INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa • University of Ottawa

Design and Implementation of Secure Communications for a Distributed Mobile Computing System

By

Zheng Cui

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirement of the degree of

M. A. Sc.
in
Electrical and Computer Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering

School of Information Technology Engineering
Faculty of Engineering
University of Ottawa
Ottawa, Ontario

December, 1999

I hereby declare that I am the sole author of this thesis.

I authorize the University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Zheng Cui

I further authorize the University of Ottawa to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Zheng Cui

Contents

List of Figures

List of Tables

Abstract

Mobile computing, which combines the traditional data network and telecommunication networks, has been a very attractive and active paradigm in the field of network communications. The mobile computing environment not only provides communication between fixed and/or mobile devices, but also seeks to provide anytime, anywhere, personalized services and resource access to its mobile users. Mobile computing offers more flexibility to the mobile users, but it also raises new concerns to the field of *Network Security*, which has been gaining more and more attention for the last two decades.

This thesis introduces a mobile computing system, the Personal Mobility Management System (PMMS), which has been implemented in the Multimedia and Mobile Agent Research Laboratory at University of Ottawa. The PMMS is a mobile computing application based on Agent technology and Blackboard Messaging Server. The thesis captures and analyzes the security weaknesses exposed in the system, and presents feasible approaches to overcome these concerns. The main purpose of the thesis work is to design and implement frameworks to ensure secure communications for the PMMS. Designs that will be discussed in details include user authentication, access control and secure communication framework for the system. Suggestions on future work are also presented at the end of the document.

Acknowledgement

I would like first to thank my supervisor Dr. Ahmed Karmouch for his continuous guidance and support throughout my thesis work. His directions were very helpful to me.

I am grateful to Tom Gray, Serguei Mankovski from Mitel Corporation and Roger Impey from National Research Council of Canada for their valuable feedback and technical suggestions on my work.

I would like to thank Hamid Harroud, Mohamed Ahmed, Magdi Amer, Amin Hooda, Yaoping Wang and many other members of the Multimedia and Mobile Agent Research Lab for their generous helps and wonderful ideas during the implementation stage of my work. Their friendship and kindness also helped to make my stay at the University of Ottawa very enjoyable.

My special thanks are due to my family for their constant support, without which this work would not have been possible.

Last, I would like to acknowledge National Research Council of Canada and CITO for their partial financial support that helped me to complete this master program.

Acronyms

CORBA	Common Object Request Broker Architecture
DASS	Distributed Authentication Security Service
DES	Data Encryption Standard
DIB	Directory Information Base
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
IPSec	Internet Protocol Security
JVM	Java Virtual Machine
L2TP	Layer 2 Tunnel Protocol
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MAC	Message Authentication Code
MD4, MD5	Message Digest 4 (5)
MIME	Multipurpose Internet Mail Extensions
NSA	National Security Agency
OTPS	One Time Password System
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
PPTP	Point to Point Tunneling Protocol
PRNG	Pseudo Random Number Generator
RC2, RC4, RC5	Rivest Cipher 2 (4, 5)
RFC	Request For Comments
RMI	Remote Method Invocation
SET	Secure Electronic Transaction
SHA	Secure Hash Algorithm
SSL	Secure Socket Layer
UML	Unified Modeling Language
VPN	Virtual Private Network
WAN	Wide Area Network

Chapter 1

Security Warfare in Mobile Computing

1.1 Network Security and Mobile Computing

With the development of the computer networks, today's people are becoming more and more dependent on network communications, especially the Internet. Companies have already established their various kinds of home pages and put their attractive ads on the WWW. This does provide a lot of convenience to our modern lives. However, on the other hand, thousands of hackers are lurking around the net, recording your every transmission and trying to take possession of your secrets even your bank account.

Therefore, in network-based communication systems, technologies, tools and procedures concerning security are essential both to assure system continuity and reliability and to protect data and programs from intrusions, modifications, theft and unauthorized disclosure. *Network Security*, consequently, has become one of the most important issues in the field of network communication.

Mobile computing [SPA 95], which combines the traditional data network and telecommunication networks, has been a very attractive and active paradigm in the field of network communications. The mobile computing environment not only provides communication between fixed and/or mobile devices, but also seeks to provide anytime, anywhere, personalized services and resource access to its mobile users. Mobile computing offers more flexibility to the mobile users, but it also raises new issues in the area of network security.

1.2 Security Threats in Mobile Communication

The communication network is the home to many hackers who pose threats to the security of the communications. Due to the dynamic characteristics, a mobile computing system normally has to face the following security threats performed by hackers or malicious users.

- ***Disclosure***

  This is the danger of releasing the content of communication messages exchanged between two or more communication parties. Sometimes data transferred on the network is of confidential nature. For example, a user's credit card number or a file describing a company's expansion strategy. Usually, the nature of the communication (whether it is confidential or not) should be defined locally by the sender prior to the communication. Once the message is on the air, a hacker can easily capture it just by listening to the communication channel. (This attacking technique is called

eavesdropping or ***passive wiretapping***.) So, protecting against the disclosure threat may be required as a matter of local policy.

- ***Modification of Information***

  Some parties may alter in-transit messages rather than just listen to them. This type of attack is referred to as ***active wiretapping***. Since the messages are usually generated by an authorized party or mobile user, they could be modified/altered in such a way as to effect unauthorized operations, including falsifying the value of an object.

- ***Masquerade/Impersonation***

  It is the danger that some operations or services not authorized for some party may be attempted by the party by assuming the identity of another party that has the appropriate authorizations. For instance, an unregistered mobile user may try to gain the access to a network service by pretending to be a registered user.

- ***Message Stream Modification***

  Communication messages may be maliciously re-ordered, delayed or replayed to an extent which is greater than can occur through the natural operations, in order to effect unauthorized operations. A typical example is to capture a successful login stream and replay it in a later time to gain the access to a system. Although the login stream might be encrypted, replaying this stream would be a very effective way to access the system, if it doesn't contain any one-time tokens.

- ***Abuse of Devices***

  A mobile device that belongs to a registered user could be used or abused by some other parties to gain the network access and obtain the services provided by the system for free. The owner of the device will be charged for whatever it costs.

- ***Denial of Services***

  Sometimes, attackers may try to prevent the normal use of communication facilities in order to interrupt communications between some parties. They could destroy the transmission process by various means such as destroying the communication line, disabling the file system, or paralyzing/bombing the server by sending a tremendous amount of garbage information or requests. Such denial-of-service attacks are in many cases indistinguishable from the type of network failures with which any mobile computing system must cope as a matter of course.

- ***Traffic Analysis***

  The attacker can eavesdrop on the data transfer and by analysis of the volume and frequency of the communications, deduct information that can be used in a harmful way to the computer system or users. For instance, the attacker can figure out when there is a high volume of data exchanged between two companies.

- ***Social Engineering***

  This type of threats is very difficult to detect and prevent. The attacker may spend $1000 to bribe an officer for a piece of confidential information. He may dig into the

garbage bin of a hi-tech company to find out the expansion strategy, or pretend to be the system admin claiming that there is a security hole in the company's network, and ask all the employees for their account information in order to recover the system. Statistics shows that this type of attack plays a significant role among all the hacking techniques, and has gained a lot of success. Unfortunately, there is no technical way available to solve this problem. The only way to reduce the risk is to establish strict management policy and educate system users.

1.3 Thesis Objectives and Outline

This thesis focuses on our work involved in a large project supported by Mobile Agent Alliance, which is composed of Mitel Corporation, National Research Council of Canada and Multimedia and Mobile Agent Research Laboratory (MMARL) at University of Ottawa.

More specifically, we participated in the design and development of a mobile computing system prototype called Personal Mobility Management System (PMMS). The PMMS is an agent-based distributed mobile computing application that seeks to provide personalized services and resources access to its nomadic users within a virtual network across different organizations. One of our primary responsibilities in the project is to design and to implement secure communication framework for the PMMS, including user authentication, authorization and secure communication, which are also the main objectives of this thesis.

The remainder of the thesis is organized as follow: Chapter 2 reviews some cryptographic techniques widely used to ensure communication security with examples. Chapter 3 first introduces the PMMS, then captures and analyzes the security weaknesses of the PMMS, next, it spends a lot of effort on presenting the feasible approaches to overcome those security concerns. Some other important security issues such as key management are also discussed in this chapter. Chapter 4 focuses on the implementation issues of the security design. Conclusions of the thesis and suggestions on future work are followed and presented in the last chapter.

1.4 Main Contributions

Our main contributions of this thesis work are the design and implementation of several security frameworks for the PMMS, including Mobile User Authentication Framework, User Authorization/Access Control Framework and Secure Communication Framework. Deploying the PMMS according to these frameworks eliminates some security problems of a mobile computing environment, and reduces the risk of being compromised by hackers. Eventually, the PMMS is able to provide personalized services to registered mobile users over the network in a secure manner, without compromising either confidential user information or system policies and profiles.

