of CSCW has been confirmed by Teufel *et al.* (1995).

The security areas relevant to the CSCW applications were identified in the literature. There is no specific security model for CSCW but the distributed security models described in literature were adopted as a basis.

Specifically in the area of software inspection, the lack of security in existing software inspection was identified and the need to add security to the software inspection process raised (Sahibuddin, 1999).

9.1.2 Enabling Technology and Security Services

On the topic of enabling technology, the literature has identified that cryptography is the main tool to support security. The literature suggests adoption of established and publicised ciphers because they are cryptographically strong (Schneier, 1996). The only limitation is their 'expiration dates' which relate to their key length and the feasibility of an exhaustive search. The ability of Java technology to support security functionality on Internet has been identified.

Security services defined by ISO were adopted in this research, and established and published security algorithm was adopted to implement the security services.

Cryptographic key management, particularly key distribution, was implemented by self-certified public key distribution based on X.509.

9.2 RESEARCH PROBLEM, DESIGN AND PROCEDURE

The literature review showed that the security in CSCW has been little explore, so the purpose of this research is to investigate security in CSCW, based on the particular CSCW application of software inspection. The work provides a security extension to FlexSIG software inspection model while maintaining the flexibility, the process model and the roles of the participants involved in FlexSIG model. The security additions to the model seek to provide secure data transmission, to secure data stored and provide secure access control to the system.

Specifically, the aim of this research is to develop the Secure Software Inspection Groupware (SecureSIG) system which includes the development of secure software inspection process model, the development of a prototype based on the model, and the evaluation of the prototype in order to measure system suitability and the transparency.

SecureSIG provides security in the context of:

- The ability to provide a secure software inspection process to use all the four quadrant of the Johansen (1988) space-time matrix and the time and space taxonomy proposed by Ellis *et al.* (1991).
- The ability to use a secure tool across the Internet.

This research was design to be experimental and used the one-shot case study design. Three stages involved in the development of this research, with each stage correspond to a specific aim of the SecureSIG development mentioned above. Simple statistical method is use in the analysis, which includes, mean, median, and standard deviation.

9.3 THE MODEL

The development of the SecureSIG model is an extension of the FlexSIG model developed by Sahibuddin (1999) in combination with the security model and security services found in the literature. The model aims is to support a secure software inspection process to use all the four quadrant of the Johansen (1988) space-time matrix and the time and space taxonomy proposed by Ellis *et al.* (1991).

To support the secure inspection process two securities layers, the External Security Layer (ESL) and Internal Security Layer (ISL), were implemented on top of the FlexSIG model. The ESL provides a secure access control process, while the ISL provides the secure information flow and information storage processes of the model. The model developed provides two-levels of protections (external and internal) which are expected to be secure because each layer was built with established technology that is considered to be secured. The independence of the layers should avoid compromising the security of each layer when they are combined.

The secure process of SecureSIG consists of the Set-up phase, follow by secure briefing phase. The process flow merges again for the secure individual inspection phase. For the group inspection phase, there is a choice between secure synchronous and secure asynchronous group inspection. In the model the main three roles involved are, the moderator, the author and several inspectors.

9.4 THE PROTOTYPE

The prototype developed is based on the SecureSIG model. The prototype aims to test security in synchronous and asynchronous modes and also the ability to use a secure tool across the Internet.

The functional architecture of the secure internet-based groupware system was presented. The secure architecture of the prototype is described by the cryptographic protocol architecture, secure access control architecture, key pair generation architecture, file encryption architecture, and secure databases. The components of the prototype includes set-up, secure system access, key pair generation, file encryption, secure briefing, secure document inspection, secure communication, and comment log components.

Among the technologies that were used to achieve the aim are: cryptography, Java technology, and the Internet. These technologies proved suitable to provide the security functionality and the platform need in the implementation of the prototype. The limitations brought out by the prototype were the speed of performing the encryption/decryption processing and inconsistency in the features support in different versions of Java technology.

9.5 RESEARCH EVALUATIONS AND FUTURE WORK

From the user evaluation of the prototype analysis, it can be concluded that this research has met the aim outlined in the chapter four and repeated in section 9.3. The findings, based on the analysis and the evaluations, relate to the original aim of this research as follows:

- *Develop a secure software inspection model.* The secure software inspection groupware (SecureSIG) model was developed based on FlexSIG model. The lack of security in FlexSIG model has been overcomes by providing the secure process flow to the model.

- *Implement a prototype based on the model.* The SecureSIG prototype was constructed based on the model developed. Secure components were developed using Java technology cryptography toolkits (Java Cryptography

Extension) to provide secure environment for the software inspection groupware process.

- *Evaluate the prototype*. Test users have evaluated the SecureSIG prototype. Analysis from the questionnaires verified the suitability and transparency of the prototype.

This research is limited to the analysis and user evaluation of the prototype due to the limited time available; the benchmarking of SecureSIG is proposed for future work. Despite the successful achieving of the aim, there are some problems and limitations which would benefit further research.

The research has opened up a number of possibilities for future work enhancement of the SecureSIG model, prototype development, and evaluation method. Other proposals that were suggested are the development of the SecureSIG system using more advanced enabling technology (e.g. audio and video), improvement of the key management and the issues of security in databases and documents related to software inspections. Extending to securing graphical documents was also proposed.

9.6 SUMMARY

As a conclusion, this research has demonstrated that it is possible to create a secure system for groupware that is suitable and transparent to the user. A secure software inspection groupware model has been developed. The prototype based on this model has been developed and evaluated by users.

As a whole this research has extended current technology of groupware, particularly software inspection groupware, by providing a secure environment

to the software inspection process. The main contributions of this research are as follows:

- A secure software inspection groupware system that is not limited to any one of the four categories of the time and space taxonomy.
- A model of a secure software inspection groupware.
- A secure software inspection groupware system that is provided with a secure access control mechanism to protect the system from unauthorised user gain access to the system and resources.
- A secure software inspection groupware that is provided with an encryption mechanism to provide a safeguard to the information stored.
- A secure software inspection groupware that is provided with an encryption mechanism to provide protection to the information flow to and from the system.
- A secure software inspection groupware that is provided with two layers of protection - the internal and external protection.
- A secure software inspection groupware that is provided with a security shell architecture for protection of CSCW environments.

The results and the findings obtained have indicated that the objectives outlined have all been met.

GLOSSARY

Access Control : The prevention of unauthorised access to resources including the prevention on their use in an authorised manner.

Asynchronous : Occurring at different times.

Authentication : Providing assurance regarding the identity of subject or object, for example, ensuring that a particular user is who he claims to be.

Author : A person or team who has written something (product) using a set of source documents, in accordance with a set of rules.

Certificate : Data recorded that provides the public key of a principle, together with some other information related to the name of the principle and the certification authority that has issued the certificate. The certificate is rendered unforgeable by appending a digital signature from a certification authority.

Certificate Authority : Trusted third party that creates, assigns, and distributes public key certificates.

Ciphertext : Data produced through the use of encryption. The semantic content of the resulting data is not available. Encryption transforms plaintext into ciphertext.

Confidentiality : The property that ensures that confidential information is not made available or disclosed to unauthorised parties.

Cryptography : The discipline that embodies principles, means, and the methods for the transformation of data in order to hide its information content, and prevent its undetected modification or unauthorised use. The choice of cryptographic mechanisms determines the methods used in encryption and decryption.

CSCW : *Computer Supported Cooperative Work.* A term which combines the understanding of the way people work in groups with the enabling technologies of computer networking and associated hardware, software, services and techniques.

Decryption : The opposite of encryption.

Defect : An error made in writing a document or code which violates a rule.

DES : Secret key cryptosystem (Data Encryption Standard).

Differential Cryptanalysis : A statistical attack that can be applied to any iterated mapping (i.e., any mapping which is based on a repeated round function).

Digital Signature : Data appended to or a cryptographic transformation of a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and to protect against forgery.

Document : A written set of information, which can be the subject for inspection.

Electronic Mail : Enable message to be sent to one or more people. The messages are delivered to an electronic mailbox and are read at the time and the location of recipient choosing.

Encryption : The transformation of original text (*plaintext*) into unintelligible text (*cipertext*).

Groupware : A generic term for specialised computer aids that are designed for the use of collaborative work.

IDEA : Secret key cryptosystem (International Data Encryption Algorithm).

Inspector : A person who examines a set related documents with the primary objective of finding potential defects.

Integrity : The property that ensures that data is not altered undetected.

Internet : Internet(work) based on TCP/IP communications protocol suite.

Java : An object-oriented programming language for creating distributed, executable applications.

Key generation - The act of creating a key.

Key Management : The generation, storage, distribution, deletion, archiving, and applications of keys in accordance with a specific security policy.

Key pair - The full key information in a public-key cryptosystem, consisting of the public key and private key.

Linear cryptanalysis - A known plaintext attack that uses linear approximations to describe the behavior of the block cipher. See known plaintext attack.

Login : The process of identifying oneself to, and having one's identity authenticated by, a computer system.

Man-in-the-middle-attack : Attacks that includes interception, insertion, deletion, and modification of messages, reflecting messages back to the sender, replaying old messages, and redirecting messages.

Masquerade : Posing as an authorised user, usually in an attempt to gain access to a system.

Message digest - The result of applying a hash function to a message.

Moderator : The person who lead the inspection process.

Password : A secret sequence of characters that's used to authenticate a user's identity.

Plaintext : Unencrypted text. Contrast with ciphertext.

Private Key : Cryptographic key used in public key cryptography to sign and/or decrypt messages.

Private-key encryption : An encryption algorithm that uses only secret keys.

Public Key. The key used in an asymmetric cryptosystem that is publicly available.

Public-key encryption : An encryption algorithm that uses a public key to encrypt data and corresponding secret key to decrypt data.

RSA : Public key cryptosystem invented by Rivest, Shamir, and Adleman.

Secret Key : The key used in a symmetric cryptosystem that is shared between the communicating parties.

Self-certified public key : Public key that is certified with its corresponding private key.

Session Key : A temporary key shared between two or more principals, with a limited lifetime.

Software Inspection : An evaluation technique to find defects and problems in a product.

Synchronous : Occurring at the same time.

TCP/IP : Entire suite of data communications protocols. The suite gets its name from two of its most important protocol, namely the transmission control protocol (TCP) and the Internet protocol (IP).

Trusted Third Party : A security authority or its agent, trusted by other entities with respect to security-related activities.

Workgroup Computing : Activities undertaken on a network using software application programmes designed to support the members of a group.

REFERENCE

Abdullah, A. H. (1994), "Accessing Networked Services: A User Interface Design Problem", PhD Thesis, Aston University, Birmingham, United Kingdom.

Ackerman, A. F. (1984), "Software Inspections and the Industrial Production of Software", in Hause, H.L. (ed), *Software Validation*, Amsterdam: Elsevier, Science Publishers.

Andelman, D. & Reeds, J. (1982), "On the Cryptanalysis of Rotor Machines and Substitution-Permutation Networks", *IEEE Transactions of Information Theory*, **IT-28**(4), July, 578-584.

Bannon, L. and Schmidt, K. (1991), "CSCW: Four Characters in Search of Context" in J.M. Bowers and S.D. Benford (eds): *Studies in Computer Supported Cooperative Work. Theory, Practice and Design*, Amsterdam: North-Holland,.pp. 3-16.

Biham, E. & Shamir, A. (1991), "Differential cryptanalysis of DES-like cryptosystems", *Journal of Cryptology*, 4(1991), 3-72.

Biham, E. & Shamir, A. (1993), "Differential cryptanalysis of the full 16-round DES", *Advances in Cryptology-CRYPTO'92*, Berlin: Spring-Verlag, pp. 487-496.

Black, U. (1993), *Computer Networks: Protocol, Standards, and Interface* (2nd Edition), New Jersey: Prentice Hall.

Blair, G. & Rodden, T. (1994), *The Challenges of CSCW for Open Distributed Processing*, Bailrigg, Lancaster: Department of Computing, Lancaster University.

Blakley, G. R. (1979), "Safeguarding Cryptographic Keys", *Proceedings of the National Computer Conference, 1979,* American Federation of Information Processing Societies, v.48, 242-268.

Brothers, L., Sembugamoorty, V., & Muller, M. (1990), "ICICLE: Groupware for Code Inspection", *CSW 90 Proceedings*, October, ACM, pp. 169-181.

CCITT Recommendation X.509 (1989), "The Directory-Authentication Framework", Consultation Committee, *International Telephone and Telegraph, International Telecommunication Union*, Geneva.

Chaum, D. & Evertse, J. H. (1986), "Cryptanalysis of DES with a Reduced Number of Rounds; Sequences of Linear Factors in Block Ciphers", *Advances in Cryptology-CRYPTO'85 Proceedings*, Berlin: Spring-Verlag, pp. 192-211.

Cheswick, W. R. & Bellovin, S. M., (1994), *Firewalls and Internet Security: Repelling the Wily Hacker*, Reading: Addison-Wesley.

Ciancarini, P., Knoche, A., Tolksdorf, R., & Vitali, F. (1996), "PageSpace: An Architecture to Coordinate Distributed Applications on the Web", *Fifth International World Wide Web Conference*, Paris, France, May 6-10, 1996.

Coulouris, G. & Dollimore, J. (1994), "A Security Model for Cooperative Work", *Technical Report 674*, Department of Computer Science, Queen Mary and Westfield College.

Daeman, J., Govaerts, R. & Vandewalle, J. (1994), "Weak Keys for IDEA", *Advances in Cryptology-CRYPTO'93 Proceedings*, Berlin: Springer-Verlag, pp. 224-230.

Davies, D. W. & Price, W. L. (1989), *Security for computer networks, an introduction to data security in teleprocessing and electronic funds transfer* (2nd Edition), Chichester: Wiley & Sons.

Diffie, W. & Hellman, M. E. (1976), "New Directions in cryptography", *IEEE Transactions on Information Theory*, 22, pp. 644-654.

Doherty, B. S., and Maarof, M. A., (1997), "Integrating Security Services Into Collaborative Systems", *Proceedings of the World Conference of the WWW, Internet, and Intranet (WebNet 97)*, Charlottesville: Association for the

Advancement of Computing Education Publications, October, Toronto, pp. 761-763.

Doherty, B. S., & Sahibuddin, S. (1996), "Modelling Distributed Code Inspection System", *Proceedings of the REDECS'96*, Serdang, Malaysia: Universiti Putra Malaysia Publications, June, pp. 69-73.

Doherty, B. S., & Sahibuddin, S. (1997), "Software Quality through Distributed Code Inspection Groupware", in Tasso, S., Adey, R. A., & Pighin, M. (eds.), *Software Quality Engineering*, Southampton: Computational Mechanics Publications, pp. 159-168.

ElGamal, T. (1985), "A Public_Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Transaction on Information Theory*, **IT-31**(4), 469-472.

Ellis, C. A., Gibbs, S. J. & Rein, G. L. (1991), "Groupware Some Issues and Experiences", *Communications of the ACM*, **34**(1), January, 39-58.

Eltoweissy, M. Y (1993), "A framework for data sharing in computer-supported co-operative environments", Ph.D., Old Dominion University, U.S.A.

Engelbart, D. & Lehtman, H. (1988), "In Depth Groupware: Working Together", *BYTE Magazine*, New York: McGraw-Hill, December 1988, pp. 245-252.

Fagan, M. E. (1976), "Design and Code Inspections to Reduce Errors in Program Development", *IBM System Journal,* International Business Machines, **15**(1), 182-211.

Fagan, M. E. (1986), "Advances in Software Inspection", *IEEE Transactions on Software Engineering*, New York: IEEE Press, July, **12**(7), 744-751.

Flynn, J. & Clarke, B. (1995), "The World Wakes Up To Java", *Computer Technology Review*, Fall/Winter 1995, pp. 34-37.

Foley, S. N. & Jacob, J. (1995), "Specifying Security for CSCW Systems", *Eight IEEE Computer Security Foundations Workshop*, June 13-15, IEEE Computer Science Press, pp. 136-45.

Gilb, T. & Graham, D. (1993), *Software Inspection*, Wokingham, England: Addison-Wesley.

Gintell, J. W., Arnold, J., Houde, M., Kruszelnicki, J., McKenny, R., & Memmi, G. (1993), "Scrutiny: A Collaborative Inspection and Review System", *Proceedings of the 4th European Software Engineering Conference*.

Girault, M. (1991), "Self-certified public keys", *Advances in Cryptology-EUROCRYPT '91, Proceedings*, Berlin: Springer-Verlag, pp 490-497.

Greif, I. (ed) (1988), *Computer Supported Cooperative Work: A Book of Reading*, San Meteo: Morgan Kaufman.

Grudin, J. (1991), "A Tale of Two Cities: Reflections on CSCW in Europe and the United States", *SIGCHI Bulletin*, July, ACM, **23**(3), 22-24.

Grudin, J. (1993), "CSCW: history and focus", Information and Computer Science Department, Irvine: University of California.

Grudin, J. (1994), "Groupware and Social Dynamics: Eight Challenges for Developers", *Communication of the ACM*, January 1994, **37**(1), 93-105.

Hanka, R. & Buchan, I. E. (1996), "Security measures in open communication systems", Medical Informatics Unit, University of Cambridge. (http://www.medinfo.cam.ac.uk/miu/papers/misc/brighton1.htm, email: hanka@medschl.com.ac.uk)

Hassler, V. (1997), "Internet Security: State-of-the-Art and Future Trends," Technical Report (TUV-1841-97-08), Distributed System Department, Technical University of Vienna.

Hix, D. & Hartson, R. (1993), *Developing User Interfaces: Ensuring usability through Products and Process*, New York: Wiley.

Hughes, M., Hughes, C., Shoffner, M., & Winslow, M. (1997), *Java Network Programming*, Greenwich: Manning.

Humphrey, W. S. (1989), *Managing the Software Process*, Reading, Massachusetts: Addison-Wesley.

IAIK (1998), Institute for Applied Information Processing and Communications (IAIK), Graz, Austria (http://jcewww.iaik.tu-graz.ac.at, email:jce-info@iaik.tu-graz.ac.at).

IBM (1997), "IBM SecureWay Internet Technical Direction", IBM Security White Paper, International Business Machine, January (http://www.ibm.com/security/html/wp_techdir.html).

Idris, N. B. (1995), "Dynamic secrecy control for a meta-integrated heterogeneous database environment", PhD Thesis, University of Wales College of Cardiff, United Kingdom.

ISO (1989), ISO 7498-2 - "Information Processing Systems – Basic Reference Model – Part 2: Security Architecture", *International Organisation for Standardisation*, Geneva, Switzerland (first edition).

Johansen, R. (1988), *Groupware: Computer Supported for Business Teams*, New York: The free Press, Macmillan.

Johnson, Philip M. (1994), "Supporting Technology Transfer of Formal Technical Review Through a Computer Supported Collaborative Review System", *Proceeding of the 16th International Conference on Software Engineering*, New York: ACM Press.

Johnson-Lenz, P., and Johnson-Lenz, T. (1982), "Groupware: The process and impact of design choices", In Kerr and Hiltz (eds), *Studies of Computer Mediated Communication Systems*, pp. 45-55.

Kanawati, R. & Riveill, M. (1995), "Access Control Model for Groupware Applications", , *HCI'95 People and Computer*, G. Allen, J. Wilkinson et P. Wright (ed), pp. 66-71 (Adjunct Proceedings), School of Computing & Mathematics, University of Huddersfield-UK.

Karila, A. T. (1991), "Open System Security – an Architectural Framework", *Telecom Findland, Business Systems R & D*, Helsinki: Telecom Findland.

Knight, J. C. & Myers, E. A. (1991), "Phased Inspection and their Implementation", *Software Engineering Notes*, New York: ACM SIGSOFT, July, **16**(3), 29-35.

Knight, J. C. & Myers, E. A. (1993), "An Improved Inspection Technique", *Communication of the ACM*, New York: ACM Press, **36**(11), 50-61.

Knudsen, J. (1998), *Java Cryptography*, Sebastopol: O'Reilly & Associates.

Kohnfelder, L. (1978), *Towards a Practical Public-Key Cryptosystem*, Bachelor's Thesis, Massachusetts Institute of Technology, May 1978.

Lai, X. & Massey, J. (1990), "A proposal for a new Block Encryption Standard", *Advances in Cryptology-EUROCRYPT'90 Proceedings*, Berlin: Springer-Verlag, pp. 389-404.

Lai, X. & Massey, J. (1991), "Markov Ciphers and Differential Cryptanalysis", *Advances in Cryptology-EUROCRYPT'91 Proceedings*, Berlin: Springer-Verlag.

Lai, X. (1992), *On the Design and Security of Block Cipher*, ETH Series in Information Processing, Konstanz: Hartung-Gorre Verlag.

LANL (1997), "Generic TCP-Based Client-Server Security Model", Technical Standard IA-7701, Los Alamos National Laboratory, U. S. (http://www.lanl.gov/projects/ia/stds/ia770112.html).

Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch D. C., Postel, J., Roberts, L. G., & Wolff, S. (1998), "A Brief History of the Internet", *Internet Society ISOC – All About The Internet*, February 1998, (http://www.isoc.org/internet/history/brief.html, email: bleiner@computer.org).

Linden, Peter van der (1996), *just Java*, Mountain View, California: SunSoft Press, A Prentice Hall Title.

Lynch, K. J, Snyder, J. M., Vogel, D. R., & McHenry, W. K. (1990), "The Arizona Analyst Information System: Supporting Collaborative Research on International Technological Trends", in Gibbs, S, and Verrijn-Stuart, A.A. (eds): *Multi-User Interfaces and Applications*, North-Holland, Amsterdam, pp 157-174.

Markovitz, P. (1994), "X.400 Message Handling Services", Security in Open System, *National Institute of Standard and Technology*, NIST Special Publication 800-7, Barkley, J. (ed), pp. 145-176.

Mashayekhi, V., Drake, J. M., Tsai, W. T., & Reidl, J. (1993), "Distributed, Collaborative Software Inspection", *IEEE Software*, **10**(5), New York: IEEE Press, 66-75.

Massey, J. L. (1994), "SAFER K-64: A Byte Oriented Block-Ciphering Algorithm", *Fast Software Encryption, Cambridge Security Workshop Proceedings*, Berlin: Springer-Verlag, pp. 1-17.

Matsui, M. (1994), "The First Experimental Cryptanalysis of the Data Encryption Standard", *Advances in Cryptology-CRYPTO'94 Proceedings*, Berlin: Spring-Verlag, pp. 1-11.

McGhie, L. (1994), "A model for a secure distributed computing environment", *Computer Security Journal*, **10**(2), 27-36.

Meier, W. (1994), "On the Security of the IDEA block cipher", *Advances in Cryptology-EUROCRYPT'93*, Berlin: Springer-Verlag, pp. 371-385.

