Improving the Security Services of the Message Handling Systems in X.400

by

Jingsong Liu

A M.Sc. Thesis

submitted to the School of Graduate Studies and Research
in partial fulfillment of the requirements for the
Master of Computer Science Degree*

University of Ottawa
Ottawa, Ontario
Canada

*The Master of Computer Science Program is a joint program with Carleton University, administrated by the Ottawa-Carleton Institute for Computer Science

ISBN 0-612-00481-3

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

To my parents
Tingwen Liu and Qifan Zhou

ACKNOWLEDGMENT

I am very grateful to my two supervisors, Dr. George White and Dr. To-yat Cheung, for their valuable time, patience and for their guidance and advice throughout my graduate studies. Their instructions for revising the drafts of the thesis has greatly improved its contents and presentation. The discussions we had are of great influence and assistance.

I would like to thank Prof. Lingyuan Ge for his kindness and assistance. I also owe thanks to Dr. Paul Van Oorschot for answering my questions and the encouraging discussions.

I acknowledge with gratitude the financial support provided by Telecommunications Research Institute of Ontario and the Natural Sciences and Engineering Research Council of Canada.

ABSTRACT

This thesis studies some security problems in the X.400 Message Handling System (MHS). It focuses on the so-called message-token, a cryptological data structure, used to convey the security-related information and support several basic security services in X.400 MHS. The current X.400 message-token which conforms to the X.509 one-way authentication protocol cannot protect X.400 messages from the attack of impersonation. An existing solution in terms of the one-way authentication protocol still cannot solve this problem when it is applied as a message-token in MHS. This thesis develops three new message-tokens for solving the problem. Correctness of the new message-tokens is proved by means of the authentication logic, a formal analysis technique for checking if authentication protocols are able to achieve their goals. Through the study, some weaknesses have been found in the postulates known as the Hash-Function Rules. New Hash-Function Rules are proposed for improving the authentication logic. The newly proposed message-tokens are proved to be capable of ensuring the integrity and originality of the conveyed information and thus prevent the impersonation threat for X.400 messages.

Contents

List of Figures

Chapter 1

Introduction

1.1 Security in Message Handling Systems

Message Handling System (MHS) is a special service in the application layer of the OSI General Reference Model. It was originally designed for exchanging interpersonal messages, i.e., electronic mail [CCITT420]. Recently, this technology is increasingly being used for some serious commercial applications [GEN90] [CCITT435], such as electronic data interchange and electronic fund transfer. This leads to more and more concern about the security of MHS.

Wherever information is transmitted the communication partners expect it to arrive unaltered. Security features are important prerequisites for the adoption of an MHS technology. They include guaranteed integrity of the transmitted messages, verification of the originator, verification of submission and receipt, confidentiality of the transmitted messages and protection against loss or duplication of messages.

The OSI Security Architecture Draft International Standard 7498/2 [ISO7498-2] is a basis for building a secure open network system. In this standard, the term *security*

is defined as *"a means for minimizing the vulnerabilities of assets and resources"*, and security is achieved by providing *security services*. This standard specifies a set of security services with which a communication system structured according to OSI7498/2 may be made secure. Some of the important security services at the application layer are listed below:

- *Origin Authentication* provides the corroboration that a communicating peer entity or the source of data is really the one claimed.
- *Access control* provides protection against unauthorized access to data.
- *Data confidentiality* protects data from unauthorized disclosure.
- *Data integrity* protects data from unauthorized modification.
- *Non-repudiation* provides the proof of the origin of data in order to prevent any attempts by the sender to falsely deny having sent the data.

In most of the literature, the terminology *Message Authentication* (or simply *Authentication*) means to ensure the integrity and origin authenticity of a transferred message, i.e. both integrity service and origin authentication service are provided together. In fact, from the recipient's point of view, it does not make much sense to provide one service without the other.

X.400 provides numerous security services for defence against various threats, described in detail in Section 2.2. These security services are specified in recommendations X.400 (Overview) [CCITT400], X.402 (Architecture) [CCITT402] and X.411 (MTS) [CCITT411]. Essentially, X.400 provides two classes of security services: those which protect connections between systems and those which protect individual messages. The end users are mainly interested in the security services at the message level. More details of X.400 security features will be described in Section 2.4.

1.2 Motivation

Through many years of development, the security model of X.400 MHS has reached a reasonable level of stability. However, it still needs improvement in some aspects.

Protection of security-related information

When exchanging a message in a distributed environment, the security related information (e.g., encryption keys, cryptological checksums, etc.) also has to be transferred and protected from unauthorized disclosure and modification. In X.400 MHS, such information is conveyed and protected by a cryptological data structure called the *message-token*. A message-token has three portions: the confidential data field, the non-confidential data field and the digital signature of the sender. The confidential data are encrypted. The signature is derived from a combination of the confidential data and the non-confidential data.

The structure of message-tokens influences the security of transferred messages. In the current X.400 message-token, the signature is derived on a token which contains encrypted confidential data (i.e., the signature is the function of the combination of non-confidential data and encrypted confidential data). It conforms to the X.509 one-way authentication protocol [CCITT509] (also see Section 2.3.4). As noted in Burrows, Abadi and Needham [BAN90], it is unable to defend against the impersonation threat. When such a message-token is employed in MHS, a malicious third party is able to modify the message-token and thereby deceive the recipient into believing that the third party is the source of the message.

[BAN90] also mentions that a possible solution is to perform the signature and encryption operations in reverse order. C. I'Anson and C. J. Mitchell in [IAM90]

give a new form of the one-way authentication protocol, which refines the idea of [BAN90] and practically employs a one-way function for calculating the signature. However, we find that applying this solution as the message-token cannot solve the above impersonation problem in MHS. Therefore, other solutions are needed to solve the problem.

Formal analysis of message-tokens

Message-tokens are considered as the application of one-way authentication protocols. The purposes of using message-tokens are to ensure the integrity and originality of the transmitted security-related information. To ensure that these purposes are achieved, the correctness of message-tokens needs to be proved. Burrows, *et al.* proposed the logic of authentication for the formal analysis of the authentication protocols. This logic in particular has several important virtues. First, it helps formalize reasoning about useful abstract properties of the protocol; second, it forces designers to make explicit security assumptions and a well-defined set of authentication goals; and third, it helps to verify whether those goals can be achieved at the end of protocol run.

However, when we try to use the logic to analyze the authentication protocols, we find that there are some mistakes in the postulates, the Hash-Function Rules. They inaccurately present the implicit meaning of using one-way hash functions in the signature technique. Therefore, the Hash-Function Rules should be redefined.

1.3 Summary of Contributions

In this thesis, we make suggestions for improving X.400 MHS security on the structure of message-tokens. To ensure their correctness, a formal technique, the authentication logic, is used to analyze the new message-tokens. Here, we also make a contribution to the authentication logic by redefining the postulate called the Hash Function Rules. In summary, the main contributions of this thesis are listed below:

(a) It is found that, when applying the solution proposed by [IAM90] as a message-token, the impersonation problem cannot be solved in MHS (Chapter 3).

(b) Three new message-tokens, Message-Tokens 3, 4 and 5, are proposed to protect messages from the threat of impersonation in MHS (Chapter 3).

(c) The proof of correctness for the new message-tokens is provided, based on the logic of authentication created by Burrows *et al.* [BAN90]. The proof shows that these new message-tokens can achieve the expected authentication goals (Chapter 4).

(d) Through applying the authentication logic, we find some weaknesses in the postulates (Hash-Function Rules). New Hash-Function Rules are proposed for more accurately describing the implicit meaning of using one-way hash functions in the digital signature technique (Chapter 4).

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows:

- Chapter 2 is an introduction to the basic concepts of X.400 MHS, the possible threats to MHS, and security mechanisms used by X.400 security service. It also gives an outline of the X.400 security services.
- Chapter 3 discusses the problems of the current X.400 message-token and the problem in the solution of [IAM90], and proposes three new message-tokens to solve the problem.
- Chapter 4 introduces the basic concepts in the logic of authentication, improves the Hashing-Function Rules, and analyzes the old message-token schemes and proves the correctness of the new message-tokens.
- Chapter 5 contains the conclusion and possible future work.

Chapter 2

Background

Before presenting a detailed description of the security of Message Handling System (MHS), some background knowledge is given in this chapter. First, the functional model of X.400 MHS is briefly introduced in Section 2.1. Section 2.2 explains the security threats in MHS, and Section 2.3 briefly introduces the security mechanisms. Finally, in Section 2.4, the security services in X.400 MHS are outlined.

2.1 The Functional Model of X.400 MHS

The X.400 MHS enables MHS-users to exchange messages on a store-and-forward basis. The major components are shown in Figure 2.1:

- User agent (UA)
- Message transfer agent (MTA)
- Message transfer system (MTS)
- Message store (MS)

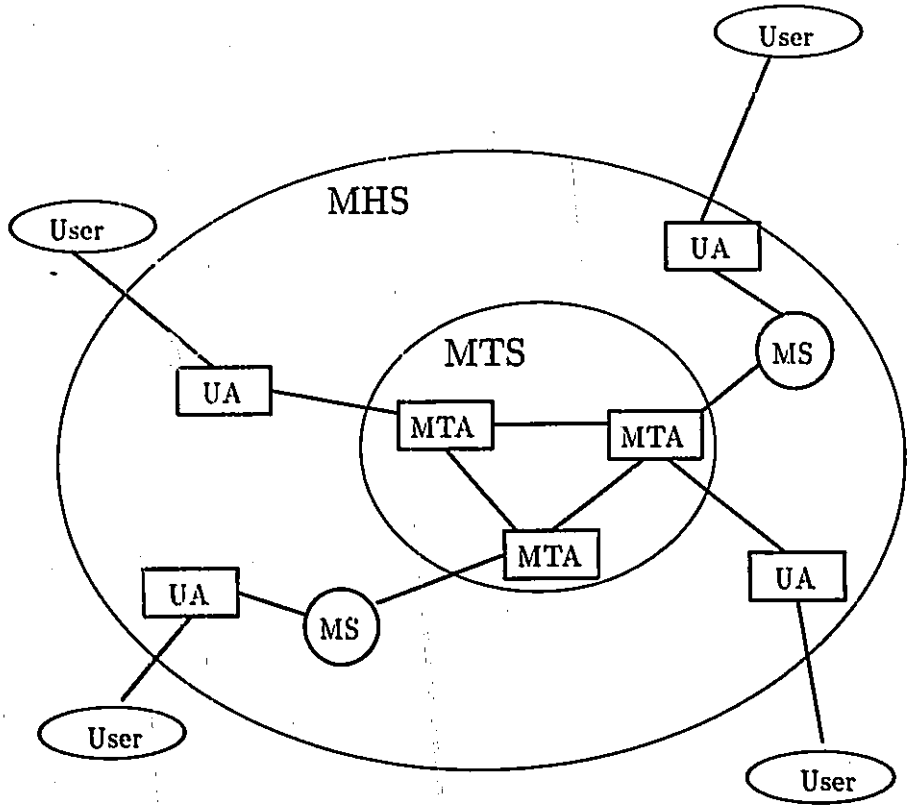


Figure 2.1: Components of Message Handling System

The *user agent* (UA) is responsible for directly interfacing with the MHS-users. It prepares, submits and receives messages for them. Besides text editing and presentation services, it also provides several other services, such as security, priority provision, delivery notification, and document distribution.

The *message transfer system* (MTS) provides the distributed store-and-forward message transfer service and thus has the task of transferring messages of all types from the sender UA to the recipient UA. The MTS neither examines nor modifies the content part of a message, except for the encode conversion requested by the

originator. An important point is that, within MHS, the MTS is considered as application independent. The *message transfer agents* (MTAs) collectively provide the MTS service. MTA provides the routing and relaying of messages. Its functions are responsible for the store-and-forward path, channel security, and the actual message routing through the communication media. An MTA is also responsible for the conversion of the encoded information types (e.g. ASCII code, Group 3 FAX). A message originator may request a particular conversion to be performed on the message by the originator MTA before it is delivered. The purpose of conversion is to improve the communication possibilities among MHS users who use terminals of different capabilities.

