Security Architecture for the TEAMDEC System

Haiyuan Wang

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical and Computer Engineering

July 7, 1999

Blacksburg, Virginia

Keywords: Network Security, Cryptography, Authentication, Digital Signature, Key Agreement, Java, TEAMDEC

Security Architecture for the TEAMDEC System

By

Haiyuan Wang

Mark T. Jones, Chairman

Department of Electrical and Computer Engineering

(ABSTRACT)

The prevalence of the Internet, client/server applications, Java, e-commerce, and electronic communications offers tremendous opportunities for business, education and communication, while simultaneously presenting big challenges to network security. In general, the web was designed with little concern for security. Thus, the issue of security is important in the design of network-based applications. The software architecture proposed in this thesis allows for the secure and efficient running of a team-based decision support system, specifically TEAMDEC. Based on the system's requirements and architecture, three types of possible attacks to the system are identified and a security solution is proposed that allows for user authentication, secure communication, and script access control. The implementation of these features will reduce security risk and allow effective use of the valuable system information data.

Acknowledgements

I am deeply grateful to my advisor, Dr. Mark T Jones, for his continued guidance and support throughout this work. Dr. Jones's extensive knowledge and creative thinking were truly invaluable.

I would like to express my gratitude to Dr. Eloise Coupey and Dr. Scott F. Midkiff for their comments and valuable contributions as members of my advisory committee.

My special thanks to Ms. Qian Chen and Ms. Jing Ma for their contributions to this research project.

Finally I want to express my gratitude to my husband, Yu Wang, who gave me his love and emotional support during the past years.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS AND TABLES

Chapter 1 Introduction

1.1 Introduction

The growth of the Internet and the World Wide Web (WWW) during the past few years has been dramatic. Most business and government institutions have their own web pages, and web browsing is quickly becoming the primary source of information for people of all ages. Current estimates indicate that there will be 500 million users connected to the Internet by 2001[1]. The prevalence of the Internet, client/server applications, Java, e-commerce, and electronic communications offers tremendous opportunities for business and education, while it simultaneously presents big challenges to network security because cyberspace is an open, distributed, and insecure environment. An organization's key asset, business information, becomes exposed and more vulnerable. If no security measures are taken, unauthorized access to this information is easy to obtain and difficult to detect. In this expanding electronic world, security issues take on new importance and complexity. Enforcing security in such a complex environment requires a security model and administration. Unfortunately, the web was designed with little concern for security. Thus, the issue of security is essential to network-based application.

The software architecture proposed in this thesis allows for the secure and efficient running of a team-based decision support system, specifically **TEAMDEC**[2].

1.2 Overview of TEAMDEC

TEAMDEC is an intelligent decision support tool that allows people to coordinate information from many different information sources to facilitate decision-making. People can make decisions using information from the Internet, or from scripts previously stored on the database, or even through real-time communications with other members of the decision team. A key element of **TEAMDEC** is the script database, which can be used to guide team decision-making. The script database can be used to provide suggestions about possible actions (information sources, personnel to contact) based on similar instances of past actions that are stored in the action database[3].

TEAMDEC can be used to help individuals or teams make better decisions more efficiently, whether working alone, or with input from members of a team. The security challenges arising from these features are substantial.

The following are the main requirements influencing the security architecture for **TEAMDEC**:

(1) **TEAMDEC** is for military use. It demands a higher level of security than a simple cooperative system. Thus, to guarantee that only an authorized user can access **TEAMDEC**, user authorization should be based on strong cryptography rather than on a simple password-based authentication scheme.

(2) To provide guarantees of the integrity and the secrecy of scripts in **TEAMDEC**, there is a need for fine-grained access control to individual scripts and their attributes. Thus a generic access model is required that enables access rights to be based on an individual's role in the system.

(3) **The TEAMDEC** runtime environment involves dynamic user groups. In such an open environment, communication among group members must be secure and, at the same time, efficient. Authenticated key agreement protocol based on the Diffie-Hellman[4] method can be employed to achieve that purpose.

1.3 Thesis Statement

1.3.1 Objectives of Research

Due to the sensitivity of the data related to decision-making, especially before a final decision is reached, security has been one of the main concerns of **TEAMDEC**. These issues demand a systematic response to making safe and controlled access to **TEAMDEC** along with its significant information - scripts. Thus, appropriate security mechanisms should be identified in order to maintain the confidentiality and integrity of system data. The research focuses on implementation of a security architecture that will provide the following security functions:

(1) user authentication by using public key cryptography that offers powerful tools for proving identity,

(2) role-based, fine-grained access control model that provides a finer level of security control for individual scripts, and

(3) authenticated key agreement protocol based on Diffie-Hellman[4] for safe communication among team members.

1.3.2 Contributions

First, the focus of this research is on how to develop a high quality Group Decision Support System(GDSSs). Qian Chen's research investigates the use of GDSSs as a practical aid for an organization and concerns with the factors influencing the quality of an interactive software system and concentrates on the design of interactive GDSSs. Based on the theories and concepts of human decision-making, the design principles of interactive software and GDSSs are applied to the development of **TEAMDEC**. Several concerns are taken into account, primarily including usability, efficiency, effectiveness, and security, and these concerns influence the quality of **TEAMDEC**[2].

Second, the desired collaborative features of **TEAMDEC** have been implemented. These features include interactive, simultaneous communications among team members of **TEAMDEC**, for example, group discussion and videoconferencing functions. Real-time search and integration capability, for example, retrieving search results from commercial search engines on the Internet and displaying them. The most important contribution is that the implementation of **TEAMDEC** script system. The system is able to record and track the user's actions and provide the action suggestion scripts and thus improve quality and efficiency of decision-making. All these features are necessary functions in group decision support systems

Third, the contribution of this thesis is that the security architecture is proposed and implemented. These security features enable the secure operation of **TEAMDEC**. The proposed security features include user authentication to prevent unauthorized access, encryption/decryption mechanism based on the Diffie-Hellamn[4] key agreement protocol for safe communication among decision team members and script access rules to maintain the confidentiality and integrity of system data.

1.4 Thesis Organization

The thesis is divided into the following chapters. The first chapter begins with an introduction and overview of **TEAMDEC** along with relevant security issues and identifies the security architecture. It then gives the thesis statement and organization. The second chapter first gives an introduction to the objectives of network security. It then reviews literature on areas such as a user authentication system, cryptography and access control. The advantages and disadvantages of different algorithms and protocols are also examined and compared. At the end of this chapter, new security features of Java are reviewed. Chapter Three provides an overview of **TEAMDEC** and its architecture, then gives a detailed security requirement analysis of **TEAMDEC**. The analysis involves the examination of security objectives related to confidentiality, integrity, and availability. It then defines three types of attacks and provides the proposed security architecture. The fourth chapter goes into more depth on the implementation of the security architecture of **TEAMDEC**. Topics include user authentication, data encryption, and script access rules. The evaluation of the security implementation is presented in chapter five. The analysis is based on two factors: computation speed and the ability to prevent different attacks. In addition, the advantages and disadvantages of the security algorithm are evaluated. The chapter concludes with a brief discussion of the possible improvement in the future. Chapter Six provides conclusions and closes the thesis.

Chapter 2 Review and Background

This chapter presents a review of some areas in network security. The chapter begins with an overview of network security. It then moves into user authentication and cryptography and presents some of the algorithms and protocols, such as public key cryptography and digital signatures. It continues with a review of access control and database security. Finally, new security features in Java are discussed.

2.1 Overviews of Network Security

2.1.1 Objectives of Network Security

With usage of the Internet and Intranets, today's networks are complex and diverse. This diversity, both technically and geographically, means that network security becomes a critical, ongoing issue for an information technology environment. System data and other sensitive information are exposed to threats such as unauthorized access, unauthorized modification, or unauthorized denial of services. Following are a few fundamental objectives of network security [1].

(1) Integrity

Integrity ensures that data is transmitted from the source to the destination without undetected alteration [6]. For example, if one wants to load a script from the database in **TEAMDEC**, one wants to be assured that the script is exactly what is wanted and has not been altered by a third party.

(2) Confidentiality

In communication, confidentiality requires that the intended recipients know what was being sent but unintended parties cannot determine what was sent[6]. For example, when a user shops online, he must ensure that his credit card information must be kept secret when transferred over the Internet. Other examples include critical business information or personal data covered by privacy laws. Mechanisms commonly used to provide confidentiality are authentication[5] and encryption[6]. For example, "*Hello, world!*" can be encrypted to "*xyOoLnWOH0eqRwUu3rQHJw*" by using a specific key[7].

(3) Availability

Availability ensures that the authorized users of the system will not be denied access when access is desired. An online stock trade system must be accessible to the investor when the stock market is open.

In addition to the fundamental objectives addressed above, several other objectives are frequently mentioned including accountability and disaster recovery. Accountability ensures that managers can determine who made what changes, when, and from where[1]. Summing up, these objectives provide the needed foundation for network security.

The proposed software architecture for **TEAMDEC** uses public key-based user authentication and data encryption to ensure data confidentiality, and uses role-based script access rules to ensure data integrity. All these schemes are used to achieve the objectives of network security.

2.1.2 Threats to the Network System

As the computer networks and distributed systems expand, they not only offer endless opportunities, but also create a number of threats. The threats are increasing as fast as the growth of the networks. Some threats are mainly due to people who deliberately try to steal critical information and secrets or simply change data. Others are virus threats and physical threats[1]. Each of these types of threats is discussed below.

2.1.2.1 People Threats

People threats usually include unauthorized use, deletion, and/or modification of system data. Some people are hackers who are motivated by the desire to break into secure systems. The most common form of access is through repeated login attempts[5][8]. The other biggest threat is when an individual of an organization supplies information (like account details) to a hacker in order to obtain confidential information. Using password protection often prevents the normal hacker. Using encryption schemes can provide more secure protection. These schemes will be discussed in detail later in this chapter. In TEAMDEC, to provide system data confidentiality and integrity, public key based user authentication and script access rules are employed to prevent people threats.

2.1.2.2 Virus Threats

Computer viruses are the most widely recognized example of a class of programs written to cause some form of intentional disruption or damage to computer systems or networks[1]. A virus is a piece of programming code inserted into other programming to cause some unexpected and, for the victim, usually undesirable event[9]. Viruses can be transmitted by downloading files from other web sites or by files on a disk. The source of the file being downloaded or received is often unaware of the virus. The virus lies dormant until circumstances cause its code to be executed by the computer. Some can be quite harmful, erasing data or causing the hard disk to require reformatting. The best protection to prevent a virus is to know the origin of each program or file you load into your computer. Typically anti-virus software can help to remove any viruses that are found.

2.1.2.3 Physical Threats

Physical threats usually originate from the physical environment. They can include electrical power failure, hardware failure or environment failures that cause damage to the whole system. Possible consequences include loss of system data, data integrity, and/or interruption of services[1].

To achieve secure and reliable operation of the network system, the threats to the system must first be identified and adequately addressed. Otherwise a threat, whether of natural causes or the result of human behavior, may cause substantial harm or loss to the whole system. The proposed security architecture for **TEAMDEC** is mainly used to prevent people threats. The following discussions give a review of different schemes to act against people threats.