Moreover, these security frameworks are very flexible, they can be deployed in a very large scale and in distributed fashion. They also have the capability to support different security protocols (authentication protocols, key exchange protocols, etc) already available. They can be applied to other communication systems as well.

Chapter 2

Cryptography and Network Security

2.1 Important Network Security Terminologies

There are five basic but important security concepts in the territory of network communications as listed below [SIM 92] [SCH 95]:

- **Authentication:** Assurance that the entity at the other end of a communication session really is the one it claims to be, including user authentication and terminal/device authentication.

- **Access Control:** Also called **Authorization,** it is the assurance that an entity is permitted to do what it asks for. Authorization protects against accidental and malicious threats to the system secrecy, authenticity and availability. Furthermore, by providing specific application access control, organizations are now able to grant access to their most sensitive data without compromising the security of the system.

- **Privacy:** Most concerned by the user, this is the assurance that sensitive information is not visible to an eavesdropper. It is usually achieved using encryption. Privacy can range from the protection of all user-transmitted data to specific fields within a message. It also applies to protection of traffic analysis so that the attacker should not be able to observe the source and destination, frequency, real length or other characteristics of the transmitted data.

- **Integrity:** Normally combined with encryption, it is the assurance that the received information is the same as when it was sent. The integrity requirement prevents the transmitted data from being modified, duplicated, reordered or replayed. Integrity can also be applied to the whole transmitted message or selected fields within the message.

- **Accountability:** Also called **Non-repudiation**. It is the assurance that any transaction that takes place can subsequently be proved to have taken place. Both the sender and the receiver agree that the exchange took place. This prevents either the sender or receiver from denying a transmitted message or committed transaction.

2.2 Cryptography in Network Communication

Cryptography [DEN 82] is a field of science concerned with encrypting information. The information, called in the cryptographic terminology **plaintext**, is provided as input to an encrypting algorithm. The algorithm uses certain value, call a **key**, to generate output

ciphertext. The ciphertext cannot be read or modified by any intruder, because they do not have the key, which is required to decipher the message. The target of the transmission uses a matching key, to decrypt the data.

2.2.1 Symmetric (secret key) Cryptosystem

In symmetric cryptosystem [DEN 82] [SIM 92], both the sender and the receiver share a secret key, which is unknown to anyone else. The secret key is critical. Anyone knowing it can decrypt the message. Figure 2.2-1 shows the diagram of a standard symmetric (conventional) cryptosystem.

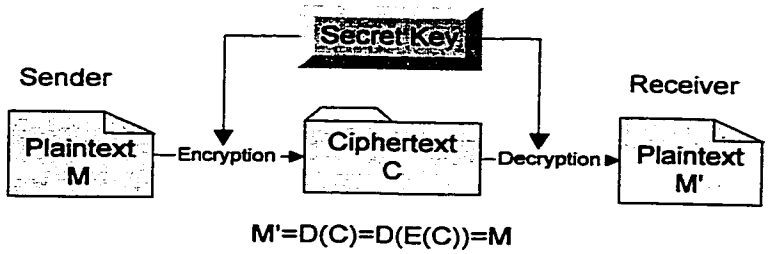


Figure 2.2-1. Conventional symmetric cryptosystem

Note: In this thesis, E(.) and D(.) are used to stand for encryption and decryption operation respectively.

The security of conventional cryptosystem depends on several factors. First, the encryption algorithm must be strong enough so that it is impractical to decrypt the ciphertext given the knowledge of the encryption and decryption algorithms and the

ciphertext. Another factor also the main challenge is how secure the shared key is kept and distributed.

2.2.2 Asymmetric (public key) Cryptosystem

In asymmetric (public key) [DEN 82] [SIM 92] cryptosystem, every entity (sender or receiver) has two keys, one public key and one private key. As they are named, the public key contains no secret and is made public to everyone, while the private key contains secret information that is only known to the owner. Here is how the system operates:

- Public key is used for encryption, while private key is for decryption;
- The sender encrypt the message using the public key of the intended receiver;
- The receiver decrypt the message using his/her own private key;
- No other recipient can decrypt the encrypted message without knowing the private key which belongs to the intended recipient;
- An entity can change his or her keys at any time by publishing the new public key and switching to the corresponding new private key.

Obviously, the algorithm involved in public key cryptosystem is much more complex than that of the secret key system. Figure 2.2-2 shows the diagram.

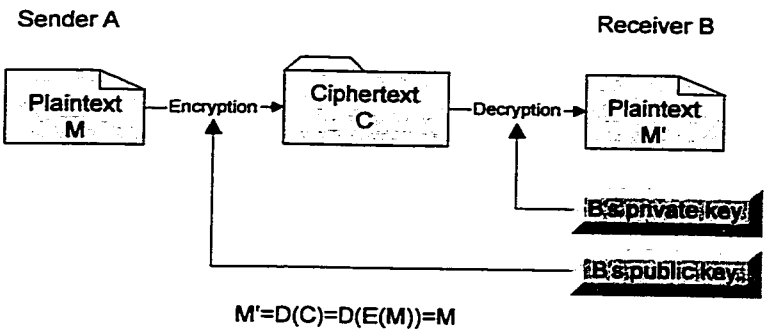


Figure 2.2-2. Public key cryptosystem

A public key cryptosystem can be configured to provide authentication and/or privacy. Figure 2.2-3 shows how public-key encryption is used to provide authentication. Suppose A is going to send a message to B. In order to assure B that the message is in fact from A, A "decrypts" the message using his/her own private key, and B "encrypts" the cipher using A's public key. If B can get a meaningful result, then he/she can be assured that the message is sent by A, since only A could have access to the private key. Moreover, if the message were modified during transmission, B would not be able to recover meaningful result. So, this approach provides authentication cause the message can be verified both in its origin and in its integrity. However, because everybody could recover the original message using A's public key, this configuration doesn't offer privacy.

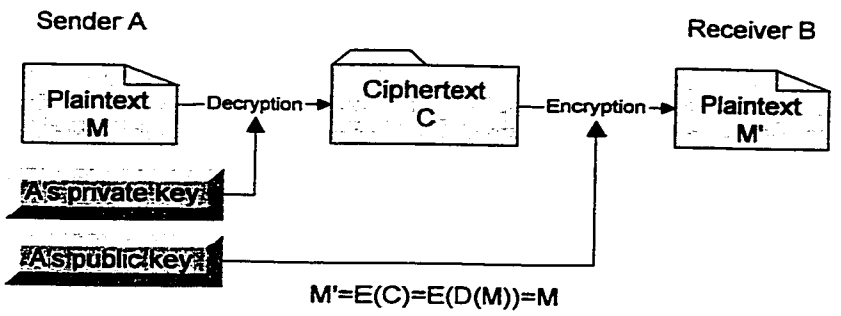


Figure 2.2-3: Authentication using public key cryptosystem

In order to provide both privacy and authentication, we can configure the system as shown in figure 2.2-4. In this case, the message from A to B is first "decrypted" using the A's private key, providing authentication, then immediately encrypted using B's public key, providing privacy. Upon receiving the cipher, B has to reverse this procedure in order to get the original message. One disadvantage of this system is that the public-key algorithms are very complex and time consuming, and they must be performed four times in a row rather than two times.

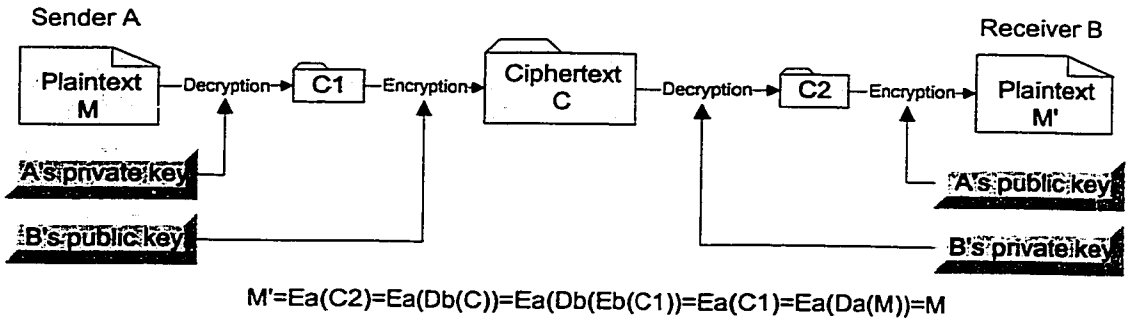


Figure 2.2-4. Authentication and privacy using public key cryptosystem

2.2.3 Secure Hash Functions (Message Digest)

Hash function [DEN 82] [SCH 95] is used to verify the integrity of the data. When we apply Hash transformation to a piece of message, the result is called Message Digest (MD). Hash transformation is a one way function, which has the following properties as shown in Figure 2.2-5: (Assume A and B are different messages.)