The *message store* (MS) provides the storage of messages and supports their submission and retrieval. It supplements the UA when the users' terminals are not available.

Transmission of a message through MHS needs several steps (Figure 2.2). When a user wishes to send a message to its peer, it makes the message available to its UA. This step is called *message origination*. Subsequently, the UA submits the message to a MTA within this MTS. It is the responsibility of the MTS to move the message from one MTA to the next, depending upon the selected route. Finally, the MTS delivers the message to a UA.

An X.400 message includes two parts: the *content* and the *envelope*. The message *content* is the information that the originating UA wishes to be delivered to other UAs. It is conveyed transparently by the MTS. The *envelope* is composed of several pieces of information, including the name of the originating user and intended recipients, a summary of past actions concerning the message, routing information, and a

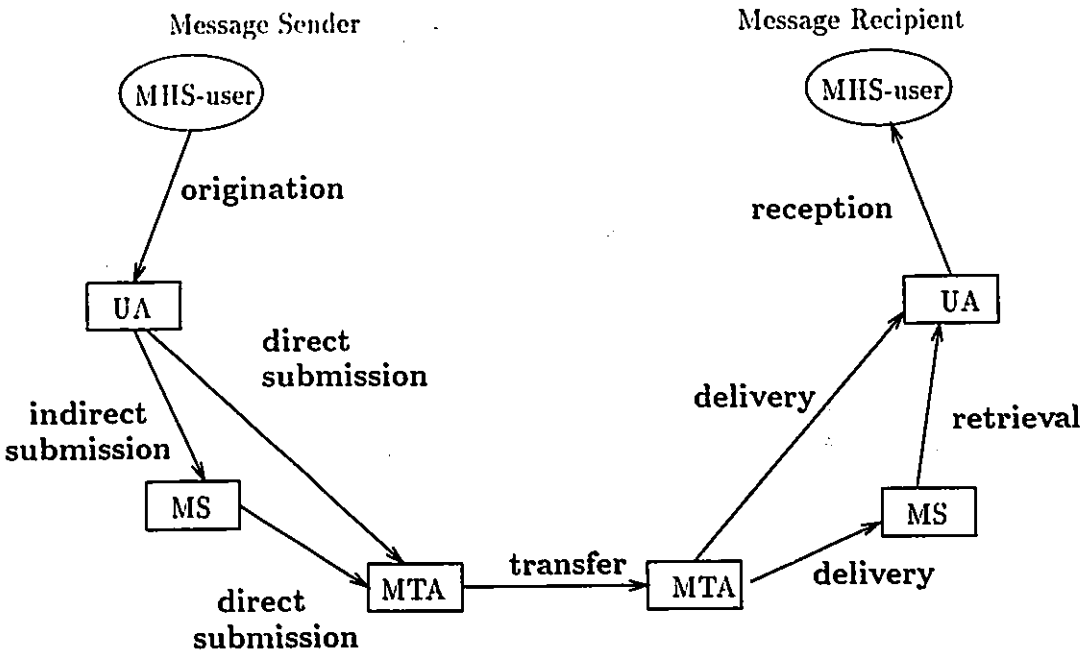


Figure 2.2: Transmission of a message

characterization of the contents. The MTAs deliver the message without modifying or examining their content.

2.2 Threats in MHS

An overview of the security threats in MHS is given in Recommendation X.402. They can be classified into the following forms:

- *Masquerade*
- *Modification of messages*
- *Leakage of information*
- *Message de-sequencing*

- *Repudiation.*

Masquerade

Masquerade occurs when a communicating entity successfully pretends to be an entity with a different identity. This can take place in a number of ways. For example, an unauthorized MTS-user may impersonate another to gain unauthorized access to MTS facilities or to discard its messages. In other situations, an MTS-user may impersonate another user or falsely acknowledge receipt of a message.

In later chapters, we will study the methods of preventing the so-called impersonation attack, in which an intruder falsely claims himself to be the source of a message. This particular impersonation attack belongs to the masquerade category.

Modification of Information

Message modification may occur to any aspect of a message, such as its contents, attributes, recipient's or originator's identity. Corruption of routing or other management information, stored in the MTA or used by the MTA, may cause the MTS to lose messages or to operate incorrectly.

Leakage of Information

Leakage of information occurs when information is disclosed to intruders. It may lead to loss of confidentiality, loss of anonymity, misappropriation of messages, and traffic analysis.

Message De-sequencing

Message de-sequencing occurs when part or the entire message is repeated, time-shifted, reordered or delayed.

Repudiation

This kind of threat occurs when an actual communication participant denies having originated, submitted or received a message.

2.3 Security Mechanisms

This section attempts to carry out a concise survey of the security mechanisms that may be used in MHS security on the basis of the existing security mechanisms for networks. For details, refer to [ISO7498-2] and [VOYS3].

2.3.1 Encipherment Techniques

Encipherment is considered the most important mechanism for the security of communication data. Encipherment techniques can be used to provide data confidentiality services or, in combination with other techniques, to provide such services as data origin authentication and integrity .

Encipherment includes two complementary processes. *Encryption* takes the plaintext P of a message as its input and applies an encryption function E on P to produce the ciphertext C. *Decryption* applies a decryption function D on C to recover the original plaintext of the message. These two processes can be expressed as follows:

$$\textbf{Encryption}: \quad C = E(K_e, P),$$

Decryption : $P = D(K_d, C)$.

where K_e is the encryption key held by the message originator, and K_d is the decryption key held by the message recipient. The algorithms D and E are publicly available, so the security mechanism rests solely on the secrecy of the two keys.

There are two general classes of encipherment algorithms: symmetric encipherment (i.e. conventional cipher) and asymmetric encipherment.

In symmetric encipherment, the key used to decipher a message is the same as that used to encipher it, i.e. $K_e = K_d$. Such a key must be kept secret. The Data Encryption Standards (DES) is one of the conventional ciphers, and was adopted by the U.S. National Bureau of Standards in 1977 [DES77].

In asymmetric encipherment, a pair of distinct keys are used: a public key K for encryption and a private key K^{-1} for decryption. Of course, it must not be practically possible to derive K^{-1} from K. Anyone can send an encrypted message to the holder of K^{-1}, by employing the holder's K. The best known public encryption system is called RSA [RSA78].

2.3.2 Integrity

The message integrity mechanisms allow recipients to determine whether or not the received messages have been modified by any interceptor during its transmission.

To achieve the message integrity, the sender of a message computes a checksum, called the *integrity check* to be appended to the message. The integrity check is a function of the message itself and may be derived by means of either symmetric or asymmetric encryption. Figure 2.3 shows the principle of the integrity checking.

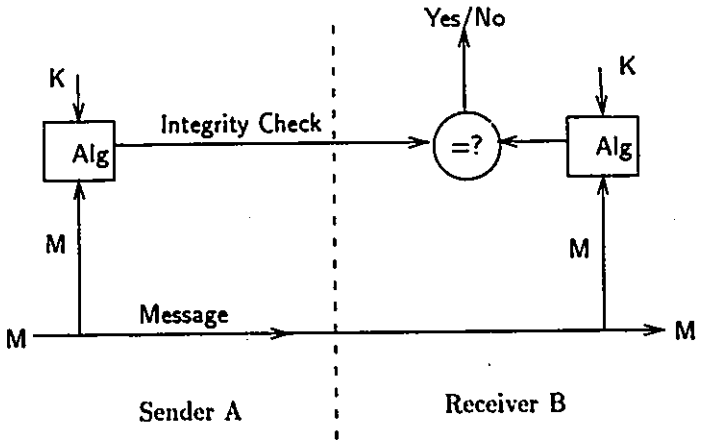


Figure 2.3: Principle of integrity checking

Using the encryption scheme employed by the sender, the recipient generates another checksum from the received message. By comparing this checksum with the received integrity check, any modification to the message can be detected.

2.3.3 Digital Signature

The concept of *digital signature* was first discussed by Diffie and Hellman in their classic paper "New Directions in Cryptography" [DIF76]. In general terms, a digital signature is a digital signal, usually represented as a string of bits. It may be appended to a message or the message may form an integral part of it. Digital signature is for the recipient to authenticate the message, i.e. to verify the integrity and origin authentication of the message. Furthermore, it can be used by a third party to solve a dispute between a sender and a receiver about the origin of a message. In the dispute, the receiver could forge a message and claim that the message come from the sender. Conversely the sender can deny an authenticated message and claim that the receiver produced a forged message.

A digital signature should have the following properties [SIM92]:

(1) It is generated from the plaintext of the message based on the sender's secret information (e.g., the private key).

(2) It can be easily verified by the receiver.

(3) It is computationally difficult to forge.

There are two schemes of digital signature: (1) a completely encrypted message; and (2) a message dependent data unit appended to the message.

The encrypted message as the signature

In this signature scheme, a sender signs a message by enciphering it with the sender's private key. The intended receiver obtains the original message by deciphering the signed message with the sender's public key. The receiver can be sure that the message is originated from the purported sender since only the sender has access to this particular private key. In this case, the signature is the encrypted message which can be transformed back by the receiver into readable form. This signature scheme is the basic concept of digital signature.

Signature appended to the message

In practice, the signature scheme most frequently used is a separate data unit appended to the message. X.400 MHS employs this signature scheme. The principle of generating a separate signature is shown in Figure 2.4. Sender A is transmitting a message M to receiver B. In this case, the message can be sent in the plaintext form. To generate the signature for a message, a one-way hash function $h(M)$ is formed using all the bits of the message and then this hash value is encrypted under the sender's secret key K_A^{-1}. At the receiving end, the signature is decrypted using

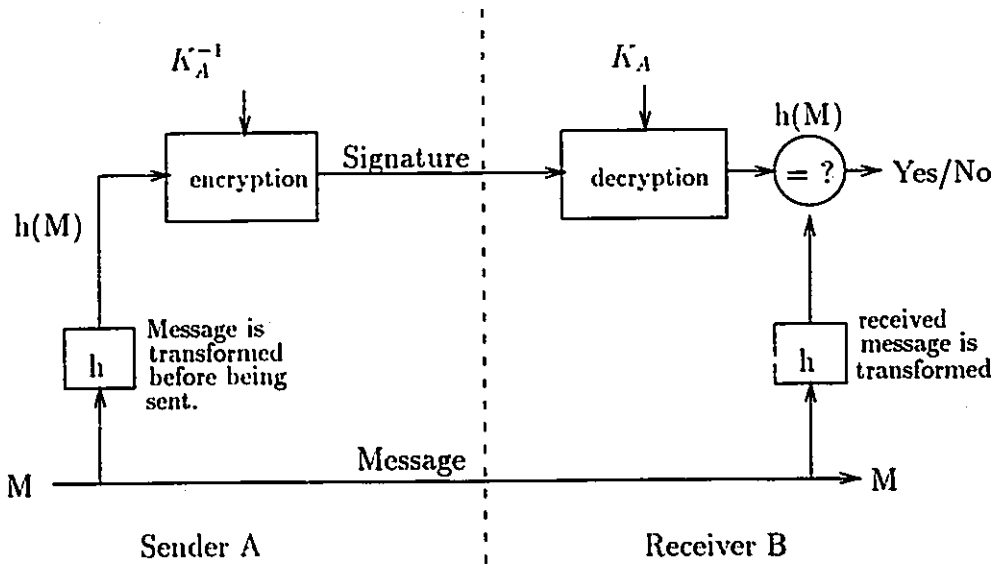


Figure 2.4: The signature appended to the message

the sender's public key K_A. This should return the value of $h(M)$. The receiver applies the one-way hash function h to the plaintext message he has received. The result should also be the value $h(M)$. If these two values are equal, the signature is pronounced valid and the message is authenticated.

A one-way hash function h employs no key and is public. It satisfies three main properties [SIM92]:

- **H1.** It must be capable of converting a message M of arbitrary length into a fixed-length "digest", $h(M)$, which has the length required to be input to the secret signature transformation.
- **H2.** It must be *one-way*. That is, given an arbitrary value y in the domain of h, it must be computationally infeasible to find a message M such that $h(M) = y$.