2.2 User Authentication

The user authentication system is the first line to prevent unauthorized users from gaining access to the system. Authentication is the process of reliably verifying the identity of someone and ensuring the person is who he claims to be and not an impostor. Some authentication systems are accomplished with the use of a password that must be presented to the system. Other authentication systems require the user to present a physical key or physical characteristics, such as fingerprints. There are lots of examples of authentication in our everyday life:

(1) By presenting the student ID card, the student is authorized to enter the VISC lab at Virginia Tech.
(2) At an automated machine, you identify yourself using your ATM card. You must input a personal identification number (PIN) to authenticate yourself.
(3) An online order company might accept as authentication the fact that you know the expiration date on your credit card.

2.2.1 Authentication Objectives

By providing only authorized users with an authentication to access the system, unauthorized users have no means of access. But this does not mean that unauthorized users are unable to gain access to the system because the system does not authenticate the identity of a user, but who the user claims to be. That is, unauthorized users are able to access the system by appearing to the system as an authorized user.

Authentication is tremendously important in computer and network applications. The other party you communicate with may be in the next room or another city. You have none of the visual or aural clues that are helpful in everyday communication. Since the system cannot verify the user's true identity, measures must be taken to defend the system. That is, the primary objective of an authentication system is to prevent unauthorized users from gaining access to the system[10].

2.2.2 Authentication Methods

The authentication methods have been categorized into three different types based on something the user knows (e.g. a password), has (e.g. a physical card) or is (e.g. fingerprints)[11].

The first type is currently the most commonly used. A system requires the user to provide specific information such as password or pass phrases to access the system. This method has the advantage of both simplicity and easy implementation. However, passwords often provide an unreliable basis for authentication. The big problem with password-based authentication is eavesdropping[8].

The second type requires physical objects that a user must have to access the system. Examples of physical objects include smart cards and magnetic cards[5][8]. This method provides a higher level of security than passwords alone and also is simple to use.

The third type is advantageous as it may not be easily guessed or stolen. It is using a user's physical characteristics such as a fingerprint or hand geometry and matching them against a profile to grant or deny access[5]. However, the technology required to implement most biometrics methods is very expensive.

2.2.2.1 Passwords

Passwords or pass phrases are predefined words that an authorized user knows and provides to the system for granting access. The passwords or pass phrases may be assigned by the system or selected by the user. This type of method is one of the most widely used authentication methods. When a user presents the password that matches the password stored in the system for that user, the system grants access.

On some systems, passwords are administratively set to a fixed attribute of a person, such as their birth of date or their email address. Although they are easier to remember, they are often easily guessed. Many systems allow users to select their own passwords that are difficult to guess. They also may be guessable if the imposter guesses enough times. In general, the password should be both easy to remember, but difficult to guess. In fact, any password, no matter how carefully chosen, can be guessed by enumerating all the finite characters sequences until you get the correct password.

There are many real problems with password-based authentication. For example, an eavesdropper might see the password when a user is using it to log in, the password might be easy to guess, or an intruder might read the file where the system stores passwords. Several approaches can be taken to reduce such threats and to prevent unauthorized people from gaining access to passwords. First, people should never write a password down, and should change the password on a regular basis. Second, the system could keep track of the number of consecutive incorrect passwords entered for an account, and when the number exceeds a threshold, "lock" the account and refuse access, even with a correct password, until the system is administratively reset[8]. Third, the user’s password file could be encrypted[12].

2.2.2.2 Physical Keys

Physical keys are objects that a user must have in his possession to access the system[13]. The most common is the key that we use everyday to unlock our home or car. In computer and network systems, the three commonly used physical keys are magnetic cards, smart cards, and specialized calculators[5].

A credit card is one example of magnetic cards. It is a piece of non-conductive material with an additional piece of magnetic recording material attached[13][14]. The advantages of using magnetic cards are that they are not easy to forge and they can hold information that is not easily memorized. There are disadvantages to using magnetic cards. First, in order to use the magnetic card technology, the system requires hardware (e.g. card reader) on each access device, such as an access terminal or a physical access path (e.g. a door). Second, the card can be stolen or lost. To improve security, the card must be used with a PIN or password for authentication.

A smart card is a better type of physical key. It is a device similar to the credit card but is capable of processing, storing, and manipulating data. The elements of a smart card are a microprocessor, memory, input/output devices, and a power source[5][15]. Due to the processing capability, smart cards have a lot of applications such as phone cards, vending machines, and road tolls. Smart cards are also being used in cellular phones and satellite TV's. Attempts are now being made to integrate all the applications into a single card. Like magnetic cards, the serious practical problem with smart cards is the need for card readers at every access terminal. But the information stored on a smart card is safer than that stored on a magnetic card because smart cards are more difficult to duplicate and offer substantial protection against eavesdropping.

2.2.2.3 Biometrics

Biometric keys are used in the user authentication process due to their characteristics and recent advancements in technology. Biometric keys provide many advantages over password-based keys or physical keys because they belong to the authorized user and are difficult to reproduce. There are various biometric keys. Some commonly used ones are fingerprint, hand geometry, and retinal prints. The authentication process allows the granting or denying of user system access based on the user's unique physical characteristics. The whole process is accomplished by obtaining an unverified sample from a user and comparing it to a previously authenticated sample. Biometric key authentication is very accurate and very difficult to reproduce. With the rapid advances in technology, biometric authentication is becoming an attractive authentication method.

2.3 Access Controls

Access control is the mechanism used to protect information from unauthorized access. Access control can define not only who or what process may have access to a specific resource, but also the type of access that is permitted[15]. Permissions are defined by the system access control policy. An access control policy basically includes a set of rules that describe the methods in which a user can access the system resource. Various schemes can be used to implement access control and protect system resources, such as a password-based scheme, a capabilities based scheme, an access control list, and protection bits. Only two of these are discussed below.

2.3.1 Password-based Scheme

A password-based scheme is similar to its usage for user authentication. In order to gain access to information (e.g., a file), the user must present the information's password to the system. In this technique, the password for specific information is different from the password for the system. The system manager or the owner of the information originally assigns a password to the information. Then the information's password must be told to the user who is to be given access to the information.

Conceptually, this technique is easy to understand and easy to implement. It also provides a fine grained security level. But it has several problems and is not suitable for most environments today. According to this scheme, users have to remember each password for each specific information they will access. The first problem is that the user needs to remember a large number of passwords. Also, with this technique there is no way to determine who has access to specific information because anyone who knows the password could access the information.

2.3.2 Access Control List

An access control list (ACL) is a table that tells a system which access rights each user has to a particular system object, such as a file directory or individual file. Each object has a security attribute that identifies its access control list. The list has an entry for each system user with access privileges. The most common privileges include the ability to read a file (or all the files in a directory), to write to the file or files, and to execute the file (if it is an executable file, or program). Windows NT[30], Novell's NetWare 5[31], Digital's OpenVMS[32], and UNIX-based systems are among the operating systems that use access control lists. The list is implemented differently by each operating system.

There are several advantages in using this method. Since a list is associated with each file instead of each user, it is easier to determine who has access to a specific object and easier to change a user's access permission. This technique also has two disadvantages. First, it is very restrictive because a user can only access the objects explicitly named in the list and an object can only accessed by a specified set of users in its ACL. Second, it becomes difficult to update and check all the object's ACLs when objects are changed.

2.4 Database Security

Database security, as a whole, is considered to deal with the confidentiality, integrity, and availability of the data stored in a database system. Confidentiality means information is only disclosed to those users who are authorized to have access to it. Integrity means information is modified only by those users who have the right to do so. For example, an employee should not be able to modify his own salary or change data concerning other payments. Availability means that information and other resources can be accessed by authorized users when needed.

For most environments, the confidentiality and integrity are often taken as a combination. Normally this is the case for systems where incorrect data could result in vast loss. A database may suffer from any of four different types of vulnerabilities[5]. These vulnerabilities are inference, aggregation, data integrity, and Trojan Horses. The following paragraph gives definitions of these terms.

Inference means the derivation of new information from known information. The problem is that the new information may be classified at a level for which the user has no permission. To properly protect a database from inference is not an easy task because inference vulnerabilities allow a user, either authorized or unauthorized, to determine the contents of the data without performing any malicious actions. Aggregation is the result of assembling or combining different tables of data when handling sensitive information. Aggregation of data might result in the dissemination of what would otherwise be considered hidden, or private, information. . Data integrity vulnerabilities allow the unauthorized or inappropriate modification, addition, or deletion of database data. A Trojan Horse is a program that performs a task that a user does not expect and does not want completed[5]. Unlike inference and aggregation, a Trojan Horse attacks the database operations and data.

Several security mechanisms can be used to prevent such vulnerabilities. These mechanisms are often similar to those employed in non-database systems, and they emphasize different applications of cryptography.

2.5 Cryptography

Cryptography is the science of writing messages that no one except the intended receiver can read[15][16]. Cryptography provides stronger methods of authentication. Two examples of using cryptography are given.

A user can encrypt the data on his computer so that even if some people gain physical access to the computer, they cannot access the data. Most web browsers (e.g., Netscape Communicator or Internet Explorer) now support SSL (Secure Sockets

Layer)[17], a cryptographic protocol that encrypts information before it is transmitted over the Internet. Thus, a user can shop online using his credit card number without worrying too much that the number will be stolen.

2.5.1 Public Key Cryptography

Public key cryptography is a system based on a pair of keys. The idea is to use two separate keys: a private key that need not be revealed to anyone, and a public key that is preferably known to the entire world. These two keys are used for encryption and decryption, and the decryption key cannot be derived from the encryption key. All known methods are quite slow, and they are usually only used to encrypt session keys that are then used to encrypt the bulk of the data using a symmetric cipher.

The algorithm could work in two different ways (shown in Figure 2.1). For example, the public key could be used to encrypt a message by someone who has never met the recipient, and only the recipient with the private key could read the message[15][16]. Or the private key could be used to sign a document, and anyone could use the public key to verify the signature[15][16]. The second way is often called a digital signature. Either way, two people do not need prior contact to communicate privately over an insecure channel.