- Non-invertable: As stated in the definition, a one way operation can not be reversed in theory.
- Collision-resistant: Given a Hash function, different messages will not produce same message digest.

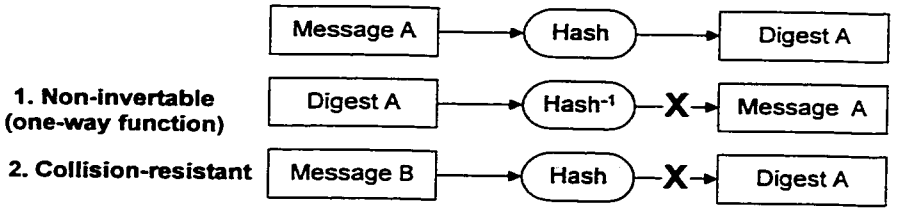


Figure 2.2-5. Properties of a cryptographic Hash function

Authentication can also be achieved by applying Hash function (see Figure 2.2-6). Here is a brief explanation: (Assume both parties know the secret key S)

- On the sender side: S is concatenated with Message, then the extended message is passed to a Hash function to generate digest D. The sender only sends the Message and the digest D to the receiver;
- On the receiver side: The digest D is separated from the Message, and saved. The Message is concatenated with S and the result is passed to the same Hash function to

produce digest D'. If D' differs from the saved digest D, then the receiver knows that the Message has been altered during transmission. Otherwise, he is assured that the Message is from the sender who knows the secret key S, and it hasn't been modified.

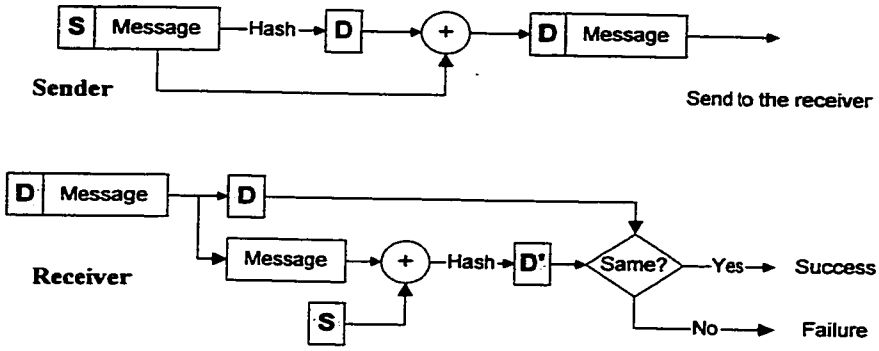


Figure 2.2-6. Authentication using Message Digest

2.3 Authentication Systems and Protocols

In this section, we will concentrate on the systems and protocols for user authentication. This is the most important issue in the system design. It's impossible to address or apply access control and other security mechanisms without authenticating a user.

There are lots of systems and protocols available for authentication purpose. Basically, they can be summarized into the following categories:

2.3.1 Physical Authentication

A user can be identified by means of checking the physical characteristics of the user such as voice, image, fingerprints etc. Also, a user can be verified based on some physical devices that posses the user's confidential information. This type of

authentication is often called *Physical Authentication*. Applied properly, it will provide much stronger security performance than any other software based authentication method (see next). However, systems that employ physical authentication mechanism always require higher computation power, larger storage capability and more complex management procedure. This, in turn, affects the deployment of physical authentication mechanism in a large distributed environment such as the Internet.

2.3.2 Software Authentication

2.3.2.1 Password-based authentication

In most computer networks and distributed systems, protection of resources is achieved by direct login to each host accessed using passwords, with users selecting the passwords and transmitting them in the clear and unprotected. This password-based authentication has several drawbacks, some of them are listed below.

1) Users always tend to select passwords that are not randomly distributed. This problem is well-known and not necessarily related to computer networks and distributed systems.
2) It is not convenient for a user who has several accounts on different hosts to remember a password for each of them, as well as to enter it each time when he or she changes the host. Instead, the user should be known as a single user to the computer network or distributed system as a whole.
3) The transmission of a password itself is exposed to passive eavesdropping and subsequent replay attacks.

Mainly because of the last drawback mentioned above, password-based authentication is not suitable for computer networks and distributed systems. Passwords sent across the network can be easily interpreted then used in a malicious way by attackers.

2.3.2.2 Address-based authentication

One way to overcome the problems of password-based authentication is address-based authentication. Address-based authentication does not rely on sending passwords around the network, but rather assumes that the identity of the source can be inferred based on the network address from where packets originate. The basic idea is that each host stores information that specifies accounts on other hosts that should have access to its resources. So this type of authentication is also known as host authentication.

Note that the idea of trusted hosts is not a general solution to the authentication problem in computer networks and distributed systems. As a matter of fact, trusted hosts can even pose a serious security threat. The point is that host authentication mechanisms can always be defeated, and if an attacker is able to break into an account in a host that is trusted by other hosts, the accounts on other hosts are compromised too.

Depending on the environment, address-based authentication may be more or less secure than sending passwords in the clear. But since it is more convenient, address based authentication is therefore the choice of many computer networks and distributed systems today.

2.3.2.3 Cryptographic authentication

The basic idea of a cryptographic authentication is that a claimant A proves his/her identity to a verifier B by performing a cryptographic operation on a quantity that either both know or B supplies. The cryptographic operation performed by A is based on a cryptographic key. This key can either be a secret key in a symmetric cryptosystem or a private key in a public key infrastructure.

In general, cryptographic authentication can be more reliable than either password-based or address-based authentication. However, designing realistic cryptographic authentication protocols is notoriously difficult, and several published protocols have exhibited substantial or subtle security problems. During the last decade, research efforts have focused on providing tools needed for developing authentication and key distribution protocols with some formal assurance of security.

2.3.3 Protocols and Systems

2.3.3.1 Kerberos

Kerberos [OPP 96] [NEU 94] is a trusted third-party authentication protocol designed for TCP/IP networks. Kerberos performs both authentication and key distribution and is based on symmetric cryptography. Kerberos keeps a database of clients and their secret keys so that it can create messages used to convince one entity of another's identity.

The components of a typical Kerberos system and message flow for authentication are shown in figure 2.3-1. Detail information can be obtained from [OPP 96][NEU 94].

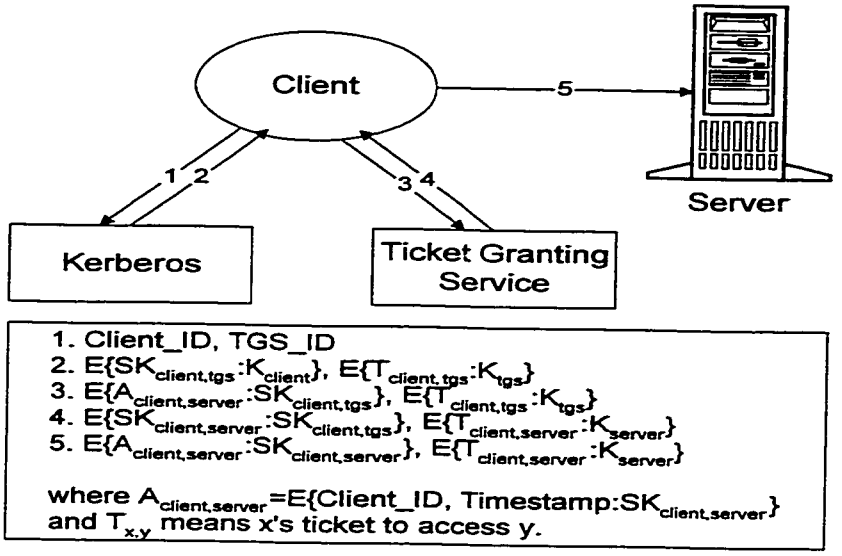


Figure 2.3-1. Kerberos Authentication Procedure

A security weakness of Kerberos is that it may be possible to launch a message replay attack using old authenticators. Although timestamps are employed here to prevent this type of attack, replays can be made during the lifetime (typically several hours) of the ticket. Also, authenticators rely on the synchronization of the network clocks, and this can provide security holes to attackers since most network time protocols are not perfect.

2.3.3.2 One Time Password System (OTPS)

As it is named, a One-Time Password [NET 98] will only be used once, as a result, a compromised OTP will not compromise future communications. This is the main advantage of an OTPS. In a typical OTPS (see Figure 2.3-2), the server records the most recent OTP, the OTP sequence number and the random seed of each user. Figure 2.3-2 also illustrates the basic message flow for the authentication procedure using OTP:

(1) A user provides his/her identifier to the server;

(2) The server retrieves the user's OTP sequence number N and the seed S, sends them back to the user;

(3) The user types in his/her Secure Pass Phrase (SPP), then the OTP generator at the client side will compute the next OTP (OTP_{N-1}) for the user, and transmit it back to the server. OTP_{N-1} is calculated by performing one way hash function on SPP and S for N-1 times;

(4) The server verifies the OTP_{N-1} by comparing its hashed value with OTP_N stored on the server. If and only if the outcome is positive, the user is authenticated, and server updates the OTP as well as the sequence number.