- **H3.** It must be *collision-free*, that is, it must be computationally infeasible to construct two messages M, M' with the property that $h(M) = h(M')$.

These three properties are necessary. The need for $H1$ is clear. $H2$ is present to prevent the interceptor of a message and its signature replacing the valid message M with a fraudulent one M' with the property that $h(M) = h(M')$. $H3$ is to prevent the event that a malicious party could persuade a sender to sign message M and then attach this signature to another message M' such that $h(M) = h(M')$. It can be observed that $H3$ implies $H2$.

2.3.4 Authentication Exchange Mechanisms

Authentication exchange mechanisms allow each communicating party to be sure of the other's identity. They are divided into two classes:

- *Weak authentication*, where a password without encryption is used, and
- *Strong authentication*, where cryptographic techniques are used.

Weak authentication is only for *entity authentication* which ensures the entity is the claimed entity. Strong authentication supports the *message authentication* in addition to entity authentication. It provides the protection of the exchanged information (e.g. password, secret key). X.509 Recommendation [CCITT509] gives one-way, two-way and three-way handshake authentication protocols for reliable mutual authentication. We only describe the X.509 one-way authentication protocol here.

One-way authentication involves a single transfer of information from one user (A) to another (B), and its purpose is to establish the following:

- A is the originator of the information.

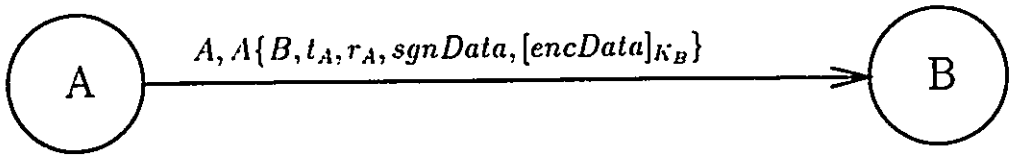


Figure 2.5: The X.509 one-way authentication protocol

- B is the intended receiver of the information.
- The integrity and originality of the information are maintained during the transmission.

The X.509 one-way authentication protocol involves the following steps, as represented by Figure 2.5:

1. A generates the token :

$$A, A\{B, t_A, r_A, sgnData, [encData]_{K_B}\}$$

where, t_A is the time-stamp; r_A is a non-repeating number; and *sgnData* is non-confidential information. $[encData]_{K_B}$ denotes that the confidential information, *encData*, is encrypted by the receiver's public key K_B. $A\{B, t_A, r_A, sgnData, [encData]_{K_B}\}$ means that A signs $(B, t_A, r_A, sgnData, [encData]_{K_B})$.

2. A sends the token to B.

3. B carries out the following actions:

   (a) checks that B itself is the intended recipient;

   (b) checks that the time-stamp t_A is current;

   (c) obtains A's public key;

(d) verifies the signature, and thus the integrity of the signed information;

(e) optionally, checks that the message token has not been replayed.

The X.509 one-way authentication protocol is employed by X.400 as message-tokens to convey the security-related information. We will have more discussion about the message-token in the following chapters.

2.4 Overview of the Security Services in X.400 MHS

Figure 2.6 shows the security services offered by X.400 MHS. Each security service is conceived as an elementary component of the entire system's security. Details of these security services are provided in the X.402 and X.411 Recommendations. More discussions on the X.400 security services and their implementations can be found in the literature [KIN92] [MRW89] [TC90] [WU92]. In the rest of this section, we briefly explain the security services which will be the subjects of later chapters of the thesis. They are Content Confidentiality Service, Content Integrity Service, and Message Origin Authentication Service.

2.4.1 Provision of Content Confidentiality Service

The **content confidentiality service** aims to insure that the message content is not disclosed to intruders. This service is an end-to-end service between an originator UA and a recipient UA.

Assume that the originator UA is A and the recipient UA is B. First, if using the symmetric type of encryption algorithm, the key used for the encryption process

Service	Threats to be countered	Mechanisms to be used
Origin authentication service group Message origin authentication Probe origin authentication Report origin authentication Proof of submission Proof of delivery	Masquerade	Digital signature
Secure access management service Peer entity authentication	Masquerade	Authentication exchange
Data confidentiality service group Content confidentiality Message flow confidentiality	Leakage of information	Encipherment
Data integrity service group Content integrity Message sequence integrity	Modification of information Message de-sequencing	Message integrity
Non-repudiation service group Non-repudiation of origin Non-repudiation of submission Non-repudiation of delivery	Repudiation	Digital signature + trusted party
Message security labelling	Differing labeling policies	

Figure 2.6: Basic MHS security services

is selected at random by A every time a confidential message is sent. This key is called Content Confidential Key (CCK). A encrypts the message content using the CCK and then sends the encrypted message together with the CCK and the content-confidentiality-algorithm-identifier to B. Upon receipt of the message, B uses the CCK to decrypt the encrypted message. The CCK must be sent via a secure channel from A to B, so that the message content can be known to A and B only.

In the case of an asymmetric cryptosystem being used, A encrypts the message content using receiver B's public key, and A sends the encrypted message and the content-confidentiality-algorithm-identifier to B. B then uses its secret key to decrypt the encrypted message. Since B is the only holder of its secret key, the message content can be known to A and B only.

2.4.2 Provision of Content Integrity Service

The **content integrity service** provides integrity for the message content. It uses a cryptographic checksum called Content Integrity Check (CIC). The CIC of a message is generated by the originator UA from the plaintext of the message content. The recipients will use the CIC to verify whether the message content has been modified during transmission, and to ensure that the originator cannot subsequently repudiate the message (see Section 2.3.2).

CIC can be derived by either a symmetric encryption algorithm or an asymmetric encryption algorithm. If symmetric encryption algorithms are employed, a Content Integrity Key (CIK) will also be sent to the recipient. CIC and CIK are protected by a message-token when they are sent to the recipients.

2.4.3 Provision of Message Origin Authentication Service

The **message origin authentication service** enables the corroboration of the source of an information object. In X.400 Recommendations, there are two methods to provide this service. The first method is using a cryptographic checksum called Message Origin Authentication Check (MOAC). It is basically the originator's signature over the message content. MOAC is a component of the message envelope.

The second way of providing the origin authentication service is using a message-token which contains a CIC for the message content. The signature on the message-token will ensure the originality of the message, since the message-token should be generated by the originator who generates the message content. This thesis is focused on this way of providing the message origin authentication service.

Chapter 3

X.400 Message-Tokens, Problems and Solutions

3.1 Introduction

In order to provide the security services discussed in the last chapter, some pieces of security-related information must also be transferred from the origination UA to the destination UA, such as the CCK for the Content Confidentiality Services (see Section 2.5.1), the CIC and the CIK for the Content Integrity Service (see Section 2.5.2). More security-related information is listed in Section 3.2.1 .

It is important that this information be protected during transmission. In X.400 MHS, the security mechanism used to provide this protection is called the **message-token**. Therefore, the message-token becomes the fundamental mechanism of providing most of the UA level security services.

A brief explanation about message-tokens has been given in Section 1.2. We will provide a more detailed description in this chapter (Section 3.2). It will be shown

that the current message-token structure in the X.400 Recommendations (hereafter called Message-Token 1) can cause a security problem. An intruder can successfully pretend to be the source of a message by modifying its message-token. (Section 3.3)

An existent solution proposed by [IAM90] uses a different structure, here called Message-Token 2. We will point out, in Section 3.4, that this solution cannot protect a message from the above impersonation attack when none of the components of the token characterizes the sender, because an intruder is still able to make undetectable modifications on the token to claim to be the originator.

As the main contribution of the thesis, three new message-tokens are proposed, named Message-Token 3, Message-Token 4 and Message-Token 5, to eliminate this threat (Section 3.5). All these new token structures are designed in a way such that any unauthorized modifications of the token can be detected by receivers. These new message-tokens can solve the impersonation problem of Message-Token 1.

3.2 Message-Token in X.400 Recommendations

3.2.1 Security-Related Information

Before discussing the message-tokens, we first describe the following security-related information which has to be transferred from originator UAs to recipient UAs for the purpose of providing the end-to-end security services [CCITT402] [CCITT411].

Content Confidentiality Key

Content Confidentiality Key (CCK) is used by the Content Confidentiality Service. It is used by the message originator to encrypt the message content, and by the

recipient to make the decryption. If a symmetric encryption algorithm is used, the CCK must be transferred to the recipient UAs, guarding its confidentiality, integrity and authenticity.

Content Integrity Check

Content Integrity Check (CIC) is used in the Content Integrity Service, and is calculated by the originator UA for the recipient UA to verify the message integrity. CIC must not be changed during the transmission.

Content Integrity Key

Content Integrity Key (CIK) is used to calculate the CIC. The recipient UA also uses it along with the CIC to verify the integrity of the message. If symmetric encipherment is employed, the CIK should be sent to the recipient UA via a secure channel.

Time-stamp

In general, a time-stamp contains an expiry time and an optional generation time. A time-stamp hinders an intruder from having enough time to manipulate the message content. The time-stamp should have its integrity and authenticity protected.

Message Security Label

A security label of a message is an indication of its sensitivity. This may be used by the MTS to guide its handling of the message. For example, it may refrain from delivering the message to a recipient whose security clearance is below the requirement stated in the label.

Message Sequence Number

A message sequence number is sent to a recipient for the detection of message duplication. It must not be changed during transmission.

3.2.2 The Structure of X.400 Message-Token

In order to send the above security-related information to the recipients safely, a signed data structure called **message-token** is employed. This can be clearly illustrated by the ASN.1 definition of the message-token in the X.400 Recommendations [CCITT411], as shown below:

```
message-token ::= SIGNED SEQUENCE{
                  signature-algorithm-identifier,
                  recipient-name,
                  time-stamp,
                  signed-data OPTIONAL,
                  encryption-algorithm-identifier OPTIONAL,
                  ENCRYPTED encrypted-data OPTIONAL }
```

where

```
signed-data ::= SEQUENCE{
                content-confidentiality-algorithm-identifier OPTIONAL,
                content-integrity-check OPTIONAL,
                message-security-label OPTIONAL,
                proof-of-delivery-request OPTIONAL,
                message-sequence-number OPTIONAL }

encrypted-data ::= SEQUENCE{
```

```
content-confidentiality-key OPTIONAL,
content-integrity-check OPTIONAL,
message-security-label OPTIONAL,
content-integrity-key OPTIONAL,
message-sequence-number OPTIONAL }
```

In order to simplify our later discussion, we give a more abstract definition of the message-token. First, the following notations are defined.

I_P : The identity of entity P.

t_P : The time-stamp of entity P.

K_P : The public key of entity P.

K_P^{-1} : The secret key of P. The matching public key is K_P .

$[M]_K$: Information M encrypted under key K.

h : One-way hash function (see Section 2.3.2).

$P\{M\}$: Information M signed by entity P. This is another way of writing $M, [h(M)]_{K_P^{-1}}$, which states that M is appended by P's signature $[h(M)]_{K_P^{-1}}$ (see Section 2.3.3).

Suppose a message-token is sent from A to B. Using these notations, the above message-token definition can be written as:

$$A\{sgnAlg,\ I_B,\ t_A,\ sgnData,\ encAlg,\ [encData]_{K_B}\}$$

Here, *sgnAlg* is **signature-algorithm-identifier** which identifies the algorithm used by the originator A to compute the signature. I_B is the identity (**name**) of B. t_A is the **time-stamp** of A. *encAlg* is **encryption-algorithm-identifier** which

identifies the algorithm used to perform this encryption. *sgnData* and *encData* are collections of security-related information, corresponding to `signed-data` and `encrypted-data` in the ASN.1 definition. The contents of *sgnData* and *encData* depend on the security services being provided.

For this message-token structure, the signature of originator A is calculated over the entire message-token, including $[encData]_{K_B}$ which is encrypted under B's public key K_B. To simplify the definition, the algorithm-identifiers, *sgnAlg* and *encAlg*, can also be represented by *sgnData*. The resulting token will still conform to the X.400 specification and lead to a simpler definition shown as follows:

Message-Token 1:

$$A\{T\}$$

where $T = I_B,\ t_A,\ sgnData,\ [encData]_{K_B}$.