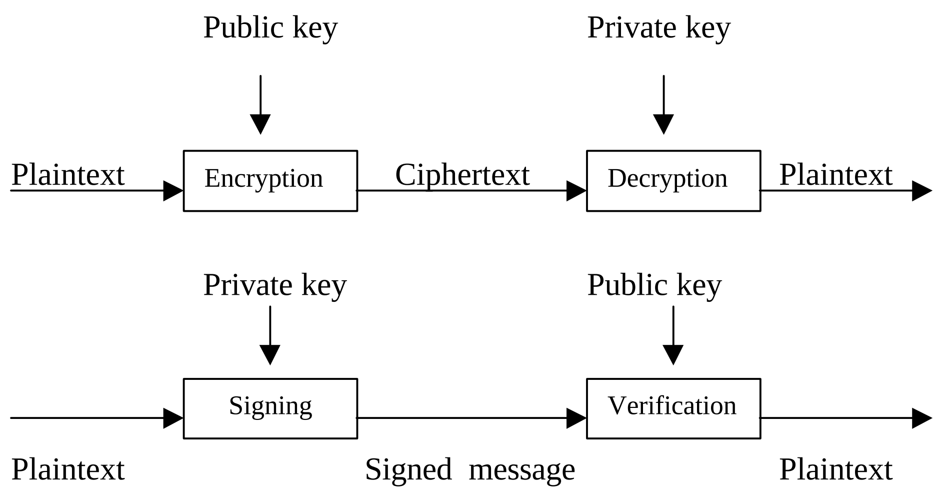

Figure 2.1 Two Methods of Public Key Cryptography

The basic idea is that public key cryptography provides a means by which two ends of the network can trust each other. For example, in a simple client/server application, the client application could send the user's credentials encrypted with the server's public key and send them to the server. Then the server could decrypt the user's credentials and authenticate the user's identity. If someone acquires the encrypted messages by listening in on the entire process and has the server's public key, however, he can not recover either the client's credentials or the server's private key. This offers both confidentiality and integrity for network messages.

In addition to the authentication example described above, public key cryptography also can solve many of the security problems of large, heterogeneous networks. In the proposed software architecture in this thesis, public key based cryptography has been built into **TEAMDEC** to provide user authentication.

Two commonly used public key algorithms, RSA[18] and Diffie-Hellman[4], are discussed below. Detailed descriptions of these two algorithms are given in [16].

2.5.1.1 RSA Algorithm

RSA (Rivest-Shamir-Adelman)[18] is the most commonly used public key algorithm. It can be used both for encryption and for signing. It is generally considered to be secure when sufficiently long keys are used (512 bits is insecure, 768 bits is moderately secure, and 1024 bits is secure, see Table 5.2). The security of RSA relies on the difficulty of factoring large integers. Dramatic advances in factoring large integers would make RSA vulnerable. RSA is currently the most important public key algorithm. At present, 512 bit keys are considered weak, 1024 bit keys are probably secure enough for most purposes, and 2048 bit keys are likely to remain secure for decades. One should know that RSA is very vulnerable to chosen plaintext attacks - a form of cryptanalysis where the cryptanalyst may choose the plaintext to be encrypted[19]. In a chosen plaintext attack, the cryptanalyst not only has access to the ciphertext and associated plaintext for several messages, but he also chooses the plaintext that gets encrypted. Because the cryptanalyst can choose specific plaintext blocks to encrypt, they might yield more information about the key[16]. If the key used to encrypt or decrypt the messages is deduced, the algorithm is broken. The RSA algorithm is believed to be safe when used properly, but one must be very careful when using it to avoid these attacks. Many implementations of RSA are freely available.

2.5.1.2 Diffie-Hellman Algorithm

Diffie-Hellman[4] is a commonly used public-key algorithm for key exchange. It is generally considered to be secure when sufficiently long keys and proper generators are used. The security of Diffie-Hellman relies on the difficulty of the discrete logarithm problem (which is believed to be computationally equivalent to factoring large integers)[16]. Diffie-Hellman is sensitive to the choice of the strong prime and the generator. The size of the secret exponent is also important for its security. Conservative advice is to make the random exponent twice as long as the intended session key. In practice, if the same prime is used for a large number of exchanges, it should be larger than 512 bits in size, and preferably 1024 bits.

2.5.2 Digital Signature

A digital signature is used to verify that a message really comes from the claimed sender. It is completely analogous to a handwritten signature used for authentication. In cryptography, a digital signature is a block of data that was created using some private key, and there is a public key that can be used to verify that the signature was really generated using the corresponding private key. It is fully described in[20]. Digital signatures provide two important functions: authentication and integrity.

The algorithm used to generate the signature must be such that without knowing the private key it is impossible to create a signature that would be verified as valid. Digital signatures can also be used to timestamp documents, a trusted party signs the message and its timestamp with the private key, thus testifying that the message existed at the stated time[16].

A digital signature of an arbitrary document is typically created by computing a message digest from the document and concatenating it with information about the signer and a timestamp. The resulting string is then encrypted using the private key of the signer and a suitable algorithm. The resulting encrypted block of bits is the signature. It is often distributed together with information about the public key that was used to sign it. To verify a signature, the recipient first determines whether it trusts that the key belongs to the person it is supposed to belong to and then decrypts the signature using the public key of the person. If the signature decrypts properly and the information matches that of the message (proper message digest), then the signature is accepted as valid. Several methods for making and verifying digital signatures are freely available. The most widely known algorithm is RSA.

2.5.3 Message Digest

A message digest is also known as a cryptographic hash function[15]. It is basically a mathematical transformation that takes a message of arbitrary length, transforms it into a string of bits, and computes from it a fixed-length number. A cryptographic hash function does this transformation in a way that makes it extremely difficult to come up with a message that would hash to a particular hash value. Cryptographic hash functions typically produce hash values of 128 or more bits. This number is vastly larger than the number of different messages likely ever to be exchanged in the world. Many good cryptographic hash functions are freely available. Well-known ones include MD5[33] and SHA[25].

Detailed algorithm descriptions on MD5, SHA and digital signatures are discussed in[16].

2.6 The Java Security API

The Java security API is a set of packages that are used for writing secure programs in Java. In particular, JDK 1.2 contains substantial security features enhancements: policy-based, easily-configurable, fine-grained access control; new cryptographic services and certificate and key management classes and interfaces[21]. Two major components, JCA and JCE, will be discussed below.

2.6.1 JCA

JCA stands for Java Cryptography Architecture. It was first introduced in JDK1.1 and refers to a framework for accessing and developing cryptographic functionality for the Java platform[22]. It specifies design patterns and an extensive architecture for defining cryptographic concepts and algorithms.

The JCA includes a provider architecture that allows for multiple and interoperable cryptography implementations. The term cryptographic service provider (CSP), or simply provider, refers to a package (or a set of packages) that supplies a concrete implementation of a subset of the cryptography aspects of the JDK Security API[22]. In JDK1.1 a provider contains an implementation of one or more digital signature algorithms, message digest algorithms, and key-generation algorithms. JDK 1.2 adds five more types of services: keystore creation and management, algorithm parameter management, algorithm parameter generation, key factory support to convert between different key representations, and certificate factory support to generate certificates and certificate revocation lists (CRLs) from their encodings. The JDK1.2 comes with a default provider, named SUN, that implements a few cryptographic algorithms including a number of DSA services, MD5 and SHA-1 message digest algorithms, and X.509 certificates.

2.6.2 JCE

JCE means the Java Cryptography Extension. Sun divided its cryptographic classes into two groups because the U.S. government limits their export. The first group is included in standard JDK1.2 and can be exported without limitation. The second group is called JCE, and is for U.S and Canadian distribution only.

JCE extends the JDK to include API's for encryption/decryption, key exchange, and message authentication code (MAC)[23]. JCE and the cryptography aspects of the JDK provide a complete, platform-independent cryptography API. The algorithms supported in JCE include encryption/decryption, password-based encryption, and Diffie-Hellman key agreement. By employing this package, one can efficiently reduce development time and prevent errors in applications.

The Java security API is used to implement the security features of **TEAMDEC**. Because the API provides extensive support for cryptographic algorithms and protocols, hides a lot of implementation details, and exposes cryptographic concepts, it is suitable for **TEAMDEC**'s design and implementation. A security requirement analysis of **TEAMDEC** is provided in Chapter 3. Detailed descriptions of these algorithms and implementations are provided in Chapter 4.

Chapter 3 Security Issues in TEAMDEC System

The development of distributed computing has recently accelerated toward a system for sharing data, applications, and computing resources across networks. As applications become more distributed, more useful features are introduced, such as those that enable data sharing and information gathering. However, distributed systems also introduce an increased security hazard. Each site could have a security risk similar to that of a centralized system, as well as the problems introduced by connecting to a network. For successful use of the application, one critical element - security - must be considered.

This chapter first gives a detailed description of **TEAMDEC**. Then the system is analyzed with respect to security requirements and the type of security measures required to be taken.

3.1 Overview of TEAMDEC

3.1.1 What is TEAMDEC?

TEAMDEC is a team based intelligent decision support tool, which integrates information from different sources to facilitate decision-making[2][3]. With **TEAMDEC**, people can make decisions using multiple information sources such as scripts from the script database, information from the commercial search engines (e.g., Yahoo, AltaVista) on the Internet, or through interactive real-time audio/video communication (e.g., video conferencing) with other decision team members. All these features can help people to make better decisions more efficiently, whether working alone or with a decision team.

A key element of **TEAMDEC** is the script database, which can be used to guide team decision-making. The script database enables a user to specify a set of information/actions that should be taken, and to label this "script" with a scenario name. Thus, when faced with the real situation, the user activates the script and follows the desired procedure. For example, a user intends to carry out a discussion with a number of on-line users about the issues of flight safety. The user must first set the option for suggestion acceptance to “On”, which indicates that the user wishes to get action guidance. After that step, the user’s activities are traced. Simultaneously, the system invokes its inference engine to derive timely action suggestions based on the user’s real-time activities in the **TEAMDEC** system, knowledge from the action database and other databases, and rules which are related to predictive judgment, evaluative judgment, and decision making. The system consequently produces advice on possible actions. Generally, the suggestion information is organized in the structure of the script as shown in Table 3.1. The script consists of two scenarios: the development of a discussion group and the group discussion[2].

Develop a discussion group	Open a window for selecting users
	Display the user information
	Select the user and add to the discussion group
	Remove a user from the discussion group
	Confirm and quit
Begin a group discussion	Open a window for group discussion
	Give a topic
	Send an invitation with the topic to group members
	Get agreement replies from members
	Setup communication channels and begin discussion
	Edit the opinion expression and pose it out
	Send an off-line message
	End the discussion

Table 3.1 Components of a Group Discussion Script[2]

The functionality of the scripts is one of the significant characteristics of **TEAMDEC**. This unique feature makes **TEAMDEC** a flexible, useful, decision making tool. From the table, we know that a script is a set of information that is stored in the script database to guide a user's possible actions. Thus the script database is the most important system asset in **TEAMDEC**.

3.1.2 TEAMDEC's Architecture

TEAMDEC is a team-based decision support system, in which a leader makes a decision with the input of a flexible team of people[3]. Figure 3.1 below shows the main components of **TEAMDEC** and their connections.