Menezes, Alfred. J., van Oorschot Paul. C., & Vanstone, Scott A. (1997), *Handbook of Applied Cryptography*, London: CRC Press.

Merkle, R. C. & Hellman, M. (1978), "Hiding Information and Signatures in Trapdoor Knapsacks", *IEEE Transactions on Information Theory*, **24**(5), September, 525-530.

NBS (1977), "Data Encryption Standard", FIPS PUB46, January.

Nechvatal, J. (1992), "Public Key Cryptography", G.J. Simmons, editor, *Contemporary Cryptology: The Science of Information Integrity*, New York: IEEE Press, pp. 177-284.

Nielsen, J. (1993), *Usability Engineering*, UK: Academic Press Limited.

NIST (1993), "Secure Hash Standard", *National Institute of Standards and Technology*, NIST FIPS PUB 180, U.S. Department of Commerce, May 93.

NIST (1994), "An Introduction to Computer Security: The NIST Handbook", *National Institute of Standard and Technology*, Special Report 800-12, Technology Administration, U.S Department of Commerce.

Odlyzko, A. M. (1994), "Public key cryptography", *AT&T Technology Journal*, **73**(5), (Sept.-Oct. 1994), pp. 17-23.

Olson, J. S., Card, S. K, Landauer, T. K, Olson, G. M, Malone, T., & Leggett, J. (1993), "Computer-supported co-operative work: research issues for the 90's", *Behaviour and Information Technology*, **12**(2), 115-129.

Opper, S. & Fersko-Weiss, H. (1992), *Technology for Teams: Enhancing Productivity in Networked Organizations*, New York: Van Norstrand Reinhold.

Oppliger, R. (1998), *Internet and Intranet Security*, London: Artech House.

Pfleeger, C. P. (1989), *Security in Computing*, New Jersey: Prentice Hall.

Power, R. (1995), *Current and Future Danger*, Computer Security Institute, San Francisco, California.

Rabin, M. O. (1979), "Digital Signature and Public-Key Functions as Intractable as Factorization"" *MIT Laboratory for Computer Science*, Technical Report, MIT/LCS/TR212, January.

Rita, C. S. (1997), *Secure Computing: Threats and Safeguards*, San Francisco: McGraw-Hill.

Rivest, R. L, Shamir, A., & Adleman, L. M. (1978), "A method for obtaining digital signature and public-key cryptosystem", *Communcation of the ACM*, 21, pp. 120-126.

Rivest, R. L. (1992), "The MD5 Message Digest Algorithm", RFC 1320, April.

Roberts, B. (1996), "Groupware Strategies", *BYTE Magazine*, New York: McGraw-Hill, July, pp. 68-78.

Rogers, A.S. (1994), "An introduction to groupware and CSCW", *BT Technology Journal*, **12**(3), 7-11.

RSA (1993), "PKCS #5: Password-Based Encryption Standard", *an RSA Laboratories Technical Note*, a division of RSA Data Security, Inc., (http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs-5.html, email: pkcs-editor@rsa.com)

Russel, D. and Gangemi Sr., G. T. (1991), *Computer Security Basics*, California: O'Reilly and Associates.

Sahibuddin, S. (1999), "FlexSIG: Flexible Software Inspection Groupware", PhD Thesis, Aston University, Birmingham, United Kingdom.

Sakibara, K., Seki, K., Okada, K. & Matsushita, Y. (1994), "The ID-based Non-interactive Group Communication Key Sharing Scheme using Smart Cards", *Proceeding of 1994 International Conference on Network Protocols*, Boston, USA, October 1994, pp. 91-98.

Schneier, B. (1996), *Applied Cryptography* (2nd Edition), New York: Wiley & Son.

Scott, R. (1985), "Wide Open Encryption Design Offers Flexible Implementations", *Cryptologia*, **9**(1), January, 75-90.

Shamir, A. (1979), "How to Share a Secret", *Communication of the ACM*, **24**(11), Nov. 1979, 612-613.

Shen, H. H. & Dewan, P. (1992), "Access Control for Collaborative Environments", *Proceeding of ACM CSCW '92*, pp. 51-58.

Shimizu, A. & Miyaguchi, S. (1988), "Fast Data Encryption Algorithm FEAL", *Advances in Cryptology – EUROCRYPT'87 Proceedings*, Berlin: Springer-Verlag, pp. 267-278.

Smid, M.E. & Branstad, D. K. (1992), "The Data Encryption Standard: Past and Future", G.J. Simmons, editor, *Contemporary Cryptology: The Science of Information Integrity*, New York: IEEE Press, pp. 43-64.

Stalling, W. (1995), *Network and Internetwork Security*, London: Prentice Hall.

Stallings, W. (1999), *Cryptography and Network Security (2nd Edition)*, London: Prentice-Hall.

Stein, Micheal *et al.* (1997), "A Case Study of Distributed, Asynchronous Software Inspection", *Proceeding of the 19th International Conference on Software Engineering*, New York: ACM Press, pp. 107-117.

Stevens, R. (1990), *Unix Network Programming*, London: Prentice Hall.

Sun (1995), *A Java Language Environment: A White Paper*, Mountain View, California: Sun Microsystems Computer Company, October.

Sun (1997), *Java Cryptography Architecture API Specification & Reference*, Sun Microsystem, December, (http://java.sun.com/products/jdk/1.1/docs/guide/security/CryptoSpec.html, email: java-security@java.sun.com).

Takizawa, M., & Mita, H. (1993), "Secure Group Communication Protocol for Distributed Systems", *Proceedings of 7th International Computer Software and Applications Conference (COMPSAC '93)*, pp. 159-165.

Tanenbaum, A. S. (1981), *Computer Networks*, New Jersey: Prentice Hall.

Teufal, S., Eloff, J. H. P., Bauknecht, K., & Karagiannis, D. (1995), "Information Security Concepts in Computer Supported Cooperative Work", Norman Revell and A. Min Tjoa (eds), In *Proceedings of 6th International Conference on Database and Expert System Application*, September 1995, Berlin Springer Verlag, pp. 621-631.

Vaudenay, S. (1995), "On the need for mulipermutaion: Cryptanalysis of MD4 and SAFER", B. Preneel (ed), *Fast Software Encryption, Third International Workshop*, Berlin: Springer-Verlag, 1996, pp. 27-32.

Voydock V. L. & Kent S. T. (1983), "Security mechanisms in high level network protocols", *ACM Computing Surveys*, **15**(2), June.

Williams, H. C. (1980), "A modification of the RSA Public-Key Encryption Procedure", *IEEE Transactions on Information Theory*, **IT-26**(6), November 1980, 726-729.

Wilson, P. (1990), *Computer Supported Cooperative Work: An Introduction*, Oxford: Intellect Books.

Wilson, P. (1991), "Computer supported cooperative work: an overview", *Intelligent Tutoring Media,* **1**(3), 1990, 103-116.

Wu, T. (1988), "The Secure Remote Password Protocol", in Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium, San Diego, CA, pp. 97-111.

BIBLIOGRAPHY

Fayad, M. E., & Tsai W. (1995), "Object-Oriented Experiences", *Communication of the ACM*, **38**(10), October, 1995.

Ganesan, R., & Sandhu, R. (1994), "Securing Cyberspace", *Communication of the ACM*, **37**(11), 30-32.

Huff, S. L. (1993), "Object-Oriented Programming", *Business Quarterly*, **58**(2), Winter 1993, 85-89.

Hughes, K. (1994), "Entering the World-Wide Web: A Guide to Cyberspace", Enterprise Integration Technologies, May 1994 (http://www.afn.org/web/guide.61/).

Ingham, D., Little, M., Caughey, S., & Shrivastava, S. (1995), "W3Objects: Bringing Object-Oriented Technology to the Web", *Proceeding of the Fourth International World Wide Web Conference*, Boston, Mass., U.S.A, Dec. 1995.

Jerram, P. (1995), "Groupware Taps the Internet", *BYTE Magazine*, New York: McGraw-Hill, December 1995.

Johnson, J. T. (1995), "Enterprise Security: Better Safe Than Sorry", *Data Communications*, **24**(3), 110-127.

Kennedy, A. J. (1995), "The Internet and World Wide Web", London: Rough Guide Ltd..

Krakowiak, S. (1993), "Issues in Object-Oriented Distributed Systems", *International Conf. on Decentralized and Distributed Systems*, IFIP WG 10.3, Palma de Mallorca, pp. 1-12.

Kydd, C., & Ferry, D. (1991), "A behavioral view of computer-supported cooperative Work tools", *Management Systems*, **3**(1), 55-67.

McGregor, J. D., & Sykes, D. A. (1992), *Object-Oriented Software Development: Engineering Software for Reuse*, Van Nostrand Reinhold, New York.

Meyer, B. (1987), "Reusability: The Case for Object-Oriented Design", *IEEE Software*, March 1987, pp. 50-64.

Mullender, S. J., & Tanenbaum, A. S. (1986), "The design of a capability-based distributed operating system", *Computer Journal*, **29**(8), 289-299.

Neuman, B. C. (1993), "Proxy-based authorization and accounting for distributed systems", In *Proceeding of the 13th International Conference on Distributed Computing Systems*, Pittsburgh, Pennsylvania, May 1993.

Nierstrasz, O. (1989), "A survey of Object-Oriented Concepts", In Won Kim and Federick H. Lochovsky, editors, *Object-Oriented Concepts, Database, and Applications*, pp. 3-22. Reading: Addison-Wesley.

Pernul, G. (1995), "Information systems security: scope, state-of-the-art, and evaluation of techniques", *International Journal of Information Systems*, **15**(3), 165-180.

Pinson, L. J, & Wiener, R. S. (1988), *An Introduction to Object-Oriented Programming and Smalltalk*, Reading: Addison-Wesley.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1991), *Object-Oriented Modelling & Design*, New Jersey: Prentice-Hall.

Appendix A

CLASSIFICATION OF CSCW

A.0 INTRODUCTION

This appendix describes Wilson's classification of CSCW. Other naming of CSCW is also given in Section A.2.

A.1 WILSON'S CLASSIFICATION

Wilson (1990, 1991) divides the field of CSCW into two distinct but interrelated fields: the *group working process* and the *enabling technology* employ to support it.

A.1.1 Group Working Process

Group working process was subdivided into four categories: *individual aspects*, *organisational aspects*, *group work design aspects*, and *group dynamics aspects*.

Individual Aspects

In individual aspects, the characteristics, skills, knowledge and artefact that individuals bring with them to the group process have a crucial impact on group's effectiveness. Research in which work on individual aspects is being done include human communication characteristics, individual work patterns and interface design for group support systems. In human characteristic research, the way humans talk to each other and take action is investigated. Individual work pattern factors need to be taken into account in the design of group support tools and working practices because when an alternative way of working is adopted, the individual's habits and predilections show through. Interface design

for group support systems depend on a combination of ingenuity and knowledge in fields such as human perceptions and cognitive psychology.

Organisational Aspects

In the organisational aspects, representation of organisational knowledge, organisation design, and management issues are investigated. In the area of representation of organisational knowledge, knowing how an organisation is structured is needed to get things done, but problems arise when too much organisational information and information frequently changes. Organisation design concerns the provision of tools to support organisational changes by integrating pre-specified goals together with knowledge about how groups and individuals work best. In management issues, as new support tools have been created, new requirements appear for the management of activities, people and resources.

Group Design Aspects

In the group design aspects, user involvement, prototyping and usability, and group work design procedures are investigated. Investigation of user involvement is needed because people who have been involved in organising their own work usually have more positive attitudes and are keener to make eventual solution work. Prototyping and usability testing is needed because user involvement is not enough to ensure success because human interaction within work groups is complex and unpredictable. Design must be carried out under as near real-world working situations as possible to ensure effective solutions. Awareness of the need to provide clear guidelines for analysis and design of CSCW systems resulted in research in group work design procedures.

Group Dynamics Aspects

The group dynamics aspects deal with the way individuals behave within a group, and the way groups perform. In these aspects, the collaboration process, group performance and group behaviour are considered. The collaboration process need to be understood because it is of great value in contributing to the design of group work tools. In group performance issues, relative effectiveness of group performance with different media has been investigated. Group behaviour research is closely allied to group performance and is done in real-world, non-laboratory conditions, and in studies using prototypes.

A.1.2 Enabling Technologies

Enabling technologies were subdivided into four categories: *communication systems*, *shared work space facilities*, *shared information facilities*, and group *activity support facilities*. These enabling technologies are not mutually exclusive. A CSCW system might possess facilities in any combination of the categories.

Communication Systems

The main aims of this area are to make sure that technological support can be provided so that informal communication can continue. Examples of communications systems are advanced electronic mail systems, X.500 electronic mail directories incorporating group and organisational information, real-time desktop video conference systems and room-based video systems.

Shared Workspace Facilities

In this area, the role that shared workspaces play in group, and tools, which support the process, is investigated. Examples of shared work space tools are remote screen-sharing, face-to-face meeting support using shared individual screens and large public screen, and electronically aided white board.

Shared Information Facilities

These facilities are a starting point for people working together. The facilities are required to support the input, storage, navigation and retrieval of that information by all members of the group. Examples of shared information are multimedia, multi-user hypertext systems, shared optical disc systems, and multi-user databases.

Group Activity Support Facilities

People who work together usually have common understanding of the work process. Group activity support facilities must be able to meet the needs of groups. Meeting these needs is a two part process, first, procedures must be established and agreed; and, second, the procedures must be visible when carried out. Examples of activity support tools are procedure processing, activity processors which allow a more general form of procedure processing,

methodologies and support tools to aid groups in analysis, procedures and equipment with which they are to carry out a group activity, and many more.

A.2 OTHER VIEWS OF NAMING CSCW

In this section other definitions and classification of CSCW are given.

A.2.1 Other Definitions

There are varieties of term that have been used in describing CSCW. Among the terms are Groupware, Workgroup Computing, Technology for Teams and Computer-Aided Teams (Wilson, 1991).

Opper & Fersko-Weiss (1992) describe groupware as any information system designed to enable groups to work together electronically. Meanwhile Johansen (1988) describes groupware as groupware as a generic term for specialised computer aids that are designed for the use of collaborative work groups. Johansen described these groups as small project-oriented teams that have important tasks and tight deadlines. Groupware can involve software, hardware, services, and group process support.

A.2.2 Other Classification

Bannon & Scmidt (1991) divide CSCW into three main issues, namely; articulating cooperative work, sharing an information space, and adapting the technology to the organisation and vice versa. In any cooperative effort, tasks are to be allocated to different members, where that worker is accountable for accomplishing that task and finally combining all the efforts of individuals and ensembles. Cooperative work can be conducted in a distributed way. Thus, any computer systems must support retrieval of information by other co-workers in order to support cooperative work.

Appendix B

OVERVIEW OF CRYPTOGRAPHY

B.0 INTRODUCTION

This appendix describes the basic terminology and definition of Cryptography. The two main category of cryptography; secret key and public key cryptography are presented. Cryptographic hash function. And the digital signatures are also presented.

B.1 BASIC TERMINOLOGY AND DEFINITIONS

In cryptographic terminology, the original message text P that we wish to transmit over the network is known as the *plaintext*. A *cryptographic algorithm* or *cipher* converts P to a form that is unintelligible to anyone monitoring the network. This conversion process is called *encryption*. The unintelligible form is known as *ciphertext*. A cryptographic algorithm is a technique or rule selected for encryption that determines how simple and how complex the process of conversion will be (Russell & Gangemi, 1991). The precise form of the ciphertext c corresponding to a plaintext P depends on an additional parameter K known as the *key*.

The intended receiver of a ciphertext c may wish to recover the original plaintext p. To do this, a second key K^{-1} is used to reverse the process. This reverse process is known as *decryption*. A method of encryption and decryption is called a cipher. A system for encryption and decryption is called a cryptosystem. Figure B.1 shows the encryption and decryption process.

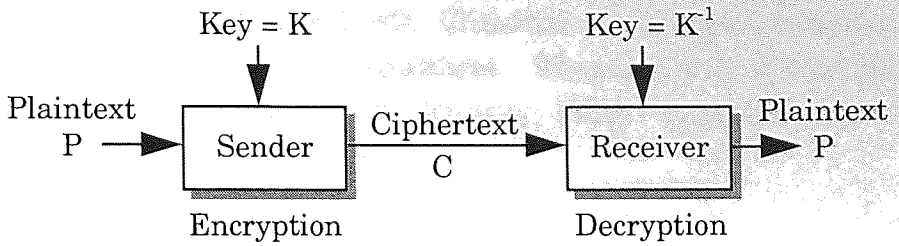

Figure B.1: Encryption and Decryption Process

The study of encryption and decryption is known as *cryptography*. The process of trying to break an encrypted message without access to the key is called *cryptanalysis*. *Cryptology* includes both cryptography and cryptanalysis (Russell & Gangemi, 1991).

Cryptographic systems fall into two general categories (identified by the types of keys they use): *secret key* and *public key* systems (Russell & Gangemi, 1991). They are discussed in detail in the next section.

B.2 SECRET KEY CRYPTOGRAPHY

In *secret key cryptography* the encryption key K and the decryption key K^{-1} are usually the same. This key is called a *secret key*. The key K is used by both parties (the sender and the receiver) to encrypt and decrypt messages to and from each other. Anyone who holds the key can create ciphertexts corresponding to arbitrary plaintexts and read the contents of arbitrary ciphertext messages. To ensure security of communication this secret key is kept secret between the communicating parties. Secret key cryptography is also refers as *symmetric cryptography*.

B.2.1 Classical Cryptography

Classical cryptography has typically used symmetric keys. Two basic components of classical cipher techniques are *substitution* and *transposition* (Davies & Price, 1994). A substitution cipher replaces the actual bits, characters, or blocks of characters with substitutes (for example one letter replaces another letter). A transposition cipher (sometimes called permutation cipher) rearranges or shuffles plaintext characters or bits. The precise substitutions and transpositions made are defined by the key. Examples includes simple, homophonic, poly-alphabetic and polygram substitution ciphers and columnar transposition ciphers and others (Davies & Price,

1989), (Pfleeger, 1989), (Schneier 1996). Elements of transposition and substitution are also included in modern-day algorithms. They are well documented already (Davies & Price, 1989; Pfleeger, 1989; Schneier, 1996). Appendix G presents some example of the Classical Cryptography.

C.2.2 Modern-day Cryptography

Modern-day secret key cryptography is principally *stream ciphers or block ciphers*. The block ciphers will be discussed in great detailed due to it adoption in the implementation of this research.

Stream Cipher

Stream cipher encrypts one bit of a plaintext message one at a time, using an encryption transformation that varies with time (Menezes, van Oorschot & Vanstone, 1997). Advantages of stream ciphers are (Pfleeger, 1989):

- *Speed of transformation*. Each symbol is encrypted without regard for any other plaintext symbols thus each symbol can be encrypted as soon as it read.
- *Low error propagation*. Each symbol is separately encoded, thus an error in the encryption process will affect only on one symbol.

By contrast, from the security perspective the independence of the symbols results in the disadvantages of the stream cipher. The disadvantages are (Pfleeger, 1989):

- *Low diffusion*. Each symbol is separately encrypted therefore all the information about that symbol is contained in one symbol of the plaintext.
- *Susceptibility to malicious insertions and modifications*. Since each symbol is separately encrypted, an active interceptor (intruder) can splice together pieces of previous messages and transmit a spurious new message that may look authentic.

Block Cipher

A *block cipher* will encrypt a block (typically 64 or 128) of plaintext bits at a time. Block ciphers operate by taking a fixed length of plaintext as one block and

generating the same amount (number of bits) of ciphertext. A very good overview of block (and other) ciphers can be found in the book by Schneier's (Schneier, 1994).

Block ciphers have advantages that stream ciphers lack, while the disadvantages of block ciphers are the strength of stream ciphers. The advantages of block ciphers are (Pfleeger, 1989):-

- *Diffusion.* Information from the plaintext is diffused into several ciphertext symbols, it affects all ciphertext symbols of the generated block.
- *Immunity to insertions.* Since blocks of symbols are encrypted, this makes it immune to insertions or modifications of ciphertext symbols in one block.

On the other hand, block ciphers have disadvantages. These disadvantages are (Pfleeger, 1989):

- *Slowness of encryption.* It has to wait for a complete block to be read before performing the encryption. This result to a slower cryptosystem.
- *Error propagation.* An error of a single symbol will effect all the transformation of all other symbols in the same block. Thus an error in a single symbol result in retransmission of the entire block.

Block Cipher Modes of Operation

The four common modes of block cipher operation are *Electronic Codebook* (ECB), *Cipher Block Chaining* (CBC), *Cipher Feedback Mode* (CFB), *and Output Feedback Mode* (OFB) (Menezes, van Oorschot & Vanstone, 1997). All of them can be used with any block cipher. The CBC mode is the interest of this research and it will be used in the development of the prototype. This mode is the most important mode of operation (Cheswick & Bellovin, 1994). Block ciphers can be either symmetric-key or public-key.

Electronic Codebook (ECB) Mode

ECB mode is the fundamental and simplest mode. This mode operates like a codebook; for a given block of plaintext and a given key, it always produces the same block of ciphertext. Because of this weakness, ECB mode should be used only for transmission of keys and initialisation vectors (Cheswick & Bellovin, 1994). Encryption and decryption in ECB are defined as: $c_j = E_K(x_j)$ and $x_j = E^{-1}{}_K(x_j)$,

respectively, where K is the key, here. In this mode each plaintext block is enciphered independently of other block. Re-ordering ciphertext blocks results in corresponding re-ordered plaintext blocks. This mode of operation has the advantage of lack of error propagation; one or more bit errors in a single ciphertext affect decipherment of that block only. Data integrity cannot be guaranteed, since data blocks may be added, removed or modified without knowledge of the secret key.

Cipher Block Chaining (CBC) Mode

This mode was discussed in more detail because this mode was used in the development of the prototype. This mode is the most important mode of operation (Cheswick & Bellovin, 1994). CBC mode was an enhanced version of the ECB that chains together blocks of ciphertext. In the CBC mode plaintext are concealed by XOR-ing of the previous ciphertext block with the plaintext, thus precluding undetected additions or removals of ciphertext blocks, hence overcome the weakness of the ECB mode. Block i of plaintext is exclusively-ored (XORed) with block i-1 of ciphertext and is then encrypted with the keyed block encryption function to form block i of ciphertext. The processes continue until the end of the message. CBC mode involves use of n-bit *initialisation vector* (IV) or value C_0 as a "seed" to initialise the process.

The algorithm of the CBC mode is given below *initialisation vector* (IV) (Menezes, van Oorschot & Vanstone, 1997):

Input : k-bit key K; n-bit IV; n-bit plaintext blocks $x_1,..., x_t$.
Summary : produce ciphertext blocks $c_1,..., c_t$; decrypt to recover plaintext.