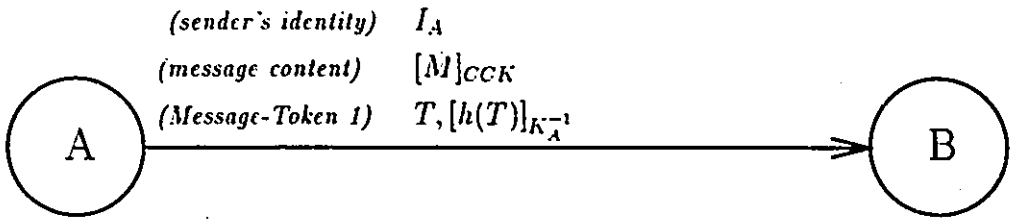
Message-Token 1 can also be presented in the following equivalent form:

$$T,\ [h(T)]_{K_A^{-1}}.$$

This form shows that the sender's signature in the message-token is $[h(T)]_{K_A^{-1}}$.

3.2.3 Protocol for Usage of Message-Tokens

We have explained, in Chapter 2, that a message sent from a UA to another UA consists of two parts: the message content and the message envelope. The message content is the real message that an MHS-user wants to deliver, and it is usually encrypted under CCK by his UA. The message envelope consists of the information about the message content, including message-tokens which are generated by the UA on a per-recipient basis. Since most of the components in the message envelope are



$$T = I_B, t_A, sgnData, [encData]_{K_B}$$

Figure 3.1: Sending a message from A to B

not related to security, we will only consider the security-related components, such as message-tokens and sender's name, in the later discussion.

The idea of using message-tokens is to apply a one-way authentication protocol in X.400 MHS. It involves a single transfer of the encrypted message and its security-related information from a sender UA to a recipient UA to achieve the integrity and origin authentication of the message being transferred. The structure of Message-Token 1 conforms to the one-way authentication protocol given by X.509 Authentication Framework [CCITT509].

Figure 3.1 shows A sending a message M to B. The protocol of using a message-token can be described as follows:

1. Sender A does the following:

   (a) encrypts M using CCK: $[M]_{CCK}$;

   (b) computes CIC from M using CIK;

   (c) generates message-token $T, [h(T)]_{K_A^{-1}}$ ($T = I_B, t_A, sgnData, [encData]_{K_B}$), where, *encData* includes CCK and CIK. *sgnData* may include CIC.

2. A sends A's identity, encrypted M and message-token $T, [h(T)]_{K_A^{-1}}$ to B.

3. At the receiver end, B carries out the following actions:

   (a) checks that B itself is the intended recipient;

   (b) checks that t_A is current;

   (c) verifies if A's signature on the message-token is valid by using A's public key (see Section 2.3.3) and the publicly available hash function h;

   (d) recovers CCK, CIC and CIK from $[encData]_{K_B}$ and deciphers $[M]_{CCK}$ using CCK to obtain M;

   (e) verifies the integrity of M using CIC and CIK (see Section 2.3.2 and Section 2.4.2).

Notice that steps (a), (b) and (c) are to authenticate the message-token; i.e. to ensure the integrity and originality of the token. If the received message can pass all these verifications, receiver B will believe in the integrity and the originality of the message. He will believe A is the originator of M and that the message was generated at the time t_A.

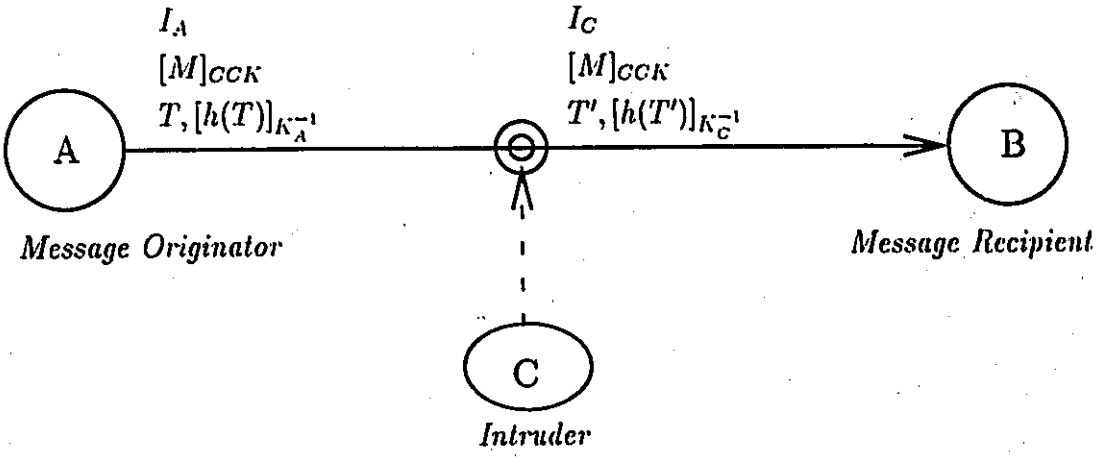
In summary, we should point out that using a message-token provides and supports the following security services for the message:

- Message origin authentication is provided by the signature on the message-token. This is because the message-token and the encrypted message are ordinarily generated by the same sender.
- Content integrity is provided by including a CIC in the *sgnData* (or *encData*) field of the token;

- Content confidentiality is provided by using a randomly chosen key CCK to encrypt the message content, and passing CCK to the recipient in the *encData* field of the token.

3.3 The Defect of Message-Token 1

Message-Token 1 can lead to a potentially serious security problem discovered by Burrows *et al.* in [BAN90]. The defect of Message-Token 1 is that such tokens can be easily modified by intruders for the purpose of impersonation, and the receiver is unable to detect this change. As a result, the receiver will wrongly believe the intruder to be the source of the message.



$$T = I_B, t_A, sgnData, [encData]_{K_B}$$
$$T' = I_B, t_C, sgnData, [encData]_{K_B}$$

Figure 3.2: Intruder illegally claims to be the originator

Figure 3.2 illustrates such an impersonating intrusion. Assume that user A sends

a message M to user B. A encrypts M under a CCK, and derives a CIC from the plaintext of M. Suppose the CCK and the CIC are included in *encData* of Message-Token 1. A then sends to receiver B the message, including A's identity, the encrypted M and the following message-token.

$$T, [h(T)]_{K_A^{-1}} .$$

where

$$T = I_B,\ t_A,\ sgnData,\ [encData]_{K_B} .$$

Now, suppose a malicious third party C intercepts the message. C is unable to read M because of his incapability to decrypt $[encData]_{K_B}$ to obtain the CCK. However, C can pretend to be the originator of M by constructing a new message-token as follows:

$$T', [h(T')]_{K_C^{-1}}$$

where

$$T' = I_B,\ t_C,\ sgnData,\ [encData]_{K_B} .$$

Note that the new message-token is signed by C and that t_C is the time-stamp of C, *encData* and *sgnData* remain unchanged. It should also be pointed out that C must replace the sender's identity, on the message envelope, by his own identity. Finally, C sends the originally encrypted M along with the new message-token to B.

Now, we examine why such a modification can lead B to believe that C is the originator of M. When B receives the modified message, according to the protocol B carries out the following actions:

(a) B checks that itself is the intended recipient.

(b) B checks that the time-stamp t_C is current.

(c) B verifies the signature on the token by using C's public key. The result of the verification shows that the signature belongs to C.

(d) B obtains CCK and CIC by deciphering $[encData]_{K_B}$ and recovers M.

(e) B verifies the integrity of M by using CIC. Because the CCK, the CIC and encrypted M haven't been changed by C, the verification will show that the integrity of M is maintained.

All the above verifications will definitely validate the results. This will lead B to wrongly believe that M is from C, instead of from its true originator A. This means that the impersonation has successfully occurred. Subsequently, undesired communication may occur between B and C.

The blame for lack of protection against such an attack can be laid against the form of the message-token used. In the above case, if B were able to detect that the received message-token had been illegally modified during the transmission, he would not believe the message-token comes from C and thus not believe message M is from C. C's attempted impersonation would not be successful. In Message-Token 1, the signature is calculated from the data including the encrypted value of *encData*. Thus there is no evidence to show to the recipient that the claimed originator (or the signer) knows *encData*. In the above case, C appears to B to be the claimed originator, and B cannot detect, by following the protocol, that C is an intruder who does not know *encData*. Therefore, the structure of message-token should be improved in order to solve the problem.

3.4 Problem in the Existing Solution

A solution has been proposed by Colin I'Anson and Chris Mitchell in [IAM90]. This solution is based on the idea that the signature of a message-token be derived from the plaintext of the data in the message-token. In other words, the sender signs the message-token before any of its data is encrypted. However, this solution cannot solve the problem in all circumstances, especially, when it is employed as a message-token in MHS. Before illustrating this point, the proposed solution is described below.

3.4.1 Existent Solution

In I'Anson and Mitchell's solution, the message-token (called Message-Token 2 in this thesis) sent from A to B is shown as follows:

Message-Token 2:

$$T,\ [h(L)]_{K_A^{-1}}.$$

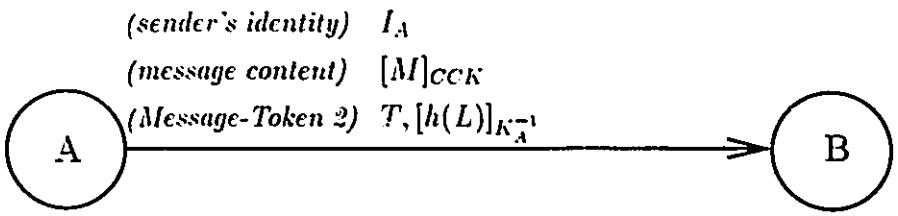
where

$$T = I_B,\ t_A,\ sgnData,\ [encData]_{K_B}$$

$$L = I_B,\ t_A,\ sgnData,\ encData$$

Here, L contains unencrypted data only. h is a one-way hash function and is known publicly. $[h(L)]_{K_A^{-1}}$ is the sender's signature calculated over L instead of T which contains $[encData]_{K_B}$. $encData$ is still kept confidential because of the one-way property of hash function h.

Figure 3.3 shows Message-Token 2 being used as a one-way authentication protocol in MHS. A encrypts M using CCK and generates Message-Token 2. A sends the



$$T = I_B, t_A, sgnData, [encData]_{K_B}$$

$$L = I_B, t_A, sgnData, encData$$

Figure 3.3: The Protocol of Using Message-Token 2

message to B. On the receiver end, receiver B carries out the following action to check and to authenticate the received message-token and the message:

(a) checks if itself is the intended recipient.

(b) checks if the time-stamp t_A is current.

(c) authenticates the message-token by verifying the claimed sender's signature.

i. obtains the hash value $h(L)$ by deciphering $[h(L)]_{K_A^{-1}}$ using the claimed sender's public key K_A;

ii. recovers *encData* using his secret key, and computes another hash value $h(I_B, t_A, sgnData, encData)$.

iii. compares these two hash values. If these two hash values are same, then $[h(L)]_{K_A^{-1}}$ is the signature of claimed originator A.

(d) recovers the message content M using CCK;

(e) checks the integrity of M using CIC.

If all these verifications are valid, then B should believe that the claimed originator A is the true originator of the message-token and the message content.

Message-Token 2 has been believed to solve the problem in [IAM90], because an intruder cannot make the kind of modification described in Section 3.3. To illustrate this, suppose A sends B a message using Message-Token 2.

If an intruder wants to illegally claim to be the source of a message, he must at least replace the following two things by his own: (1) the original sender's identity on the envelope; and (2) the original signature on the message-token. In the case of using Message-Token 2, changing the original signature becomes impossible for an intruder. That is, C cannot create a new message-token

$$T', [h(L')]_{K_C^{-1}}$$

where

$$T' = I_B, t_C, sgnData, [encData]_{K_B}$$

$$L' = I_B, t_C, sgnData, encData .$$

Here, C cannot generate $[h(L')]_{K_C^{-1}}$ as his signature, because C cannot form L' which contains a secret component, *encData*, unknown to him. *encData* is encrypted under user B's public key and can be decrypted only by B using his secret key K_B^{-1}. Therefore, it is concluded that C is unable to make any undetectable modification on Message-Token 2. The problem is solved in this case.