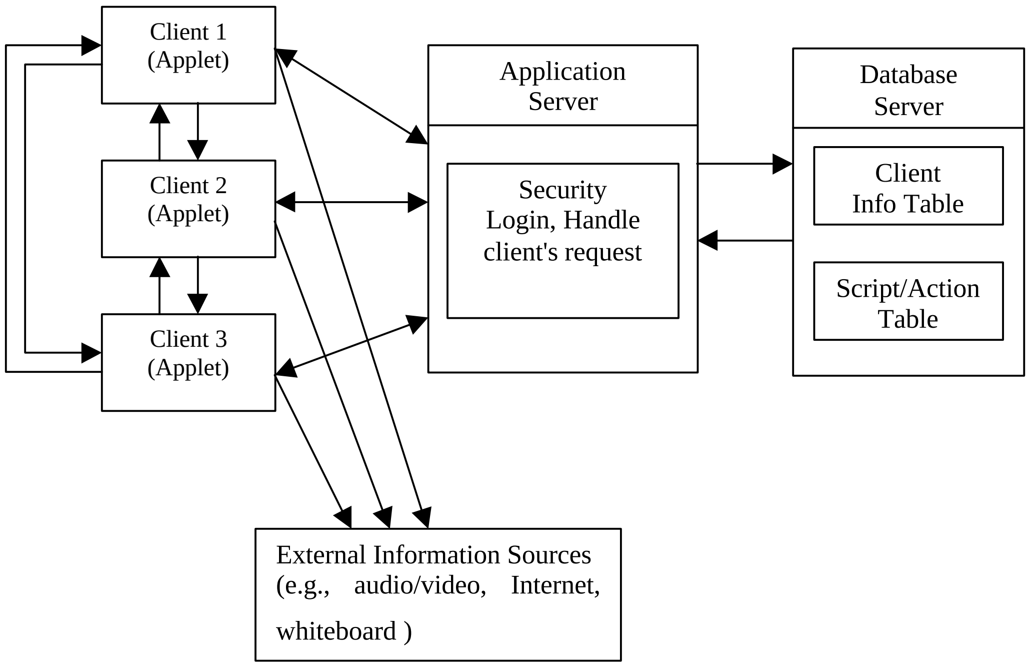

Figure 3.1 TEAMDEC System Architecture

Basically, the system is a multi-user client/server system and employs a three-tier architecture. The system has three components: the applet (also called client), the application server, and the database server. The applet is responsible for GUI display, communicating with other **TEAMDEC** applets, and sending requests (e.g., save request for saving actions, query request for querying about desired scripts) to the database through the application server. The application server is the middle-ware between the applet and the database server. The server is responsible for user authentication and handling client's requests. The database server is responsible for storing and operating system critical data including user information and actions/scripts.

3.2 Security Requirement Analysis

The process of analyzing system security requires the examination of security objectives related to confidentiality, integrity, and availability. The following section gives a detailed analysis of the system requirements in **TEAMDEC**.

3.2.1 Confidentiality Objectives

TEAMDEC is largely dependent on information transmitted among different decision team members and between clients and servers to make it a flexible decision-making system. Without reliable, confidential information, it is not possible to achieve an efficient, better quality decision among decision-team members. Thus, ensuring the confidentiality of the system information and protecting it from unauthorized access becomes the most important task in security analysis. The types of information in **TEAMDEC** are listed below.

The first type of information is that transmitted among different clients. The information would be an invitation, a request, or a response. For example, an invitation message sent by a user to his team members would be: "The discussion topic is Emergency, Would you like to join my discussion group?" The second type of information is the information transmitted between a client and the application server. The information sent by the client to the server would be the client's personal information such as user name, user ID, the client's request code, and the client's action records or scripts. The request code is a digital number that identifies the type of request that the server will accept. A record or a script is a string consisting of a series of strings, such as time, activity, description of the activity, and name of the action. These strings are concatenated together and describe the user's actions. The information sent back from the server to the client would be the server's response code. The response code is also a digital number that identifies the type of response from the server and the type of scripts retrieved from the database server according to the client's search criteria.

These critical system data are all in plain text if they are not encrypted, which means that when transmitting them from sender to receiver, they will not be secure over the Internet. The data could be accessed by unauthorized people, and could be altered or viewed by a third party. When transmitting the data over the Internet, such a lack of security would be a major flaw.

In addition, if users from inside can get out to the Internet and get valuable information, then without precautions users from outside can get into the local system. This applies not only to the Internet, but also to any system if it allows users to come in from the outside. Interconnection of networks introduces vulnerabilities as a result of network resource sharing with public users. Remote access can leave a system vulnerable to hackers, viruses, and other intruders. The client in **TEAMDEC** is implemented as a Java applet. Although Java can provide several benefits such as platform independence, it also introduces potential security issues.

Because the applet is a program written in the Java language, it can be included in an HTML page. When using a Java-enabled browser to view a page that contains an applet, the applet's code is transferred to the local system and executed by the browser. Thus, anyone could run the applet anywhere in the world and access **TEAMDEC** as long as he used a Java-enabled browser to view the page. This will not be allowed because only an authorized user should be able to access **TEAMDEC**. To prevent such a situation from happening, user authentication is essential.

In a distributed system such as **TEAMDEC**, a client sends sensitive information to another client or to a server through an insecure communication channel, which may have the following risks. The system data of **TEAMDEC** is highly sensitive. The data could be stolen and disclosed, which would do damage to the whole system and impact the decision-making capability. Someone using the Internet could disguise himself as an authorized user of **TEAMDEC or** could change the information content. For example, an intruder could look at the data packets transmitted on the network and intercept network data sent from client to server or vice versa via network eavesdropping. Eavesdroppers can operate from any point on the pathway between the two communication end points. This is one of the most difficult threats to be detected. Over the Internet, data packets may come through several nodes or even take different paths. For the end point users, there is no way to know exactly what happened during transmission. In the case of wireless communication, the media is accessible to anyone. For mission-critical information transmitted on the Internet by **TEAMDEC**, security measures must be taken to prevent eavesdropping. Data encryption is the only defense against this kind of threat.

The following example could describe the possible attacks mentioned above.

An airplane is in an emergency situation because its landing gear are locked. When the airplane approaches the airport, the pilot calls for help. With **TEAMDEC**, the controller opens the decision script for airplane landing. Following the script prompts, the controller contacts the manufacturer of the airplane and gets possible suggestions. Then the controller integrates the information from different sources and makes a decision with his decision team. After the possible decision is made, the controller communicates with the pilot and sends him the instructions for a safe landing.

During the whole rescue procedure, there are three possible attacks. First, information from the pilot is stolen or modified. If this mission-critical information is modified, then the controller will not get the correct information and may make a decision that is not suitable for a safe landing situation. Second, the scripts in the database are modified or deleted. If someone modifies the airplane landing script in the script database, for example, then the sequence of instructions from the scripts are modified and the decision team will not make a correct decision. Third, the instructions from the controller to the pilot are stolen or modified. If the important instructions are modified or lost, the pilot can not get the correct instructions and a safe landing will not be accomplished.

Under any of the attacks described above, the rescue task can not be completed successfully and the damage will be great. As seen from this example, security considerations impact the whole system. This example demonstrates the most important objectives of **TEAMDEC**, confidentiality and integrity. For secure, timely, and accurate delivery and receipt of the system data, **TEAMDEC** requires user authentication, access control, and encryption/decryption. These make up the desired security features of **TEAMDEC**.

Different types of user authentication to prevent unauthorized access exist. Different methods may provide different levels of assurance and be used to protect against different threats. In **TEAMDEC**, there is a need not only for confidentiality, but also for strong user authentication, because **TEAMDEC** is a distributed system over the Internet.

A password-based authentication scheme is a simple authentication method, but it is prone to attack by a user guessing password values. In a relatively low risk environment, it can provide protection for the system from attacks, in which an imposter cannot see, insert, or alter the information passed between the sender and the receiver during an authentication process. But using such a simple authentication method in a distributed network system can only provide weak authentication. The user authentication method in **TEAMDEC**, which will be discussed in the next chapter, uses digital signatures with public/private keys and message digest function for the communication between client and server. It provides a strong user authentication method, and it ensures mutual confirmation of identification.

3.2.2 Integrity Objectives

Integrity is an essential requirement for the proper operation of **TEAMDEC.** The scripts stored in the database are the heart of **TEAMDEC**. By using this critical information, the system helps develop a decision, and provides suggestions such as who should be contacted or which command should be issued when a specific situation happens.

Everyone's role in the decision team environment is unique. For example, one team member might be the team leader, who gets information from different sources and coordinates them before issuing a decision. Another team member might only collect information from the Internet and report it to the team leader. These different functions may dictate that access rights to the scripts must also be different.

Shared use of **TEAMDEC** by independent team members introduces the risk of unauthorized access to the critical system data and hence compromises the system's integrity. Because in today’s networks, disgruntled employees, hackers, and other forms of destruction are common, the threat may come from within the **TEAMDEC** user community or from outside the system. For an outside threat, an unauthorized user could break into **TEAMDEC** and modify some system critical information. For an inside threat, any registered users can access, distribute, and even change valuable information data if proper security measures are lacking. Consider the following examples.

A user of **TEAMDEC** executes some actions such as changing record actions or deleting scripts that he is not authorized to do. This is the kind of attack that breaks the integrity of critical information. A disgruntled user of **TEAMDEC** would retrieve the scripts from the script database and send them to other people who are not related to **TEAMDEC**. This makes confidential information open to others and destroys the security of the system.

The user authentication method can be used to prevent unauthorized access from the entry point of the system, while access control can protect highly sensitive information from being disseminated to users who are not authorized to receive it.

Several access control schemes have been suggested in the literature[24][34][35]. One of these is task and role-based access control for distributed applications[24]. In order to provide a fine-grained access control level to the critical scripts in **TEAMDEC**, a role-based access control method is employed. The access is granted by the user's role in the system. In the next chapter, the method and its implementation will be discussed in detail.

In addition to the two major security aspects of **TEAMDEC**, confidentiality and integrity, another security objective is availability. This issue is related to denial of access, data loss, or disaster recovery. This is one area that many organizations may overlook[1]. As more and more organizations begin to utilize the Internet for their business, "availability" will be a key requirement for success.

3.3 An Architecture for Achieving a Secure System

As the Internet expands its reach and scope, there are a growing number of computer hackers who attempt to penetrate computers. Computer fraud is becoming more widespread and sophisticated. The previous discussions on security requirement analysis of **TEAMDEC** indicate a critical need to protect system information, such as scripts, records, and users' personal information. The proposed security schemes in this thesis should provide confidentiality and integrity.

3.3.1 Possible System Attacks

The first step in designing a security architecture is to analyze system requirements and identify the possible risk areas in the running environment of the system. Based on previous discussions, we can examine **TEAMDEC**'s architecture again and identify the possible attacks. Figure 3.2 shows the communication among the various parties and the possible attacks.