1. Encryption: $c_0 \leftarrow IV$. For $1 \le j \le t$, $c_j \leftarrow E_k(c_{j-1} \oplus x_j)$.
2. Decryption: $c_0 \leftarrow IV$. For $1 \le j \le t$, $x_j \leftarrow c_{j-1} \oplus E^{-1}{}_k(c_j)$.

Properties of the CBC. In this mode identical ciphertext blocks will be produced when the same plaintext is enciphered under the same key and IV (Menezes, van Oorschot & Vanstone, 1997). Different ciphertext blocks will be produced when different *initialisation vector* or key is used. ECB chaining mechanism causes ciphertext c_j to depend on x and all preceding plaintext blocks. Consequently, rearranging the order of ciphertext blocks affects decryption. Proper decryption of a correct ciphertext block requires a correct preceding ciphertext block. This chaining

mechanism hides repeated patterns. If a single bit error occurs in ciphertext block c_j , it affect decipherment of block c_j and c_{j+1} (since x_j depends on c_j and c_{j-1}). The CBC mode has a property of *self-synchronising* or *ciphertext autokey* or *self-healing* (Pleeger, 1989), in the sense that if an error (including loss of one or entire blocks) occur in block c_j but not c_{j+1}, c_{j+2} is correctly decrypted to x_{j+1}.

There are some subtle attacks possible if *initialisation vectors* (IVs) are not chosen properly; to be saved *initialisation vectors* (IVs) should be chosen randomly; not used with more than one partner; and either transmitted encrypted in ECB mode or chosen a new for each separate message, even to the same partner (Voydock & Kent, 1983).

Cipher Feedback (CFB) Mode

While CBC mode processes plaintext n bit at a time (using an n-bit block cipher), some application require that r-bit plaintext units be encrypted and transmitted without delay (for $r < n$). In this case, Cipher Feedback Mode (CFB) maybe used. In this mode, the previous ciphertext block is encrypted and the output produced is combined with the plaintext block using XOR to produce current ciphertext block. An *initialisation vector* (IV) or value c_0 is used as a "seed" for the process. In CFB mode, data can be encrypted in units smaller than the size block (64-bits). Encryption and decryption in CFB are defined as: $Ci = P_i \oplus e(Ci_{-1})$ and $Pi = C_i \oplus d(C_{i-1})$, respectively. In CFB mode, patterns are also concealed in the ciphertext by the use of the XOR operation.

Output Feedback (OFB) Mode

OFB mode is a method of running a block cipher as a synchronous stream cipher. It is similar to the CFB, except that the quantity XORed with each plaintext block is generated independently of both the plaintext and ciphertext. An *initialisation vector* (IV) S_0 is used as a "seed" for a sequence of data block S_i. The encryption of a plaintext block is derives by taking the XOR of the plaintext with the relevant data block. OFD uses DES as a random number generator, by looping its output back to its input, and exclusive-OR'ing the output with the plaintext. Encryption and decryption in OFB are defined as: $Ci = B_i \oplus Si$; $Si = DES(S_{(i-1)})$ and $Bi = C_i \oplus Si$; $Si = DES(S_{(i-1)})$, respectively. Si is the state, which is independent of either plaintext or the ciphertext. In OFB mode, patterns are also concealed in the ciphertext by the use of the XOR operation.

B.3 PUBLIC KEY CRYPTOGRAPHY

The notion of *public key cryptography* (also refers as *asymmetric cryptography*) was introduced by Diffie & Hellman (Diffie & Hellman, 1976). Public key systems, also called asymmetric systems (Simmons, 1979), differ from secret key systems in that there is no longer a single secret key shared by a pair of users. In the public key systems, each user has his owns key material. Furthermore, the key material of each user is divided into two portions, a private component known as *private key* and a public component known as *public key*. The public key and private key is used for encryption and decryption respectively and the decryption cannot be derived from the encryption key. Secret key cryptography permits the public key to be public, allowing anyone to encrypt with the key, but only the intended recipient (who knows the private key) can decrypt the message

Assume that A wishes to send a message to B. Let $E_u()$ denote the encryption under the public key of user u and $D_u()$ denote the decryption under the private key of user u. If A requires to send a message M to B, A would obtain the public key of B (either by requesting it from B, or from a directory of public keys). A would then perform the encryption:

$$E_B(M) = C$$

and send the encrypted message C to B. B would be able to recover the message by performing the decryption with the corresponding private key:

$$D_B(C) = D_B(E_B(M)) = M$$

A cryptosystems that employ public key cryptography is known *as public key cryptosystem* (PKCS). The development of public key cryptography evolved due to the limitations of the secret key cryptography. These limitations are:

- *Key distribution problem.* Key must be transmitted over a completely secure channel. If the key is revealed, the interceptors can immediately decrypt all encrypted information.
- *N-square problem.* The number of keys will increase if large number of parties required secure communication, this is because the number of keys required is proportional to the square of the participating parties (for an n-party system, $n(n\text{-}1)/2$ keys will be exchanged).
- *Authentication problem.* Symmetric cryptography cannot provide an environment where a person needs to prove to his partner that he has sent a message.

Types of public key cryptosystems are discussed in Appendix E.

B.4 CRYPTOGRAPHIC HASH FUNCTIONS

All cryptographic systems discussed above were based on reversible encryption, that is, encryption in which it is possible to recover the original message by applying a decryption transformation. In contrast, a hashing function is an example of irreversible encryption (ISO, 1989). A hash function H is a keyless transformation function that, given variable-sized message as input m, produces a fixed-sized representation of the message (generally smaller) as output, called the hash value h (i.e., message digest), so that $h = H(m)$. In other words, hash function reduces a message of arbitrary length to a fixed length value. A cryptographically strong hash function should have these properties (Rita, 1997):

- It should produce a relatively large hash function (base on the current state of technology, at least 128 bits);
- The hash $H(m)$ is relatively easy to compute for any given m;
- The hash $H(m)$ is *one-way;*
- The hash $H(m)$ is *collision-free.*

A *collision-free* hash function H is one for which it is computationally infeasible to find any two messages m and m' such that $H(m) = H(m')$. In other words, it is not possible to find two messages that hash to the same digest. A hash function H is said to be *one-way* if it hard to invert, which means that given a hash value h, it is computationally infeasible to find some input m such that $H(m) = h$.

Many hash functions have been developed to meet the criteria of strong hash functions given above. Some of the more common and widely known are as follows:

- **MD5** (Rivest, 1992). MD5 is a one-way hash function invented by Rivest for RSA Data Security, Inc. "MD" stands for "Message Digest" and the algorithm produces a 128-bit hash value for a arbitrary-length input message. MD5 is the result of a redesign of MD4 with the goal of producing a slightly slower but secure hash function. MD5 processes the input text in 512-bit blocks. The block is divided into 16 32-bit sub-blocks. It output is a set of four 32-bit blocks, which concatenated to form a single 128-bit hash value. The MD5 process of hashing can be described as follow. First the message is padded so that its length is just 64 bits short of being a multiple of 512. Padding is a process of adding a single 1-

bit to the end of the message, followed by as many zeros as are required, and finally a 64-bit of the length of the message before padding. MD5 is widely regarded as secure and is widely used.

- **Secure Hash Algorithm (NIST, 1993).** SHA (Secure Hash Algorithm) is a one-way hash function developed by National Institute of Standards and Technology (NIST) along with the National Security Agency (NSA). It has been specified for public used with the Digital Signature Standard (DSS). It produces a 160-bit hash value from an arbitrary-length message that is longer than MD5. Like MD5, SHA is based on a redesign of MD4 for greater security, however, SHA uses an additional round, has an expand transformation, and produces a 25% longer hash value.

B.5 DIGITAL SIGNATURES

A *digital signature* is a protocol[1] that produces the same effect as a real signature. It is a mark that only the senders can makes, but other people can easily recognises as belonging to the sender. A digital signature is different from a hand-written signature, in that hand-written signatures are constant, regardless of the document being signed. A user's digital signature varies with the data. A digital signature is used to confirm agreement to a message. The concept of digital signature was first discussed by Diffie and Hellman in their classic paper "New Direction in Cryptography" (Diffie & Hellman, 1976). Digital signatures use asymmetric cryptography and hash function to provide simultaneous proof of origin and data integrity.

The digital signature must meet these conditions (Pfleeger 1989, Schneier 1996):

- *Unforgeable.* If person P signs message M with signature $S(P,M)$, it is impossible for anyone else to produce their pair $[M, S(P,M)]$.
- *Authentic.* If a person R receives the pair $[M, S(P,M)]$ from P, R can check that the signature is really from P. Only P could create this signature, and the signature is firmly attached to M.
- *Not alterable.* After being transmitted, M cannot be changed by S, by R, or by any interceptor.
- *Not reusable.* A previous message presented will be instantly detected by R.

[1] Protocol - an orderly sequence of steps taken by two or more parties to accomplish some task.

- *Not repudiatable.* R doesn't need P help to verify it signature.

Digital signature algorithm is public-key algorithm with secret information to sign documents and public information to verify signatures.

The creation of a digital signature involves three steps on the part of the sender:

- The sender creates a message digest of the data to be transferred using a cryptographicly strong hashing function.
- The sender encrypts this message digest with his/her private key, producing the digital signature.
- The digital signature is appended to the data. Both the data and signature are transferred to the recipient.

Any recipient having access to the claimed sender's public key can verify the digital signature by performing three steps:

- Compute the message digest over the data portion of the received message.
- Decrypt the received digital signature using the claimed sender's public key.
- Compare the deciphered and calculated message digests. If they are equal, the signature is accepted as valid; if they are not equal, the signature is rejected as invalid (either the signature was not made by the claimed sender or the message has been altered).

The steps in the method of digital signature describes above is the practice of how digital signatures are generated. RSA Data Security Incorporated has published a de facto standard for digital signatures using MD5 and RSA that prescribes this method of digital signature generation. This digital signature method is widely adopted.

Appendix C

DATA ENCRYPTION STANDARD & INTERNATIONAL DATA ENCRYPTION ALGORITHM

C.0 INTRODUCTION

This section describes the two main secrets key block ciphers adopted in the development of the prototype in this research.

C.1 DATA ENCRYPTION STANDARD (DES)

The Data Encryption Standard (DES) resulted from IBM's submission to the 1974 U.S National Bureau of Standards (NBS) solicitation for encryption algorithms for protection of computer data. The DES was released in 1977 by the NBS, for United States Government encryption of non-classified information. It is the most well-known secret key block cipher, recognised world-wide and it set the precedent in the mid 1970s as the first commercial-grade modern algorithm with openly and fully specified implementations details (Menezes, van Oorschot & Vanstone, 1997). DES is now specified in the U.S Federal Information Processing Standards Publication 46-2 (FIPS 46-2); the same cipher is defined in the American National Standard ANSI X3.92-1981 for data encryption and referred to as the Data Encryption Algorithm (DEA). The idea being to promulgates a standard DEA for world-wide adoption by commercial and financial organisations.

C.1.1 Description of DES

The DES algorithm is a combination of two of the fundamental building block of encryption: *substitution* and *permutation*. DES is a *Fiestel cipher*[2] which processes plaintext blocks of n = 64 bits, producing 64-bit cphertext blocks (Figure E.1). The input key K for DES is specified as a 64-bit key but the effective size of key K is k = 56 bits, because every 8th bit (bit 8, 16, 24, ..., 64) is used for parity bits.

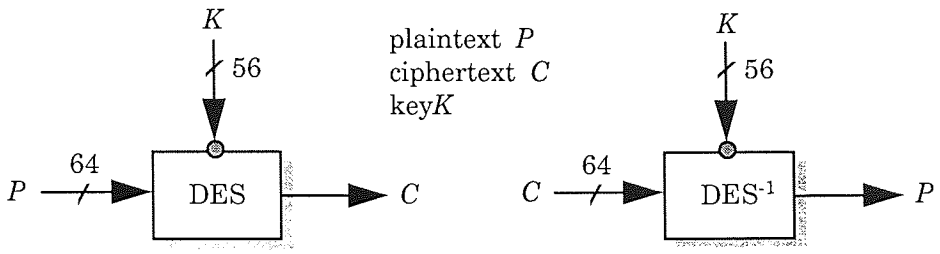

Figure C.1: DES input-output

C.1.2 The encryption and decryption process

The encryption process proceeds in 16 rounds (see Figure E.2). From the input key K, sixteen 48-bit sub-key Ki (K_1, K_2, ..., K_{16}) are generated, one for each round. At each round, 8 fixed, 6-to-4 bit substitution mapping (*S-boxes*) were selected, collectively denoted S, are used. An input data block to the DES undergoes an initial permutation, IP, and after an initial permutation, the data block is broken into two sub-blocks, the left sub-block (L_0) and the right sub-block (R_0), each 32 bits long. In each round of DES which is functionally equivalent (see Figure E.3), the 32-bit inputs L_{i-1} and R_{i-1} from the previous round is taken and producing 32-bit outputs L_i and R_i for $1 \le i \le 16$, as follows:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \text{ where } f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$$

There are actually four separate operations. First the right sub-block of the data is expanded, mapping R_{i-1} from 32 bits to 48 bits via a fixed expansion[3] permutation, E. Then it is combined with 48 bits of a shifted and permuted key via an XOR operation. The result of this operation is then sent through 8 S-boxes producing 32 new bits. The 32 bits are permuted again via a fixed permutation, P, and combined with the left sub-block. These four operations make up function

[2] A Fiestel cipher is an iterated cipher mapping a $2t$-bit plaintext (L_0, R_0), for t-bit blocks L_0, R_0, to a ciphertext (R_r, L_r), through an r-round where $r \ge 1$.
[3] Expansion - an operation of repeating certain bits.

f. The result of these operations becomes the new right sub-block; the old right sub-block becomes a new left sub-blocks. These operations are repeated 16 times, making 16 rounds of DES.

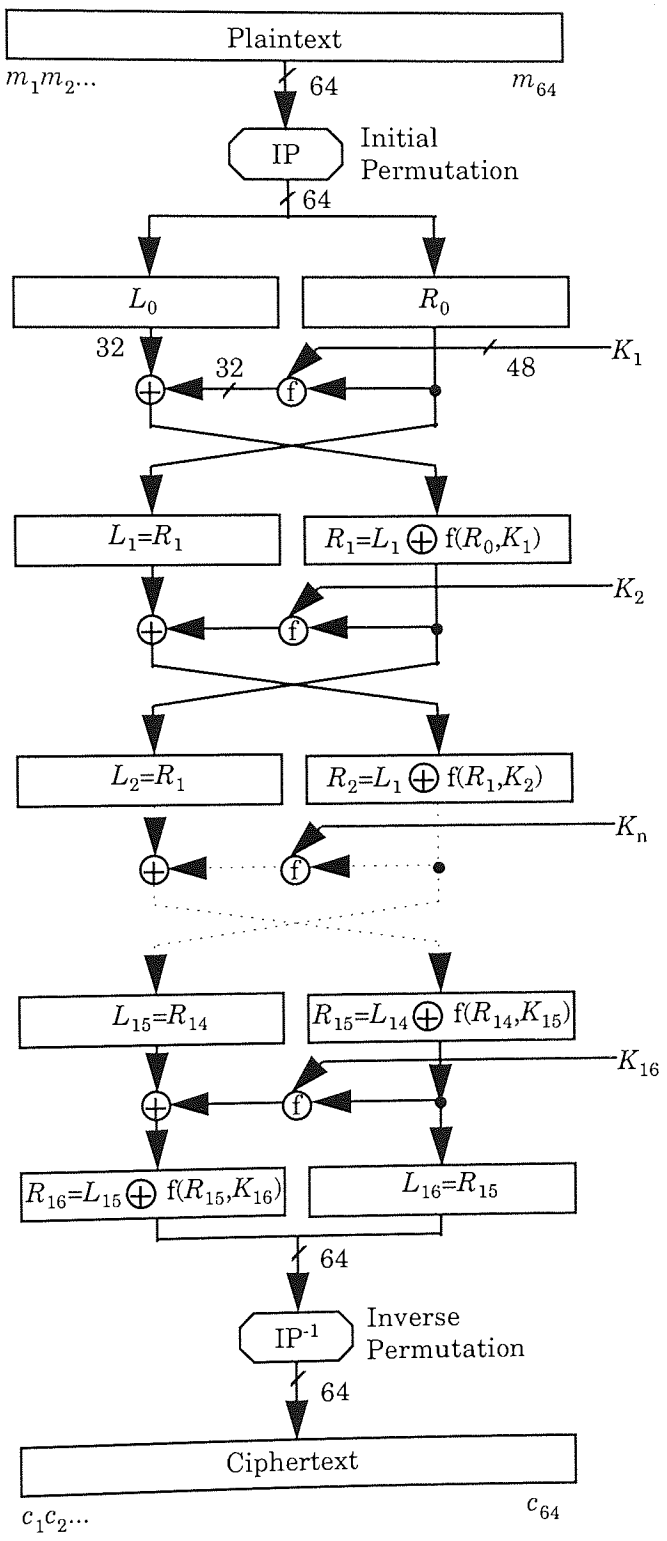

Figure C.2: DES Computation Path

The round is complete by swapping the two sub-blocks (left and right sub-block). After the sixteenth round, the right and left sub-block are joined, and then applied to inverse of initial permutation, IP^{-1}, to produce the ciphertext and finishes off the algorithm.

Decryption of the data is achieved using the same DES algorithm and key. The only changes is that the sub-keys must be used in reverse order, in which K_{16} is used in the first iteration, K_{15} in the second iteration an so forth until K_1 is used in the sixteenth iteration (K_{16}, K_{15}, ..., K_1).

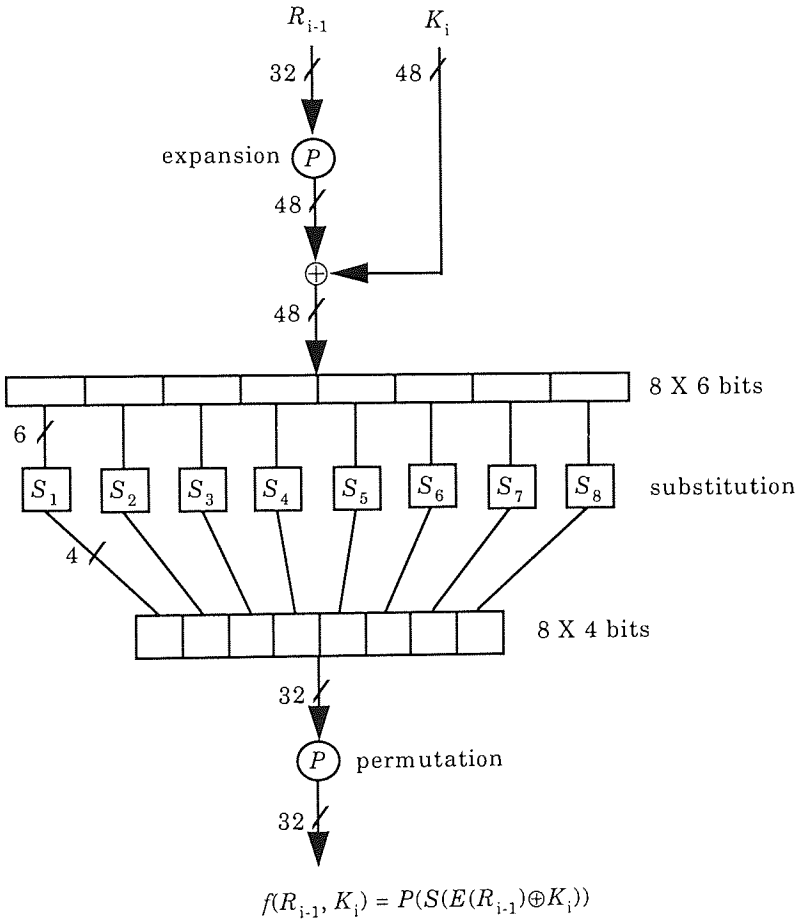

Figure C.3: DES inner function

C.2 INTERNATIONAL DATA ENCRYPTION STANDARD (IDEA)

The International Data Encryption Algorithm (IDEA[4]) was developed by Xuejia Lai and James Massey of the Swiss Federal Institute of Technology (Lai & Massey, 1990). It previous version was called PES (Proposed Encryption Standard) and later improved in 1991 (Lai & Massey, 1991) after been broken

[4] IDEA is a registered trade mark

with differential cryptanalysis and called Improved PES (IPES). IPES changed its name to IDEA in 1992 (Lai, 1992). IDEA is a symmetric-key block cipher. The IDEA cipher is an improved version of PES and was developed to increase the security against differential cryptanalysis. It encrypts 64-bit plaintext to 64-bit ciphertext blocks, using a 128-bit input key K, which is larger than 56-bit DES key. It based on the new design concept of "mixing operations from different algebraic groups". As for other block ciphers, IDEA uses both confusion and diffusion. The required confusion was achieved by successively using three "incompatible" group operations on pairs of 16-bit sub-blocks and the diffusion was achieved by the multiplication addition structure (see Figure C.4).

C.2.1 Description of IDEA

The block cipher IDEA is an iterated cipher consisting of 8 rounds followed by an output transformation (see Figure C.4). The 64-bit data block is divided into 16-bit sub-block X_1, X_2, X_3, and X_4. These four sub-blocks became the input to the first round of the algorithm. There are eight rounds total. In each round the four sub-blocks are XORed, added, and multiplied with one another and with six 16-bit sub-keys. Specifically, three algebraic groups are being mixed in this round, they are:

- bit-by-bit exclusive-or (XOR), denoted as $\oplus$;
- integer addition modulo 2^{16}, the operation is denoted as +;
- integer multiplication modulo $2^{16} + 1$, this operation could be viewed as IDEA's substitution block (S-block), the operation is denoted as .

Between rounds, the second and third sub-blocks are swapped. Finally the four sub-blocks are combined with four sub-keys in an output transformation. The same algorithm is used for both encryption and decryption.

C.2.2 IDEA Design Priciples

The design goals for IDEA can be grouped into those related to cryptographic strength and those related to ease of implementation (Stallings, 1999).

Cryptographic Strength

The following characteristics of IDEA relate to its cryptographic strength:

- ***Block length***. The block length should be long enough to deter statistical

analysis (that is, to deny the opponent any advantage that some blocks appear more often than others). The use of block size 64 bits is generally recognised as sufficiently strong. Furthermore, the use of cipher feedback mode of operation further strengthens this aspect of the algorithm

- ***Key length***. The key length should be long enough to prevent exhaustive key searches. With a length of 128 bits, IDEA seems to be secure.