However, we will explain next that in certain other cases Message-Token 2 cannot solve the impersonation problem of Message-Token 1.

3.4.2 Problem in Message-Token 2

In the above discussion, there is an implicit assumption that the time-stamps t_A and t_C are distinguishable in terms of their originators. In fact, the X.400 time-stamp contains only the expiry time and the optional generation time of a message. The receiver cannot determine who actually generated the time-stamp from its contents alone. Thus, Message-Token 2 sent from A to B can be represented as follows:

$$T,\ [h(L)]_{K_A^{-1}}$$

where

$$T = I_B,\ t,\ sgnData,\ [encData]_{K_B}$$

$$L = I_B,\ t,\ sgnData,\ encData\ .$$

Here, t is the time-stamp.

In this situation, if the time-stamp need not be changed, intruder C can easily replace the original signature by his own, and create the following new token:

$$T,\ [h(L)]_{K_C^{-1}}$$

where

$$T = I_B,\ t,\ sgnData,\ [encData]_{K_B},$$

$$L = I_B,\ t,\ sgnData,\ encData\ .$$

The only change in the token is that C has replaced the signature $[h(L)]_{K_A^{-1}}$ with $[h(L)]_{K_C^{-1}}$. This can be done by C because he is able to recover the hash value $h(L)$ from $[h(L)]_{K_A^{-1}}$ by using A's public key K_A.

Upon receiving the modified message-token, B carries out the following verifications on the received token:

(a) checks that B itself is the intended receiver;

(b) checks that the time-stamp t has not expired;

(c) checks that the signature $[h(L)]_{K_C^{-1}}$ belongs to the claimed sender C.

It is obvious that the signature is valid. In the case where the time-stamp is still current, B will believe in the authenticity of the received message-token, and then continue to verify the content integrity. The result will lead B to believe that C generates the message-token and the message content at the time shown by the time-stamp. This means that Message-Token 2 cannot protect itself from such unauthorized modification, and thus cannot protect messages from the impersonating threat discussed in Section 3.3.

Therefore, "signing unencrypted data" may not prevent the impersonating attack when a one-way hash function is used in the signature calculation. Other solutions must be found to solve the problem of Message-Token 1.

3.5 New Solutions to the Problem

In this section, we propose three new solutions involving new message-token structures, namely Message-Token 3, Message-Token 4 and Message-Token 5. In order to counter the impersonation threat for messages, the new message-tokens have the property of protecting themselves against illegal modification for the purpose of impersonation. The solutions are outlined below; and their proofs are presented in Chapter 4.

3.5.1 The First New Solution: Message-Token 3

This new solution is to include the sender's identity in Message-Token 2 so that an intruder cannot replace the original signature without knowing the identity of the real message originator.

Figure 3.4 shows the protocol of using Message-Token 3. Suppose A is the message originator. A generates a time-stamp t, encrypts $encData$, then creates and sends the following message-token:

Message-Token 3:

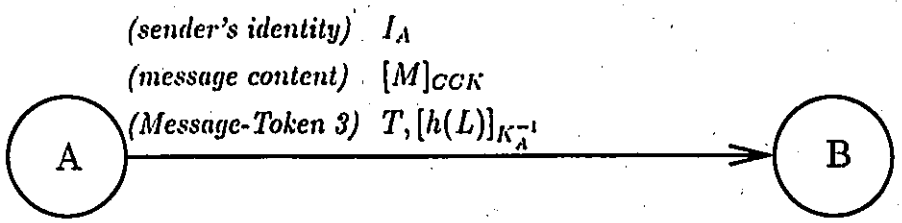
$$T,\ [h(L)]_{K_A^{-1}}\ .$$

where

$$T = I_B,\ t,\ sgnData,\ [encData]_{K_B}$$

$$L = I_A,\ I_B,\ t,\ sgnData,\ encData$$

Here, I_A is the identity of A.



$$T = I_B, t, sgnData, [encData]_{K_B}$$

$$L = I_A, I_B, t, sgnData, encData$$

Figure 3.4: The Protocol of Using Message-Token 3

At the receiving end, B authenticates the received message-token and the message content by carrying out the following steps:

(a) checks if B itself is the intended receiver;

(b) checks if the time-stamp t is current;

(c) verifies the signature in the message-token, i.e.

i. recovers the hash value $h(L)$ from $[h(L)]_{K_A^{-1}}$ by using the claimed originator's public key K_A;

ii. decrypts $[encData]_{K_B}$, obtains the claimed originator's identity I_A from the envelope, and then computes another hash value $h(I_B, I_A, t, sgnData, encData)$;

iii. compares these two hash values. If they are the same, the signature belongs to the claimed originator.

(d) recovers the message content M using CCK (included in $encData$);

(e) verifies the integrity of M using CIC.

If such verification indicates that nothing is wrong with the received token and the message content, then B believes that the claimed originator A is the true originator of the message-token and the message content.

The sender's identity in Message-Token 3 makes the signature computationally infeasible by any intruder who doesn't know $encData$. More specifically, intruder C is unable to create the following new message-token:

$$T', [h(L')]_{K_C^{-1}}$$

where

$$T' = I_B, t, sgnData, [encData]_{K_B}$$

$$L' = I_C,\ I_B,\ t,\ sgnData,\ encData$$

because it is impossible for C to generate $[h(L')]_{K_C^{-1}}$ as his signature. He cannot obtain *encData* which is an element of L'. *encData* is encrypted under user B's public key and can be decrypted only by B using his secret key K_B^{-1}.

On the other hand, the intruder C also cannot perpetrate successful impersonation by making the following change in the message-token:

$$T,\ [h(L)]_{K_C^{-1}}\ .$$

where

$$T = I_B,\ t,\ sgnData,\ [encData]_{K_B}$$

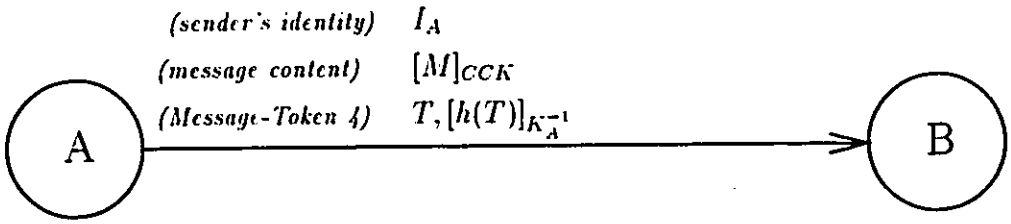
$$L = I_A,\ I_B,\ t,\ sgnData,\ encData$$

The reason is that B can find out that, at Step (c) of the protocol for Message-Token 3, the signature in the message-token does not belong to the claimed sender. It reveals that the message-token has been illegally modified.

Therefore, by using Message-Token 3 in MHS, an intruder is unable to make any undetectable modification on the message-token, and thus he cannot successfully pretend to be the source of the token and the message. The previous masquerade cannot occur if the Message-Token 3 scheme is used.

3.5.2 The Second New Solution: Message-Token 4

The second new solution applies a new message-token called Message-Token 4. In Message-Token 4, the originator's identity is encrypted together with *encData* using the receiver's public key. This ensures that an intruder is unable to make a valid



$$T = I_B, t_A, sgnData, [I_A, encData]_{K_B}$$

Figure 3.5: The Protocol of Using Message-Token 4

signature to replace the original signature, and thus an attempt of impersonating the message cannot succeed.

Figure 3.5 describes using Message-Token 4 in MHS. Originator A generates a time-stamp t and encrypts his identity together with *encData* using receiver B's public key K_B, Then A creates and sends the following token to B:

Message-Token 4:

$$T,\ [h(T)]_{K_A^{-1}}\ .$$

where

$$T = I_B,\ t,\ sgnData,\ [I_A,\ encData]_{K_B}$$

At the receiving end, B takes the following steps to authenticate the received message-token and the message:

(a) checks if B itself is the intended receiver;

(b) checks if the time-stamp t is current;

(c) verifies if the signature belongs to the claimed sender A.

(d) obtains an originator's identity by deciphering $[I_A, encData]_{K_B}$ and checks if this identity is same as the one from the envelope.

(e) obtains CCK from *encData* and recovers the message content M;

(f) verifies the integrity of M using CIC.

If all these steps yield the positive results, then B believes that the claimed originator A is the true originator of the message-token and the message.

It is obvious that I_A in $[I_A, encData]_{K_B}$ plays an important role in Message-Token 4. If intruder C tries to sign the message-token as follows:

$$T, [h(T)]_{K_C^{-1}} .$$

where

$$T = I_B, t, sgnData, [I_A, encData]_{K_B}$$

B can easily find out, at step (d), that *encData* is encrypted by A, but the signature belongs to C.

On the other hand, it is also impossible for C to change the original token into the following form:

$$T', [h(T')]_{K_C^{-1}}$$

where

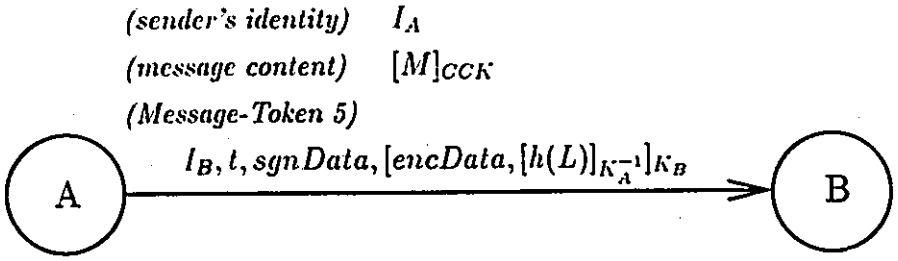
$$T' = I_B, t, sgnData, [I_C, encData]_{K_B} .$$

This is because C cannot change $[I_A, encData]_{K_B}$ into $[I_C, encData]_{K_B}$. To do this, he must hold B's private key K_B^{-1}.

Therefore, an intruder's attempt of impersonation cannot succeed. The problem is solved by using Message-Token 4.

3.5.3 The Third New Solution: Message-Token 5

In this new solution, the sender's signature is derived from the plaintext of data and then encrypted together with the secret information *encData*. This ensures that the signer (the claimed originator) knows *encData*, and also makes it impossible for any intruder to replace the original signature without knowing *encData*.



$$L = I_B, t_A, sgnData, encData$$

Figure 3.6: The Protocol of Using Message-Token 5

Figure 3.6 shows that Message-Token 5 is used as a one-way authentication protocol in MHS. Suppose A is the message originator. A encrypts M, creates the following message-token and sends the message to B:

Message-Token 5:

$$I_B,\ t,\ sgnData,\ [encData,\ [h(L)]_{K_A^{-1}}]_{K_B}$$

where

$$L = I_B,\ t,\ sgnData,\ encData$$

At the receiving end, B authenticates the received message by carrying out the following steps:

(a) checks if B itself is the intended receiver;

(b) checks if the time-stamp t is current;

(c) verifies the signature in the message-token, i.e.

i. decrypts $[encData, [h(L)]_{K_A^{-1}}]_{K_B}$ using K_B;

ii. recovers the hash value $h(L)$ from $[h(L)]_{K_A^{-1}}$ by using the claimed sender's public key K_A;

iii. computes another hash value $h(I_B, t, sgnData, encData)$;

iv. compares these two hash values. If they are the same, the signature belongs to the claimed sender.

(d) obtains CCK from *encData* and recovers the message-content M;

(e) verifies the integrity of the message content.

If all these verifications indicate nothing is wrong with the received token, then B believes that the claimed sender A is the true originator of the message-token and the message, M.