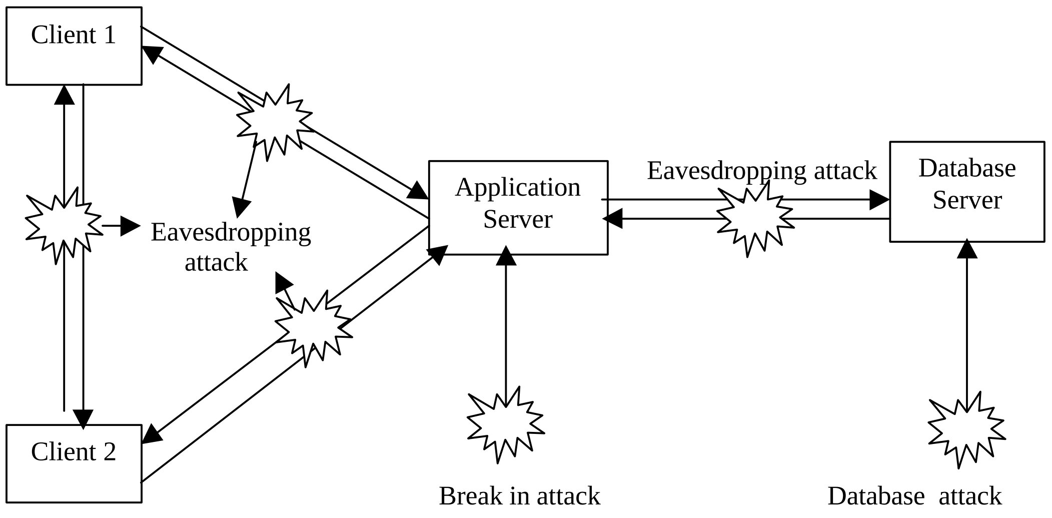

Figure 3.2 System Architecture and Possible Attacks

A *Break in attack* is a type of attack in which the attacker tries to break into the system and obtain unauthorized access to system information. It can be prevented by a user authentication method. An *eavesdropping attack* is a type of attack in which the eavesdropper captures or modifies the information between the two end points of a communication channel. In **TEAMDEC**, this type of attack is possible between the clients, between the client and the application server, and between the application server and the database server. This type of attack can be prevented by using an access control scheme, and/or by using an encryption/decryption method. The *database attack* is a type of attack in which the attacker tries to gain access to the information database directly. An access control method implemented in the application server and in the database server can prevent such attacks.

Once the possible attacks have been identified, the next step is to protect the system from such attacks by using user authentication, access control, and encryption/decryption methods.

3.3.2 Security Methods

TEAMDEC employs a three-tier architecture. A three-tier architecture is a special type of client/server architecture consisting of three well-defined and separate processes including a user interface, a middle tier, and a database management system, each possibly running on a different platform. The user interface runs on the user's computer (the client). The middle tier runs on a server and is often called the application server and actually processes data. A database management system (DBMS) runs on another server and stores the data required by the middle tier[36].

The main drawback to this scheme is the decreased speed compared to a two-tier system in which the clients are communicating directly with the database server. However, the three-tier architecture makes the system more secure. Since the database server can only talk to the application server, it greatly limits the threat of someone connecting directly to the database server and causing damage.

Figure 3.3 shows the security methods proposed in this thesis to prevent the previously described possible attacks.

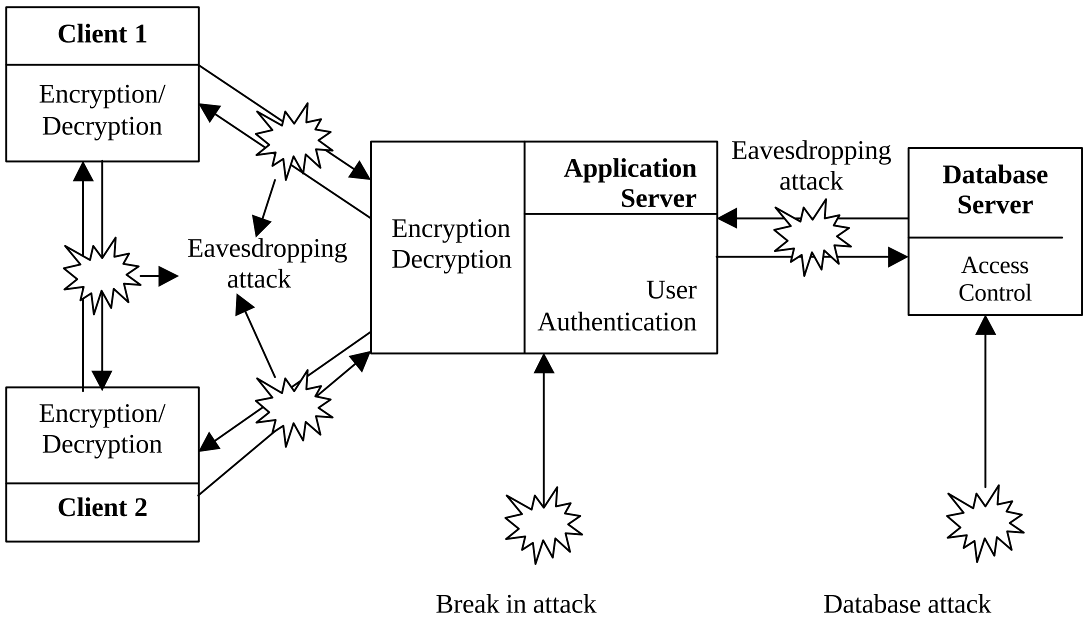

Figure 3.3 TEAMDEC Security Architecture

Figure 3.3 also shows the proposed security architecture for **TEAMDEC**. All the three possible attacks can be protected by the proposed architecture.

The user authentication scheme is based on public key cryptography[15][16][18]. Each new user is first registered by the system administrator and given a pair of DSA keys. The DSA private key is encrypted with the user's password. When the user logs into **TEAMDEC**, he must present his password and private key for the client application to generate a signature and send to the server for authentication. The server verifies the signature by using the user's public key. If the signature has been successfully verified, then the user has been authenticated and will be granted access to **TEAMDEC**.

The encryption/decryption method employed in this architecture prevents an eavesdropping attack. At the beginning of the communication, two end points agree on a session key by using the Diffie-Hellman key agreement protocol[4][16]. The remainder of the communication is encrypted using the session key at the sender and is decrypted using the session key at the receiver. The detailed algorithm description and implementation are presented in the next chapter. Due to the communication between two end points being encrypted, the information on the network is no longer plain text. Although an eavesdropper could capture the message, he can not recover the message.

There are two types of access control implementation. One is the access control scheme that is implemented in the database server by the vendor. The access control list can be configured using a database provided function. The other is proposed and implemented in the application server for fine-grained access control to scripts. This kind of access control is based on the user's position in the system. It can be flexibly configured and updated.

3.4 Summary

This chapter provides an overview of TEAMDEC and its architecture, then gives a detailed security requirement analysis of TEAMDEC. The analysis involves the examination of security objectives related to confidentiality and integrity. It then identifies the three types of attacks in the running environment of the system. Based on the analysis, user authentication, access control, and encryption/decryption methods are proposed to protect the system from such attacks. In the next chapter, a detailed implementation of the three security methods is presented.

Chapter 4 Implementation

In the previous chapter, **TEAMDEC's** security risks are analyzed, and three possible attacks are identified according to its security requirements. In this chapter, a detailed implementation is presented, including the user authentication scheme for preventing break-in attacks, the key agreement protocol for preventing eavesdropping attacks, and the access control model for preventing database attacks.

The implementation is accomplished by using existing Java security packages, specifically, the Java Cryptography Architecture (JCA) and the Java Cryptography Extension (JCE). The choice of Java provides the following advantages. The Java platform is a great development environment and provides a large number of libraries and programs for network application development. Java is portable to nearly every platform, including Windows 95/98, Windows NT, and most UNIX systems. Java also provides a framework and implementations for encryption, digital signature, strong random number generating, key management algorithms and support for encryption including symmetric, asymmetric ciphers in the Java Cryptography Architecture, which makes it an excellent tool for the development of secure applications. Java also enables software reuse and integration capabilities because of its object model.

4.1 User Authentication

User authentication is implemented in **TEAMDEC** to prevent unauthorized users from stealing confidential system information not intended for their eyes and modifying and executing scripts stored in **TEAMDEC.**

Through the user authentication method, the system can assure that the remote user is, in fact, an authorized user. Passwords are a simple solution to authentication; they are easy to implement, but they are not considered very secure, especially when people choose easy-to-guess passwords, or write down passwords in obvious places. A public key based authentication scheme, which is used in this project, is a stronger form of authentication. The whole authentication procedure including key generation and key distribution is outlined below.

(1) Prepare public/private key pair

In **TEAMDEC**, a new user first must be registered. The **TEAMDEC** system administrator generates a pair of public and private keys for each user using the DSA algorithm[20].

(2) Encrypt the user's public key

The user's DSA private key is encrypted with the user's password. The name of the encryption algorithm implemented in JCE is PBEWithMD5AndDES. PBE stands for passphrase-based encryption. PBEWithMD5AndDES is a particular variant of PBE; a MD5 message digest is used to digest the passphrase. The digest value is then used as a DES key. The detailed algorithm is described in PKCS#5, a document published by RSA Data Security, Inc[12].

(3) Client generates signature

Upon logging in, the user is prompted for his user name and password. The password is used to decrypt the user's private key. The client generates a timestamp and a random number. Then the client creates a signature using this data and his private key. After the digital signature is generated, the client sends this information to the server for verification. Figure 4.1 shows the method[7] on the client side:

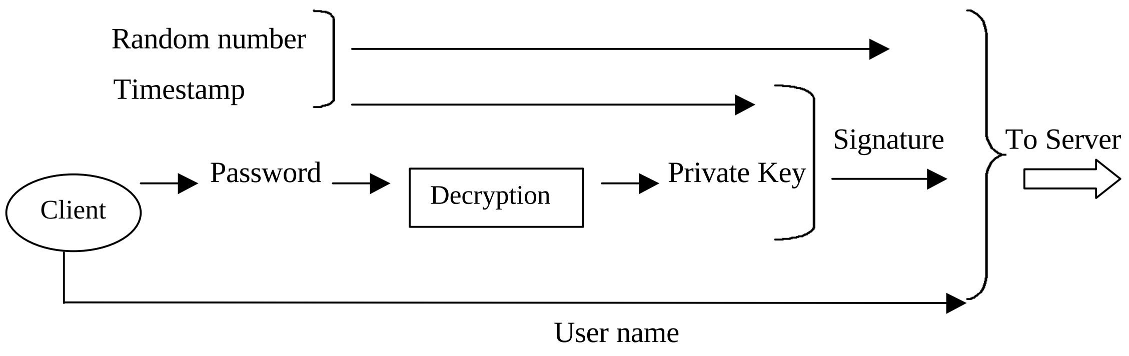

Figure 4.1 User Authentication Method - Client Side

(4) Server verifies signature

After receiving the information from the client, the server retrieves the user's public key according to the user's user name. The server verifies the signature with the client's public key and decides either to allow the user into the system or deny access.

A signature provides two security services, authentication and integrity. Recall from Chapter 2 that a signature is a message digest that is encrypted with the signer's private key. Only the signer's public key can decrypt the signature, which provides authentication. If the message digest matches the decrypted message digest from the signature, then integrity is also assured.

The name of the algorithm used to generate the message digest in JCA is SHA. SHA stands for the Secure Hash Algorithm that was developed by the National Institute of Standards and Technology (NIST)[25]. Figure 4.2 shows the method[7] on the server side.