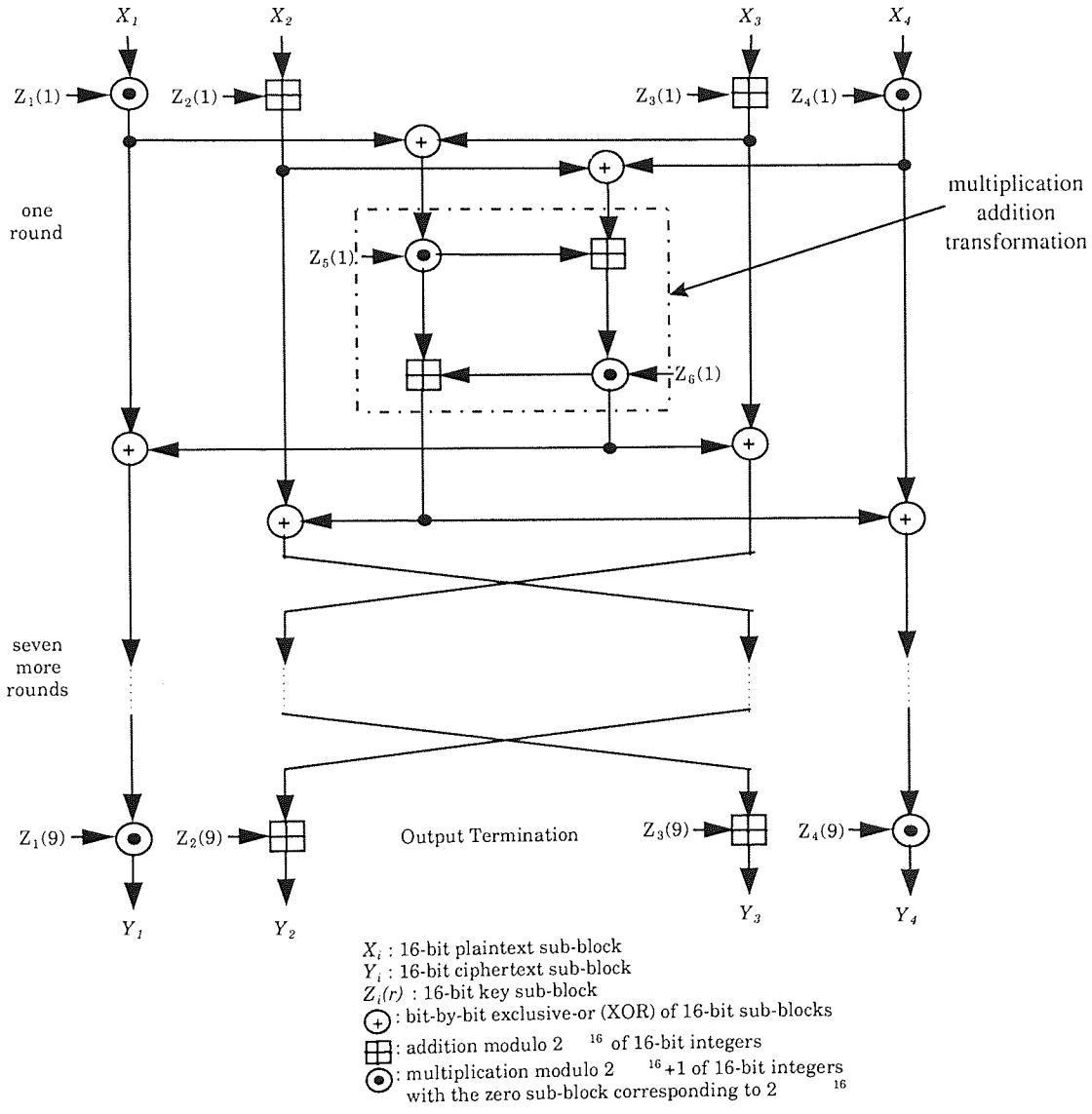

Figure C.4: The International Data Encryption Algorithm (IDEA)

- ***Diffusion***. Each plaintext bit should influence every ciphertext bit, and each key bit should influence every ciphertext bit. The spreading out of a single

plaintext bit over many ciphertext bits hides the statistical structure of the plaintext. IDEA is very effective in this regard by using the multiplication addition (MA) structure (see Figure C.4).

- ***Confusion***. The ciphertext should depend on the plaintext and key in a complicated and involved way. The objective is to complicate the determination of how statistics of the ciphertext depend on the statistics of the plaintext. IDEA achieves this goal by using three different operations, as mentioned in Section C.2.1. This is contrast to DES, which relies principally on the XOR operation and small non-linear S-boxes.

Implementation Considerations

IDEA is designed to facilitate both software and hardware implementation (Stallings, 1999). Lai & Massey (1990) cites the following design principles:

- Design principles for software implementation:
  - ◊ *Use sub-blocks*. Cipher operations should operate on sub-blocks that are "natural" for software, such as 8, 16, or 32 bits. IDEA uses 16-bit sub-blocks.
  - ◊ *Use simple operation*. Cipher should be easily programmed using addition, shifting, and so on. The three basic elements of IDEA meet this requirement.
- Design principles for hardware implementations:
  - ◊ *Similarity of encryption and decryption*. Encryption and decryption should differ only in the way of using the key so that the same device can be used for both encryption and decryption. Like DES, IDEA has a structure that satisfies this requirement.
  - ◊ *Regular structure*. The cipher should have a regular modular structure to facilitate VLSI implementation. IDEA is constructed from two basic modular building blocks repeated multiple times.

C.2.3 The encryption and decryption process

In the encryption process, three different group operations on pair of 16-bit sub-block are used. These three different group operations are:

- bit-by-bit exclusive-OR (XOR) of two 16-bit sub-blocks, denoted as ⊕.
- addition of integer modulo 2^{16} where the 16-bit sub-blocks is treated as the usual
- multiplication of integer modulo $2^{16} + 1$, this operation could be viewed as IDEA's substitution block (S-block)

The 64-bit plaintext block X is partitioned into four 16-bit sub-blocks X_1, X_2, X_3, and X_4., i.e., X = (X_1 X_2, X_3,,X_4.) (see Figure C.4). These four sub-blocks became the input to the first round of the algorithm. The four plaintext sub-blocks are then transformed into four 16-bit ciphertext sub-blocks Y_1, Y_2, Y_3, Y_4 [i.e., the ciphertext block is Y = (Y_1, Y_2, Y_3, Y_4)] using the 52 key sub-blocks of 16 bits that are formed from the 128-bit secret key.

Appendix D

OTHER SECRET KEY BLOCK CIPHERS

D.0 INTRODUCTION

This appendix describes secret key block ciphers found in the literature. Among the secret key block ciphers discussed are NewDES (Scott, 1985), FEAL (Shimizu & Miyaguchi, 1988), and SAFER (Massey, 1994).

D.1 NEWDES

NewDES was designed by Robert Scott (Scott, 1985) as a proposal to replace DES. It operates on a 64-bit block, with a relatively large 120-bit key. NewDES is simpler than DES with no initial or final permutations. Although newDES having a much larger key than DES, it was shown that NewDES was less secure compare to DES (Schneier, 1996).

In the NewDES, the input plaintext block is divided into eight 1-byte sub-blocks. The cipher consists of 17 round, and each round has eight steps. In each step, one of the sub-blocks is XORed with key material substituted with another byte via a function,

and then XORed with another sub-block to become that sub-block.

D.2 FEAL

FEAL (Fast Data Encipherment Algorithm) (Shimizu & Miyaguchi, 1988) is a DES-like block cipher which operates on 64-bit blocks using 64-bit key. The design goal of FEAL was to make a stronger round function than DES, needing fewer rounds, that results a faster cipher.

In FEAL, the encryption process starts with the 64-bit data block is

XORed with 64 key bit, and then split into two half, left- and right-half. The left-half is XORed with the right-half to form a new right-half. The left-half and the new right-half go through n rounds (4, initially) where in each round the right-half is combined with 16 bits key and XORed with the left-half to form new right-half. After n rounds the left-half (original right-half before the round) is again XORed with the right-half to form a new right-half. The left-half and the new right-half are concatenated together to form a 64-bit whole. The data block is XORed with another 64-bits of key material before terminating.

D.3 SAFER

SAFER K-64 (Secure And Fast Encryption Routine with 64-bit key) is an *iterated block cipher*[5] with 64-bit plaintext and ciphertext blocks (Massey, 1994). The algorithm has a block and key size of 64 bits.

In SAFER K-64 the plaintext is divided into eight byte-length sub-blocks, and these sub-blocks go through n rounds using two subkeys: K_{2n-1} and K_{2n} for each round. Finally, an output transformation is applied to the sub-blocks

Knudsen found a weakness in the key schedule of SAFER K-64: in every key, there exists at least one (sometimes as many as nine) other key that encrypts some different plaintext to identical ciphertexts (Knudsen, 1995). This attack did not affect SAFER's security when used as an encryption algorithm, but it greatly reduces its security when used as a one-way hash function. Vaudenay (1995) showed SAFER K-64 is weakened if the S-box mapping is replaced by a random permutation. Schneier (1996) recommend years of intense cryptanalysis of SAFER before using it in any form.

[5] Iterated block cipher is a block cipher involving the sequential repetition of an internal function called a round function.

Appendix E

PUBLIC KEY CRYPTOSYSTEM

E.0 INTRODUCTION

This section describes the RSA public key cryptosystem (Rivest, Shamir & Adleman, 1978) and the other existing public key cryptosystem found in the literature namely, Knapsack cryptosystem (Merkle & Hellman, 1978), ElGamal cryptosystem (ElGamal, 1985), and Rabin cryptosystem (Rabin, 1979).

E.1 RSA CRYPTOSYSTEM

RSA is the most popular public key cipher in use today. It is named after its inventors: Ron Rivest, Adi Shamir and Leonard Adleman. They obtained the best-known and most versatile public key cryptosystem (Rivest, Shamir & Adleman, 1978) and their work was first published in 1978 and it was the first working public key cryptosystem. It supports both secrecy and authentication, and hence can provide complete and self-contained support for public key distribution and signatures (Nechvatal, 1992). Since it is proposed RSA has withstood extensive cryptanalysis (Schneier, 1996).

E.1.1 Description of the RSA Algorithm

RSA gets it security from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large (100 to 200 digits or even larger) prime numbers. To generate the two keys

- choose p and q, two random large prime (e.g., 1024-bit) number and computes $n = pq$.
- randomly choose the encryption key, e, such that e and $(p-1)(q-1)$ are relatively prime, which means they have no prime factors in common.

- finally use extended Euclidean algorithm to compute the decryption key, d, such that $(ed-1)$ is evenly divisible by $(p-1)(q-1)$. Mathematically it is written as

$$ed = 1 \ (\text{mod}((p-1)(q-1)),$$

or in other words

$$d = e^{-1} \ \text{mod}((p-1)(q-1)).$$

The number e and n are the public key; the number d is the private key. The public key can be publish freely, because there is no known easy methods of calculating d, p, or q given only e and n (the public key). The two primes, p and q, are no longer needed but must remain secret (not to be revealed).

- The encryption function is $encrypt(t) = (t^e) \ \text{mod} \ n$, where t is the plaintext.

The decryption function is $decrypt(c) = (c^d) \ \text{mod} \ n$, where c is the ciphertext

E.2 KNAPSACK PUBLIC KEY CRYPTOSYSTEMS

The first algorithm for generalised public-key encryption was the knapsack algorithm developed by Ralph Merkle and Martin Hellman (Merkle & Hellman, 1978). The knapsack problem examines a sequence $a_1, a_2, ..., a_n$ of integers and a target sum, T. The problem is to find a vector of 0s and 1s such that the sum of integers associated with 1s equals T. That is, given $S = [a_1, a_2, ..., a_n]$, and T, find a vector V of 0s and 1s such that

$$\sum_i a_i * v_i = T$$

A **superincreasing knapsack** is a knapsack where the integer of S must form a *superincreasing sequence*, that is, one where each integer is greater than the sum of all preceding integers. Then, every integer a_k would be of the form

$$a_k \underset{i=1}{\overset{k-1}{>}} \Sigma a_j$$

The solution of a superincreasing knapsack (also called a ***simple knapsack***) is easy to find. Start with T. Compare the largest integer in S to it. If the is larger than T, it is not in the sum, so let the corresponding position in V be 0. If the largest integer is less than or equal to T, that integer is in the sum, so let the corresponding position in V be 1 and reduce T by the integer. Repeat for all remaining integers in S.

E.3 THE ELGAMAL CYPTOSYSTEM

The security of ElGamal (1985) public key scheme is based on calculating discrete logarithms in a finite field. ElGamal scheme can be used for both digital signature and encryption.

Encryption. First, prime number, p, and two random number, g and x (g and x are less than p). Then calculate

$$y = g^x \bmod p$$

The value y, g, and p are the public key and x is the private key.

Encryption. To encrypt a message, M, a random number k, such that k is relatively prime to p -1 is selected. Then compute

$$a = g^k \bmod p$$

$$b = y^k M \bmod p$$

The pair a and b, is the ciphertext. To decryption of a and b is obtained by compute $M = b/a^x \bmod p$. Since $a^x \equiv g^{kx} \bmod p$,

$$\begin{aligned} b/a^x &\equiv y^k M/ a^x \bmod p \\ &\equiv g^{xk} M/g^{xk} \bmod p \\ &\equiv M \bmod p. \end{aligned}$$

E.4 THE RABIN CYPTOSYSTEM

The Rabin cryptosystem was proposed in 1979 by Rabin (Rabin, 1979). Rabin's scheme gets it security from the difficulty of finding square roots modulo a composite number.

Key generation. First, choose two large random primes p and q, each roughly the same size. Then compute $n = pq$. n is the public key and (p,q) is the private key.

Encryption. To encrypt a message m, compute $c = m^2 \bmod n$.

Decryption. To recover a plaintext m from c, using the Chinese remainder theorem to find the four square roots m_1, m_2, m_3, and m_4 of c modulo n as follow:

$$m_1 = c^{(p+1)/4} \bmod p$$

$$m_2 = (p - c^{(p+1)/4}) \bmod p$$

$$m_3 = c^{(q+1)/4} \bmod q$$

$m_4 = (p - c^{(q+1)/4}) \bmod q$

Then chose an integer $x = q(q^{-1} \bmod p)$ and an integer $y = p(p^{-1} \bmod q)$. Then one of the four results produced below is equal to m.

$M_1 = (xm_1 + ym_3) \bmod n$

$M_2 = (xm_1 + ym_4) \bmod n$

$M_3 = (xm_2 + ym_3) \bmod n$

$M_4 = (xm_2 + ym_4) \bmod n$

Williams (1980) proposed a modified that has the advantage over Rabin's scheme that there is an easy way procedure to identify the intended message from the four roots of a quadratic polynomial. The Rabin's and modified Rabin's scheme by Williams are vulnerable to chosen ciphertext attack. According to Schneier (1996), if it is used for digitally signing others messages, an attacker may request a signature and obtain the secret key.

Appendix F

pbeWithMD5andDES-CBC

F.0 INTRODUCTION

This appendix discusses one of the key-encryption algorithm defines by PKCS #5: Password-Based Encryption Standard known as MD5 with DES-CBC (pbeWithMD5andDES-CBC) (RSA, 1993). The standard describes a method for encrypting an octet string with a secret key derived from a password. The result of the method is an octet string. The pbeWithMD5andDES-CBC algorithms employ DES secret-key encryption in cipher-block chaining mode, where the secret key is derived from a password with a MD5 message-digest algorithm.

F.1 GENERAL OVERVIEW

The next section specifies the encryption and the decryption process. Each entity shall privately select an octet P as its "password". The password is an arbitrary octet string, and it need not be a printable "word" in the usual sense. The octet string may contain zero or more octets.

Each entity shall also select an eight-octet string S as its salt value and a positive integer c as its iteration count. The 'salt' value S and the iteration count c are parameters of the algorithm identifier for the password-based encryption algorithm.

An entity's password, salt value, and iteration count may be different for each message the entity encrypts.

The encryption and decryption processes shall both be performed with the password, salt value, and iteration count as a key for the password-based encryption algorithm. Both processes transform an octet string to another octet string. The processes are inverses of one another if they use the same key.

F.2 ENCRYPTION PROCESS

The encryption process consists of three steps: DES key generation, encryption-block formatting, and DES encryption. The input to the encryption process shall be an octet string M, the message; an octet string P, the password; an octet string S, the salt value; and an integer c, the iteration count. The output from the encryption process shall be an octet string C, the ciphertext.

F.2.1 DES key generation

A DES key K and an initialising vector IV shall be generated from the password P, the salt value S, and the iteration count c. The key generation process shall involve the following steps:

1• The octet string $P \,||\, S$ (concatenation of P, S) shall be digested with c iterations of the selected message-digest algorithm. "One iteration" of the message-digest algorithm is just the ordinary message digest; "two iterations" is the message digest of the message digest and so on.

2• The least significant bit of each of the first eight octets of the result of step 1 shall be changed if necessary to give the octet odd parity, as required by DES. The resulting eight octets shall become the DES key K.

3• The last eight octets of the result of step 1 shall become the initialising vector IV.

F.2.2 Encryption-block Formatting

The message M and a padding string PS shall be formatted into an octet string EB, the encryption block.

$$EB = M \,||\, PS. \tag{1}$$

The padding string PS shall consist of $8 - (||M||^{6} \bmod 8)$ octets all is having value $8 - (||M|| \bmod 8)$. This makes the length of the encryption block EB a multiple of eight octets. In other words, the encryption block EB satisfies one of the following statements:

$$EB = M \,||\, 01 \text{ — if } ||M|| \bmod 8 = 7;$$
$$EB = M \,||\, 02\ 02 \text{ — if } ||M|| \bmod 8 = 6;$$

[6] $||M||$ is length in octets of M.

.

.

$$EB = M \,||\, 08\ 08\ 08\ 08\ 08\ 08\ 080\ 08 \text{ — if } ||\, M \,||\, \text{mod } 8 = 0;$$

Note. The encryption block can be parsed unambiguously since every encryption block ends with a padding string and no padding string is a suffix of another.

F.2.3 DES Encryption

The encryption block *EB* shall be encrypted under DES in cipher-block chaining mode with key *K* and initialising vector *IV*. The result of encryption shall be an octet string *C*, the ciphertext.

Note. The length of the ciphertext *C* is a multiple of eight octets.

F.3 DECRYPTION PROCESS

The decryption process consists of three steps: DES key generation, DES decryption, and encryption-block parsing. The input to the decryption process shall be an octet string *C*, the ciphertext; an octet string *P*, the password; an octet string *S*, the salt value; and an integer *c*, the iteration count. The output from the decryption process shall an octet string *M*, the message.

For brevity, the decryption process is described in terms of the encryption process.

F.3.1 DES Key Generation

A DES key *K* and an initialising vector *IV* shall be generated from the password *P*, the salt value *S*, and the iteration count *c* as described for the encryption process.

F.3.2 DES Decryption

The ciphertext *C* shall be decrypted under DES in cipher-block chaining mode with key *K* and initialising vector *IV*. The result of decryption shall be an octet string *EB*, the encryption block.

It is an error if the length of the ciphertext *C* is not a multiple of eight octets.

F.3.3 Encryption-block Parsing

The encryption block *EB* shall be parsed into an octet string *M*, the message, and an octet string *PS*, the padding string, according to Equation (1).

It is an error if the encryption block cannot be parsed according to Equation (1), i.e., if the encryption block does not end with k octets all having value k for some k between 1 and 8.

Appendix G

CLASSICAL CIPHERS

G.0 INTRODUCTION

There are two basic components of classical cipher techniques: *substitution* and *transposition.*

G.1 SUBSTITUTION CIPHERS

A substitution cipher is one in which character in the plaintext is substituted for another character in the ciphertext (Schneier, 1996). Examples of substitution ciphers are the *Caesar cipher*, *mono-alphabetic substitution cipher*, and *poly-alphabetic substitution cipher.*

G.1.1 The Caesar Cipher

Caesar cipher is the simplest substitution cipher. Caesar cipher replace letters of a message by others letters in a symmetric fashion. In this cipher each plaintext character is replaced by the character three to the right modulo 26, "A" is replaced by "D", "B" is replaced by "E", "W" is replaced by "Z", ..., "X" is replaced by "A", "Y" is replaced by "B" and so on in the cyclic form.

If, in the Caesar cipher, the letters A, B, C, ... are given numbers 0, 1,2, ... the process of encipherment can be expressed as $c = p + 3$ (modulo 26), where the addition requires 26 to be substituted if the result exceeds 25.

Crytanalysis of Caesar cipher is effectively a key search; the displacement is the key and possible values are tested until the right one is found by inspecting the plaintext produced in each case.

G.1.2 Mono-alphabetic Substitution Cipher

Mono-alphabetic substitution cipher or simple substitution cipher is one in which each character of the plaintext is replaced with corresponding character of ciphertext (Scheier, 1996). Thus the alphabetic displacement is dependent upon the letter concerned. For mono-alphabetic substitution cipher, for English language, there are 26! or about 4×10^{26} possible alphabets, whereas the Caesar cipher has only 25 possible alphabets for the English language.

Cryptanalysis of such a mono-alphabetic substitution cipher is achieved by counting the letter frequency of the plaintext, which will be of the same frequency as the language of the plaintext, but in shuffled order.

G.1.3 Poly-alphabetic Substitution Cipher

Polyalphabetic substitution is made up of multiple simple substitution cipher. An example of poly-alphabetic substitution cipher is Vigenére cipher. Application of Vigenére cipher is facilitated by using tableau, in which all the possible displaced alphabets are tabulated as shown in Table G.1.

In Vigenére encipherment the key word determines which displaced alphabet is used to encipher each successive letter of the plaintext. A keyword is used for selecting the column of the matrix and is repeated as many times as necessary to encrypt the whole message. The plaintext character at the left-hand-side of the table specifies the alphabet and the ciphertext character is determined by the selected column by the character of the key word.

For example using a key "cscw", the plaintext message "three black mice" is enciphered as follows:

Plaintext	:	three black mice
Keyword	:	cscwc scwcs cwcs
Ciphertext	:	jllsy rrwai qoao

Crytanalysis on a Vigenére cipher can be carried out by choosing a word which likely to be in the plaintext messages and then carries out modulo 26 subtraction of that word from the ciphertext in possible location, i.e. a scan is made of the entire ciphertext. If at any position the result of the subtraction produce a word of natural language or a fragment of such word, then it is likely that the keyword or part of it has been discovered.