In Message-Token 5, the signature calculated from the plaintext data, L, shows that the signer knows *encData*. Furthermore, the signature encrypted together with *encData* makes it impossible for any intruder who doesn't know *encData* to replace it. By using Message-Token 5 in MHS, an intruder is unable to make any undetectable modification on the message-token, and thus he cannot successfully pretend to be the source of the token and the message. The previous impersonation cannot occur if the Message-Token 5 scheme is used.

3.6 Summary

In this chapter, we have stated that the nature of an impersonation attack described here is that an intruder can replace an original signature by his own even though he may not be aware of the whole message. One way to prevent this attack is to ensure that a receiver can detect such an unauthorized modification on the message-token being used. All of Message-Tokens 3, 4 and 5 meet this requirement, and thus can prevent the impersonation attack.

Someone may have the following arguments: Among the new message-tokens, only Message-Token 5 carries the evidence that the claimed message originator actually encrypts *encData*. Message-Tokens 3 and 4 can prevent the impersonation attack but carry no evidence that the token originator knows *encData*. It may happen that someone else helps the token originator create *encData* and encrypt it. However, since only the methods of preventing the impersonation attack are studied in the thesis, this scenario will not be considered, i.e., the token originator should always create and encrypt *encData* himself. Or, it can be considered as an originating group acting like a single originator.

Chapter 4

Logical Proof of Authentication for the New Message-Tokens

4.1 Introduction

In this chapter, as the main contribution of this thesis, we shall use *authentication logic* to prove that Meessage-Tokens 3, 4 and 5 function correctly. This authentication logic was created by Burrows, Abadi and Needham in [BAN90]. It was developed for specifying the authentication goals and for verifying that authentication protocols satisfy such goals. In other words, it allows us to describe the beliefs of trustworthy parties involved in authentication protocols and the evolution of these beliefs as the consequence of communication. There are also several papers which improve and extend the syntax and semantics of the logic ([ABT91] [GNY90]).

This chapter not only proves that the new message-tokens can achieve their authentication goals, but also improves the logic postulates, called Hash-Function Rules. The improvement makes them more accurate in describing the reasoning behind the

use of one-way hash functions for digital signatures. More discussion about the improved Hash-Function Rules can be found in Section 4.2.3.

The rest of the chapter is arranged as follows: Section 4.2 introduces the basic concepts of the logic, including notations, preliminaries and postulates and the improved Hash-Function Rule. Section 4.3 discusses some issues on using the logic, such as the authentication goals and idealized protocols. Section 4.4 analyzes Message-Tokens 1 & 2 and shows that the authentication goals are not achieved. Section 4.5, Section 4.6 and Section 4.7 analyze Message-Tokens 3, 4 and 5 respectively, and prove their correctness.

4.2 The Basic Concepts

In this section, we introduce the syntax and semantics of the logic, its rules, and the transformations which will be applied to prove the correctness of the new message-tokens. Those concepts were given by [BAN90].

4.2.1 Syntax and Semantics

In authentication logic, there are several sorts of objects: principals, statements and encryption keys. Principals are the parties who participate in the communication. Statements may be included in any kind of messages or information exchanged between principals. The symbols P and Q represent the principals, X and Y are statements, and K denotes an encryption key.

The logical proof of authentication employs the following constructs:

P believes X: P *believes* in X or P would be entitled to believe X. In particular, P may act as if X is true.

P **sees** X: *P sees X.* P has received X in a message. P is able to read X (possibly after decryption) and repeat it.

P **said** X: *P once said X.* P has sent X in a message. P believed that X was true when it sent the message. It is not known whether the message was sent long ago or during the current protocol run.

fresh (X): *"X is fresh."* Time is devised into two *epochs*, the *past* and the *present*; the present begins with the start of the current execution of the current protocol. X is fresh if it has not been sent in a message at any time in the past (before the current protocol run). Verifying the freshness of messages guards against the replay of old messages.

$P \overset{K}{\leftrightarrow} Q$: *"$K$ is a shared key for P and Q."*. It will never be discovered by any principal except P and Q, or a principal trusted by either P or Q.

$\overset{K}{\mapsto} P$: *"P has K as public key."* K^{-1} as the matching secret key will never be discovered by any principal except P or a principal trusted by P .

$P \overset{X}{=} Q$: *"X is a shared secret between P and Q."* X is known only to P and Q, and possibly by principals trusted by them. P and Q can use X to prove their identities to one another.

$[X]_K$: *"X encrypted under K"*. $[X]_K$ represents the message X encrypted using the key K. This is a shorthand for $[X^P]_K$, where P is the principal encrypting the message. The mention of P is used only in implementing an assumption that each principal can recognize and ignore its own messages.

$\langle X \rangle_Y$: *"X combined with Y."* $\langle X \rangle_Y$ represents the message X combined in some way with Y, usually by concatenation. Y is intended to be a secret and its presence proves the identity of the sender.

4.2.2 Logical Postulates

Some informal preliminaries are useful for understanding the rules of the authentication logic. Firstly, an encrypted message cannot be understood by a principal who does not know the decryption key; and the decryption key cannot be deduced from the encrypted message. Messages contain sufficient information for a principal to identify his own messages. Secondly, in the study of authentication, two epochs are considered : the *past* and the *present*. The present epoch begins at the start of the particular protocol run under consideration. All messages sent before this moment are considered to be in the past; and the authentication protocol should be careful to prevent any such messages from being accepted as belonging to the current protocol run. All beliefs held in the present are stable for the entirety of the protocol run, while beliefs held in the past are not necessarily carried forward into the present.

After these informal preliminaries, we are now ready to discuss the main logical postulates used in our proofs.

Message-Meaning Rules

The message-meaning rules are concerned with the interpretation of messages. Two of them concern the interpretation of encrypted messages, and the third concerns the interpretation of messages in cases where the principals share a secret. They are all used to derive beliefs about the origin of messages.

(a) For shared keys:

$$\frac{P \text{ believes } (Q \overset{K}{\leftrightarrow} P),\ P \text{ sees } [X^R]_K}{P \text{ believes } Q \text{ said } X}$$

This is, if P believes that the key K is shared with Q and sees that X is encrypted under K, then P believes that Q sent the message X. However,

P must be certain that it did not simply send the message to itself, and the side condition that $P \neq R$ reflects the assumption that a principal can detect and ignore its own messages.

In the case of $P = R$, [ABT91] gives the following argument. If P has said $[X]_K$, then either P has received $[X]_K$ in a previous message, or P has the key K to encrypt the message X and hence is also considered to have said the content X. These situations should be decided according to each individual protocol.

(b) For public keys:

$$\frac{P \text{ believes } \overset{K}{\mapsto} Q,\ P \text{ sees } [X]_{K^{-1}}}{P \text{ believes } Q \text{ said } X}$$

This rule represents the basic concept of digital signature (see Section 2.3.3, the encrypted message as the signature).

(c) For shared secrets:

$$\frac{P \text{ believes } (Q \overset{Y}{\rightleftharpoons} P),\ P \text{ sees } \langle X \rangle_Y}{P \text{ believes } Q \text{ said } X}$$

This is, if P believes that the secret Y is shared with Q and sees $\langle X \rangle_Y$, then P believes that Q once said X. In particular, Y can be the identity of Q.

Nonce-Verification Rule

A *nonce* is an expression (such as a time-stamp) invented for the purpose of being fresh, and included in messages to prove their freshness. The Nonce-Verification Rule

$$\frac{P \text{ believes fresh}(X),\ P \text{ believes } Q \text{ said } X}{P \text{ believes } Q \text{ believes } X}$$

says that if P believes a message is fresh and that Q sent the message, then P believes that Q sent the message recently, and still believes the content of the message. The name "nonce-verification" comes from the fact that the freshness of X is typically proven by including a nonce in the message X.

Freshness Rule

If one part of a statement is fresh, then the entire statement must also be fresh:

$$\frac{P \text{ believes fresh}(X)}{P \text{ believes fresh}(X,\ Y)}$$

Seeing Rules

If a principal sees a statement, then he also sees its components if he knows the necessary keys. In the following rules, $P \neq R$.

(a)

$$\frac{P \text{ sees } (X,\ Y)}{P \text{ sees } X}$$

(b)

$$\frac{P \text{ sees } \langle X \rangle_Y}{P \text{ sees } X}$$

(c)

$$\frac{P \text{ believes } (Q \overset{K}{\leftrightarrow} P),\ P \text{ sees } [X^R]_K}{P \text{ sees } X}$$

(d)

$$\frac{P \text{ believes } \overset{K}{\mapsto} P,\ P \text{ sees } [X^R]_K}{P \text{ sees } X}$$

Note that if P sees X and P sees Y, it does not follow that P sees $(X,\ Y)$, since this means that X and Y were uttered at the same time.

Conjunction-Breaking Rules

(a)

$$\frac{P \text{ believes } Q \text{ believes } (X,\ Y)}{P \text{ believes } Q \text{ believes } X}$$

(b)

$$\frac{P \text{ believes } Q \text{ said } (X,\ Y)}{P \text{ believes } Q \text{ said } X}$$

Note that if P believes Q said X and P believes Q said Y, it does not always follow that P believes Q believes$(X,\ Y)$, since this would imply that the two parts X and Y were uttered at the same time.

4.2.3 Improved Hash-Function Rule

In practice, principals often sign the hash value of a message instead of the entire message, mainly for the reason of efficiency (see Section 2.3.3). A one-way hash function is used here. It is intended that, given a hash value, it should be computationally infeasible to obtain information yielding the same hash value. Moreover, it should be computationally infeasible to generate two pieces of information having the same hash value, to avoid uncertainty in the identity of the data being signed.

For digital signature schemes using one-way hash functions, the underlying assumption is that signing a hash value is equivalent to signing the entire message. We have already stated that the form of a signature on the message X (signed by Q) is

$$[h(X)]_{K_Q^{-1}}.$$

To capture the reasoning implicit in the use of a one-way hash function h, the following Hash-Function Rule is employed [BAN89]:

$$\frac{P \text{ believes } Q \text{ said } h(X),\ P \text{ sees } X}{P \text{ believes } Q \text{ said } X}$$

In this thesis, however, we claim that the Hash-Function Rule [BAN89] is not always sound. To explain this fact, suppose A sends the ciphertext $[X]_{K_B}$ with A's signature $[h(X)]_{K_A^{-1}}$ to B. An intruder C can replace the signature by its own signature during the transmission. When B receives $[X]_{K_B}$ with C's signature $[h(X)]_{K_C^{-1}}$, B believes that C said $h(X)$. B also obtains X from the decryption. According to the [BAN89] Hash-Function Rule, B should believe that C said X. Actually, C doesn't even know (or see) X. What C does know is $[X]_{K_B}$ and $h(X)$.

So, BAN Hash-Function Rule is not universally true. Some conditions must be added, as descripted in the following improved Hash-Function Rules.

Hash-Function Rule (a)

$$\frac{P \text{ sees } [h(X)]_{K_Q^{-1}},\ P \text{ believes } \overset{K_Q}{\mapsto} Q,\ P \text{ sees } X}{P \text{ believes } Q \text{ said } X}$$

For this rule to be sound, one of the following conditions must be satisfied:

- **X is not a result from decryption; or**
- **X and $[h(X)]_{K_Q^{-1}}$ are encrypted together under one encryption key.**

Furthermore, [BAN90] gives another postulate in the case where a piece of information contains several components.

$$\frac{P \text{ believes } Q \text{ said } h(X_1, ..., X_n,),\ P\text{sees}X_1, ..., P\text{sees}X_n}{P \text{ believes } Q \text{ said } (X_1, ..., X_n)}$$

It says that some components of the message can be the results of decryption. Again, this may lead to the same error described above. So we redefine it as follows:

Hash-Function Rule (b)

$$\frac{P \text{ sees } [h(X_1,...,X_n)]_{K_Q^{-1}},\ P \text{ believes } \overset{K_Q}{\mapsto} Q,\ \ P \text{ sees } X_1,...,P \text{ sees } X_n}{P \text{ believes } Q \text{ said } (X_1,...,X_n)}$$

For this rule to be sound, the following condition must be satisfied:

- **At least one of X_i $(i = 1,2,...,n)$ is encrypted together with $[h(X_1,...,X_n)]_{K_Q^{-1}}$ under one encryption key.**

This rule says that if one component of the message is encrypted together with the signature, then the sender (signer) said the whole message.