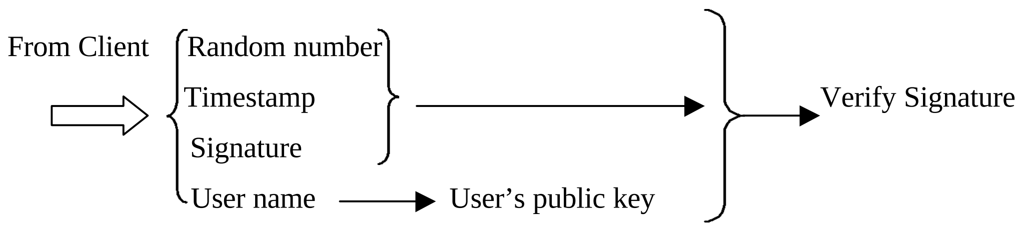

Figure 4.2 User Authentication Methods - Server Side

Safely maintaining and distributing the private/public key files is the most important task. Usually, it involves saving the private key file on a removable media, and carrying it when logging in. Detailed discussions on public/private key security and a possibility of storing the private key file on a smart card are presented in Chapter 5.

4.2 Data Encryption to Prevent Eavesdropping Attack

In **TEAMDEC**, communication among clients, or between client and server are on an insecure medium, the Internet. It is not hard to eavesdrop because data is sent in plain text over the network. In order to guarantee safe communication between different clients and between client and server, cryptography must be added to the system, providing authentication for each end of the conversation and encryption for the communication itself. A common way to protect information is to encrypt it at the sending end and decrypt it at the receiving end.

The Diffie-Hellman (DH) key agreement protocol[4] is employed in **TEAMDEC** to secure conversations on networks. At the beginning of the conversation, each end point agrees on a session key that will be used to encrypt each data packet sent between them. The remainder of the conversation is encrypted using the session key. The following procedures show how the protocol works between two communication end points, called Client A and Client B. The whole key agreement procedure is presented in Figure 4.3.

(1) At the beginning, create new DH parameters - base and modulus.

(2) Client A creates his own DH key pair using the DH parameters above. Then he executes Phase1 of his version of the DH protocol. He encodes his public key and sends it to Client B.

(3) B has received A's public key in encoded format. He creates a DH public key from the encoded key material. B gets the DH parameters associated with A's public key. He uses the same parameters to generate his own key pair. B executes Phase1 of his version of the DH protocol and encodes his public key, and sends it over to A.

(4) Client A uses B's public key for Phase2 of his version of the DH protocol. Before he can do so, he has to create a DH public key from B's encoded key material.

(5) B uses A's public key for Phase2 of his version of the DH protocol.

(6) At this stage, both A and B have completed the DH key agreement protocol. Each generates the (same) shared secret.

During the subsequent conversation, each end can use the generated secret key to create a DES[26] key, which can be used to encrypt their messages. Although someone is able to hear the entire exchange between A and B, he will not know the shared secret and subsequent communications between them.

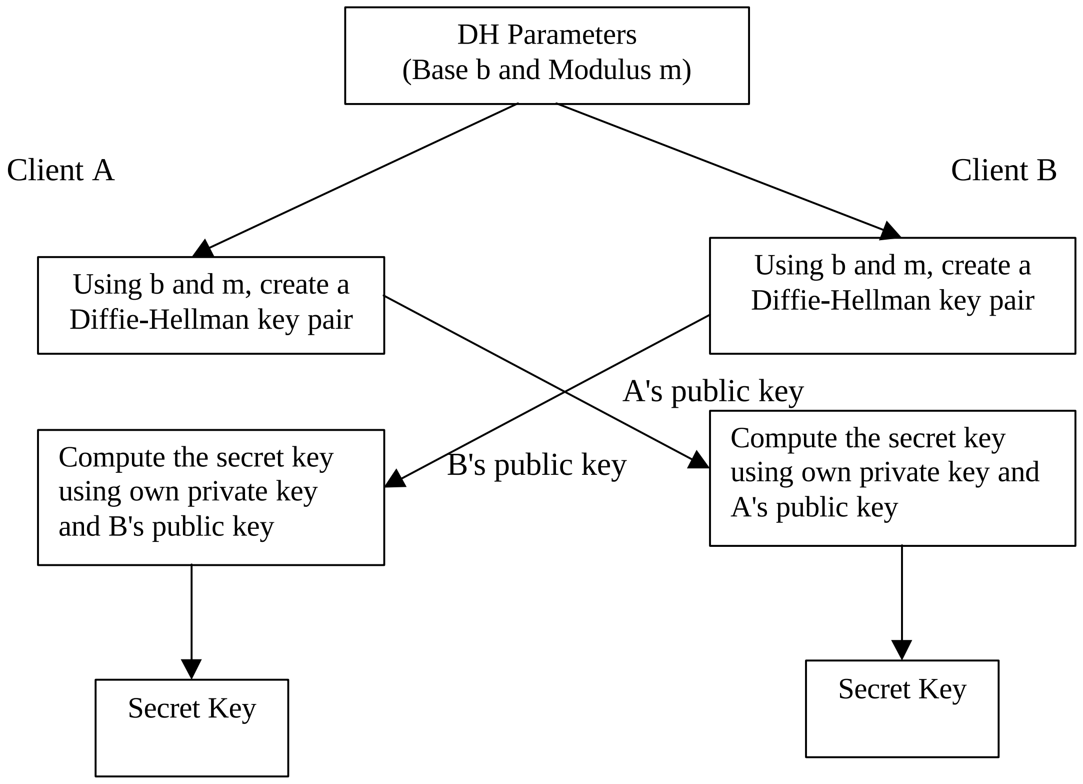
DH Parameters
(Base b and Modulus m)
Client A
Client B
Using b and m, create a Diffie-Hellman key pair
Using b and m, create a Diffie-Hellman key pair
A's public key
B's public key
Compute the secret key using own private key and B's public key
Compute the secret key using own private key and A's public key
Secret Key
Secret Key

Figure 4.3 Key Exchange between Client A and B

This key agreement protocol can easily be expanded to include three or more participants. For example, if A, B and C participate in a communication, imagine them sitting around a table. Each one picks DH parameters and computes his own DH key pair.

Then each one passes his public key to the person sitting to his right. Thus, each person can calculate the secret value based on the other person's public key and his own private key. If there are 5 participants, 4 exchanges of information are required before the secret value can be calculated[7]. During the whole procedure, five session keys will be calculated that all of these have the same value. The Diffie-Hellman algorithm is implemented in JCE.

4.3 Script Access Rules

4.3.1 User Classification

In many systems, misuse of the administrative procedures could corrupt the system data or deny service to users. In **TEAMDEC**, measures are taken to prevent such things from happening. The system decides whether to allow procedures based on the user's identity, the system scripts' attributes, and the type of operation the user is attempting. In **TEAMDEC**, users are divided into two categories: System Administrator and General Users.

(1) System Administrator

The administrator is a trusted user, authorized to perform system management and maintenance functions essential to the smooth and secure running of the system. These functions are not available to the general user; only the administrator may access them through a separate administration program. The **TEAMDEC** system administrator has exclusive control over the registered user database and scripts database. The administrator has the following responsibilities:

a. Setting up a new user's account

   To establish a user account, the administrator must run the system administration program, type certain items, such as user name, user ID, password, user position, user position ID, and group ID to describe the new user, and create an entry in the registered user database and public/private files, when placing the group ID in the user database asserts that the user is a member of the specified group.

b. Deleting an existing user's account

   When the user leaves the organization or ceases to require access to **TEAMDEC**, deleting the inactive account improves security. To do this, the system administrator may run the administration program and remove the entire entry that defined for the user.

c. Setting up user groups

   To create user groups, the administrator needs to run the administration program, type the group name, the group ID, and a list of names of the users who are authorized to be members of this group, and then place an entry in the group database.

d. Deleting user groups

   A group is deleted by removing an entry from the group database.

(2) General User

When a user logs into the system, the user needs his user name, user ID, group ID, and password for authentication. That means the user can only work in one group at a time, even though he may be a member of several groups.

The user database is used to hold information about registered users of the system. Each entry in the user database includes each user's login name, user ID, position, position ID, and group ID. The user database is totally controlled by the system administrator.

Each user must be a member of at least one group and can be a member of several groups. If one user is a member of several groups, the administrator may need to place more than one entry in the user database for this user. In this situation, the system treats the user as different users.

4.3.2 Script Access Rules

Scripts are the critical data in **TEAMDEC**. Each script has several associate attributes that are related to security. These attributes specify the script's access control level. In **TEAMDEC**, the system administrator will specify different permissions for different users and groups. The primary reason is to identify those who can read the script, who can modify the script, and who can execute script, thus, efficiently preventing unauthorized use of the system data and preventing database attack. The following paragraphs discuss this point in detail.

4.3.2.1 Access Control List

An access control list (ACL) is a table that specifies which permission each user has to a particular system object. In **TEAMDEC**, the system objects are the scripts. Each script has a security attribute that identifies its access control list. The list has an entry for each system user with access privileges.

4.3.2.2 Permissions on a Script

The three common permissions on each script are as follows:

Read Permission: The permission to display the name, commands, and description of the script in a table, regardless of whether the user has permission to access the script.

Modify permission: The permission to change the script's name, edit, and delete the script only if the user or the group has appropriate privileges to do so.

Execute permission: The permission to execute the commands of the script only if the user or the group has appropriate privileges to do that.

An access control list is associated with each script. Each access control list has one or more access control entries (ACEs) consisting of the name or ID of a user or group of users. The user ID is assigned according to the user's role in the system. For each of these users, groups, or roles, the access privileges are stated in a string. The system administrator creates the access control list for each script.

Table 4.1 shows a script table. For simplicity, the table does not list all the columns in the script table. Other items, such as Sub Command and parameters, are not listed.

Time	User ID	User Name	Command	Description	Script Name
4/12/99	alice	Alice	Group Discussion	Group Discussion session	Script a
5/03/99	bob	Bob	Send Notice	Send online notice to alice	Script b

Table 4.1 Script Table

The first column is the script name and the second column is the name of the access list. In this three-entry table, each entry in the table specifies the access control list for the script. The system would look at the access control list and determine whether the user who made the request had access privilege. In Script b, for example, the access privilege is defined in ACL2.

Though it is possible to put all the data into one table, it is not logical to do this all of the time. For example, the access control list information could be added in table1; however, the purpose of the script table is to store data on scripts. The solution is to create another table, called Table 4.2, which will contain information about the specific access control list name and associate it with Table 4.1. Table 4.2 shows a table that tells the access control list about scripts.

Script Name	Access Control List Name
Script a	ACL1
Script b	ACL2
Script c	ACL3

Table 4.2 Script Access Control List

Table 4.3 is another table that shows the content of an access list for each script. Suppose this table shows the content for ACL2. Each entry in the table determines the privileges for each system user.