	ABCDEFGHIJKLMNOPQRSTUVWXYZ
A	ABCDEFGHIJKLMNOPQRSTUVWXYZ
B	BCDEFGHIJKLMNOPQRSTUVWXYZA
C	CDEFGHIJKLMNOPQRSTUVWXYZAB
D	DEFGHIJKLMNOPQRSTUVWXYZABC
E	EFGHIJKLMNOPQRSTUVWXYZABCD
F	FGHIJKLMNOPQRSTUVWXYZABCDE
G	GHIJKLMNOPQRSTUVWXYZABCDEF
H	HIJKLMNOPQRSTUVWXYZABCDEFG
I	IJKLMNOPQRSTUVWXYZABCDEFGH
J	JKLMNOPQRSTUVWXYZABCDEFGHI
K	KLMNOPQRSTUVWXYZABCDEFGHIJ
L	LMNOPQRSTUVWXYZABCDEFGHIJK
M	MNOPQRSTUVWXYZABCDEFGHIJKL
N	NOPQRSTUVWXYZABCDEFGHIJKLM
O	OPQRSTUVWXYZABCDEFGHIJKLMN
P	PQRSTUVWXYZABCDEFGHIJKLMNO
Q	QRSTUVWXYZABCDEFGHIJKLMNOP
R	RSTUVWXYZABCDEFGHIJKLMNOPQ
S	STUVWXYZABCDEFGHIJKLMNOPQR
T	TUVWXYZABCDEFGHIJKLMNOPQRS
U	UVWXYZABCDEFGHIJKLMNOPQRST
V	VWXYZABCDEFGHIJKLMNOPQRSTU
W	WXYZABCDEFGHIJKLMNOPQRSTUV
X	XYZABCDEFGHIJKLMNOPQRSTUVW
Y	YZABCDEFGHIJKLMNOPQRSTUVWX
Z	ZABCDEFGHIJKLMNOPQRSTUVWXY

Table G.1: A Vigenére Tableau

G.2 TRANSPOSITION CIPHERS

Transposition ciphers aim to hide the contents of a message by taking the individual characters or bits and rearranging their order (Davies & Price, 1989). Examples of transposition ciphers are *simple columnar transposition cipher* and *the Nihilist cipher*.

G.2.1 Simple Columnar Transposition Cipher

The simple columnar transposition cipher takes successive letter, groups into block of fixed length and rearranges horizontally, to produce a table. The

ciphertext is read off vertically. Encryption is done by writing the ciphertext vertically of the same fixed length and then reading the plaintext horizontally.

For example, the following plaintext message: ASTON UNIVERSITY BIRMINGHAM is grouped in blocks of five and arranged as follows:

Plaintext : ASTON UNIVERSITY BIRMINGHAM

c1	*c2*	*c3*	*c4*	*c5*	(*c* column)
A	S	T	O	N	
U	N	I	V	E	
R	S	I	T	Y	
B	I	R	M	I	
N	G	H	A	M	

Ciphertext : AURBNSNSIGTIIRHOVTMANEYIM

Furthermore, the group can be rearranged in the sequence of specified columns. This sequence constitutes the encipherment key. If the order of 41532 is used in the above example, the ciphertext will be as follow:

	c4	*c1*	*c5*	*c3*	*c2*	(*c* column)
key	4	1	5	3	2	
	O	A	N	T	S	
	V	U	E	I	N	
	T	R	Y	I	S	
	M	B	I	R	I	
	A	N	M	H	G	

Ciphertext : OVTMAAURBNNEYIMTIIRHSNSIG

When a plaintext is not in multiple of the block length, a character such as "X" is padded at the last block. This cipher could be applied more than one times with different key in very round.

G.2.2 The Nihilist Cipher

The Nihilist is a combination of columnar and row transposition (Davies & Price, 1994). The plaintext is written in rows under a key word, but the key word is

also written vertically at the side of the columns. The same key word can be used in the case of two transpositions. Ciphertext is read out row by row similar to columnar transposition, but with the order of row selection determined by the key word written vertically alongside.

For example, the following plaintext message: ASTON UNIVERSITY BIRMINGHAM is grouped in blocks of five and rearranged in the order 41532 as follows:

Plaintext :	ASTON UNIVERSITY BIRMINGHAM					
Key word	L	E	M	O	N	
	c4	*c1*	*c5*	*c3*	*c2*	(*c* column)
L 4	O	A	N	T	S	
E 1	V	U	E	I	N	
M 5	T	R	Y	I	S	
O 3	M	B	I	R	I	
N 2	A	N	M	H	G	
Ciphertext :	VUEINANMHGMBIRIOANTSTRYIS					

Appendix H

SECURITY ATTACKS

H.0 INTRODUCTION

This appendix describes some of the existing security attacks. Among the attacks are brute-force attack, masquerade attack, dictionary attack, man-in-the-middle-attack, replay attack, and cryptanalytic attack.

H.1 BRUTE FORCE ATTACK.

A brute force attack is one in which an attacker searches the entire cryptographic key space until the correct key is found. It is called a brute force because it does not require any particular knowledge about the key being used, but requires the ability to try all the possible permutations for a particular length of key, given knowledge (or assumptions) about the particular cryptographic algorithm employed. This will result in a trade-off between the required processing power and the time required to performed the calculations. Any realistic cryptosystem will inevitably require substantial amounts of both processing power and time in order for a brute force to be mounted. Some cryptosystems may prevent certain brute force attacks by means of protocols which prevent repetitive attempts to use a key in a given period of time e.g. Bank automatic teller machines prevent any more than three attempts to enter the correct PIN.

H.2 DICTIONARY ATTACK

Dictionary attack is a general threat to all passwords. An attacker who obtains some sensitive password-derived data, such as a hashed-password, performs a series of

computations using every possible guess for the password. Since passwords are typically small by cryptographic standards, the password can often be determined by brute-force. Depending on the system, the password, and the skills of the attacker, such an attack can be completed in days, hours, or perhaps only a few seconds.

The term dictionary attack initially referred to finding passwords in a specific list, such as an English dictionary.

A password database should always be kept secret to prevent dictionary attack on the data. Obsolete password methods also permit dictionary attack by someone who eavesdrops on the network. Strong methods prevent this.

H.3 MAN-IN-THE-MIDDLE ATTACK

The Man-In-The-Middle attack occurs when an adversary acts as a third party in a two party conversation. Both legitimate parties assume that they are talking securely with each other; in fact the adversary is intercepting the entire conversation, decrypting it, re-encrypting it and sending it on to the intended recipient.

For example, in public-key protected communication: Bob wants to talk to Alice, but Bob needs Alice's public-key first. Eve intercepts Bob's request for Alice's public-key and instead sends Bob her public-key. Bob uses Eve's public key to encrypt his conversation, thinking that it is Alice's. Eve can now intercept all Bob's messages to Alice, decrypt them, re-encrypt them using Alice's real public-key and send them on to Alice. An entire conversation can pass like this without anyone realising what is happening.

To get round this problem a certificate is used to identify a person (or e-mail address) with a public key. What makes the certificate binding is that it is digitally signed by a certification authority. A certificate will typically consist of the person's name and public key, the certification authoritiy name and public key and the digital signature. The signature is calculated by hashing the data on the certificate and encrypting the result with the certification authority's private key. The certificate can then be verified by a third party by encrypting the signature block with the certification authority's public key (this amounts to decryption) and comparing the result with the third party's own hash calculation of the certificate. Note that more than one person can sign a certificate and there is nothing to prevent someone signing their own certificate.

H.4 MASQUERADE ATTACK

A masquerade attack is one in which an attacker poses as a legitimate entity, in order to either gain access to the particular data belonging to the user or to gain access to the system in general.

H.5 REPLAY ATTACK

Replay attack is an attack that comprises the recording and replaying of previously sent messages. Any constant authentication information, such as a password, a one-way hash of a password, or electronically transmitted biometric data, can be recorded and replayed.

H.6 CRYPTANALYTIC ATTACKS

A cryptographic attack is where an attacker attempts to gain knowledge about the cryptosystem and/or the transmitted data by statistical analyses of the transmitted data. This may or may not be complemented by a prior knowledge of details of the particular cryptographic algorithm(s) employed. Such statistical analyses may be categorised as follows:

- ***Ciphertext-only attack.*** In this type of attack, the attacker has knowledge only of the ciphertext that has been intercepted. There is no knowledge of the plaintext corresponding to the ciphertext, although there may be some knowledge about the algorithms employed in the cyptosystem concerned. Any such attack on a cryptosystem would have to rely purely on a statistical analysis of whatever ciphertext was recovered, perhaps on the basis of assumptions regarding the nature of the corresponding plaintext.

- ***Known-plaintext attack.*** In this type of attack, the attacker has knowledge both of the recovered ciphertext and some of all of its corresponding plaintext. Such knowledge would afford the attacker greater opportunity for statistical analysis of the cryptographic algorithm(s) employed by the cryptosystem, of the cryptographic key(s) used, and of any ciphertext to which the corresponding plaintext is known.

- ***Chosen-plaintext attack.*** In this type of attack, the attacker not only has knowledge both of the ciphertext and its corresponding plaintext, but also is able

to choose what the plaintext shall be. This would afford the attacker the greatest opportunity for statistical analysis of the cyrptographic algorithm(s) employed by the cyrptosystem and of the cryptographic key(s) used, and would hence lead to a possible situation of the attacker being able to recover plaintext from a subsequent ciphertext-only attack on the same cryptosystem, perhaps not even using the same cryptographic key(s).

- ***Adative-chosen-plaintext attack.*** This is a special case of a chosen-plaintext attack. Not only can the cryptanalyst choose the plaintext that is encrypted, but he can also modify his choice based on the results of previous encryption. In a chosen-plaintext attact, a cryptanalyst might just be able to choose one large block of plaintext to be encrypted; in an adaptive-chosen-plaintext he can choose a smaller block of plaintext and then choose another based on the results of the first, and so forth.

- ***Chosen-ciphertext attack.*** The cryptanalyst can choose different ciphertexts to be decrypted and has access to the decrypted plaintext.

Appendix I

SECURESIG USER MANUAL

I.0 INTRODUCTION

This manual provides information on how to use the Secure-SIG prototype. This manual is divided into two parts:

1. **Server Setup** and
2. **System Login**.

The *Server Setup* part is the setting up of the Secure-SIG server. The *System Login* part is the step-by-step of how user can use the system. This part is intended for the member of the inspection team. SecureSIG is a UNIX platform system.

I.1 SERVER SETUP

There are eight (8) servers needed to be setup before the system can be used. These servers are *AccessServer*, *BriefingServer*, *CodeLoaderServer*, *LoadedCodeServer*, *RemoteLoadedCodeServer*, *LogEntryServer*, *ChatServer*, *EmailServer*.

All the servers are running using UNIX host at `kojak.aston.ac.uk`. Follow the procedures given below to setup all the servers:

1. Open system console (*New system console should be opened for each server*).
2. Telnet server host by typing

   ```
   > telnet kojak
   ```

   and press RETURN key.

The system will ask for *username* and *password*. Key in the following username: *maarofma* and password: *jaisal95*.

To run all the servers follow the following steps:

```
> cd project/classes <ret>
> SSIG_Server <ret>
```

I.2 SYSTEM LOGIN

This section decribes how user (member of the inspection team) can login to the SecureSIG system.

I.2.1 Start the System

To section describes how to generate user key pair, and then how to access the system.

Getting the StartUp Frame

To start the SecureSIG system, open a new console, telnet **kojak** (used the above username and password in Section I.1), and type the following command at the prompt:

```
> StartSSIG
```

and press RETURN key.

The system will display a welcome frame. Three options will be given. These options are **SETUP**, **LOGIN** and **CANCEL LOGIN**.

*** **IMPORTANT** *** : It is important to perform the SETUP option before proceeding with LOGIN option.

- **SETUP** button is used to generate key pair
- **LOGIN** button is used to start login to the system, and
- **CANCEL LOGIN** button is used to cancel login.

Click the option button to activate the option.

SETUP Option

This option describes how to generate user key pair and data file encryption. If this option is press, a SETUP frame will be displayed consists of three options, **Key Pair Generation**, **Data File Encryption**, and **Exit**.

***Note**: Data File Encryption only for moderator. This option must be perform first by the moderator before other members of the group can login to the system.*

Key Pair Generation

To generate key pair, follow these instructions:

1. Key in *username* and *passphrase key* at the username and passphrase field allocated (all characters, symbols and numbers are allowed for both username and passphrase). The passphrase key is used as key to save the private key. A maximum of 60 characters can be used as a passphrase key. A choice of username and passphrase key in Table I.1 below can be used. This same passphrase key or the passphrase key of your choice should be used when requested in the login process.

role	*username*	*passphrase key*
moderator	zaini	twinkle twinkle little star
author	salwa	humpty dumpty set on the wall
inspector	doherty	ba ba black sheep have you any wool
inspector	paul	jack and jill went up the hill
inspector	zahid	incy wincy spider
inspector	vasilas	three blind mouse

Table I.1: List of username and passphrase key

2. To create the key pair, click

   Create

3. To exit from the frame, click

   Exit

Data File Encryption

Two files need to be encrypted before proceed using the system. These files are:

- `usrPasswd.dat` and
- `currentUsr.dat`

The files can be encrypted by using the following step:

1. Click the following option

   Encrypt Data File

2. A StoreDataFile frame will be displayed.

   Enter data file to be encrypted, `usrPasswd.dat`, at the filename field allocated.

3. To activate the encryption process, click

   Encrypt

4. To clear the filename field, click

   Reset

   Enter the second the data file to be encrypted, `currentUsr.dat`, by repeating step 3 and 4.

5. To exit from the frame, click

   Exit

Exit

This option is used to exit from this frame.

LOGIN Option

This option describes how to access the system. If this option is press, a system login frame will be displayed. To access the system, follow these instructions:

1. Enter your username and the password in the space allocated (see Table I.2 for a valid username and password).

username	*password*
zaini	alpha
salwa	beta
doherty	gamma
paul	tango
zahid	vectra
vasilas	epsilon

Table I.2: List of username and password

2. To activate the login process, click

   Submit

3. To clear the field, click

   Reset

4. A new frame, request for the passphrase key will be displayed,

5. To proceed, enter the passphrase key to the space allocated.

   (***Important***: The same passphrase key used when the pre-generated key pair was generated during the SETUP process (in SETUP option) must be entered.

6. To proceed, click

   Submit

The systems will only allowed a user to attempt to access the system for maximum of three times. Every invalid login will be displayed with a denied access frame. The third invalid login will result the system to exit automatically.

For a valid login, the system will displayed a frame according to the role of the valid user (e.g. *moderator*, *author*, or *inspector*).

Moderator Frame

There are nine (8) buttons on this frame. They are **Generate Key Pair** button, **File Encryption** button, **Briefing** button, **Code Viewer** button, **Remote Viewer** button, **Log Entry** button, **Chat** button and **Exit** button.

Generate Key Pair

This option is used to regenerate a new key pair to replace the old key pair. To regenerate the key pair, do the following:

1. Click the following option from the menu

   Generate Key Pair

2. The regenerate key pair frame will be displayed. Key in *username* and *passphrase key* at the username and passphrase field allocated.

3. To activate the regeneration of the key pair, click

   Submit

4. To cancel regeneration of the key pair, click

   Cancel

File Encryption

The moderator must perform file encryption first before other members of the group can participate in the inspection process.

To activate the file encryption process click the following option from the menu

File Encryption

A new frame with four (4) options will be displayed. These four options are the **Encrypt Briefing File**, the **Encrypt Program File**, the **Encrypt Data File**, and the **Exit**.

Encrypt Briefing File

To encrypt a briefing file:

1. Click the following option from the menu

   Briefing File

2. A new frame will be displayed, request for the file to be encrypted. Type in the briefing file name, `Introduction`, at the filename field allocated.

3. To activate the encryption process, click

   Encrypt

4. To clear the filename field, click

   Reset

   Enter the other briefing filename, `Briefing-1`, `Briefing-2`, and `Briefing-3` one at a time, by repeating step 3 and 4.

5. To exit from the frame, click

   Exit

Encrypt Program File

To encrypt a program file:

1. Click the following option from the menu

   Program File

2. A new frame will be displayed, request for the file to be encrypted. Type in the program file name, `Client.java`, at the filename field allocated.

3. To activate the encryption process, click

   Encrypt

4. To clear the filename field, click

   Reset

   Enter the other program filename, `Server.java`, `Kunci.java`, and `CipherIS.java` one at a time, by repeating step 3 and 4.

5. To exit from the frame, click

   Exit

Briefing

To access the briefing material, follow the following step:

1. Click the following option from the main menu

   Briefing

2. A briefing session frame will be displayed.

3. Select the briefing filename by using the filename button on this frame. Click the filename to confirm selecting the file.

4. To load the file, click

   Fetch

5. The document will be displayed. The windows can be left open or closed down after finishing with the materials.

6. To exit from this frame, click

   Exit

Code Viewer

The document viewer can be accessed using the following step:

1. Click the following option from the main menu

   Code Viewer

2. A document viewer session frame will be displayed.

3. Select the program file by using the filename button on this frame. Click the filename to confirm selecting the file.

3. To load the file, click

   Fetch

4. The document will be displayed. The windows can be left open or closed down after finishing with the materials.

5. To exit from this frame, click

   Exit

Remote Viewer

The remote document viewer can be accessed using the following step:

1. Click the following option from the main menu

   Remote Viewer

2. A remote code viewer session frame will be displayed.
3. Select the name of another user from the list of user currently using the system.
4. To proceed with the process, click the username to fetch the program file viewed by this user.
5. The document will be displayed. The windows can be left open or closed down after finishing with the materials.
6. To exit from this frame, click

   Exit

Log Entry

All the defects and queries found during the inspection process is stored in the log file. To add a log entry, follow the following step:

1. Click the following option from the main menu frame

   Log Entry

2. The log entry frame will be displayed.
3. Enter the defect information at the allocated field in the frame.

   The following information has to be entered.

   - Username field
   - Log filename
   - Program line number
   - Select the option from the Critically option. There are three (3) options: *Major*, *Minor*, and *Warning*.
   - Select the option from the Type option. There are nine (9) options given. They are *Data*, *Design*, *Documentation*, *Language*, *Logic*, *Performance*, *Test & Branch*, *Maintainability* and *Other*.

- Enter comment at the Comment text area
- Enter suggestion at the Suggestion text area

4. To proceed with the process, click the following button

   Submit

5. To clear the entry, click the following button

   Clear

6. To exit from this frame, click

   Exit

Chat

The chat can be accessed by using the following steps:

1. Click the following option from the main menu frame

   Chat

2. A chat frame will be displayed.
3. Enter your name and press **Return** key.
4. To start the chat session, click

   Start Chat

A frame for chatting will be displayed. To start chatting type message at the field provided at the bottom of the frame and press **Return** key. To see the active users in the chat session type ?who and press **Return** key. To exit from this chat session frame press the **Exit** button on the top of the frame.

Author Frame

There are seven (7) buttons on this frame. They are **Generate Key Pair** button, **Briefing** button, **Code Viewer** button, **Remote Viewer** button, **Log Entry** button, **Chat** button and **Exit** button. All these buttons perform the same actions as in the Moderator Frame.

Inspector Frame

There are seven (7) buttons on this frame. They are **Generate Key Pair** button, **Briefing** button, **Code Viewer** button, **Remote Viewer** button, **Log Entry** button, **Chat** button and **Exit** button. All these buttons perform the same actions as in the Moderator Frame.

Appendix J

SECURESIG EVALUATION GUIDELINE

1. Start Secure Software Inspection Groupware (SecureSIG)
   - i. Open console
   - ii. Type **StartSSIG**

2. Select **SETUP** from menu
   - i. Select Generate Key Pair button
     - a. Enter Username
     - b. Enter Passphrase Character
     - c. Click EXIT button

3. Select **LOGIN** from menu
   - i. Enter Username
   - ii. Enter Password
   - iii. Click Submit
   - iv. Enter Passphrase Character
   - v. Click Submit button

4. Select **Generate Key Pair** from menu
   - i. Enter username
   - ii. Enter passphrase character

iii. Click Generate button

5. Select **File Encrytion** from menu (only moderator has this option)

i. Select **Encrypt Briefing File** from menu

a. Enter Briefing filename

b. Click Encrypt button

c. Click Reset button

d. Repeat step a to c for other Briefing filname

e. Click Exit button

ii. Select **Encrypt Program File** from menu

a. Enter Program filename

b. Click Encrypt button

c. Click Reset button

d. Repeat step a to c for other Program filname

e. Click Exit button

6. Select Briefing from menu

i. Select Code inspection process description

ii. Select SecureSIG description

iii. Select inspection group information

iv. Select overall project description

v. Select current code inspection description

Asynchronous Evaluation

7. Select Code Viewer from menu

i. Enter Username

ii. Select the file to view

iii. Fetch the file

8. Select Log Entry from menu
   - i. Enter Username
   - ii. Enter Filename
   - iii. Create a comment and enter into log
   - iv. Log another comment

9. Select E-mail from menu
   - i. Send e-mail to yourself
     - a. Enter your e-mail address
     - b. Select your key
   - ii. Select e-mail to your colleague
     - a. Enter your colleague e-mail address
     - b. Select his/her key
   - iii. Open mail box and read messages send previously in step 7i

Synchronous Evaluation

10. Select Remote Viewer from menu
    - i. Select file which is currently viewed by other group member
    - ii. Select other file

11. Select Chat from menu
    - i. Enter your name
    - ii. Press RETURN key
    - iii. Click Start Chat
    - iv. Chat with your colleague

**** End of evaluation. Thank you. ****

Appendix K

USER DETAILS

Please tick ([ ]) the following which is/are relevant to you

[ ] I am an undergraduate [ ] I am a postgraduate

[ ] I am staff (Please specify: _____)

[ ] Others (Please specify: _____)

Gender: [ ] Male [ ] Female

Qualification/Course Attended: _____

Which platforms do you usually work on?

[ ] UNIX [ ] PC [ ] Macintosh [ ] Others (_____)

How many years in IT/Computer related fields:

[ ] 1 - 2 [ ] 3 - 4 [ ] 4 - 5 [ ] 7 -

Which area/s do you have experience in:

[ ] Programming
[ ] System Analysis
[ ] Network Administrator
[ ] S/W Project Manager
[ ] Others (Please specify: _____)

[ ] Data Security
[ ] Computer Security
[ ] Network Security
[ ] Cryptography

Have you used any inspection or review method in S/W quality process?

[ ] Yes [ ] No

If Yes, what type of inspection of review method have you been using so far?

[ ] Software Review/Inspection
[ ] Code Inspection
Others (Please specify: _____)

[ ] Format Technical Review
[ ] Software Audit

Appendix L

PROTOTYPE EVALUATION QUESTIONNAIRE

Please answers the following question by circle your choice.

Use the following rating scale for question 1 – 15:

1 – Strongly Agree
2 – Agree
3 – Not Sure
4 –Disagree
5 – Strongly Disagree

1. Do you think that it is require to protect the briefing material by encrypted all the related briefing material in an asynchronous meeting?

   (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

2. Do you think that it is important to secure the information flow that is related to the briefing material in an asynchronous meeting?

   (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

3. Do you think that the remote viewer that helps you in accessing the file viewed by other group member needs to secure it files by encrypted all the related files?

   (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

4. Do you think that there is a need to secure all the information flow involved in the remote viewer process?

   (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

5. Do you think that there is a need to secure all the information flow in the e-mail facility in SecureSIG?

   (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

6. Do you think that there is a need to protect the log entry files and provide secure information flow in the log entry process?