Hash-Function Rule (c)

$$\frac{P \text{ sees } [h(X_1,...,X_n)]_{K_Q^{-1}},\ P \text{ believes } \overset{K}{\mapsto} Q,\ \ P \text{ sees } X_1,...,P \text{ sees } X_n}{P \text{ believes } Q \text{ said } X_i}$$

where, X_i $(X_i \in \{X_1,...,X_n\})$ **is not the result of decryption.**

This rule says that the unencrypted parts of the message are always known by the signer.

4.3 Using the Logic

To use the authentication logic to analyze a protocol, one first abstracts the protocol into a special idealized form, writes down a list of formulas describing the assumptions about the system and the authentication goals to be achieved at the end of the protocol's execution. Then, the inference rules can be used to derive the belief formulas holding at various points in the protocol's execution. Inability to derive the formulas stating the authentication goals for a protocol suggests the existence of an error in the protocol.

4.3.1 The Goals of Authentication

There is room for debate about what should be the goals of authentication protocols. Different protocols have different authentication goals. Usually authentication is a precursor to some communication protected by a shared session key, so we might desire conclusions that describe the situation at the start of such a communication. Thus, the authentication is considered to be complete between P and Q if there is a K such that

$$P \text{ believes } (Q \overset{K}{\leftrightarrow} P), \quad Q \text{ believes } (Q \overset{K}{\leftrightarrow} P)$$

Some authentication protocols achieve stronger goals of authentication, as shown below:

$$P \text{ believes } Q \text{ believes } (Q \overset{K}{\leftrightarrow} P), \quad Q \text{ believes } P \text{ believes } (Q \overset{K}{\leftrightarrow} P)$$

Some public-key protocols are not intended to result in the exchange of shared keys, but instead to transfer some other pieces of information. In these cases, the required goals are generally obvious from the context. For example, the interaction of principal P with the certification authority may be intended by P to transfer a single public key to P, such that:

$$P \text{ believes } \overset{K}{\mapsto} Q$$

In addition, principals may establish some shared secrets:

$$P \text{ believes } (P \overset{Y}{\rightleftharpoons} Q), \quad Q \text{ believes } (P \overset{Y}{\rightleftharpoons} Q)$$

Some protocols may attain the final states, such as:

$$P \text{ believes } Q \text{ believes } X .$$

That is, P believes that Q has recently sent X. The authentication goal of the MHS message-tokens belongs to this category.

4.3.2 Idealized Form of Protocols

In most of the literature, an authentication protocol is described as a sequence of steps of the form

$$P \rightarrow Q: \ X$$

intended to denote the fact that P sends X to Q.

Many representations of protocols are not in the appropriate basis for formal analysis. Consequently, instead of analyzing the informal description of these protocols as they appear in the literature, we analyze protocols written in an idealized form. For instance, the protocol step

$$A \rightarrow B: \ [A, K_{ab}]_{K_{bs}}$$

may tell B, who knows the key K_{bs}, that K_{ab} is a key to communicate with A. This step should then be idealized as

$$A \rightarrow B: \ [A \overset{K_{ab}}{\leftrightarrow} B]_{K_{bs}} .$$

Furthermore, the idealized protocols often omit some of the message parts, such as the plaintext of data, because they can be forged anyway and do not contribute anything of interest to the beliefs of the principals taking part in the protocol.

4.4 Analysis of Message-Tokens 1 & 2

The authentication goal of the message-tokens is to provide the integrity and origin authentication of the transferred information, *sgnData* and *encData*. That is:

$$B \text{ believes } A \text{ believes } sgnData, \quad B \text{ believes } A \text{ believes } encData \ .$$

where A is the principal sending the message-token and B is the principal receiving the token.

For all the message-tokens, it is assumed that each principal knows his own secret key and the other's public key, and the recipient believes the sender's time-stamp to be fresh. These assumptions are presented as follows:

A believes $\overset{K_A}{\mapsto} A$	(a1)
A believes $\overset{K_B}{\mapsto} B$	(a2)
B believes $\overset{K_B}{\mapsto} B$	(a3)
B believes $\overset{K_A}{\mapsto} A$	(a4)
B believes fresh(t)	(a5)

4.4.1 Analysis of Message-Token 1 [BAN90]

The one-way authentication protocol using Message-Token 1 can be represented as follows:

$$A \rightarrow B: \ T, [h(T)]_{K_A^{-1}} \ .$$

where

$$T = I_B,\ t,\ sgnData,\ [encData]_{K_B}$$

Now, we analyze the protocol using the rules and the above assumptions. According to Seeing Rule (b), when B receives Message-Token 1,

$$B \text{ sees } [h(I_B,\ t,\ sgnData,\ [encData]_{K_B})]_{K_A^{-1}} .$$

Message-Meaning Rule (B) and assumption (a4) yield

$$B \text{ believes } A \text{ said } h(I_B,\ t,\ sgnData,\ [encData]_{K_B}) .$$

The following formula is also true when B receives the Message-Token 1.

$$B \text{ sees } (I_B,\ t,\ sgnData,\ [encData]_{K_B})$$

Applying the Hash-Function Rule (c) yields

$$B \text{ believes } A \text{ said } (I_B,\ t,\ sgnData,\ [encData]_{K_B}) .$$

Using assumption (a5) B **believes fresh**(t); and the Freshness Rule yields

$$B \text{ believes fresh}(I_B,\ t,\ sgnData,\ [encData]_{K_B}) .$$

From the Nonce-Verification Rule, we can get

$$B \text{ believes } A \text{ believes } (I_B,\ t,\ sgnData,\ [encData]_{K_B}) .$$

Breaking the above conjunction, we obtain

$$B \text{ believes } A \text{ believes } sgnData .$$

For *encData*, we only can obtain B **believes** A **believes** $[encData]_{K_B}$. This does not imply that B **believes** A **believes** *encData* . It suggests that the authentication goal may be unattainable.

For a better understanding of the above analysis given by [BAN90], we should notice that, for receiver B, principal A is a claimed originator (who may be either

the true originator or an intruder). The receiver can make sure that the claimed originator is the true originator of the message if all the authentication goals can be achieved. In other words, since the above analysis shows that one of the authentication goals cannot be reached, the claimed originator A may be an intruder who does not know *encData* (only the true originator knows *encData*). So, the analysis gives an explanation of why the impersonation can occur when using Message-Token 1 in MHS.

4.4.2 Analysis of Message-Token 2

The one-way authentication protocol using Message-Token 2 can be represented as:

$$A \rightarrow B: \ T, [h(L)]_{K_A^{-1}} \ .$$

where

$$T = I_B,\ t,\ sgnData,\ [encData]_{K_B}$$

$$I_\cdot = I_B,\ t,\ sgnData,\ encData$$

For Message-Token 2, there are the same assumptions as for Message-token 1. When B receives Message-Token 2 from A, we have

$$B \textbf{ sees } [h(I_B,\ t,\ sgnData,\ encData)]_{K_A^{-1}}$$

Message-Meaning Rule (b) yields

$$B \textbf{ believes } A \textbf{ said } h(I_B,\ t,\ sgnData,\ encData)\ .$$

When B receives Message-Token 2, the following formula is also true.

$$B \textbf{ sees } (I_B,\ t,\ sgnData,\ [encData]_{K_B})\ .$$

Since B can decipher $[encData]_{K_B}$ and all of the data I_B, t, $sgnData$ and $[encData]_{K_B}$ are received together, we then have

$$B \text{ sees } (I_B,\ t,\ sgnData,\ encData)\ .$$

Because $encData$ is the result of deciphering $[encData]_{K_B}$, only Hash-Function Rule (c) can be applied. So, B can ensure that A knows I_B, t and $sgnData$ rather than $encData$. That is

$$B \text{ believes } A \text{ said } (I_B,\ t,\ sgnData)\ .$$

Again, we have the hypothesis (a5) B **believes fresh**(t) , and the Freshness Rule applies and yields

$$B \text{ believes } A \text{ believes } (I_B,\ t,\ sgnData)\ .$$

So far, we can only obtain

$$B \text{ believes } A \text{ believes } sgnData\ .$$

We have not been able to obtain another authentication goal B **believes** A **believes** $encData$. This suggests the weakness in Message-Token 2.

4.5 Analysis of Message-Token 3

The use of Message-Token 3 as a one-way authentication protocol can be represented as:

$$A \rightarrow B:\ T, [h(L)]_{K_A^{-1}}$$

where

$$T = I_B,\ t,\ sgnData,\ [encData]_{K_B}$$

$$L = I_A,\ I_B,\ t,\ sgnData,\ encData$$

The protocol can be idealized into the following form:

$$A \rightarrow B: \ T, [h(A \overset{Y}{\rightleftharpoons} B)]_{K_A^{-1}}$$

where

$$Y = I_B,\ t,\ sgnData,\ encData$$

Here, $h(L)$ can be considered as $h(A \overset{Y}{\rightleftharpoons} B)$. This is because, in the protocol of Message-Token 3, I_A is the identity of the originator who actually derived hash value $h(I_A, Y)$. On the other hand, only B can obtain *encData* from the message-token, and thus obtain the entire Y. Therefore, Y is the shared secret between A and B.

For Message-Token 3, the assumptions are the same as for the previous message-tokens. We analyze the protocol by applying the logic postulates to its idealized form. When B receives Message-Token 3,

$$B \textbf{ sees} [h(A \overset{Y}{\rightleftharpoons} B)]_{K_A^{-1}} .$$

Then, Message-Meaning Rule (b) and assumption (a4) yield

$$B \textbf{ believes } A \textbf{ said } h(A \overset{Y}{\rightleftharpoons} B) \ .$$

By deciphering $[encData]_{K_B}$, B can obtain Y, i.e

$$B \textbf{ sees } Y \ .$$

Since Y is the secret shared by A and B, the formula, A **said** $h(A \overset{Y}{\rightleftharpoons} B)$, means that the hash value $h(A \overset{Y}{\rightleftharpoons} B)$ must be computed by A itself. In this case, the following formula is true:

$$B \textbf{ believes } A \textbf{ said } (A \overset{Y}{\rightleftharpoons} B) ,$$

i.e.

$$B \textbf{ believes } A \textbf{ said } (I_B,\ t,\ sgnData,\ encData) \ .$$

Similar to the previous proof, we can obtain

$$B \textbf{ believes fresh}(t,\ sgnData,\ encData)\,.$$

From the Nonce-Verification Rule, we can get

$$B \textbf{ believes } A \textbf{ believes } (t,\ sgnData,\ encData)\,.$$

Conjunction-Breaking Rule applies and yields the result:

$$B \textbf{ believes } A \textbf{ believes } sgnData,\quad B \textbf{ believes } A \textbf{ believes } encData\,.$$

Therefore, the authentication goal is achieved by Message-Token 3.