User ID/ Position ID	Permission to read	Permission to modify	Permission to execute
bob	Read	modify	execute
alice	Read		execute
System			execute

Table 4.3 Access Control List Content

In this three-entry ACL example, before reading, modifying, or executing the script, the system would look at the list and determine whether the user who made the request had the access privilege. The ID bob, for example, could read the script or modify and execute it. But the user whose ID is alice could read the script or execute it, but would not be allowed to modify it.

As seen in the previous example, the three tables are related to each other. Table 4.1 contains a column that has the script name. This script name also appears in Table 4.2, which relates to Table 4.3. Table 4.3 lists the content of the access control list. Suppose a user whose ID is Tom needs to access Script b. The system would look at Table 4.2 first and identify that ACL2 specifies the access rules for Script b. Then the system scans every row in Table 4.3 and tries to find a match. If there exists a corresponding entry in Table 4.3 that matches the user's ID, then the system will grant Tom access privilege to Script b. In this example, the system cannot find an entry in Table 4.3, so Tom's request to Script b is denied.

4.4 Database Protection

In addition to the three measures discussed above, **TEAMDEC**'s database server must also be considered for security concerns. As the repository for critical and sensitive information of **TEAMDEC**, a secure database server is the key technical component in addressing the overall security requirements of **TEAMDEC**. Although network services and encryption devices also provide important measures, the database server is chiefly responsible for processing the most valuable and vital portions of the whole system.

TEAMDEC employs a three-tier architecture, adding a middle application server between the client and the database server. The rationale for this architecture is to isolate the database server for security. This kind of architecture introduces four advantages. First the database server can only handle a limited number of connections. The middle application server can have several clients on one side, while using only one connection to the actual database server, thus reducing the likelihood of breaks-ins into the system data repository. Second, moving the access-control list from the database server to the middle application server will further secure the database server, so that only certain users can access certain databases based on the privileges the system gives them. Third, the database server can be physically isolated. Finally, using a commercial off-the-shelf database package that has more security mechanisms can further improve system security.

4.5 Summary

Based on the analysis of TEAMDEC's security risks, this chapter goes into more depth on the implementation of the security architecture of TEAMDEC. The three security methods, the user authentication scheme, the key agreement protocol, and the script access control model, are presented. Finally, database protection is discussed for security concerns. In the next chapter, the security implementation of the system is evaluated and possible suggestions are provided for future development.

Chapter 5 Security Evaluation

Security evaluation is one of the important stages in the whole security architecture development. With security evaluation, the strengths and weaknesses of the implementation can be identified, and the quality of software architecture can be determined. Thus, the system can be continually enhanced in the future.

This chapter analyzes the security implementation, which is described in the previous chapter. The analysis is based on functionality and the ability to prevent different types of attacks. In addition, the advantages and disadvantages of the security algorithms are evaluated, and possible improvements are suggested for future developments.

5.1 User Authentication

5.1.1 Functionality

Public key based user authentication has been implemented in **TEAMDEC** to prevent unauthorized access. The obvious advantage of this scheme is that it is more convenient than secret key cryptography, because it is not necessary for two parties to authenticate each other by sharing a secret key. In addition, one of the most important aspects of public key cryptography is that it enables digital signatures, which are used for user authentication.

However, the disadvantage of the public key based method is the performance penalty incurred. As noted in Chapter 4, to use this method, public/private keys must be generated, the private key needs to be encrypted and later decrypted, and the signature must be generated and verified. The cost of this method is affected by the key size. Table 5.1 shows the time needed for the whole authentication procedure. These results are tested on a Pentium II 450MHz system with 128M memory running the Windows 98 operating system. The time required to generate DSA Key Pair is obtained by running the system administration program in which a pair of DSA keys are generated for each user. The time required for client signing is observed when a client logs into the system. It includes the following periods: the time to generate a timestamp, the time to generate a random number, and the time to generate the signature using his private key. The time required for server verifying is examined at the server side when the server receives the information and verifies the signature.

Key Size (# of bits)	Generating DSA Key Pair	Client Signing	Server Verifying
	Time (*ms*)		
512	33400	33120	33740
768	33450	33170	33890
1024	33560	35700	36410

Table 5.1 Testing Results

Public key algorithms are based on the difficulty of factoring large numbers that are the product of two large primes[16]. Factoring large numbers is difficult, but with the advances in computer technology and mathematics, factoring large numbers is requiring less time than in the past. Table 5.2 lists some results from[16].

# of bits	Mips-years required to factor
512	30,000
768	$2*10^{8}$
1024	$3*10^{11}$
1280	$1*10^{14}$
1536	$3*10^{16}$
2048	$3*10^{20}$

Table 5.2 Factoring Using the General Number Field Sieve

Mips-year: a one-**m**illion-**i**nstruction-**p**er-**s**econd (mips) computer running for one year.

The table above shows that when the size of the key only increases once or twice, the time to factor increases dramatically. Thus, the length of the key plays an important role in the security of public key based authentication method. It is very difficult to determine which key size is appropriate. The shorter the key size, the likelier it is to be factored and attacked. The longer the key size, the more difficult it is to attack, but the greater the computation time needed to generate key pairs and to perform signing and verification.

From the testing results in Table 5.2, large key length doesn’t impose excessive overheads on computing time. When the key length increases from 512 bits to 1024 bits, the key generating time, client signing time, and server verifying time only increased 0.48%, 7.8% and 7.9%, respectively. But the security benefit is very impressive: the Mips-years required to factor increased from 30,000 to $3*10^{11}$.

5.1.2 The Difficulty of Attack

Another issue in evaluating an algorithm is the algorithm's ability to resist different types of attacks. The figure below shows the entire authentication procedure.

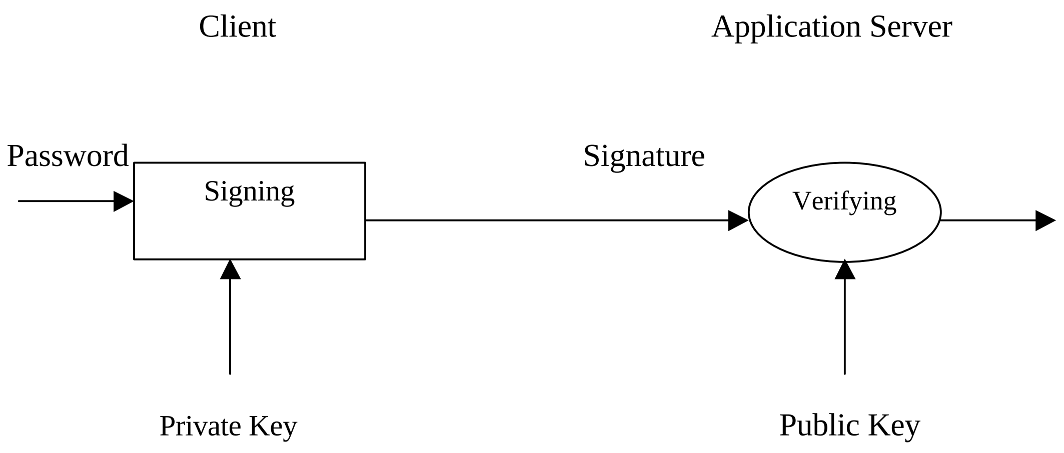

Figure 5.1 Operation of the Authentication Algorithm

Consider the following types of possible attacks.

First the attacker gets only the user's password. Suppose the user's password is stolen or guessed, can the attacker break into **TEAMDEC**? This possibility does not exist, because the authentication procedure depends not only on the user's password, but also on the user's private key. Without the private key, all the following authentication procedures can not be performed. This is the obvious advantage of the public key based authentication method over the traditional password-based authentication method.

Second the attacker gets the user's private key. If the attacker gets the user's private key, can the attacker break into the system? At this time, the possibility of breaking into the system is greater than in the previous attack. Because the attacker has stolen the user's private key, he may try different methods to get the user's password, and then successfully break into the system. Usually people choose easy-to-remember passwords. For example, some people choose their birthdays or other relevant personal information as passwords. Sometimes people write down their passwords and keep this record on a little piece of paper in their wallets. All these habits give the attacker opportunities to attack the system. There is a type of attack called "dictionary attack"[16] because the attacker uses a dictionary of common keys. By using this system, 40 percent of the passwords on the average computer can be cracked[16] . Hence, how to store and protect the users' private keys becomes one of the most important issues in the **TEAMDEC** to improve system security.

As seen in Chapter 2, the physical and electrical properties of the smart card make it an ideal medium for safely storing private keys. If the user's private key is on a smart card, and the private key never leaves the smart card, then the system would never be in a situation where it is easily compromised. With the smart card the private key is portable and can be used in different places no matter whether the user is at work, or at home. In addition, storing the private key in a smart card is more secure. For example, if the private key is stored in a floppy disk and protected by a password, then the private key file can probably be attacked by a "dictionary attack"[16]. But if the private key is stored in a smart card, the card will normally lock itself up after several consecutive bad password attempts. Thus "dictionary attack" is no longer a feasible way to get the private key. However, any security system, including smart cards, is breakable. There always exists a trade-off between cost and performance. Choosing a smart card to protect a private key depends on the cost of the equipment and the security level to be achieved.

Third, the attacker gets the user's public key. If this situation happens, the attacker has the user's public key, but it does not do him any good. Based on the theory of public key cryptography[4][15][16], it is computationally hard to deduce the private key from the public key, because public key cryptography relies on one-way functions; functions which are easy to calculate but hard to reverse without prior knowledge. Factorization is one example of a one-way function. It is often difficult to factor large numbers, but easy to verify a factorization. For example, it is harder to factor 6231 than to verify that 67*93 = 6231. With the development in computer technology and networks, no one can predict advances in mathematical theory. The table below shows the number of decimal digits factored [16] in different years.

Year	# of decimal digits factored	How many times harder to factor a 512-bit number
1983	71	> 20 million
1985	80	> 2 million
1988	90	250,000
1989	100	30,000
1993	120	500
1994	129	100

Table 5.3 Factoring Using the Quadratic Sieve

This is a very fast-developing field. Recently, RSA Data Security, Inc. published the following information[27]:

"On February 2, 1999, a group of researchers completed the factorization of the 140 digit RSA Challenge Number. The work was accomplished with the General Number Field Sieve.

Sieving was performed on about 125 SGI and Sun workstations running at 175 MHz on average, and on about 60 PCs running at 300 MHz on average. The elapsed time was one month and the total amount of CPU-time spent on sieving was 8.9 CPU years."

With the advances in mathematical theory and computer technology, computational algorithms, once considered infeasible to break, may be breakable in the future.

5.2 Secure Communication

Secure communications between two end points of an unsafe communication channel are achieved through the Diffie-Hellman key agreement protocol[4] which generates a session key first, and then encrypts the remainder of the conversation using that session key.

Although using the Diffie-Hellman key agreement protocol[4] is efficient for key agreement and eliminates the need for prior communication between users, the possibility of attacks still exists.

If an eavesdropper sits between the two parties and is able to hear the entire key exchange procedure, will he be able to know the value of the secret key? This question involves the security of the Diffie-Hellman algorithm[4]. During the entire key exchange, the two parties exchange their DH public keys. From the previous discussion, we know it is computationally hard to deduce the private key from the public key. If the time needed to figure out the private key from the public key is too long, the session maybe already be finished. Thus, the attacker will not succeed.