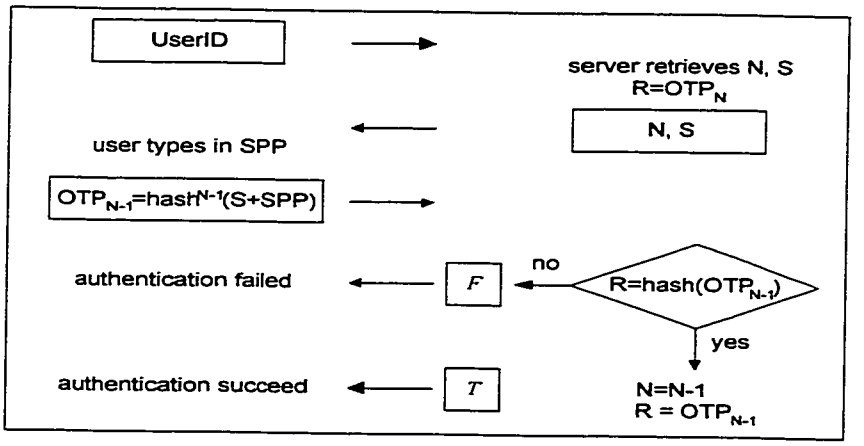


Figure 2.3-2. One Time Password System

One-time Password is primarily designed to counter "replay attack" (see chapter 1). It is based on the hypothesis that it is infeasible to recover the original message from its digest. That is to say it is infeasible to figure out OTP_{N-1} from OTP_N, thus, revealing the user's OTP will not compromise the user's secret. The only secret information is the

user's Secure Pass Phrase (SPP), which is never sent to or kept in the server. This leads to a very important feature of the OTP system: the user's OTP doesn't need to be stored secretly or encrypted. The loss of the previous OTP will not compromise the user's secure pass phrase, neither will it compromise the next OTP. However, this OTP system does not provide privacy of transmitted data, and it does not provide effective protection against most active attacks such as modification of information.

2.3.3.3 Other Authentication Protocols

There are many other protocols designed for authentication. Many of these protocols normally realize authentication using either one way hash function or public key cryptography or both. Besides OTPS, SKID2 and SKID3 [RES 92] are also typical symmetric cryptography authentication protocols based on the concept of hash function. While Distributed Authentication Security Service (DASS) [GAS 89], Secure Socket Layer (SSL) [MAC 96] and Secure Electronic Transaction (SET) [ANS 81] are typical authentication (and key exchange) protocols using public key cryptography.

In general, authentication protocols based on hash function are very efficient and easy to implement, while those using public key infrastructure (PKI) are more secure and more complex. One should choose the proper protocol by analyzing the actual situations and requirements of a system.

2.4 Data Encryption and Encoding Algorithms

2.4.1 Data Encryption Standard (DES) and Triple-DES

DES [ANS 81] is a well known private-key-only (also called single-key or symmetric) encryption method that uses the same secret 56-bit key to encrypt and later decrypt the message. It is a block cipher [DEN 82] algorithm, taking 64-bit input data and generating 64-bit output cipher. Since most input messages are not exact multiple of 64-bit in length, there is usually a short block of zeros or ones or their combination, which is referred as *Padding*, at the end to make it complete.

To help keep DES a secure method of data encryption, the algorithm it uses was designed to be difficult to implement in software (and it therefore runs slowly in software). This makes software-only methods of breaking DES codes undesirable. Moreover, the U.S. government restricts sales of DES hardware, so obtaining DES hardware for the purposes of breaking codes would also be difficult.

However, DES, which was initially designed with a 64-bit key but was changed to 56-bit by the U.S. government's National Security Agency (NSA), is nearing the end of its useful life. Its 56-bit key size is vulnerable to a brute-force attack. In the meantime, low cost DES integrated circuits are available, and hundreds or thousands of them could be (or maybe already have been) built into machines which would be able to apply all the DES ICs in parallel--to more quickly decipher messages. Also, recent advances in differential cryptanalysis and linear cryptanalysis [LAN 94] indicate that DES is vulnerable to other attacks as well.

A very good feature of DES is that many DES blocks can be concatenated to form a large encryption block with longer key length (multiple of 56 bit). Triple-DES [ANS 85] is such an encryption chain comprised of three standard DES blocks (see figure 2.4-1 blow).

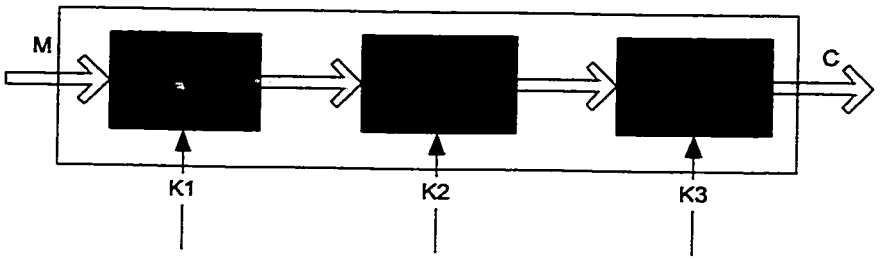


Figure 2.4-1: Triple-DES Encryption

Where M and C stand for Message and Cipher respectively. K1, K2 *and* K3 *are encryption keys each of which is 56 bit in length.*

As the reader may notice, the encryption of Triple-DES is actually composed of two encryption blocks (the first and the last) and a decryption block. The advantage of this configuration is that we can use the same model to implement 56-bit encryption by assigning $K1 = K2 = K3$, 112-bit encryption by assigning $K1 = K2 \neq K3$ and 168-bit encryption if we choose $K1 \neq K2 \neq K3$.

Although the maximum length of the key used for Triple-DES is 168 (56x3) bits, in practice, a 112-bit key is more preferable. The reason for this is obvious: long enough to provide strong protection yet short enough to be computed quickly.

2.4.2 RSA

RSA [RIV 78] [RIV 79] is a patented public key (also called dual-key or asymmetric) data encryption scheme that can provide both encryption and authentication.

The technique was developed at Stanford University in 1977, but was named after three Massachusetts Institute of Technology professors (Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman). They made it a useful system and started a company (RSA Data Security, Inc.) to license the technology and sell software toolkits for application developers so that they can add these capabilities to their software.

Here are the steps and a very simple example to illustrate how RSA works:

- First, choose two large primes P and Q; [for instance: let P=3, Q=5]
- Then, calculate two products: N = P x Q and PHI = (P-1) x (Q-1); [N=15, PHI=8]
- The public key E will be generated so that the greater common divisor of E and PHI is 1. In other words, E is relatively prime with PHI; [E can be 3,5,..., choose E=3]
- N and E are the public keys; [public key: (N=15, E=3)]
- The private key D is the inverse of E modulo PHI, which means D x E = 1 (mod PHI), then D is the private key; [in this example, choose D=3]
- Encrypt the message M to get the cipher C using C=M^E mod N;

  [assume M=2, then C=2^3 mod 15 = 8]
- For the decryption, using M'=C^D mod N to recover the message.

  [M'=8^3 mod 15 = 2 =M]

It is difficult (presumably) to obtain the private key D from the public key (N, E) if P and Q are two large prime numbers. If one could factor N into P and Q, however, then one could obtain the private key D. Thus the security of RSA is related to the assumption that

factoring large prime number is difficult. Factoring large numbers takes more time than factoring smaller numbers. This is why the size of the modulus (N) in RSA determines how secure an actual use of RSA is; the larger the modulus, the longer it would take an attacker to factor, and thus the more resistant it is to attack the RSA.

2.4.3 Base64 Encoding

The Base64 encoding system is described in the Multipurpose Internet Mail Extensions (MIME) standard (RFC 1521). It is intended as a mechanism for converting binary data into a form that can be sent through mail gateways, some of which can only handle 7-bit ASCII data. The result of this conversion is to mask the contents of any text string, but, although it looks as though the data is encrypted, the protection that base64 provides is only an illusion. For example, "cipher" `bm9fc2VjcmV0` can be Base64-decoded back into `no_secret` (for detail, please refer to RFC 1521). In practice, Base64 encoding is mainly used as a converter to convert non-displayable binary data to text strings.

2.4.4 Other Algorithms

2.4.4.1 IDEA

Being part of PGP [ZIM 95], International Data Encryption Algorithm (IDEA) [SIM 92] [LAI 92] has the fame of one of the best and most secure block cipher algorithm available to the public at this time. It is patented and must be licensed for commercial applications.

Like many other block cipher algorithms, IDEA operates on 64-bit plaintext blocks. The key is 128 bits long, and the same algorithm is used for both encryption and decryption.