4.6 Analysis of Message-Token 4

The one-way authentication protocol using Message-Token 4 can be shown as:

$$A \rightarrow B: \ T, [h(T)]_{K_A^{-1}} \ .$$

where

$$T = I_B,\ t,\ sgnData,\ [I_A,\ encData]_{K_B}$$

According to the protocol of Message-Token 4, *encData* is encrypted together with the identity of the encryptor, i.e. $[I_A,\ encData]_{K_B}$ indicates that A encrypted *encData*. On the other hand, $[I_A,\ encData]_{K_B}$ can only be decrypted by B. This means that *encData* is a secret shared by A and B. So, $[I_A,\ encData]_{K_B}$ can be replaced by $[A \overset{encData}{\rightleftharpoons} B]_{K_B}$ in the protocol, shown as follows:

$$A \rightarrow B: \ I_B,\ t,\ sgnData,\ [A \overset{encData}{\rightleftharpoons} B]_{K_B},\ [h(I_B,\ t,\ sgnData,\ [A \overset{encData}{\rightleftharpoons} B]_{K_B})]_{K_A^{-1}}$$

The assumptions for Message-token 4 are the same as the previous tokens. Now, we analyze Message-Token 4 by applying the authentication logic rules. When B receives Message-Token 4,

$$B \text{ sees } [h(I_B,\ t,\ sgnData,\ [A \overset{encData}{\rightleftharpoons} B]_{K_B})]_{K_A^{-1}} \ .$$

The following formula is true when B receives Message-Token 4.

$$B \text{ sees } (I_B,\ t,\ sgnData,\ [A \overset{encData}{\rightleftharpoons} B]_{K_B})$$

We also have the assumption (a4)

$$B \text{ believes } \overset{K_A}{\mapsto} A \ ,$$

Hash-Function Rule (c) applies and yields

$$B \text{ believes } A \text{ said } (I_B,\ t,\ sgnData,\ [A \overset{encData}{\rightleftharpoons} B]_{K_B}) \ .$$

Again, the hypothesis B **believes fresh**(t) and Freshness Rule yield

$$B \textbf{ believes fresh}(I_B,\ t,\ sgnData,\ [A \overset{encData}{\rightleftharpoons} B]_{K_B}) .$$

From Nonce-Verification Rule, we can get

$$B \textbf{ believes } A \textbf{ believes } (I_B,\ t,\ sgnData,\ [A \overset{encData}{\rightleftharpoons} B]_{K_B}) .$$

Applying Conjunction-Breaking Rule, we have

$$B \textbf{ believes } A \textbf{ believes } sgnData ,$$

and

$$B \textbf{ believes } A \textbf{ believes } [A \overset{encData}{\rightleftharpoons} B]_{K_B} .$$

The formula A **believes** $[A \overset{encData}{\rightleftharpoons} B]_{K_B}$ means that A recently said $[A \overset{encData}{\rightleftharpoons} B]_{K_B}$. *encData* is the shared secret between A and B, and B didn't send the message. So, we can conclude that A made the encryption, i.e. A recently said *encData*. The following formula is true:

$$B \textbf{ believes } A \textbf{ believes } encData .$$

Therefore, the authentication goals are achieved by Message-Token 4.

4.7 Analysis of Message-Token 5

The use of Message-Token 5 as a one-way authentication protocol can be represented as:

$$A \rightarrow B: \ I_B,\ t,\ sgnData,\ [encData,\ [h(L)]_{K_A^{-1}}]_{K_B}\ .$$

where

$$L = I_B,\ t,\ sgnData,\ encData\ .$$

For Message-Token 5, the assumptions are the same as for the previous message-tokens. When B receives Message-Token 5,

$$B \textsf{ sees } I_B,\ t,\ sgnData,\ [encData,\ [h(L)]_{K_A^{-1}}]_{K_B}\ .$$

B can decrypt $[encData,\ [h(L)]_{K_A^{-1}}]_{K_B}$, and the following formula is true:

$$B \textsf{ sees } [h(L)]_{K_A^{-1}}\ ,$$

and

$$B \textsf{ sees } encData\ .$$

We can say

$$B \textsf{ sees } (I_B,\ t,\ sgnData,\ encData)\ ,$$

here, $encData$ is the result of decryption and it is encrypted together with the signature. So, Hash-Function Rule (b) can be applied and yields

$$B \textsf{ believes } A \textsf{ said } (I_B,\ t,\ sgnData,\ encData)\ .$$

Similarly, we can obtain

$$B \textsf{ believes fresh}(t,\ sgnData,\ encData)\ .$$

From the Nonce-Verification Rule, we can get

$$B \text{ believes } A \text{ believes } (t,\ sgnData,\ encData)\ .$$

The Conjunction-Breaking Rule applies and yields the result:

$$B \text{ believes } A \text{ believes } sgnData,\quad B \text{ believes } A \text{ believes } encData\ .$$

Therefore, the authentication goal is also achieved by Message-Token 5.

Chapter 5

Conclusions and Future Works

This thesis has studied the structure of message-tokens which influence the security of transmitted messages in X.400 MHS. To protect messages from the impersonation attack, three new message-tokens have been proposed. This thesis also has improved the logic of authentication and has proved the correctness of the new message-tokens. The results of the thesis will improve the security services of a message handling system.

The nature of the impersonation attack concerned in the thesis is that an intruder can sign the message-token again (i.e. change the original signature to his own) without knowing *encData*, and thus he can pretend to be the source of the message-token and the message content. Due to one-way functions being used for calculating the signature in the practice, "signing unencrypted data" may not be enough to defend against the impersonation threat. Message-Tokens 3, 4 and 5 have been proposed. These three message-tokens present all types of possible solutions to solve the problem at the message-token level.

Message-Token 3 type of solutions: The signature is derived from plain-

text data, which include a component characterizing the originator.

Message-Token 4 type of solutions: The signature is derived from the encrypted data, which include a component characterizing the originator.

Message-Token 5 type of solutions: The signature is derived from the plaintext data and is encrypted together with at least one of the secret components.

We can conclude that the following services and conditions must be provided to prevent such an impersonation threat in MHS:

- Content Integrity Service;
- Message Origin Authentication Service;
- Content confidentiality Service;
- The original signature in the message-token cannot be successfully replaced by others.

In this thesis, the authentication logic of Burrows *et al.* is used to prove that the new message-tokens we proposed can achieve the desired authentication goals. We found that the Hash-Function Rules do not correctly present the reasoning implicated in using hash functions for the signature calculation. They ignore that the impersonation attack could succeed in cases like Message-Token 2. At present, there are a number of proposals in the literature for improving the syntax and semantics of the logic, but none of them examined the correctness of the postulates, especially the Hash-Function Rules. Our newly defined Hash-Function Rules are able to overcome

the weaknesses of the old rules and help to detect the potential security problem (e.g. the impersonation problem in Message-Token 2) existing in authentication protocols.

In the future work, we shall study how the impersonation attack can be prevented in multi-destination messaging, in which a message is sent to a group of recipients at same time. In this case, the impersonation threat may come from a malicious group member who knows more about the message than outside intruders. The security services in one-to-one messaging may not be simply "multiplied" [MIT92] [JAN91]. Furthermore, the authentication logic should also be extended to the group communication.

X.400 security architecture itself still needs to be developed. The security features defined under the 1988 X.400 Recommendations do not fully cover all security issues on all levels. For example, they do not deal with internal security from the User Agent(UA) to the final MHS-user himself. They also cannot cover the destruction of messages, although they can help to identify that a message has gone astray.

Finally, we should point out that the security can be built into the MHS. A fundamental condition for this is the development of a consistent security architecture which should include:

- *security policy* which derives from the threats to the system and user's security requirements;
- *security services* which are used to counter the perceived security threats; and
- *security mechanisms* which are used to support the security service.

The X.400 security features must be implemented as part of an overall security messaging policy of OSI, according to OSI Security Architecture of [ISO7498-2]. The

X.400 security alone may not provide any guarantees for the security of the entire message transmission, because it is provided only within the application layer.

Bibliography

[ABT91] Martin Abadi, Mark R. Tuttle. *A Semantics for a Logic Authentication.* Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing (1991), pp. 201-216.

[BAN90] Michael Burrows, Martin Abadi and Roger Needham. *A Logic of Authentication.* ACM Transactions on Computer Systems, Vol. 8, No. 1 (1990), pp. 18-36.

[BAN89] Michael Burrows, Martin Abadi and Roger Needham. *A Logic of Authentication.* Rep. 39, Digital Equipment Corporation Systems Research Systems Research Center, Palo Alto, Calif. Fev. 1989.

[CCITT400] CCITT Recommendation X.400, *Message Handling, Part 1: System and Service Overview.* (1988)

[CCITT402] CCITT Recommendation X.402, *Message Handling, Part 2: Overall Architecture.* (1988)

[CCITT411] CCITT Recommendation X.411, *Message Handling, Part 4: Message Transfer System - Abstract Service Definition and Procedures.* (1988)

[CCITT420] CCITT Recommendation X.420, *Message Handling, Part 7: Interpersonal Messaging System.*

[CCITT435] CCITT Recommendation X.435, *MHS Application for EDI Messaging.*

[CCITT500] CCITT Recommendation X.500, *The Directory, Part1: Overview of Concepts, Models, and Services.*

[CCITT509] CCITT Recommendation X.509, *The Directory, Part8: Authentication Framework.*

[DES77] National Bureau of Standards. *Data encryption standard* Federal Information Processing Standard Publ. 46. (1977)

[DIF76] W. Diffie and M. Hellman, *New Directions in Cryptography.* IEEE Trans. Inform. Theory, Vol. IT-22, (1976), pp. 644-654.

[GEN90] Guy Genlloud. *X.400 MHS: First Steps Toward an EDI Communication Standard.* ACM SIGCOMM Computer Communication Review, Vol. 20, No. 2 (1990), pp. 72-86.

[GNY90] L. Gong, R. M. Needham, and R. Yahalom. *Reasoning About Belief in Cryptographic Protocols.* Proceedings of the 1990 IEEE Symposium on Security and Privacy (1990), pp. 234-247.

[IAM90] C.I'Anson and C. J. Mitchell, *Security defects in CCITT Recommendation X.509 The directory authentication framework,* Computer Communication Review, Vol. 20, No. 2 (1990), pp. 30-34.

[ISO7498-2] ISO, Information Processing System, *Open Systems Interconnection, Basic Reference Model, Part 2: Security Architecture,* (1989).

[JAN91] P. Janson and R. Molva. *Security in Open Networks and Distributed Systems.* Computer Networks and ISDN Systems, Vol. 22 (1991), pp. 323-346.

[KIN92] Judith King. *X.400 Security.* Computer & Security, No. 11 (1992), pp. 707-710.

[MIT87] C. Mitchell. *Multi-destination Secure Electronic Mail.* The Computer Journal, Vol. 32, No. 1 (1989), pp.13-15.

[MRW89] C. Mitchell, D. Rush and M. Walker. *A Secure Messaging Architecture Implementing the X.400-1988 Security Features.* The Computer Journal, Vol. 33, No. 4 (1989) pp. 290-295.

[MWR89] C. Mitchell, Michael Walker and David Rush. *CCITT/ISO Standards for Secure Message Handling.* IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4 (May, 1989), pp. 517-524.

[MIT92] Chris J. Mitchell. *Authentication Multicast Internet Electronic Mail Massages Using a Bidirectional MAC is Insecure.* IEEE transactions on Computers, Vol. 41, No. 4 (1992), pp. 505-507.

[MUF91] Sead Muftic. *Security in Open Distributed Processing Systems* Computer Security and Information Integrity, IFIP, (1991), pp.333-349.

[RSA78] R.L. Riverst, A. Shamir and L. Adleman, *A method for obtaining digital signature and public key cryptosystem* Communication ACM, Vol. 21, No. 2 (1978), pp. 120-126.

[SAT89] M. Satyanarayanan. *Integrating Security in a Large Distributed System.* ACM Transactions on Computer Systems, Vol. 7, No. 3 (1989), pp. 247-280.

[SIM92] IEEE PRESS, 1992 *Contemporary Cryptology. The science of information integrity.* Chapter 6, pp. 325-373.

[SHE92] Robin L. Sherman. *Distributed System Security.* Computer & Security, No. 11 (1992), pp.24-28.

[TC90] K. W. Tse and K. P. Chow. *Message Encryption in a X.400 Message Handling System.* IEEE Region 10 Conference on Computer and Communication Systems, September 1990, Hong Kong, pp.432-435.

[TSU92] Gene Tsudik. *Message Authentication with One-Way Hash Functions* ACM SIGCOMM Computer Communication Review, Vol. 22, No. 5 (1992), pp. 29-38.

[WU92] Suchun Wu. *MHS Security: A Concise Survey.* Computer Networks and ISDN Systems, Vol. 25 (1992), pp.490-495.

[VOY83] Victor L. Voydock and Stephen T. Kent. *Security Mechanisms in High-Level Network Protocols.* ACM Computing Surveys, Vol. 15, No. 2 (1983), pp. 135-171.