If an eavesdropper sits between the two parties and is able to get the encrypted message, will it be hard for him to recover the session key? This question involves the security of the DES algorithm. DES stands for Data Encryption Standard, which is described in NIST FIPS 46-2 [26]. According to the document, DES has a 64-bit block size and uses a 56-bit key during execution (8 parity bits are stripped off from the full 64-bit key). When using DES, there are several practical considerations that can affect the security of the encrypted data.

In our application, a different DES key is generated for each session, and a secure key agreement is provided by the Diffie-Hellman algorithm[4] that will improve the security of DES. Despite the efforts of researchers over many years, no easy attack on DES has been discovered. There is a fact stated in[27]:

> "Most recently, Distributed.Net, a worldwide coalition of computer enthusiasts, worked with the Electronic Frontier Foundation's (EFF) "Deep Crack," a specially designed supercomputer, and a worldwide network of nearly 100,000 PCs on the Internet, to win RSA Data Security's DES Challenge III in a record-breaking 22 hours and 15 minutes, beating the previous record of 56 hours".

Clearly, for a session this kind of attack is normally impractical. But if a session is recorded, the break in opportunity now exists because the attacker could decrypt the key in the future and thus get the secret messages between the two parties.

5.3 Database Server Issues

The database server in **TEAMDEC** system plays an important role, since the primary goal of **TEAMDEC** is to provide better decisions based on the past actions and scripts stored in the database. Ideally, database security should enable the system to have confidence that its data repository will provide the information requested and expected, while denying accessibility to those who have no right to it. However, issues exist relating to the database server, in particular, speed and security assurance.

The database server used in **TEAMDEC** is Mini SQL[28]. Mini SQL is a lightweight database server for both large and small data sets. In addition to the basic functions it provides for manipulating data, the Mini SQL server also provides functions to control unauthorized access through an access control list. It does not support encrypted connections, however, and thus is still vulnerable to security breaches. Unlike some commercial database products, the Mini SQL server does not support powerful and flexible security mechanisms to provide assurance. To prevent possible attacks on the database server, and to improve confidentiality, integrity, and availability, it is better to replace the current Mini SQL server with commercial products such as Trusted Oracle 7[29] from Oracle Corporation in the future.

5.4 Summary

This chapter gives an evaluation of the security implementation. The evaluation procedure is based on two factors: functionality and the ability to prevent different types of attacks. In addition, the advantages and disadvantages of the security algorithms are evaluated and possible improvements are suggested.

Chapter 6 Conclusion and Contributions

With the growing popularity of the Internet, network security is a foremost concern for many applications. With proprietary technology, a network security solution can secure the application running on an unsafe medium, such as the Internet.

6.1 Conclusion

The software architecture proposed in this thesis enables the secure and efficient operation of **TEAMDEC:** a team-based decision support system. Based on the system's requirements and architecture, three types of possible attacks to the system are identified, and a security solution is proposed that includes user authentication, secure communication, and script access control. The implementation of these features reduces security risk, improves the overall system security, and effectively uses the valuable system information data. Several advantages are provided by this architecture.

First, **TEAMDEC** demands a higher level of security than a simple distributed application. To guarantee that only an authorized user can access **TEAMDEC**, a public key based user authorization scheme is employed. The scheme can ensure that the remote user is, in fact, an authorized system user. In addition, with a digital signature, the scheme provides two security services, authentication and integrity.

Second, the Diffie-Hellman key agreement protocol[4] is employed to achieve secure communications among team members in such an open network environment. By exchanging keys and computing the shared secret key, the two parties can exchange the keys and messages over an unsafe communication channel.

Third, a generic script access model is defined to provide integrity and secrecy of scripts in the **TEAMDEC**. The model allows fine-grained control access to individual scripts and their attributes that is based on the individual's role in the system.

The implementation is accomplished by using existing Java security packages, specifically, JCA and JCE. By using these commercial Java security packages, it is fast and efficient to develop the security features of **TEAMDEC**.

The development of appropriate security measures for TEAMDEC necessitated the evaluation of security performance of the software. While the options adopted for TEAMDEC appear appropriate, it is important to note that computer technologies advance every day. As a result, algorithms or applications currently considered secure may not be secure in the future. The evaluation procedure is valuable and it can be used to maintain and improve the system in the future.

6.2 Contributions

The contributions of this research are twofold. First, the desired collaborative features of TEAMDEC have been implemented. These features include interactive, simultaneous communications among team members of TEAMDEC and real-time search and integration capability. The most important contribution is that the implementation of the **TEAMDEC** script system. The system is able to record and track the user's actions and provide the action suggestion scripts. The second area of contribution of this research and specifically of this thesis is that the security architecture is proposed and implemented. These security features enable the secure operation of TEAMDEC and thus improve the quality and efficiency of TEAMDEC's decision-making capability.

References

[1]. Krause, M., Tipton, H. F., *Handbook of Information Security Management*, CRC Press LLC, 2000 Corporate Blvd., N.W., Boca Raton, Florida 1999.

[2]. Qian Chen, *TEAMDEC: A Group Decision Support System*, M.S. Thesis, VPI&SU, Blacksburg, July 1998.

[3]. Eloise Coupey and Mark T. Jones, *TEAMDEC: Integrative Decision Solutions With Multiple Information Sources*, Research Proposal, VPI&SU, Blacksburg.

[4]. W. Diffie and M.E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, v. IT-22, n. 6, Nov 1976, pp. 644-654.

[5]. Gregory B. White, Eric A. Fisch and Udo W. Pooch, *Computer System and Network Security*, CRC Press, 2000 Corporate Blvd., N.W., Boca Raton, Florida 1996.

[6]. R. Atkinson, *Security Architecture for the Internet Protocol*, RFC 1825, Network Working Group, August 1995.

[7]. Jonathan Knudsen, *Java Cryptography*, O'Reilly & Associates, Inc. 101 Morris Street, Sebastopol, CA 1998.

[8]. Kaufman, C., Perlman, R. and Speciner, M., *Network Security, Private Communication in a Public World*, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[9]. "Virus", online document at http://www.whatis.com/, May 1999.

[10]. Fites, P. and Kratz, M. P. J., *Information System Security: A Practitioner's Reference*, Van Nostrand Reinhold, New York, New York, 1993.

[11]. Wood, H.M., *The Use of Passwords for Controlled Access to Computer Resources*, National Bureau of Standards Special Publication 500-9, U.S Dept. of Commerce/NBS, 1977.

[12]. "PKCS#5", online document at http://www.rsa.com/rsalabs/pubs/PKCS/html/pkcs-5.html, RSA Labs, March 1999.

[13]. Russell, D. and Gangemi Sr., G. T., *Computer Security Basics*, O'Reilly & Associates, 1991.

[14]. American National Standards Institute, *American National Standard ANSI/ISO 7810-1985: Identification cards - physical characteristics*, New York, New York, May 20, 1988.

[15]. Nichols, Randall K., *ICSA Guide to Cryptography*, McGraw-Hill, Gahanna Industrial Park, 860 Taylor Station Road, Blacklick, Ohio 1999.

[16]. Bruce Schneier, *Applied Cryptography, Second Edition, Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Inc, 605 Third Avenue, New York, 1996.

[17]. Alan O. Freier, Philip Karlton and Paul C. Kocher, "*The SSL Protocol Version 3.0*", Online document at http://home.netscape.com/eng/ssl3/draft302.txt, November 18, 1996.

[18]. R.L. Rivest, A. Shamir, and L.M. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM*, 21(2): 120-126, February 1978.

[19]. "*Glossary: chosen plaintext attacks*", published at http://www.rsa.com/rsalabs/faq/html/glossary.html, RSA Laboratories.

[20]. National Institute of Standards and Technology (NIST). FIPS Publication 186, *Digital Signature Standard (DSS)*, U.S. Department of Commerce, May 1994.

[21]. *The Java Tutorial*, *A Practical Guide for Programmers*, Published at http://java.sun.com/docs/books/tutorial/index.html, Sun Microsystems, May 28, 1999.

[22]. *Java^{TM} Cryptography Architecture API Specification & Reference*, Published at

http://java.sun.com/products/jdk/1.2/docs/guide/security/CryptoSpec.html#Introduction, Sun Microsystems, October 30, 1998.

[23]. *JavaTM Cryptography Extension 1.2 API Specification & Reference*, Sun Microsystems, July 21, 1998.

[24]. George Coulouris, Jean Dollimore, and Marcus Roberts, *Role and Task-based Access Control in the PerDis Groupware Platform*, 3rd ACM Workshop on Role-Based Access Fairfax VA 1998 1-58113-113-5/98/10.

[25]. NIST FIPS 180-1 at http://www.nist.gov/itl/div897/pubs/fip180-1.htm.

[26]. NIST FIPS 46-2, *Data Encryption Standard*.

[27]. On-line document published at http://www.rsa.com/rsalabs/, RSA Laboratories, Inc.

[28]. Brian Jepson and David J. Hughes, *Official Guide to MiniSQL 2.0*, Wiley Computer Publishing, 605 Third Avenue, New York, NY 10158-0012, 1998.

[29]. On-line document published at http://www.oracle.com/database/trusted/html/chapter1.html, Oracle Corporation.

[30]. On-line document at http://www.microsoft.com/MSJ/0599/security/security0599top.htm, Microsoft System Journal, May 1999.

[31]. On-line document at http://www.novell.com/documentation/lg/nw5/docui/index.html, Novell, Inc

[32]. OpenVMS documentation published at http://www.openvms.digital.com:8000/72final/5929/5929pro.html, Digital Equipment Corporation.

[33]. R. Rivest, *The MD5 Message-Digest Algorithm*, RFC 1321, Network Working Group, April 1992

[34]. Hyun Park, Randy Chow, *Internetwork Access Control Using Public Key Certificates, Information Systems Security, Facing the information society of the 21st century*, Chapman & Hall, 2-6 Boundary Row, London SE1 8HN, UK. 1996, pp237-246

[35]. Johab S von Solms, Martin S Olivier and Sebastiaan H von Solms, *MoFAC: A Model for Fine-grained Access Control, Information Systems Security, Facing the information society of the 21st century*, Chapman & Hall, 2-6 Boundary Row, London SE1 8HN, UK. 1996, pp295-305

[36]. Online published at http://www.zdwebopedia.com/Programming/three_tier.html, ZD Webopaedia, 1999

Vita

Haiyuan Wang received her B.S. and M.S. in Department of Electrical Engineering from Xi'an Jiaotong University, China in 1992 and 1995. From 1995 to 1996, she worked as an Engineer at the No. 41 Institute of China Aerospace Ministry.

She joined the Virginia Tech Information System Center (VISC), Virginia Tech in January of 1998 as a graduate research assistant. While doing research in the VISC lab, she became interested in network-based application design and development. After graduating from Virginia Tech, she will work as a Software Engineer to develop and design Java-based applications.