   (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

7. Do you think that the SecureSIG components provided offered a secure working environment for asynchronous mode of inspection?

   (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

8. Do you think that it is important to protect the briefing material by encrypted all the related briefing material in synchronous meeting?

   (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

9. Do you think there is a need to secure the information flow that is related to the briefing material in synchronous meeting?

   (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

10. Do you think that the document viewer that helps you viewing the document need to secure it files by encrypted all the related files?

    (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

11. Do you think that there is a need to secure all the information flow involved in the document viewer process?

    (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

12. Do you think that there is a need to secure all the information flow in the group chat facility in SecureSIG?

    (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

13. Do you think that the SecureSIG components provided offered a secure working environment for synchronous mode of inspection?

    (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

14. Overall, do you think that with all the secure components provided the SecureSIG offered a secure working environment for an asynchronous and synchronous mode of process?

    (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

15. Do you think there is a need to provide a secure System Access Control to SecureSIG?

    (Strongly Agree) 1 2 3 4 5 (Strongly Disagree)

Use the following rating scale for question 16 – 19:

1 – Not Aware
2 – Slightly Aware
3 – Don't Know
4 –Aware
5 – Fully Aware

16. During login to the SecureSIG system do you aware the encryption and decryption process involved in the login process?

(Not Aware) 1 2 3 4 5 (Fully Aware)

17. When using all the components in the SecureSIG do you aware the process of fetching the public key pair need for the encryption process?

(Not Aware) 1 2 3 4 5 (Fully Aware)

18. Do you aware the encryption processed in the system in each component of the system that you choose?

(Not Aware) 1 2 3 4 5 (Fully Aware)

19. Do you aware the process of generating and updating of the public key pair during generate and re-generate key pair process?

(Not Aware) 1 2 3 4 5 (Fully Aware)

Use the following rating scale for question 20:

1 – Highly Accessible
2 – Easily Accessible
3 – Neither Good or Bad
4 –Not Easily Accessible
5 – Difficult to Access

20. When using the SecureSIG components either the secure briefing, secure document viewer, and secure remote viewer component or during login process can you access the encrypted files related to them every time?

(Accessible) 1 2 3 4 5 (Not Accessible)

Use the following rating scale for question 21:

1 – Very Transparent
2 – Transparent
3 – Not Sure
4 – Slightly Transparent
5 – Not Transparent

21. Overall, do you think the encryption process in the SecureSIG system is transparent to you?

(Very Transparent) 1 2 3 4 5 (Not Transparent)

Please write any comment about the system in the space below, if you have any.

Appendix M

RESULT OF THE QUESTIONNAIRE

M.0 INTRODUCTION

This appendix consists of the test users details information (Section M.1) and the result from the prototype evaluation by the test users (Section M.2).

M.1 TEST USERS DETAIL INFORMATION

USER DETAILS	TU1	TU2	TU3	TU4	TU5	TU6
Post-G	X	X	X	X	X	X
Under-G						
Gender	M	M	M	M	M	M
Platform	UNIX	UNIX	UNIX	UNIX	UNIX	UNIX
Qualification/Course Attended	CS	CS	CS	CS	CS	CS
IT experience	O4 - O5	> 7	> 7	> 7	O4 - O5	O4 -O5
Information Technology Experience						
Programming	X	X	X	X	X	X
System Analysis		X	X			
Network Admins.			X	X		
S/W Proj. Man.		X				
Data Security						
Computer Security			X			
Network Sec.	X		X			
Cryptography	X		X			
Used any inspection tool		X		X		
S/W review/inspec.						
Code Inspection			*X*	X		
Format Tech. Rev.						
Software Audit		X				

Table M.1: Users Detail Information

M.2 PROTOTYPE EVALUATION RESULT

PROTOTYPE EVALUATION	TU1	TU2	TU3	TU4	TU5	TU6
Prototype Suitability						
Asynchronous						
Q1	2	2	1	2	2	2
Q2	2	2	1	2	2	2
Q3	1	1	1	2	2	2
Q4	1	2	1	2	2	2
Q5	1	2	1	2	2	2
Q6	2	2	2	2	2	3
Q7	2	2	1	2	2	2
Synchronous						
Q8	2	2	1	2	2	2
Q9	2	2	1	2	2	2
Q10	1	1	1	1	2	2
Q11	1	2	1	2	2	2
Q12	2	2	2	2	3	3
Q13	2	2	1	2	2	3
Q14	2	1	2	2	2	3
Q15	1	1	1	1	1	2
Prototype Transparency						
Q16	2	2	2	2	2	3
Q17	1	1	1	1	1	1
Q18	1	1	1	1	1	1
Q19	1	1	2	2	2	3
Q20	1	1	1	2	2	2
Q21	1	1	2	2	2	2

Table M.2: Prototype Evaluation Result

Appendix N

Program Listing

N.0 INTRODUCTION

This appendix contains java programs of Secure-SIG prototype. Section N.1 list the program modules and in Section N.2 because of the size of the whole program are big only some are included and most of the programs are kept in the diskette accompanied.

N.1 PROGRAM MODULE

In this section all the program modules for SecureSIG prototype are listed. These includes user authentication module, key pair generation module, file encryption module, briefing module, code viewer module, remote code viewer module, log entry module, e-mail module, and other utilities module.

1. User Authentication Module

   - ValidateUserFrame.java
   - AccessServer.java
   - AccessServerThread.java
   - AccessValShare.java
   - AccessValShare.java
   - Protection.java

2. Key Pair Generation Module

   - StoreUserKeyPair.java

3. File Encryption Module

   - fileEncryptFrame.java
   - StoreBRIEFINGfile.java
   - StorePROGRAMfile.java

4. Briefing Module

   - BriefingFrame.java
   - BriefingServer.java
   - BriefingServerThread.java

5. Code Viewer Module

   - CodeLF.java
   - CodeLoadedServer.java
   - CodeLoadedSvrThread.java
   - LoadedCodeServer.java
   - LoadedCodeSvrThread.java

6. Remote Code Viewer Module

   - RemoteCLF.java
   - RemoteCodeServer.java
   - RemoteCodeSvrThread.java

7. Chat Module

   - ChatServer.java
   - GroupChat.java
   - SClient.java
   - Mservice.java
   - Info.java
   - EnterString.java
   - Notify.java
   - TService.java
   - IOException.java

8. Log Entry Module

   - AddLogFrame.java
   - LogEntryServer.java
   - LogEntrySvrThread.java

9. E-mail Module

   - CipherMail.java
   - Message.java
   - Composer.java
   - SMTP.java
   - POP3.java

10. Other Utilities Module

    - StartSSIG.java
    - StoreDATAfile.java
    - GenerateUserKeyCert.java

N.2 PROGRAM LISTING

This section only included some of the Java program listed in Section N.1.

Key Pair Generation Module

storeUserKeyPair.java

```
//------------------------------------------------------
// This program generate and store RSA public key pair.
//------------------------------------------------------

import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
import java.math.*;
import java.security.*;
import java.math.BigInteger;
import java.security.cert.CertificateException;

import javax.crypto.*;
import iaik.x509.*;
import iaik.asn1.structures.*;
import iaik.asn1.*;
import iaik.x509.extensions.*;
import iaik.security.rsa.*;
import iaik.security.cipher.*;
import iaik.pkcs.*;
import iaik.pkcs.pkcs1.*;
import iaik.pkcs.pkcs8.*;
import iaik.utils.KeyAndCertificate;
```

```
import iaik.security.provider.IAIK;

public class storeUserKeyPair extends Frame
                        implements ActionListener {

  protected boolean inAnApplet = true;
  protected TextField passField = null;
  protected TextField aliasField = null;
  protected TextArea contentArea = null;
  protected TextField contentLength = null;
  protected TextField contentSize = null;
  protected TextField userField = null;
  protected Choice fileListChoice;
  protected Button bFetch, bDelete, bExit;
  protected String InputFile, OutputFile;
  protected PrivateKeyInfo priv_keyInf = null;
  protected RSAPrivateKey privKey = null;
  protected RSAPrivateKey privateKEY = null;
  protected Label mStatusLabel;
  public static int saveFormat = ASN1.PEM;
  protected String passphrase;
  protected String nameAlias;
  protected String alias = "";

  public storeUserKeyPair(String alias) {

    nameAlias = alias;
    passphrase = "Aston University";

    setLayout(new BorderLayout());

    //Add small things at the bottom of the window.
    int startPos = 0;
    int endPos;

    Panel p0 = new Panel();
    p0.add(bFetch = new Button("GENERATE KEY PAIR"));
    bFetch.addActionListener(this);
    p0.add(new Label("      "));
    p0.add(bExit = new Button("CANCEL"));
    bExit.addActionListener(this);
    add("North", p0);

    Panel p1 = new Panel();
    p1.add(mStatusLabel = new
      Label("                                                    "));
    add("South", p1);

    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        if (inAnApplet) {
          dispose();
        } else {
          System.exit(0);
        }
      }
    });
    }


 // add component into gridbaglayout
  private void addComp (Component c, GridBagLayout gbl,
        GridBagConstraints gbc, int x, int y, int w, int h) {

    gbc.gridx = x;
    gbc.gridy = y;
    gbc.gridwidth = w;
    gbc.gridheight = h;
    gbl.setConstraints (c, gbc);
    add (c);
  }


    public void actionPerformed(ActionEvent event) {

      String command = event.getActionCommand();

      if (event.getSource() instanceof TextField || command ==
                                        "GENERATE KEY PAIR") {
          doKeyAndCertificate(nameAlias);
          dispose();
      }
      if (command == "CANCEL") {
          dispose();
      }
```

```
    }


  //-----------------------------------------------------------------
  // saveKeyAndCert() method :
  // Encrypt the private key and save the key and the certificate   // chain to a file.
  //-----------------------------------------------------------------

  public static void saveKeyAndCert(KeyPair keyPair,
       X509Certificate[] chain, String fileName, String                    passphrase)
throws IOException {

    EncryptedPrivateKeyInfo epki = new
     EncryptedPrivateKeyInfo((PrivateKeyInfo)keyPair.getPrivate());

    try {
      epki.encrypt(passphrase, AlgorithmID.pbeWithMD5AndDES_CBC,
       null);
    } catch (NoSuchAlgorithmException ex) {
      throw new RuntimeException("No implementation for
                    pbeWithMD5AndDES_CBC!");
    }

    // append the correct extension
    fileName = fileName + (saveFormat == ASN1.DER ? ".der" : ".pem");

    System.out.println("save private key and certificate chain to   file
"+fileName+"...");
    new KeyAndCertificate(epki, chain).saveTo(fileName, saveFormat);
  }


  //-----------------------------------------------------------------
  // generateKeyPair() method:
  // Generates a Key pair for the requested public key algorithm.
  //-----------------------------------------------------------------

  public KeyPair generateKeyPair(String algorithm, int bits) throws
                    NoSuchAlgorithmException {

    KeyPairGenerator generator = null;
    try {
      generator = KeyPairGenerator.getInstance(algorithm, "IAIK");
    } catch (NoSuchProviderException ex) {
      throw new NoSuchAlgorithmException(ex.toString());
    }
    generator.initialize(bits);
    KeyPair kp = generator.generateKeyPair();
    return kp;
  }


  //-----------------------------------------------------------------
  // verifyCertificateChain() method:
  // Verifies a chain of certificates where the user certificate.
  //-----------------------------------------------------------------

  public boolean verifyCertificateChain(X509Certificate[] certs) {

    try {
      int anz = certs.length;
      verifyCertificate(certs[anz-1], null);
      for (int i=anz-1; i>0; i--)
        verifyCertificate(certs[i-1], certs[i]);
      System.out.println("Verify certificate chain OK!");
    } catch (SignatureException ex) {
      System.out.println("Verify certificate chain ERROR!");
      return false;
    }
    return true;
  }


  //-----------------------------------------------------------------
  // verifyCertificate() method:
  //                 Verifies the digital signature of a certificate.
  //-----------------------------------------------------------------

  public void verifyCertificate(X509Certificate userCert,
              X509Certificate caCert) throws  SignatureException {

    try {
      if (caCert == null)
        userCert.verify();   // self signed
      else
        userCert.verify(caCert.getPublicKey());
```

```
    } catch (Exception ex) {
      throw new SignatureException(ex.toString());
    }
  }


  //----------------------------------------------------------------
  // createCertificate() method:
  // Creates a test certificate according to the X.509 Notation.
  //----------------------------------------------------------------

  public X509Certificate createCertificate(Name subject, PublicKey          pk, Name
issuer, PrivateKey sk, AlgorithmID algorithm,
              int serialNumber) throws CertificateException {

    boolean extensions = false;
    X509Certificate cert = new X509Certificate();

    try {
      cert.setSerialNumber(BigInteger.valueOf(serialNumber));
      cert.setSubjectDN(subject);
      cert.setPublicKey(pk);
      cert.setIssuerDN(issuer);

      GregorianCalendar date = new GregorianCalendar();
      date.add(Calendar.DATE, -1);
      cert.setValidNotBefore(date.getTime()); // not before yesterday

      date.add(Calendar.MONTH, 6);
      cert.setValidNotAfter(date.getTime());

      if (extensions) {          // add some v3 extensions
        byte[] id = {1,2,3,23,3,4,3,23,3};
        SubjectKeyIdentifier ski = new SubjectKeyIdentifier(id);
        cert.addExtension(ski);

        BasicConstraints bc = new BasicConstraints(true, 1);
        bc.setCritical(true);
        cert.addExtension(bc);

        KeyUsage ku = new KeyUsage( KeyUsage.digitalSignature |
                                    KeyUsage.keyCertSign |
                                    KeyUsage.cRLSign);
        cert.addExtension(ku);
      }
      cert.sign(algorithm ,sk);

      return cert;

    } catch (InvalidKeyException ex) {
      throw new CertificateException(ex.toString());
    } catch (NoSuchAlgorithmException ex) {
      throw new CertificateException(ex.toString());
    }
  }

  //----------------------------------------------------------------
  // doKeyAndCertificate() method:
  // Creates some test certificate chains and writes them into .PEM        // files.
  //----------------------------------------------------------------

  public void doKeyAndCertificate(String nameAlias) {

    try {
      System.out.println("add Provider IAIK...\n");
      Security.addProvider(new IAIK());

       boolean create_rsa = true;

      // create a test directory
      File file = new File("certs");
      if (!file.exists())
        file.mkdir();

      // First create the private keys
      KeyPair caRSA = null;
      KeyPair userKey1 = null;
      KeyPair userKey2 = null;

      try {
          caRSA = generateKeyPair("RSA", 512);
      } catch (NoSuchAlgorithmException ex) {
        System.out.println("No implementation for RSA! Can't create
              RSA certificates!\n");
        create_rsa = false;
```

```
        }

        if (create_rsa) {
            userKey1 = generateKeyPair("RSA", 512);
            userKey2 = generateKeyPair("RSA", 512);
        }


        // Now create the certificates
        Name issuer = new Name();
        issuer.addRDN(ObjectID.country, "AT");
        issuer.addRDN(ObjectID.organization ,"TU Graz");
        issuer.addRDN(ObjectID.organizationalUnit ,"IAIK");
        issuer.addRDN(ObjectID.commonName ,"IAIK Test Certification
          Authority");

        Name userSubject = new Name();
        userSubject.addRDN(ObjectID.country, "AT");
        userSubject.addRDN(ObjectID.organization ,"TU Graz");
        userSubject.addRDN(ObjectID.organizationalUnit ,"IAIK");

        // create self signed CA cert
        X509Certificate caRSACert = null;
        X509Certificate caDSACert = null;
        X509Certificate[] chain = new X509Certificate[1];

        if (create_rsa) {
          System.out.println("create self signed CA certificate...");
          caRSACert = createCertificate(issuer, caRSA.getPublic(),
                issuer, caRSA.getPrivate(),
          AlgorithmID.md5WithRSAEncryption, 1);
          chain[0] = caRSACert;
          saveKeyAndCert(caRSA, chain,
          "project/classes/certs/caRSAcert", passphrase);
        }

        // create user certificates
        chain = new X509Certificate[2];
        chain[1] = caRSACert;

        if (create_rsa) {
          userSubject.addRDN(ObjectID.commonName , "User - RSA/RSA");
          System.out.println("create User certificate [RSA/RSA]...");
          chain[0] = createCertificate(userSubject,
          userKey1.getPublic(),issuer, caRSA.getPrivate(),
          AlgorithmID.md5WithRSAEncryption, 3);
          userSubject.removeRDN(ObjectID.commonName);
          verifyCertificateChain(chain);
          saveKeyAndCert(userKey1, chain, "project/classes/certs/" +
          nameAlias + "RSAcert1", passphrase);
        }

        if (create_rsa) {
          userSubject.addRDN(ObjectID.commonName , "User - RSA/RSA");
          System.out.println("create User certificate [RSA/RSA]...");
          chain[0] = createCertificate(userSubject,
          userKey2.getPublic(),issuer, caRSA.getPrivate(),
          AlgorithmID.md5WithRSAEncryption, 3);
          userSubject.removeRDN(ObjectID.commonName);
          verifyCertificateChain(chain);
          saveKeyAndCert(userKey2, chain, "project/classes/certs/" +
          nameAlias + "RSAcert2", passphrase);
        }
        System.out.println("\nKey and Certificates created.");
      } catch (Exception ex) {
        System.out.println("Exception: "+ex);
      }
    }
  } // end storUserKeyPair

// This program is an alteration of IAIK CreateCertificates.java
// Copyright (C) 1997 by DI Wolfgang Platzer, IAIK
// email: wplatzer@iaik.tu-graz.ac.at
//
// All rights reserved.
```

File Encryption Module

fileEncryptFrame.java

```
//-------------------------------------------------
// This program create a frame for file encryption
//-------------------------------------------------

import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;


public class fileEncryptFrame
             extends Frame implements ActionListener {

  protected boolean inAnApplet = true;
  protected Frame win, briefWin, progWin;

  public fileEncryptFrame() {

    // --- graphical user interface

    Button encBrief = new Button("Encrypt Briefing File");
    encBrief.addActionListener(this);

    Button encProg = new Button("Encrypt Program File");
    encProg.addActionListener(this);

    Button encData = new Button("Encrypt Data File");
    encData.addActionListener(this);

    Button exit = new Button("EXIT");
    exit.addActionListener(this);

    win = new Frame();
    win.setLayout(null);
    win.add(encBrief);
    win.add(encProg);
    win.add(encData);
    win.add(exit);

    // manually layout

    encBrief.setBounds(55, 40, 140, 40);
    encBrief.validate();

    encProg.setBounds(55, 80, 140, 40);
    encProg.validate();

    encData.setBounds(55, 120, 140, 40);
    encData.validate();

    exit.setBounds(55, 165, 140, 30);
    exit.validate();

    win.setLocation(250,170);
    win.setTitle("File Encryption");
    win.setSize(250,210);
    win.show();

    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        if (inAnApplet) {
          dispose();
        } else {
          System.exit(0);
        }
      }
    });

  }


  // Take action

   public void actionPerformed(ActionEvent evt) {
```

```
    String command = evt.getActionCommand();

    // submit request

    if (command == "Encrypt Briefing File") {
      doEncryptBriefing();
    }

    if (command == "Encrypt Program File") {
      doEncryptProgram();
    }

    if (command == "EXIT") {
      win.dispose();
    }
  }


  //-----------------------------------------
  // doEncryptBriefing() method :
  // calling frame to encrypt Briefing material
  //-----------------------------------------

  public void doEncryptBriefing() {

    briefWin = new StoreBRIEFINGfile();
    briefWin.setTitle("Briefing File Encryption");
    briefWin.setLocation(460, 220);
    briefWin.setSize(360, 110);
    briefWin.setVisible(true);
  }

  //-------------------------------------------------
  // doEncryptProgram() method :
  // calling frame to encrypt program code/document
  //-------------------------------------------------

  public void doEncryptProgram() {

    progWin = new StorePROGRAMfile();
    progWin.setTitle("Program File Encryption");
    progWin.setLocation(460, 220);
    progWin.setSize(360, 110);
    progWin.setVisible(true);
  }

}   // end fileEncryptFrame class
```