The design philosophy behind IDEA is one of "mixing operations from different algebraic groups". Three algebraic groups are being mixed, they are XOR, Addition modulo 2^{16} and Multiplication modulo $2^{16}+1$. Although cryptanalysis has made some progress against reduced-round variants, the algorithm still seems very strong.

2.4.4.2 Blowfish

Blowfish [SIM 92] [SCH 94] is a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use. Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Since then it has been analyzed considerably, and it is slowly gaining acceptance as a strong encryption algorithm. The best part is that Blowfish is un-patented and license-free, and is available free for all uses.

Blowfish uses only simple operations: addition, XOR and table lookup. It is fast and optimized for applications where the key doesn't change often, such as a communication link. It is significantly faster than DES when implemented on 32-bit microprocessors with large data caches. However, Blowfish is not suitable for applications with frequent key changes such as packet switching.

2.4.4.3 RC2, RC4 & RC5

These algorithms are designed by Ron Rivest for RSA Data Security Inc. "RC" stands for "Rivest Cipher" obviously.

RC2 is a variable-key-size 64-bit block cipher. According to RSA Data Security Inc., software implementation of RC2 is three times faster than DES. The encryption speed is independent of key size. *"RC2 is not an iterative block cipher, and offers more protection against differential and linear cryptanalysis than other block ciphers which have relied for their security on copying the design of DES "* [ROB 94]. RC2 is proprietary, and its detail has not been published.

RC4 [RIV 92a] is a variable-key-size stream cipher, which encrypts the input data bit by bit. It does not seem to have any small cycles and is highly nonlinear. There are no public cryptanalysis results. Unlike RC2, RC4 was proprietary for the first seven years, but the source code was posted to a mailing list in 1994. It is no longer a trade secret, must be licensed in order to use it in commercial applications. RC4 has been part of many commercial products including Lotus Notes and Oracle Secure SQL.

Both RC2 and RC4 have special export status if the encryption key length is 40 bits or under.

RC5 [RIV 95] is quite new, it is a block cipher with a variety of parameters: block size, key size and number of rounds. It has three operations: XOR, addition and rotation. Actually, RC5 is a family of algorithms. Rivest designates particular implementations of RC5 as RC5-w/r/b, where w is the word (block) size, r is the number of rounds, and b is the length of the key in bytes.

2.5 Virtual Private Network (VPN)

2.5.1 What is VPN?

Basically, VPNs [FEI 97] are private network overlays on a public IP network infrastructure such as the Internet. A VPN provides secured connections between several points, sort of an express lane on the information superhighway, where communications travel along the Internet but are encrypted for security.

At minimum, a VPN should encrypt data over a dynamic connection on a public network to protect the information from being revealed if intercepted. Beyond that basic function, VPN features customarily include tools for authentication, and a limited number provide integrated access control and authorization capabilities.

Typically, a VPN uses the Internet as the transport backbone to establish secure links with business partners, extend communications to regional and isolated offices, and significantly decrease the cost of communications for an increasingly mobile workforce.

2.5.2 Intranet, Internet and Extranet VPNs

There are as many types of VPN implementations [FEI 97] as there are companies taking advantage of the concept's benefits, each with its own specific set of technology requirements. However, VPN configurations can be basically divided into three categories:

Intranet VPNs: between internal corporate departments and branch offices;

Remote Access VPNs: between a corporation and remote or mobile employees;

Extranet VPNs: between a corporation and its strategic partners, customers, and suppliers.

Intranets are defined here as semi-permanent WAN connections over a public network to a branch office. These types of LAN-to-LAN connections are assumed to carry the least security risk because corporations generally trust their branch offices and view them as an extension of the corporate network. In this case, the corporation generally controls both the source and destination nodes. A typical Intranet VPN is composed of encrypted, bi-directional tunnels established between trusted LANs across the network.

Remote Access VPNs between a company and its remote and/or mobile employees have different requirements. Reliability and Quality of Service (QoS) are important, because the employees accessing the VPN are typically limited to slow modem speeds. Additionally, strong authentication is critical to ensure the remote and mobile users' identities in the most accurate and efficient manner possible. On the management side, Remote Access VPNs require centralized management and a high degree of scalability to handle the multitude of VPN links, as well as the vast number of users accessing the VPN.

Extranet VPNs between a company and its strategic partners, customers and suppliers require an open, standards-based solution to ensure interoperability with the various solutions that the business partners might implement. Equally important is traffic control to eliminate bottlenecks at network access points and guarantee swift delivery of and

rapid response times for critical data. A typical secure Extranet VPN is established by creating encrypted unidirectional links from one endpoint to another through a VPN server.

2.5.3 VPN Protocols

The VPN security market is quite young, and standards are still evolving, but a handful of protocols have emerged as the leading choices for building VPNs. Understanding the benefits of each protocol may help clarify the related strengths and weaknesses of different VPN solutions. Although there are many possible security approaches for creating a VPN, the following protocols show the most promise for lasting in the market, whether for the quality of their design or their financial backing. (As shown in Figure 2.5-1).

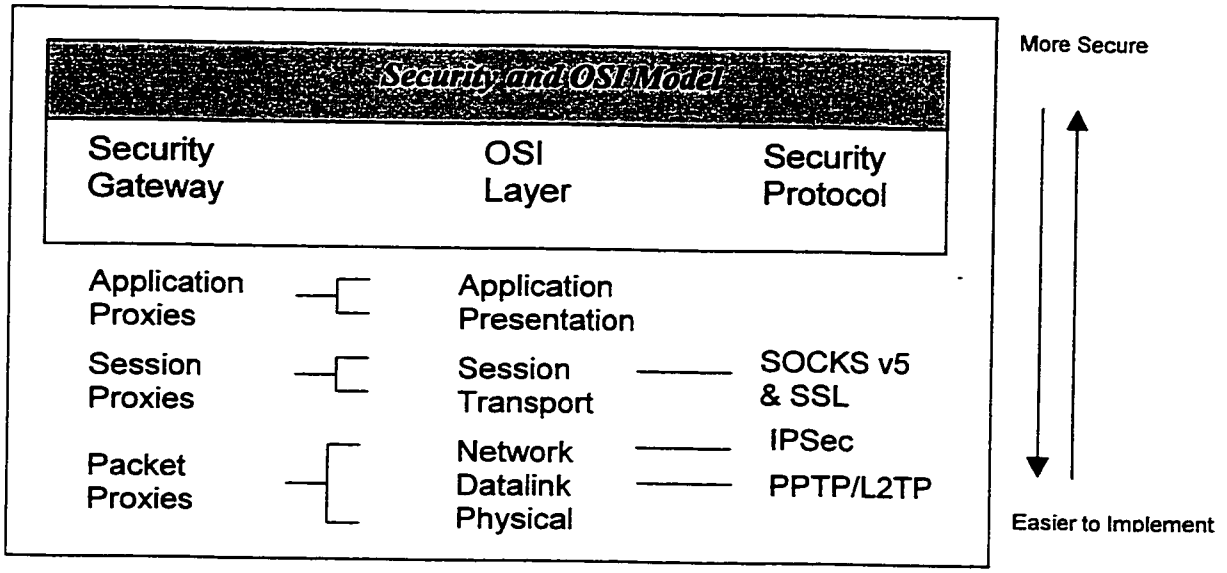


Figure 2.5-1. VPN Protocols and OSI Model

SOCKS v5 [NET 96] was originally approved by the IETF as a standard protocol for authenticated firewall traversal, and, when combined with SSL [MAC 96], it provides the

foundation for building highly secure VPNs that are compatible with any firewall. It is most appropriately applied to VPNs that require the highest degree of security, since its strength is access control.

One of the most widely known VPN security choices is Point-to-Point Tunneling Protocol (PPTP) [MIC 97] from Microsoft. It sits at the datalink layer, which maps approximately to layer 2 of the OSI model. PPTP and its successor, L2TP (Layer 2 Tunnel Protocol by IETF) [NET 99], are seen as tools to extend the current PPP dial-up infrastructure supported by Microsoft, most ISPs, and the remote access hardware vendors. Analysts predict that PPTP and L2TP will play a dominant role in the Internet-based remote access market when security requirements are relatively low.

Internet Protocol Security (IPSec) [FEI 97] has gained a lot of recent attention in the industry. It evolved from the IPv6 movement, and as a standard promoted by the IETF, IPSec will be a broad-based, open solution for VPN security that will facilitate interoperability between VPNs. IPSec can be configured to run in two distinct modes -- tunnel mode and transport mode. Analysts predict that IPSec will be the primary standard for this segment of the VPN market.

Generally speaking, upper layer protocols are usually more secure than the lower layer protocols, but the lower layer protocols are easier to implement than the upper layer protocols. Therefore most VPN providers have decided to support the lower layer protocols rather than the upper layer protocols.

Chapter 3

Security Frameworks Design for the PMMS

3.1 The PMMS and Its Prototype

The Personal Mobility Management System is an application that manages personal mobility in a virtual network and provides personalized services and resources access to its nomadic users over the network across different organizations. The PMMS implements dynamic mapping between a user and the shared devices (or services) available at any location within the network. The dynamic mapping functionality is obtained using the Internet *LDAP directory* [NET 97]. For certain functions in data management, the agent [BRA 97] concept is leveraged. Another inherent aspect of PMMS is universal messaging that deals with the communication needs of mobile users. Messaging is largely addressed by the interaction of autonomous programs (agents), representing users, services, and data resources.

Figure 3.1-1 shows the three-site virtual network scenario. This test bed would allow the exploration of the practical issues of *personal* mobility. A site where *PMMS* is available

is called here an *Enabled Site*. A set of *Enabled Site* forms an *Enabled Region*. A site where *PMMS* is not available is referred to as an *Un-Enabled Site*. Similarly, a set of *Un-Enabled Site* combine to form an *Un-Enabled Region*. Each *Enabled Site* maintains a Directory Information Base (DIB) [NET 97] for its organizational network. Formally, the directory's data model is based on a partial information strategy (nodes in the network contain database for the portion of the network) and each DIB is analogous to a "regional directory" of [AWE 95]. It should be noted in figure 3.1-1, that the three agents are posted to form a virtual network among the sites. These agents are representatives of proxy or interconnection agents that support the mobility of subscribers roaming through heterogeneous environments of these sites.

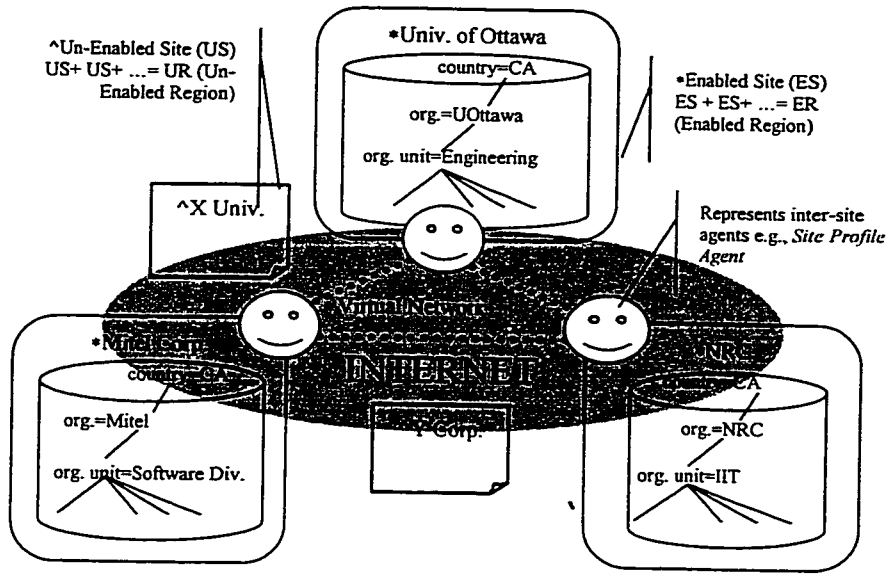


Figure 3.1-1 Three-Site Virtual Network

Each enabled site within the virtual network maintains a three-layered architecture or analogous structure (prototype) as shown in figure 3.1-2.

The first layer is the back end layer containing all the data information such as user profile, service profile and access policy. This layer is only accessible to the middle layer.

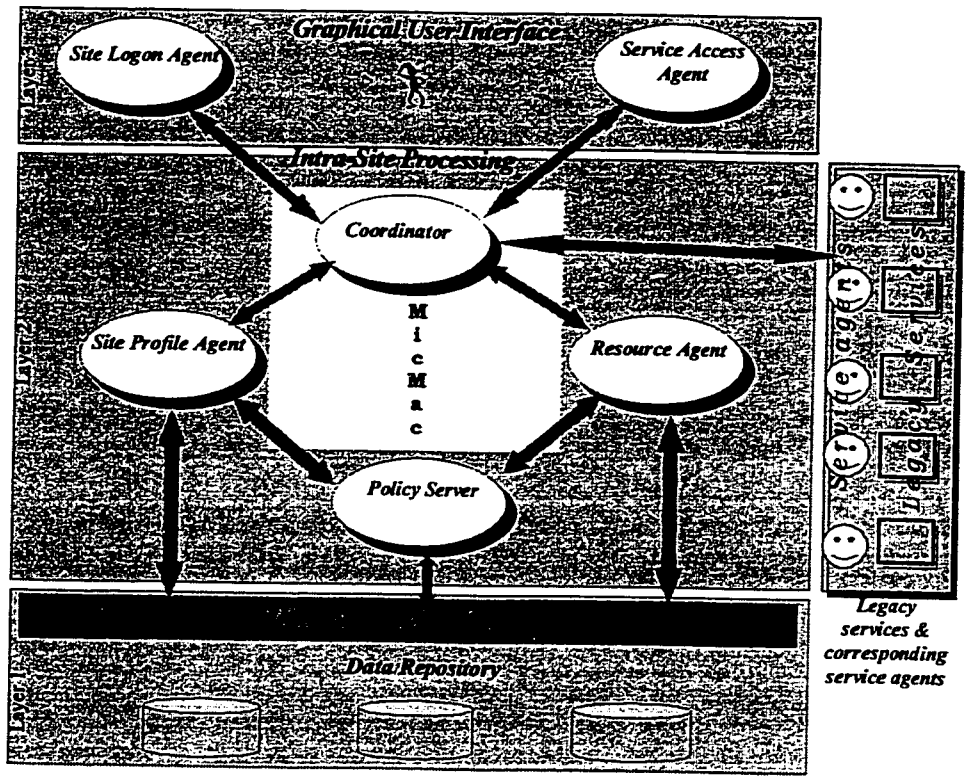


Figure 3.1-2 The Three Layered Architecture of PMMS

Layer 2 is the business logic layer, which deploys all kinds of intra-site and inter-site processing agents:

- **Site Profile Agent** negotiates, with its remote peer (in a user's home site), the user's preferences, and dynamically determines the authorized services for the user, in accordance with the two sets of policies of both concerned sites.

- **Resource Agent** negotiates, with its peer, resources for each requested and authorized service. It prepares also creates an environment for this service. Negotiation between two resource agents determines the availability and accessibility of local resources before processing the service.

- **Coordinator** manages the inter-agent communications. Its central role is to establish communication among two or more agents (locally or remotely), to locate agents and to control their activities when exchanging information. It is also responsible to activate and control appropriate legacy services when a user is authorized to access those services.

- **Policy Server** provides the interface for agents to access the system policy, which contains the private knowledge base (for each site) regarding authorized services, accessible resources and management rules (cost, delay, privileges, etc.) in the form of obligations and authorizations allowed or denied to each agent or user. The content of the policy resides physically in the first layer.

Layer 3 is the front-end interface providing service access to the nomadic users. A mobile user logs in the system via the **User Logon Agent**, which is actually the authentication client of the system. After receiving the information typed in by the user, the User Logon Agent will locate the user in the virtual network and authenticate the user as the one he/she claimed to be. **Service Access Agent** is the interface responsible for proving

dynamically located services to the authenticated user. This interface will not be displayed if the user can not be identified by his/her home site.

3.2 Communication Background — MicMac Server

The PMMS is a message oriented agent-based mobile computing system, where messaging is totally addressed by the interaction between two or more agents. In the PMMS, MicMac [MIT 96] (a blackboard messaging server developed by Mitel Corporation) provides the communication background that carries the message interactions (see Figure 3.1-2 layer 2).

MicMac is a set of software tools, which provides an environment to support hands-on experimentation with multi-agent interaction and negotiation protocols. It allows the creation and the manipulation of one or more **Agoras** (Tuple Space). An Agora/Tuple Space is an open area for exchanging information between agents. A shared Tuple Space for inter-site communication is shown in figure 3.2-1 as an example.

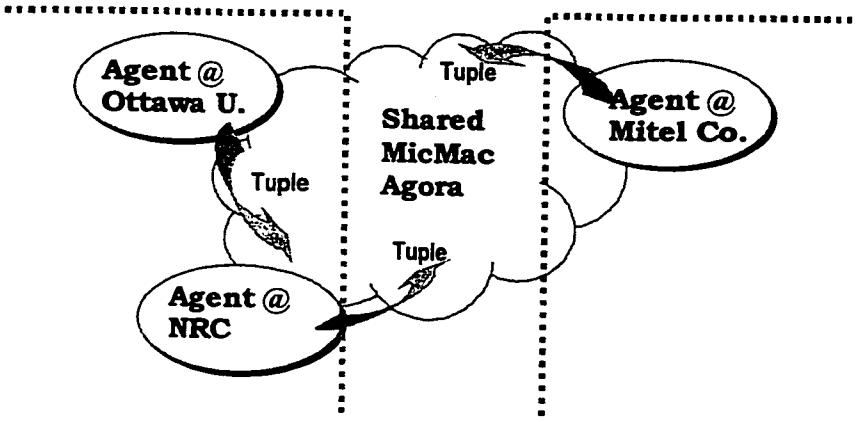


Figure 3.2-1 Inter-Site Communication with Shared Agora

In order to proceed, we need to spend some words on the mechanism of MicMac Agora. An Agora *server* implements a shared memory which stores information as *tuples* (a tuple is an unordered set of fields represented as «key-value» associations); the shared memory is then called an *Agora* or a *Tuple Space*. The server supports access by co-located clients as well as by remote (networked) clients. Network connections do not directly end up at the Agora server itself; actually they are handled by a specialized local Agora client acting as a shared proxy for networked clients. The service interface offered by an Agora server consists of a small and simple set of primitives, which trigger specific processing requests over tuple space content.