StoreBRIEFINGfile.java

```
//-------------------------------------------------------
// This program encrypt and store the briefing material
//-------------------------------------------------------

import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
import java.math.*;
import java.security.*;

import javax.crypto.*;
import javax.crypto.spec.*;
import iaik.security.rsa.*;
import iaik.x509.*;
import iaik.security.cipher.*;
import iaik.pkcs.*;
import iaik.pkcs.pkcs1.*;
import iaik.pkcs.pkcs8.*;
import iaik.asn1.*;
import iaik.asn1.structures.*;
import iaik.security.provider.IAIK;
import iaik.utils.KeyAndCertificate;

public class StoreBRIEFINGfile extends Frame
                        implements ActionListener {

  protected boolean inAnApplet = true;
```

```
protected TextField codeField = null;
protected TextArea contentArea = null;
protected TextField contentLength = null;
protected TextField contentSize = null;
protected TextField userField = null;
protected Choice fileListChoice;
protected Button bFetch, bReset, bExit;
protected String FileName;
protected RSAPrivateKey privKey = null;
protected boolean kpNotEXIST = true;
protected boolean notEXIST = true;
protected PublicKey publicKEY = null;
protected KeyPair kp = null;
protected PublicKey getPub = null;
protected Key key = null;
protected KeyAndCertificate KeyCert;
protected byte[] cipherIDEAkey = null;
protected ObjectOutputStream oos = null;
protected IvParameterSpec spec = null;
static byte[] iv = { (byte)0xfe, (byte)0xdc, (byte)0xba, (byte)0x98,
                     (byte)0x76, (byte)0x54, (byte)0x32, (byte)0x10 };

public StoreBRIEFINGfile() {

  // --- security provider
  Provider iaik = new iaik.security.provider.IAIK();
  Security.addProvider(iaik);

  // --- graphical user interface
  setLayout(new BorderLayout());

  Panel p = new Panel();
  p.add("North", new Label("Filename:"));
  codeField = new TextField(30);
  codeField.addActionListener(this);
  p.add(codeField);
  add("North", p);

  Panel p0 = new Panel();
  p0.add(new Label("    "));
  p0.add(bFetch = new Button(" ENCRYPT "));
  bFetch.addActionListener(this);
  p0.add(bReset = new Button(" RESET "));
  bReset.addActionListener(this);
  p0.add(bExit = new Button(" EXIT "));
  bExit.addActionListener(this);
  add("South", p0);

  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
      if (inAnApplet) {
        dispose();
      } else {
        System.exit(0);
      }
    }
  });
  }


//-------------------------------------------------------
// actionPerformed() method: performing the action taken
//-------------------------------------------------------

public void actionPerformed(ActionEvent event) {

  String command = event.getActionCommand();

  if (event.getSource() instanceof TextField || command == "
                                              ENCRYPT ") {

    doEncrypt();
  }

  if (command == " RESET ") {
    doResetField();
  }

  if (command == " EXIT ") {
    storeIDEAkey();
    dispose();
  }
}


//-----------------------------------
```

```
// doResetField() method: reset field
//-----------------------------------

public void doResetField() {
  codeField.setText("");
}


//-----------------------------------------------------
// doEncrypt() method:
// start create and store key used to encrypt the file
//-----------------------------------------------------

public void doEncrypt() {

  createIDEAkey();
  storeIDEAkey();

}


//-------------------------------------------
// createIDEAkey() method:
// creating key for file encryption (IDEA key)
//-------------------------------------------

public void createIDEAkey() {

  try {
    if (notEXIST) {
     try {
       KeyGenerator keygen = KeyGenerator.getInstance("IDEA",
                                                      "IAIK");
       key = keygen.generateKey();
       notEXIST = false;
     } catch (Exception ex) {
       System.out.println(ex);
     }
    } else {
       System.out.println("Use old IDEA key ...");
       notEXIST = false;
    }

    //Encrypt file
    try {
       FileName = codeField.getText();

       Cipher c = Cipher.getInstance("IDEA/CBC/PKCS5Padding");
       spec = new IvParameterSpec(iv);
       c.init(Cipher.ENCRYPT_MODE, key, spec);

       FileInputStream fis = new FileInputStream("project/classes/"
                                                 + FileName);
       CipherInputStream cis = new CipherInputStream(fis, c);
       FileOutputStream fos = new
            FileOutputStream("project/classes/files/"
                                               + FileName);

     byte[] buffer = new byte[2048];
     int r;
     while ((r = cis.read(buffer)) != -1)
       fos.write(buffer, 0, r);
     fis.close();
     cis.close();
     fos.flush();
     fos.close();
    } catch (Exception ex) {
     System.out.println(ex);
    }
  } catch (Exception e) {
    System.out.println(e);
  }
}


//---------------------------------------
// storeIDEAkey() method: store IDEA key
//---------------------------------------

public void storeIDEAkey() {

  // encrypt IDEA key

  try {
    KeyCert = new
```

```
        KeyAndCertificate("project/classes/certs/serverRSAcert.pem");
      System.out.println("Encrypting IDEA key ....");
      Cipher RSACipher = Cipher.getInstance("RSA");

      X509Certificate c = KeyCert.getCertificateChain()[0];
      PublicKey publicKey = c.getPublicKey();

      RSACipher.init(Cipher.ENCRYPT_MODE, publicKey);
      cipherIDEAkey = RSACipher.doFinal(key.getEncoded());
      System.out.println("DONE encrypting IDEA key ....");
    } catch (Exception ex) {
      System.out.println(ex);
    }

    // create DES file
    try {
      oos = new ObjectOutputStream(
            new FileOutputStream
                ("project/classes/files/briefIDEAkey.enc"));
      oos.writeObject(cipherIDEAkey);
      oos.close();
    } catch (Exception ex) {
      System.out.println(ex);
    }
  }

} // end StoreBRIEFINGfile class
```

StorePROGRAMfile.java

```
//-------------------------------------------------------
// This program encrypt and store program code/document
//-------------------------------------------------------

import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
import java.math.*;
import java.security.*;

import javax.crypto.*;
import javax.crypto.spec.*;
import iaik.security.rsa.*;
import iaik.x509.*;
import iaik.security.cipher.*;
import iaik.pkcs.*;
import iaik.pkcs.pkcs1.*;
import iaik.pkcs.pkcs8.*;
import iaik.asn1.*;
import iaik.asn1.structures.*;
import iaik.security.provider.IAIK;
import iaik.utils.KeyAndCertificate;

public class StorePROGRAMfile extends Frame
                        implements ActionListener {

  protected boolean inAnApplet = true;
  protected TextField codeField = null;
  protected TextArea contentArea = null;
  protected TextField contentLength = null;
  protected TextField contentSize = null;
  protected TextField userField = null;
  protected Choice fileListChoice;
  protected Button bFetch, bReset, bExit;
  protected String FileName;
  protected RSAPrivateKey privKey = null;
  protected boolean notEXIST;
  protected boolean kpNotEXIST = true;
  protected boolean DESnotEXIST = true;
  protected PublicKey publicKEY = null;
  protected KeyPair kp = null;
  protected PublicKey getPub = null;
  protected Key key = null;
  protected KeyAndCertificate KeyCert;
  protected IvParameterSpec spec = null;
  static byte[] iv = { (byte)0xfe, (byte)0xdc, (byte)0xba,(byte)0x98,
                       (byte)0x76, (byte)0x54, (byte)0x32, (byte)0x10 };

  public StorePROGRAMfile() {
```

```
  // --- security provider
  Provider iaik = new iaik.security.provider.IAIK();
  Security.addProvider(iaik);

  // --- graphical user interface
  setLayout(new BorderLayout());

  Panel p = new Panel();
  p.add("North", new Label("Filename:"));
  codeField = new TextField(30);
  codeField.addActionListener(this);
  p.add(codeField);
  add("North", p);

  Panel p0 = new Panel();
  p0.add(new Label("     "));
  p0.add(bFetch = new Button(" ENCRYPT "));
  bFetch.addActionListener(this);
  p0.add(bReset = new Button(" RESET "));
  bReset.addActionListener(this);
  p0.add(bExit = new Button(" EXIT "));
  bExit.addActionListener(this);
  add("South", p0);

  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
      if (inAnApplet) {
        dispose();
      } else {
        System.exit(0);
      }
    }
  });
  }


//-------------------------------------------------------
// actionPerformed() method: performing the action taken
//-------------------------------------------------------

public void actionPerformed(ActionEvent event) {

  String command = event.getActionCommand();

  if (event.getSource() instanceof TextField || command == "
                                                    ENCRYPT ") {
    doEncrypt();
  }

  if (command == " RESET ") {
    doResetField();
  }

  if (command == " EXIT ") {
    storeDESkey();
  dispose();
  }
}


//-----------------------------------
// doResetField() method: reset field
//-----------------------------------

public void doResetField() {
  codeField.setText("");
}


//-------------------------------------------------
// doEncrypt() method:
// perform the document encryption and store process
//-------------------------------------------------

public void doEncrypt() {

  createDESKey();
  storeDESkey();

}


//-----------------------------------------------
// createDESKey() method:
```

```
// creating key for document encryption (DES key)
//-----------------------------------------------

public void createDESKey() {

  try {
    if (DESnotEXIST) {
      try {
        KeyGenerator keygen = KeyGenerator.getInstance("DES",
                                                       "IAIK");
        keygen.init(new SecureRandom());
        key = keygen.generateKey();
        DESnotEXIST = false;
      } catch (Exception ex) {
        System.out.println(ex);
      }
    } else {
      DESnotEXIST = false;
    }

    //Encrypt file
    try {
        FileName = codeField.getText();

        Cipher c = Cipher.getInstance("DES/CBC/PKCS5Padding");
        spec = new IvParameterSpec(iv);
        c.init(Cipher.ENCRYPT_MODE, key, spec);

        FileInputStream fis = new FileInputStream("project/classes/"
                                                  + FileName);
        CipherInputStream cis = new CipherInputStream(fis, c);
        FileOutputStream fos = new
      FileOutputStream("project/classes/files/" + FileName);

      byte[] buffer = new byte[2048];
      int r;
      while ((r = cis.read(buffer)) != -1)
        fos.write(buffer, 0, r);
      fis.close();
      cis.close();
      fos.flush();
      fos.close();
    } catch (Exception ex) {
      System.out.println(ex);
    }
  } catch (Exception e) {
    System.out.println(e);
  }
}


//--------------
// store DES key
//--------------

public void storeDESkey() {

  // encrypt DES key
  byte[] CipherDESKey = null;
  ObjectOutputStream oos;

  try {
    KeyCert = new
      KeyAndCertificate("project/classes/certs/serverRSAcert.pem");
    System.out.println("Encrypting DES key ....");
    Cipher RSACipher = Cipher.getInstance("RSA");

    X509Certificate c = KeyCert.getCertificateChain()[0];
    PublicKey publicKey = c.getPublicKey();

    RSACipher.init(Cipher.ENCRYPT_MODE, publicKey);
    CipherDESKey = RSACipher.doFinal(key.getEncoded());
    System.out.println("DONE encrypting DES key ....");
  } catch (Exception ex) {
    System.out.println(ex);
  }

  // create DES file
  try {
    oos = new ObjectOutputStream(
          new FileOutputStream
              ("project/classes/files/desKEY.enc"));
    oos.writeObject(CipherDESKey);
    oos.close();
  } catch (Exception ex) {
```

```
      System.out.println(ex);
    }
  }

} // end StorePROGRAMfile class
```

Code Viewer Module

CodeLF.java

```
//-----------------------------------------------------------
// This program create a frame and perform the viewing of
// code/document.
//-----------------------------------------------------------

import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
import java.security.*;
import java.security.spec.*;

import javax.crypto.*;
import javax.crypto.spec.*;
import iaik.security.*;
import iaik.security.rsa.*;
import iaik.security.cipher.SecretKey;
import iaik.pkcs.*;
import iaik.x509.*;
import iaik.pkcs.pkcs1.*;
import iaik.pkcs.pkcs8.*;
import iaik.utils.KeyAndCertificate;
import iaik.security.provider.IAIK;

public class CodeLF extends Frame
                    implements ActionListener, ItemListener {

  protected boolean inAnApplet = true;
  protected TextField codeField = null;
  protected TextArea contentArea = null;
  protected TextField contentLength = null;
  protected TextField contentSize = null;
  protected TextField userField = null;
  protected Choice fileListChoice;
  protected String fileBase;
  protected int fileNum;
  protected DatagramSocket socket1, socket2, socket3;
  protected InetAddress address;
  protected int port1 = 20012;
  protected int port2 = 20013;
  DatagramPacket packet1, packet2, packet3;
  protected byte[] sendBuf1 = new byte[1024];
  protected byte[] receiveBuf1 = new byte[1024];
  protected byte[] sendBuf3 = new byte[1024];
  protected byte[] receiveBuf3 = new byte[1024];
  protected byte[] buffer = new byte[1024];
  protected byte[] cipherInfo = new byte[1024];
  protected boolean DEBUG = true;
  protected Button bFetch, bExit, bDoc;
  protected Key key = null;
  protected byte[] skey = null;
  protected RSAPrivateKey privateKEY = null;
  protected DataInputStream dis = null;
  protected FileInputStream fis = null;
  protected FileInputStream fileIn = null;
  protected CipherInputStream cis = null;
  protected Cipher cipher = null;
  protected BufferedInputStream buffIn = null;
  protected BufferedInputStream buffCtr = null;
  protected RSAPrivateKey clientPrivateKey = null;
  protected PublicKey serverPublicKey = null;
  protected KeyAndCertificate KeyCert;
  protected EncryptedPrivateKeyInfo epki;
  protected String nameAlias = " ";
  protected Key desKey = null;
  protected Key DESkey = null;
```

```
protected Key tmpDESkey = null;
protected boolean desKeyNotExist = true;
protected byte[] plainData = null;
protected byte[] dataDigest = null;
protected byte[] dataSignature = null;
protected byte[] serverPublicKeyBytes = null;
protected String Check;
protected Frame wf;
protected IvParameterSpec spec = null;
static byte[] iv = { (byte)0xfe, (byte)0xdc, (byte)0xba,(byte)0x98,
                     (byte)0x76, (byte)0x54, (byte)0x32, (byte)0x10 };

public CodeLF(String alias) {

  String host = "kojak.aston.ac.uk";
  nameAlias = alias;

  try {
    address = InetAddress.getByName (host);
  } catch (UnknownHostException e) {
    System.out.println ("Couldn't get Internet address: Unknown host!");
    return;
  }

  makeGUI();
}


public void makeGUI() {

  // --- graphical user interface

  setLayout(new BorderLayout());

  String fileInfo = getFileInfo();

  //Add small things at the bottom of the window.
  int startPos = 0;
  int endPos;

  Panel p = new Panel();
  p.add("North", new Label("Filename:"));
  codeField = new TextField(15);
  codeField.addActionListener(this);
  p.add(codeField);

  p.add("North", new Label("Select File:"));
  fileListChoice = new Choice();
  fileListChoice.addItemListener(this);
  for (int i = 0; i < fileNum; i++) {
    endPos = fileInfo.indexOf (";", startPos);
    fileListChoice.addItem(fileInfo.substring(startPos, endPos));
    startPos = endPos + 1;
  }
  fileListChoice.select(0);
  p.add (fileListChoice);

  p.add(new Label("    "));
  p.add(bFetch = new Button(" Fetch "));
  bFetch.addActionListener(this);
  p.add(bExit = new Button(" EXIT "));
  bExit.addActionListener(this);
  add("North", p);

  Panel p0 = new Panel();
  p0.add(new Label("Username:"));
  userField = new TextField(15);
  userField.addActionListener(this);
  p0.add(userField);
  p0.add(new Label("Length:"));
  contentLength = new TextField(10);
  contentLength.addActionListener(this);
  contentLength.setEditable(false);
  p0.add(contentLength);
  add("South", p0);

  //Put a label and a text area in the right column.
  Panel p1 = new Panel();
  Panel centerPanel = new Panel();
  p1.setLayout(new BorderLayout());
  contentArea = new TextArea(35, 80);
  Font fcourier = new Font("Courier", Font.PLAIN, 13);
  contentArea.setFont(fcourier);
  contentArea.setEditable(false);
  p1.add(contentArea);
```

```
  p1.add("North", new Label(">>>>> INSPECTION DOCUMENT <<<<<", Label.CENTER));
  centerPanel.add(p1);
  add("Center", centerPanel);

  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
      if (inAnApplet) {
        dispose();
      } else {
        System.exit(0);
      }
    }
  });
  }


//---------------------------------------
// actionPerformed() method: Take action
//---------------------------------------

public void actionPerformed(ActionEvent event) {
  String command = event.getActionCommand();

  if (event.getSource() instanceof TextField || command == " Fetch ") {
    doFetch();
  }
  if (command == " EXIT ") {
    dispose();
  }
}



public void itemStateChanged(ItemEvent event) {
  codeField.setText(fileListChoice.getSelectedItem());
}


//-------------------------------------------------------
// getFileInfo() method: Check for the selected document
//-------------------------------------------------------

public String getFileInfo() {

  // open socket
  try {
    socket1 = new DatagramSocket();
  } catch (SocketException e) {
    System.out.println ("Couldn't create new DatagramSocket");
    return "Error";
  }

  // send request to validate to server
  String initiateRequest = "";
  sendBuf1 = initiateRequest.getBytes();
  packet1 = new DatagramPacket (sendBuf1, sendBuf1.length , address, port1);
  try {
    if (DEBUG) {
      System.out.println (" in getFileInfo address : " + address);
      System.out.println ("port : " + port1);
    }
    socket1.send(packet1);
    if (DEBUG) {
      System.out.println("sent packet.");
    }
  } catch (IOException e) {
    System.out.println("socket.send failed:");
    e.printStackTrace();
    return "Socket.send failed";
  }

  // check file info from server
  String received = "";
  packet1 = new DatagramPacket(receiveBuf1, receiveBuf1.length);

  try {
    if (DEBUG) {
      System.out.println("About to call socket.receive().");
    }
    socket1.receive(packet1);
    if (DEBUG) {
      System.out.println("returned from socket.receive().");
    }
  } catch (IOException e) {
    System.out.println("socket.receive failed:");
```

```
    e.printStackTrace();
    return "Socket.receive failed";
  }

  received = new String(packet1.getData());

  if (DEBUG) {
    System.out.println("File info : " + received);
  }
  int curPos = received.indexOf (";");
  fileNum = Integer.parseInt (received.substring (0, curPos));
  curPos = curPos + 1;
  String fileNames = received.substring (curPos);

  if (DEBUG) {
    System.out.println("fileNum : " + fileNum);
    System.out.println("fileNames : " + fileNames);
  }
  return fileNames;
}


//-----------------------------------------------------------
// doFetch() method : Start fetching the selected document
//-----------------------------------------------------------

public void doFetch() {

  try {
    byte b[] = new byte[2048];
    String content = "";
    int nbytes;

    // get DES key
    if (desKeyNotExist) {
      tmpDESkey = getDESkey();
      desKeyNotExist = false;
    }

    DESkey = tmpDESkey;

    if (Check == "xValid") {
      doNotValidFrame1();
    } else if (Check == "xVerify") {
      doNotVerifyFrame1();
    } else {
      try {
        cipher = Cipher.getInstance("DES/CBC/PKCS5Padding");
        spec = new IvParameterSpec(iv);
        cipher.init(Cipher.DECRYPT_MODE, DESkey, spec);

        fis = new FileInputStream("project/classes/files/"
                                  + codeField.getText());
        cis = new CipherInputStream(fis, cipher);

      } catch (Exception ex) {
        System.err.println("Error: fail to get/decrypt file.");
      }

      String loadedFile = userField.getText() + ";" + codeField.getText();

      if (DEBUG) {
        System.out.println("username  : " + userField.getText());
      }

      updateLoadedCodeFile(loadedFile);

      // Read and Display content
      contentArea.setText ("");
      String lineContent;
      int line = 0;
      int strLength, ptrPos, startPos, endPos;
      boolean newLine = true;

      // Read file until eof
      while ((nbytes = cis.read(b)) != -1) {
        content = new String(b, 0, nbytes);
        strLength = content.length() - 1;
        ptrPos = 0;
        startPos = 0;
        endPos = 0;

        // Process buffer until empty
        while (ptrPos < strLength) {
```

```
            // Check eoln
            endPos = content.indexOf ("\n", startPos);
            if ((endPos != -1) && (endPos != strLength)) {
              lineContent = content.substring (startPos, (endPos + 1));
              ptrPos = endPos + 1;
            }
            else {
              lineContent = content.substring (startPos);
              ptrPos = strLength;
            }

            // Print line number and one line of code
            if (newLine) {
              line = line + 1;
              contentArea.append (Integer.toString(line) + " -- ");
            }
            else
              newLine = true;

            contentArea.append (lineContent);
            startPos = ptrPos;
          }
          if (endPos == -1)
            newLine = false;
        }

        int length = line;
        contentLength.setText (Integer.toString (length) + " lines");
      }
    } catch (IOException e) {
      return;
    }
  }


  //-------------------------------------------------------
  //  updateLoadedCodeFile() method :
  //  update LoadedCodeFile to store the selected document
  //-------------------------------------------------------

  public void updateLoadedCodeFile (String loadedFile) {
    byte[] sendBuf2 = new byte [1024];
    byte[] receiveBuf2 = new byte [1024];

    //    open socket 2
    try {
      socket2 = new DatagramSocket();
    } catch (SocketException e) {
      System.out.println ("Couldn't create new DatagramSocket");
      return;
    }

    //    send file loaded update to server
    String initiateRequest = loadedFile;
      sendBuf2 = initiateRequest.getBytes ();
    packet2 = new DatagramPacket (sendBuf2, sendBuf2.length, address, port2);

    try {
      if (DEBUG) {
          System.out.println (" in updateLoadedCodeFile address : " + address);
          System.out.println ("port2 : " + port2);
      }
      socket2.send(packet2);

      if (DEBUG) {
          System.out.println("sent packet.");
      }
    } catch (IOException e) {
      System.out.println("socket.send failed:");
      e.printStackTrace();
      return;
    }

    //    check status of update from server
    packet2 = new DatagramPacket (receiveBuf2, receiveBuf2.length );

    try {
      if (DEBUG) {
          System.out.println("About to call socket.receive().");
      }
      socket2.receive(packet2);
      if (DEBUG) {
        System.out.println("Returned from socket.receive().");
      }
    } catch (IOException e) {
```

```
      System.out.println("socket.receive failed:");
      e.printStackTrace();
      return;
    }

    String received = new String(packet2.getData());

    if (DEBUG) {
      System.out.println("Update confirm info : " + received);
    }

    if (received.equals("NotOK")) {
      System.out.println("Error updating loaded code file.");

    }
  }


  //-------------------------------------------------------
  // getDESkey() method : retrieve DES key from the server
  //-------------------------------------------------------

  public Key getDESkey() {

    try {

      // send request

      packet3 = sendPacket3(nameAlias);

      try {
        if (DEBUG) {
          System.out.println (" in getFileInfo address : " + address);
          System.out.println ("port : " + port1);
        }

        socket1.send(packet3);

        if (DEBUG) {
          System.out.println("sent packet.");
        }
      } catch (IOException e) {
        System.out.println("socket.send failed:");
        e.printStackTrace();
      }

      // receive respond

      receivePacket3();

    } catch (Exception ex) {
      System.out.println(ex);
    }
    return DESkey;
  }


  //-------------------------------------------------
  // sendPacket3() method : Encrypt packet to be send
  //-------------------------------------------------

  protected DatagramPacket sendPacket3(String nameAlias) throws Exception {

    ByteArrayOutputStream byte0 = new ByteArrayOutputStream();
    DataOutputStream data0 = new DataOutputStream(byte0);

    // plain data
    plainData = new String(nameAlias).getBytes();

    // create a session key
    KeyGenerator kg = KeyGenerator.getInstance("DES", "IAIK");
    kg.init(new SecureRandom());
    Key sessionKey = kg.generateKey();

    // encrypt data with session key
    cipher = Cipher.getInstance("DES/CBC/PKCS5Padding");
    spec = new IvParameterSpec(iv);
    cipher.init(Cipher.ENCRYPT_MODE, sessionKey, spec);
    byte[] cipherData = cipher.doFinal(plainData);

    // encrypted IV using DES session key
    cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, sessionKey);
    byte[] cipherIV = cipher.doFinal(iv);
```

```
    // --- get server public key
    KeyCert = new KeyAndCertificate("project/classes/certs/serverRSAcert.pem");
    X509Certificate c = KeyCert.getCertificateChain()[0];
    serverPublicKey = c.getPublicKey();

    // encrypt session key with server public key
    cipher = Cipher.getInstance("RSA", "IAIK");
    cipher.init(Cipher.ENCRYPT_MODE, serverPublicKey);
    byte[] cipherSessionKey = cipher.doFinal(sessionKey.getEncoded());

    // --- get our/client private key
    KeyCert = new KeyAndCertificate("project/classes/certs/"
                                    + nameAlias + "RSAcert2.pem");
    epki = (EncryptedPrivateKeyInfo)KeyCert.getPrivateKey();
    epki.decrypt("Aston University");
    clientPrivateKey = (RSAPrivateKey)epki.getPrivateKeyInfo();

    // sign message
    Signature s = Signature.getInstance("SHA/RSA", "IAIK");
    s.initSign(clientPrivateKey);
    s.update(plainData);
    byte[] dataSignature = s.sign();

    // message digest
    MessageDigest mDigest = MessageDigest.getInstance("SHA");
    mDigest.update(plainData);
    dataDigest = mDigest.digest();

    // --- Embed all information ---

    // send the session IV
    dataO.writeInt(cipherIV.length);
    dataO.write(cipherIV);

    // send the encrypted session key
    dataO.writeInt(cipherSessionKey.length);
    dataO.write(cipherSessionKey);

    // send the data signature
    dataO.writeInt(dataSignature.length);
    dataO.write(dataSignature);

    // send the message digest
    dataO.writeInt(dataDigest.length);
    dataO.write(dataDigest);

    // send the encrypted data;
    dataO.writeInt(cipherData.length);
    dataO.write(cipherData);

    byte[] cipherInfo = byteO.toByteArray();
    return new DatagramPacket(cipherInfo, cipherInfo.length, address, port1);
}


//---------------------------------------------------------
// receivePacket3() method : fetch packet from the server
//---------------------------------------------------------

public void receivePacket3() {

  try {
    packet3 =  new DatagramPacket(receiveBuf3, receiveBuf3.length);
    socket1.receive(packet3);
    ByteArrayInputStream byteI = new ByteArrayInputStream(
                                          packet3.getData(), 0, 512);
    DataInputStream dataI = new DataInputStream(byteI);

    // read encrypted DES key
    byte[] cipherDESKey = new byte[dataI.readInt()];
    dataI.read(cipherDESKey);

    // read signature
    byte[] dataSignature = new byte[dataI.readInt()];
    dataI.read(dataSignature);

    // read the message digest data
    byte[] orginalDigest = new byte[dataI.readInt()];
    dataI.read(orginalDigest);

    // *** Proses Data ***

    // --- get our/client private key ---
    KeyCert = new KeyAndCertificate("project/classes/certs/"
                                    + nameAlias + "RSAcert1.pem");
```

```
      epki = (EncryptedPrivateKeyInfo)KeyCert.getPrivateKey();
      epki.decrypt("Aston University");
      clientPrivateKey = (RSAPrivateKey)epki.getPrivateKeyInfo();

      // decrypt the session key
      cipher = Cipher.getInstance("RSA", "IAIK");
      cipher.init(Cipher.DECRYPT_MODE, clientPrivateKey);
      byte[] plainDESkey = cipher.doFinal(cipherDESKey);
      DESkey = new SecretKey(plainDESkey, "RAW");

      // verify the signature
      Signature s = Signature.getInstance("SHA/RSA", "IAIK");
      s.initVerify(serverPublicKey);
      s.update(plainDESkey);

      if (s.verify(dataSignature)) {
        System.out.println("Verified........");
      }
      else {
        System.out.println("Not Verified........");
        Check = "xVerify";
      }

      // check for data integrity
        MessageDigest mDigest = MessageDigest.getInstance("SHA");
      mDigest.update(plainDESkey);
      byte[] dataDigest = mDigest.digest();

      if (MessageDigest.isEqual(dataDigest, orginalDigest)) {
          System.out.println("Data received is valid...");
      }
      else {
          System.out.println("Data was corrupted...");
          Check = "xValid";
      }
    } catch (Exception ex) {
      System.err.println("Error : fail to read packet.");
    }
  }

  //----------------------------
  // doNotVerifyFrame1() method:
  //----------------------------

  public void doNotVerifyFrame1() {

    wf = new NotVerifyFrame1();
    wf.setTitle("Secure-SIG");
    wf.setSize(300,110);
    wf.setLocation(350,100);
    wf.setBackground (Color.red);
    wf.setVisible(true);
  }


  //----------------------------
  // doNotValidFrame1() method:
  //----------------------------

  public void doNotValidFrame1() {

    wf = new NotValidFrame1();
    wf.setTitle("Secure-SIG");
    wf.setSize(300,110);
    wf.setLocation(350,100);
    wf.setBackground (Color.red);
    wf.setVisible(true);
  }

}  // end CodeLF class
```