Table 3.2-1 Primitive Set of Agora

Primitives	Timeout [a]	Request Description
Post	Yes [b]	Add a tuple to a tuple space
Pick	Yes [c]	Extract a tuple from a tuple space
Peek	Yes [c]	Read a tuplein a tuple space
Cancel	No	Terminate a pending pick/peek request

a. optional temporal constraint (specified as an optional parameter of the primitive)

b. constraint affects tuple persistency

c. constraint affects request persistency

The *post* and *cancel* primitive return control to the caller immediately. The *pick* and *peek* primitives however remain pending at the server until an entry in the tuple space satisfies the tuple template specified by the primitive, an associated temporal constraint takes effect, or a cancel primitive specifying a matching tuple template is issued by another client. Note that the effect of a cancel primitive is not persistent i.e. it is not withheld by a server if no matching pick/peek request exist when it is processed.

Examples:

1) caller Alice sends a message ("Hello") to callee Bob by performing a **post** operation:

   post *Tuple{("sender", "Alice"), ("receiver", "Bob"), ("content", "Hello")}* (Tp 1)

2) in order to receive the message, callee Bob must register himself to the tuple space by launching a **pick** request of the same format as shown below:

   pick *Tuple{("sender", "Alice"), ("receiver", "Bob"), ("content", any)}* (Tp 2)

   or

   pick *Tuple{("sender", any), ("receiver", "Bob"), ("content", any)}* (Tp 3)

In this example, "sender", "receiver" and "content" are the **keys** of each Tuple, while "Alice", "Bob", "Hello" are the corresponding **values**. These **values** could be Tuples as well. There is another important reserved variable called **any**, which can be used as the **value** of any **key** to ignore that particular attribute field. In this case, Tp 2 can extract any message from Alice to Bob, while Tp 3 can extract any message from anyone to Bob, etc. provided that all the messages use the same "sender --- receiver --- content" format. So, both the caller and the callee must agree on a certain type of format (key combination) before they talk. However, if some other agent has the knowledge of this format, it could be able to remove or browse all the messages of this type posted on the Tuple space, this rises several security problems (see next section for detail).

3.3 Security Analysis of the PMMS

Without explicit mechanisms, personal communication over an unsecured channel like the Internet can be viewed or modified, registered users could be impersonated, their accounts could be compromised, and even the mobile devices might be compromised and abused as well (see chapter 1). The PMMS is an agent-based message-oriented mobile computing system based on MicMac. Although the communication mechanism of MicMac is somewhat different from that of the Internet, MicMac is still an open environment and has to face all these security problems.

3.3.1 Security Weakness of MicMac Agora

As we have introduced previously, the security strength of MicMac Agora depends largely on the confidentiality of Tuple format, i.e. the randomness of the keys that constitute the tuple attributes. Using the API of the MicMac Agora, an agent is not able to pick/peek messages exchanged on the Tuple space unless it has the knowledge of the key combination. Both caller and callee have to agree on a certain pattern for their communication session in order to proceed. For instance, they both use "sender --- receiver --- content" tuple format to exchange messages.

There are several security holes in this solution. First, in order to establish secure communication session, all parties involved in the communication should secretly agree on a key combination prior to exchanging messages. This gives chance to attackers. In the mean time, it causes extensive overheads. Second, the key combination must be changed frequently, otherwise, a compromised combination will also compromise future

communication using this key combination. More importantly, if a malicious agent can access the physical memory on the computer where the MicMac server is running, as a result, the complete content of any tuple could be eavesdropped or modified arbitrarily.

Now let's only consider the consequences of a compromised key combination. In this case, messages exchanged on top of MicMac can be recorded, replayed, modified and removed by some entity other than the expected receiver. For instance, assume that the format of the Tuple in a communication session between Alice and Bob is "sender --- receiver --- content" and Mallory happens to know this format, then she can issue the following request to achieve different malicious purposes (see table 3.3-1).

Table 3.3-1 Attacks on MicMac Tuple

Attack	Action
Record/Eavesdropping	**peek** *Tuple{("sender", any), ("receiver", any), ("content", any)}*
Remove/Denial of Service	**pick** *Tuple{("sender", any), ("receiver", any), ("content", any)}*
Bomb/Denial of Service	**post** *Tuple{("sender", "Alice"), ("receiver", "Bob"), ("content", "It's party time!")}*
Message Replay	**peek**, then **post** the original Tuple in a later time
Message Reorder	**pick** all, then post those messages in a different order
Modify/Active Wiretapping	**pick**, change the *values*, then **post** the modified version

Although in order to launch any of the above requests requires the knowledge of the Tuple format between Alice and Bob, it is not an impossible task for Mallory. In fact, Alice and Bob have to negotiate the format before they exchange the actual information or just use the default "sender --- receiver --- content" format. Of course Alice can send a secured e-mail or a certified mail to Bob about the format of their next communication, however, such a system is not fully automatic and practical. It requires human

intervention in order to change the program from time to time to fit different communication formats. Another way is to negotiate the format just using the default "sender --- receiver --- content" form. This is automatic, but Mallory has the chance to get (**peek**) the format then launch attacks. We will see how to overcome this limitation and ensure secure communication in the coming section.

3.3.2 User Authentication and Authorization

Before a user accesses the system services and resources, his/her identity must be verified. This is the job of user authentication process. *User Authentication* is the mechanism used to make sure that the user who is accessing the system really is the one he or she claims to be. It establishes the identity of the person accessing the system and eliminates the possibility of unauthorized access.

Basically, users can be identified using either physical or software authentication mechanism (see chapter 2). In the PMMS, all the communications are presented by Tuples rather than packages. IP address or network address is not a built attribute for a Tuple, furthermore, even if we specify an attribute termed "IP" or "Network Address", it is much easier to modify the value of this attribute in a Tuple than modify the address header in a TCP/IP package. Thus, we've chosen to use cryptographic mechanism in our current system for mobile user authentication.

When a mobile user comes to a visiting location, he/she is going to use the resources at the visiting site to access the services, this may cause conflicts with the visiting site

policies. Protection mechanisms must be considered to ensure that the use of these resources will not compromise the system security of the visiting location. *User Authorization*, also called *User Access Control*, is the procedure to ensure that the user is allowed to do what he/she asks for. It protects against accidental and malicious threats to the system secrecy, authenticity and availability. Furthermore, by providing specific application access control, organizations are now able to grant access to their most sensitive data without compromising the security of the system. Access control requires user authentication in advance. User Authorization and authentication constitute the kernel of personal security in a mobile computing system.

3.3.3 Terminal Security Issues in Mobile Computing

Another interesting yet important issue in a mobile computing system is *Terminal Security*. A particular mobile device (terminal) should be only accessible to a particular mobile user or a certain category of mobile users. The system must ensure that users can not use any mobile terminals not authorized to them. Furthermore, the mobile computing system should verify if a given mobile device is allowed to access the system (*Terminal Authentication*) and what privilege is applicable to this device (*Terminal Authorization*). To accomplish terminal authentication and authorization, normally, a special kind of information called *hardware token* is usually used. By reading the hardware token stored/installed on a particular mobile device, the server is able to verify the identity of the device and map the appropriate access rights and mobility services to it.

In our current implementation of PMMS, all the mobile terminals are network access enabled computers, which are statically located in the Enabled Locations. We delegate the terminal authentication and authorization procedures to the enabled site a terminal belongs to, and trust the *Coordinators* of the enabled site. Each enabled site could maintain a profile containing information about all the terminals, assign a unique ID and a serial number to each terminal, and store this information in the profile. It is recommended that this profile also include platform information of the terminal such as the type of the browser and the type of its operating system. Access policies for each terminal should also be defined for authorization and mission control.

3.4 Security Frameworks Design for the PMMS

In this section, we present our security frameworks developed for current PMMS implementation.

3.4.1 Secure Communication Framework for MicMac

In order to ensure secure communication on top of MicMac Agora, first, we need to overcome the security weakness of the MicMac Agora as we analyzed previously. This is very essential to the entire system. Based on the concept of Virtual Private Network (VPN), an information fortress model (as shown in figure 3.4-1) has been developed as the communication framework of the system based on MicMac Agora.