CodeLoaderServer.java

```
//--------------------------------------------
// Main server program for code viewer process
//--------------------------------------------

public class CodeLoaderServer {
  public static void main(String[] args) throws java.io.IOException {
    new CodeLoaderServerThread().start();
  }
```

```
}
```

CodeLoaderSeverThread.java

```
//-------------------------------------------------------------
// This program is a server program to perform the code viewer
// process.  This program is called by main server program
// (CodeLoaderServer.java).
//-------------------------------------------------------------

import java.io.*;
import java.net.*;
import java.util.*;
import java.math.*;
import java.security.*;
import java.security.spec.*;

import javax.crypto.*;
import javax.crypto.spec.*;
import iaik.security.*;
import iaik.security.rsa.*;
import iaik.security.cipher.*;
import iaik.security.cipher.SecretKey;
import iaik.asn1.structures.*;
import iaik.x509.*;
import iaik.asn1.*;
import iaik.security.provider.IAIK;
import iaik.pkcs.*;
import iaik.pkcs.pkcs8.*;
import iaik.utils.KeyAndCertificate;

class CodeLoaderServerThread extends Thread {

  protected DatagramSocket socket = null;
  protected DatagramSocket socket1 = null;
  protected BufferedReader clfs = null;
  protected BufferedReader flLst = null;
  protected boolean DEBUG = true;
  protected int port = 20012;
  protected int maxLine = 5;
  protected FileInputStream fis = null;
  protected CipherInputStream cis = null;
  protected RSAPrivateKey privateKEY = null;
  protected Cipher cipher = null;
  protected byte[] skey = null;
  protected String fileLoc[];
  protected char EOLN = '\n';
  protected char EOF = (char) -1;
  protected RSAPrivateKey serverPrivateKey = null;
  protected KeyAndCertificate KeyCert;
  protected EncryptedPrivateKeyInfo epki;
  protected Key key = null;
  protected PublicKey clientPublicKey = null;
  protected byte[] receivebuf = new byte[1024];
  protected byte[] sendbuf = new byte[1024];
  protected byte[] receivebuf1 = new byte[1024];
  protected byte[] sendbuf1 = new byte[1024];
  protected DatagramPacket packet;
  protected DatagramPacket packet1;
  protected InetAddress address;
  protected String alias;
  protected byte[] PlainDESKey;
  protected Cipher RSACipher;
  protected byte[] dataDigest = null;
  protected byte[] plainData = null;
  protected byte[] dataSignature = null;
  protected boolean OK = true;
  protected String check = null;
  protected byte[] iv = null;
  protected IvParameterSpec spec = null;

  // open a socket
  public CodeLoaderServerThread() throws IOException {
    this("CodeLoaderServerThread");
  }

  public CodeLoaderServerThread(String name) throws IOException {
    super(name);
    socket = new DatagramSocket(port);
    System.out.println("CodeLoaderServerThread listening on port: "
                       + socket.getLocalPort());
```

```
}

// run() :  wait for a packet, open a file and send info
public void run() {

  if (socket == null)
    return;

  while (true) {
    try {

      String fileInfo = null;

      // receive request
      this.openInputFile();
      packet = new DatagramPacket(receivebuf, receivebuf.length);
      if (DEBUG) {
        System.out.println("Server about to call socket.receive().");
      }
      socket.receive(packet);
      if (DEBUG) {
        System.out.println("Server received packet.");
      }
      address = packet.getAddress();
      System.out.println("address : " + address);
      port = packet.getPort();
      System.out.println("port : " + port);
      String received= new String (packet.getData());

      // --- get file info
      if (flLst == null)
        fileInfo = "XNO";
      else {
        fileInfo = getFileInfo();
      }

      // send response
      sendbuf = fileInfo.getBytes();
      packet = new DatagramPacket(sendbuf, sendbuf.length, address, port);
      if (DEBUG) {
        System.out.println("Before send packet.");
        System.out.println("fileInfo.length : "+ fileInfo.length());
        System.out.println("sbuf.length : "+ sendbuf.length);
        System.out.println("fileInfo : "+ fileInfo);
        System.out.println("Server send packet.");
      }

      socket.send(packet);

      try {
        flLst.close();
      } catch (IOException e) {
        System.out.println("Error closing file : " + e);
      }

      // receive request
      try {
        receivePacket1();

        // send response
        // --- get DES and send to client ---

        packet1 = sendPacket1();

        socket.send(packet1);

      } catch (Exception ex) {
        System.err.println("Error : fail to build packet.");
      }

    } catch (IOException e) {
      System.err.println("IOException:  " + e);
      e.printStackTrace();
    }
  }
}


//-------------------------------------------------
// openInputFile() method :  open configuration file
//-------------------------------------------------
private void openInputFile() {
```

```
  try {
    flLst = new BufferedReader(new FileReader("fileLocList.dat"));
  } catch (Exception e) {
    System.err.println("Could not open data file location..");
  }
}


//-------------------------------------
// getFileInfo() method : get all data
//-------------------------------------

private String getFileInfo() {

  String returnValue = null;
  String readValue = null;
  int fileNum, ctr = 0;
  String temp;

  try {

    readValue = flLst.readLine();
    temp = readValue.substring(0, 1);
    fileNum = Integer.parseInt (temp);
    returnValue = readValue + ";";

    while (ctr < fileNum) {
      readValue = flLst.readLine();
      returnValue = returnValue.concat (readValue);
      returnValue = returnValue.concat (";");
      ctr++;
      if (DEBUG) {
        System.out.println("in read readValue : " + readValue);
        System.out.println("file-returnValue : " + returnValue);
      }

    }
  } catch (IOException e) {
    returnValue = "IOException occurred in server.";
  }

  if (DEBUG) {
    System.out.println("returnValue : " + returnValue);
  }
  return returnValue;
}


//-----------------------------------------------------
// receivePacket1() method : receive packet from client
//-----------------------------------------------------

public void receivePacket1() {

  try {
    DatagramPacket packet = new DatagramPacket(receivebuf1,
                                               receivebuf1.length);
    socket.receive(packet);
    ByteArrayInputStream byteI = new ByteArrayInputStream(
                                          packet.getData(), 0, 256);
    DataInputStream dataI = new DataInputStream(byteI);

    // read the encrypted session IV
    byte[] cipherIV = new byte[dataI.readInt()];
    dataI.read(cipherIV);

    // read encrypted session key
    byte[] cipherSessionKey = new byte[dataI.readInt()];
    dataI.read(cipherSessionKey);

    // read signature
    dataSignature = new byte[dataI.readInt()];
    dataI.read(dataSignature);

    // read the message digest data
    byte[] orginalDigest = new byte[dataI.readInt()];
    dataI.read(orginalDigest);

    // read encrypted data
    byte[] cipherData = new byte[dataI.readInt()];
    dataI.read(cipherData);

    // *** Process all Data ***

    // --- get our/server private key
```

```
    KeyCert = new KeyAndCertificate("certs/serverRSAcert.pem");
    epki = (EncryptedPrivateKeyInfo)KeyCert.getPrivateKey();
    epki.decrypt("Aston University");
    serverPrivateKey = (RSAPrivateKey)epki.getPrivateKeyInfo();

    // decrypt the session key
    cipher = Cipher.getInstance("RSA", "IAIK");
    cipher.init(Cipher.DECRYPT_MODE, serverPrivateKey);
    byte[] plainDESkey = cipher.doFinal(cipherSessionKey);
    Key desKey = new SecretKey(plainDESkey, "RAW");
    Key desKey1 = new SecretKey(plainDESkey, 0, 8, "DES");

    // decrypt session IV
    cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
    cipher.init(Cipher.DECRYPT_MODE, desKey1);
    iv = cipher.doFinal(cipherIV);

    // decrypt the data
    cipher = Cipher.getInstance("DES/CBC/PKCS5Padding");
    spec = new IvParameterSpec(iv);
    cipher.init(Cipher.DECRYPT_MODE, desKey, spec);
    plainData = cipher.doFinal(cipherData);

    // --- get client public key ---
    alias = new String(plainData);
    KeyCert = new KeyAndCertificate("certs/" + alias + "RSAcert2.pem");
    X509Certificate c = KeyCert.getCertificateChain()[0];
    PublicKey clientPublicKey = c.getPublicKey();

    // verify the signature
    Signature s = Signature.getInstance("SHA/RSA", "IAIK");
    s.initVerify(clientPublicKey);
    s.update(plainData);

    if (s.verify(dataSignature))
      System.out.println("Verified........");
    else {
      System.out.println("Not Verified........");
      check = "xVerify";
      OK = false;
    }

    // check for data integrity
    System.out.println("try to check for data integrity...");
    MessageDigest mDigest = MessageDigest.getInstance("SHA");
    mDigest.update(plainData);
    byte[] dataDigest = mDigest.digest();

    if (MessageDigest.isEqual(dataDigest, orginalDigest)) {
      System.out.println("Data received is valid...");
    }
    else {
      System.out.println("Data was corrupted...");
      check = "xValid";
      OK = false;
    }
  } catch (Exception ex) {
    System.err.println("Error : fail to read packet.");
  }
}


//-------------------------------------------------
// sendPacket1() method : Encrypt packet to be send
//-------------------------------------------------

protected DatagramPacket sendPacket1() throws Exception {

  ByteArrayOutputStream byteO = new ByteArrayOutputStream();
  DataOutputStream dataO = new DataOutputStream(byteO);

  // get DES key
  retriveDESkey();

  // --- get client public key
  KeyCert = new KeyAndCertificate("certs/" + alias + "RSAcert1.pem");
  X509Certificate c = KeyCert.getCertificateChain()[0];
  PublicKey clientPublicKey = c.getPublicKey();

  // encrypt session key with client public key
  cipher = Cipher.getInstance("RSA", "IAIK");
  cipher.init(Cipher.ENCRYPT_MODE, clientPublicKey);
  byte[] cipherDESkey = cipher.doFinal(key.getEncoded());

  // --- get our/server private key
```

```
    KeyCert = new KeyAndCertificate("certs/serverRSAcert.pem");
    epki = (EncryptedPrivateKeyInfo)KeyCert.getPrivateKey();
    epki.decrypt("Aston University");
    serverPrivateKey = (RSAPrivateKey)epki.getPrivateKeyInfo();

    // sign message
    Signature s = Signature.getInstance("SHA/RSA", "IAIK");
    s.initSign(serverPrivateKey);
    s.update(PlainDESKey);
    byte[] desSignature = s.sign();

    // message digest
    MessageDigest mDigest = MessageDigest.getInstance("SHA");
    mDigest.update(PlainDESKey);
    byte[] desDigest = mDigest.digest();


    // *** Send all the data ***

    // send the encrypted des key
    dataO.writeInt(cipherDESkey.length);
    dataO.write(cipherDESkey);

    // send the data signature
    dataO.writeInt(desSignature.length);
    dataO.write(desSignature);

    // send message digest
    dataO.writeInt(desDigest.length);
    dataO.write(desDigest);

    byte[] cipherBuff = byteO.toByteArray();
    return new DatagramPacket(cipherBuff, cipherBuff.length, address, port);

  }


  //---------------------------------------
  // retriveDESkey() method : retreive DES key
  //---------------------------------------

  public void retriveDESkey() {
    try {
      ObjectInputStream ois = new ObjectInputStream(
                              new FileInputStream("files/desKEY.enc"));
      skey = (byte[])ois.readObject();

      // --- get our/server private key
      KeyCert = new KeyAndCertificate("certs/serverRSAcert.pem");
      epki = (EncryptedPrivateKeyInfo)KeyCert.getPrivateKey();
      epki.decrypt("Aston University");
      serverPrivateKey = (RSAPrivateKey)epki.getPrivateKeyInfo();

      // decrypt DES key
      RSACipher = Cipher.getInstance("RSA");
      RSACipher.init(Cipher.DECRYPT_MODE, serverPrivateKey);
      PlainDESKey = RSACipher.doFinal(skey);
      System.out.println("Decrypting DES key ... DONE");

      // generate key
      key = new SecretKey(PlainDESKey, "RAW");

      ois.close();
    } catch (Exception e) {
      System.err.println("Error: fail to get/decrypt DES key.");
    }
  }
}   // end CodeLoaderServerThread class
```

LoadedCodeServer.java

```
// Main Server Program for storing information used by remote viewer // process.

class LoadedCodeServer {

   public static void main (String args[]) {
      new LoadedCodeServerThread().start();
   }
}
```

LoadedCodeSeverThread.java

```
// This program server store information into a file that going to be
// used in remote viewer process.  This program server is called by
// LoadedCodeServer.java.

import java.io.*;
import java.net.*;
import java.util.*;

class LoadedCodeServerThread extends Thread {

  protected DatagramSocket socket = null;
  protected RandomAccessFile lcrf = null;
  protected boolean DEBUG = true;
  protected int port = 20013;

  LoadedCodeServerThread() {

    super("LoadedCodeServer");
    try {
      socket = new DatagramSocket(port);
      System.out.println("LoadedCodeServer listening on port: "
                         + socket.getLocalPort());
    } catch (java.net.SocketException e) {
      System.err.println("Could not create datagram socket.");
    }
  }


  public void run() {

    if (socket == null)
      return;

    while (true) {
      try {
        byte[] buf = new byte[256];
        DatagramPacket packet;
        InetAddress address;
        int port;
        String fileInfo = null;

        // receive request
        packet = new DatagramPacket(buf, 256);
        if (DEBUG) {
          System.out.println("Server about to call socket.receive().");
        }
        socket.receive(packet);
        if (DEBUG) {
          System.out.println("Server received packet.");
        }
        address = packet.getAddress();
        if (DEBUG) {
          System.out.println("address : " + address);
        }
        port = packet.getPort();
        String received = null;
        received = new String (packet.getData());
        if (DEBUG) {
          System.out.println("port : " + port);
          System.out.println("received : " + received);
        }

        // store loaded file info
        String confirmInfo = writeFileInfo(received);

        // send response
        buf = confirmInfo.getBytes();
        packet = new DatagramPacket(buf, buf.length, address, port);

        if (DEBUG) {
          System.out.println("buf.length : "+ buf.length);
          System.out.println("confirmInfo : "+ confirmInfo);
          System.out.println("Server send packet.");
        }

        socket.send(packet);
        try {
          lcrf.close();
        } catch (IOException e) {
          System.out.println("Error closing file : " + e);
        }
```

```
        } catch (IOException e) {
          System.err.println("IOException:  " + e);
          e.printStackTrace();
        }
      }
    }


    //-----------------------------------------------------
    // writeFileInfo() method: write information to a file
    //-----------------------------------------------------

     private String writeFileInfo (String receivedInfo)
                                          throws java.io.IOException {
        String returnValue = null;
        boolean notFound = true;
        boolean removeUser = false;
        String readValue = null;
        int pos = 0, pos2 = 0, locPtr = 0, endIndx = 0;
        String fileLoc[] = new String[20];
        String userName = null;
        String fileUrl, inFileLoc = null;
        String inUserName = null;
        long prevFilePtr = 0, userLocPtr = 0;
        char EOF = (char) -1;

        for (int j = 0; j < 20; j++) {
           fileLoc[j] = null;
        }

          pos = receivedInfo.indexOf (";");
        String temp2 = receivedInfo.substring (0, pos);
        if (temp2.equals ("rm")) {
           removeUser = true;
           pos2 = receivedInfo.indexOf (";", pos + 1);
           inUserName = receivedInfo.substring (pos + 1, pos2);
        }
        else {
           inUserName = temp2;
           inFileLoc = receivedInfo.substring (pos + 1);
        }
        String fileName = "loadedCodeInfo.dat";
        if (DEBUG) {
           System.out.println("filename : " + fileName);
           System.out.println("inUserName : " + inUserName);
           System.out.println("inFileLoc : " + inFileLoc);
        }
        this.openOutputFile (fileName);
        if (lcrf == null) {
           returnValue = "NotOK";
           return returnValue;
        }

        //    check if info is already in file and load all info
        int i = 0;
        char c;
        prevFilePtr = lcrf.getFilePointer();
        while ((lcrf.getFilePointer() < lcrf.length()) && ((c = lcrf.readChar()) !=
EOF)) {
           StringBuffer temp = new StringBuffer();
           while (c != '\n') {
              temp.append (c);
              c = lcrf.readChar();
           }
           readValue = temp.toString();

           if (DEBUG) {
              System.out.println ("readValue : " + readValue);
           }

           pos = readValue.indexOf (";");
           userName = readValue.substring (0, pos);
           fileLoc[i] = readValue;

           if (DEBUG) {
              System.out.println ("username : *" + userName + "*");
              System.out.println ("inusername : *" + inUserName + "*");
           }

           if ((notFound) && (inUserName.equalsIgnoreCase (userName))) {
              notFound = false;
              locPtr = i;
              userLocPtr = prevFilePtr;
              fileLoc[i] = receivedInfo;
```

```
            if (DEBUG) {
               System.out.println ("in if-notFound : " + notFound);
            }
         }

         endIndx = i;
         i++;
         prevFilePtr = lcrf.getFilePointer();

         if (DEBUG) {
            System.out.println ("prevFilePtr : " + prevFilePtr);
            System.out.println ("userName : " + userName);
            System.out.println ("i : " + i);
            System.out.println ("notFound : " + notFound);
         }
      }
      if (notFound) {
         userLocPtr = prevFilePtr;
      }
      if (DEBUG) {
         System.out.println ("userLocPtr : " + userLocPtr);
         System.out.println ("locPtr : " + locPtr);
         System.out.println ("i : " + i);
      }

      // write file info into random access file

      if (notFound) {
         if (removeUser) {
            System.out.println ("Error : File empty, cannot find user..");
            returnValue = "NotOK";
         }
         else {
            lcrf.seek(userLocPtr);

            if (DEBUG) {
               System.out.println ("length : " + lcrf.length());
            }
            lcrf.writeChars (receivedInfo);
            lcrf.writeChar ('\n');
            lcrf.writeChar (EOF);
            if (DEBUG) {
               System.out.println ("length : " + lcrf.length());
            }
            returnValue = "OK";
         }
      }
      else {
         lcrf.seek (userLocPtr);
         if (removeUser)
            i = locPtr + 1;
         else
            i = locPtr;
         for (; i <= endIndx; i++) {
            lcrf.writeChars (fileLoc[i]);
            lcrf.writeChar ('\n');
            if (DEBUG) {
               System.out.println ("i : " + i);
            }
         }
         lcrf.writeChar (EOF);
         returnValue = "OK";
      }
      if (DEBUG) {
         System.out.println("returnValue : " + returnValue);
      }
      lcrf.close();
      return returnValue;
   }

   //-----------------------------------------------
   // openOutputFile() method: open loaded code file
   //-----------------------------------------------

   private void openOutputFile (String fileName) {
   try {
        lcrf = new RandomAccessFile (fileName, "rw");
      } catch (java.io.FileNotFoundException e) {
         System.err.println("Could not open data file location..");
      } catch (java.io.IOException e) {
         System.err.println("IO Exception when opening file info ..");
      }
   }

} // end class LoadedCodeServerThread
```