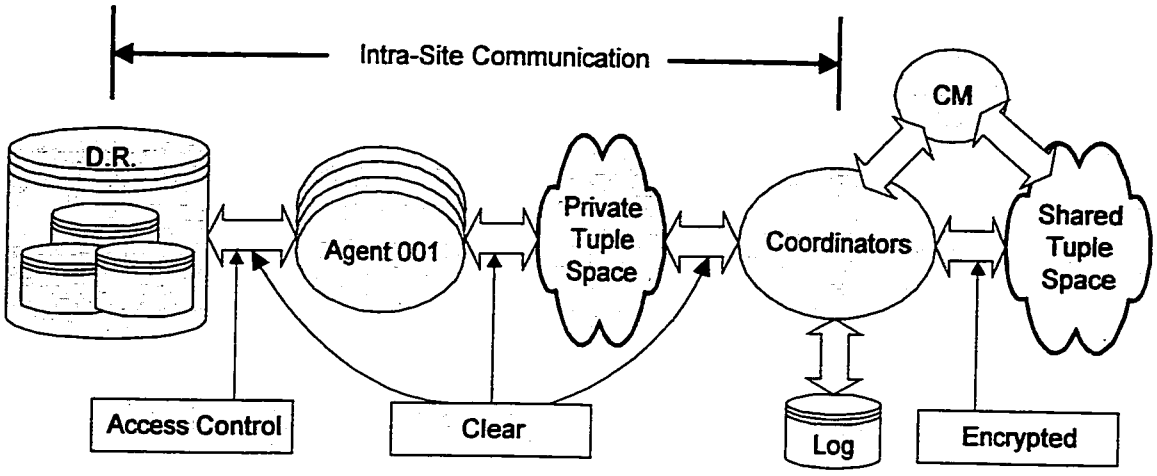


Figure 3.4-1 Secure Communication Framework

As illustrated in figure 3.4-1, the kernel of the framework is called **Coordinator**. This is the software component that manages the inter-agent communications. Coordinator resides in each site/organization where MicMac server is enabled, and its central role is to route communication among two or more agents (locally or remotely), to locate agents and to control their activities when exchanging information. Coordinators behave very much like "software routers". They form a "virtual firewall" separating inter-site communications from intra-site communications.

Two types of Agoras (Tuple Spaces) are established at each site to handle inter-agent communications. All the outgoing messages are encrypted then posted to the *Shared Tuple Space* known to all the sites/organizations, while the intra-site communication messages are exchanged on the *Private Tuple Space* known only to the local agents. Local message exchange might or might not be encrypted. If encrypted, the local encryption keys should be completely irrelevant to those used by coordinators. If the site

has strong and robust protection of its memory access, intra-site communication could be in the clear form.

This framework allows the system to ensure secure communication between sites independently of agents. As a result, agents could be developed without worrying about the inter-site communication privacy. It is the responsibility of the coordinators to implement inter-site data encryption, decryption and other security operations such as digital signature. This feature makes it very easy to deploy this architecture to a very large scale. All we need to do is to properly install/configure one copy of coordinator for each site.

Coordinator Manager (CM) is a software component responsible for initializing and managing site coordinators. There is only one CM at each site. The identity/name of the site CM is the only public information to the outside world and it is globally unique. One pair of coordinators is created/initiated for every inter-site communication session by the CMs of the two sites based on successful user/agent authentication. This framework also enables the Coordinators to screen the sender's information (sender screening) such as the caller's ID of each outgoing message. This feature is particular important in countering *Traffic Analysis* (chapter 1) attacks performed on the MicMac Tuple Space.

To illustrate how this framework works, let's consider an example where Alice from Mitel sends a "Hello" message to Bob at NRC (figure 3.4-2 shows the message flow).

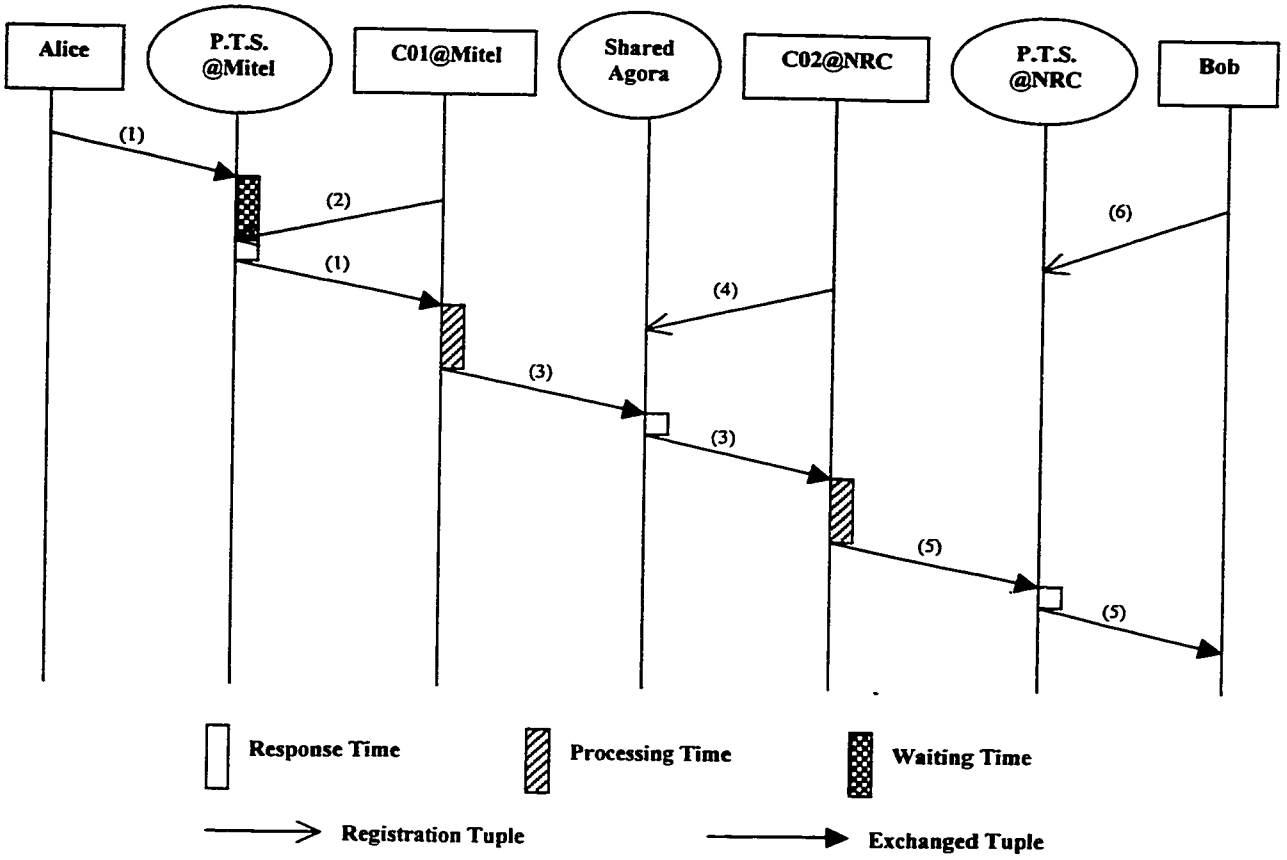


Figure 3.4-2 A Message Flow Example based on the Secure Communication Framework

According to the framework, first, Alice **posts** a tuple to the Private Tuple Sapce at Mitel:

(1) Tuple{("sender", "Aclice@Mitel"), ("receiver", "C01@Mitel"), ("forwardTo", "Bob@NRC"), ("content", "Hello")}

The coordinator "C01@Mitel" will **pick** up this tuple by listening to the Private Tuple Space of:

(2) Tuple{("sender", any), ("receiver", "C01@Mitel"), ("forwardTo", any), ("content", any)}

Then, the coordinator will convert this tuple to another, and **post** it to the Shared Tuple Space:

(3) Tuple{("sender", "C01@Mitel"), ("receiver", "C02@NRC"), ("content", aTuple)}

Where aTuple = Tuple{("sender", "Aclice@Mitel"), ("receiver", "Bob@NRC"), ("content", "Hello")}

Next, the coordinator called "C02@NRC" will **pick** up this tuple by registering the following request at the Shared Tuple Space:

(4) Tuple{("sender", "C01@Mitel"), ("receiver", "C02@NRC"), ("content", any)}

After stripping off the unwanted headers, C02@NRC forwards the content to Bob via the Private Tuple Space at NRC in form of Tuple:

(5) Tuple{("sender", "C02@NRC"), ("receiver", "Bob@NRC"), ("content", aTuple)}

Finally, Bob could retrieve the message form the Private Tuple Space by sending a **pick** request of:

(6) Tuple{("sender", "C02@NRC"), ("receiver", "Bob@NRC"), ("content", any)}

Now, Bob can parse the Tuple and extract the value of *content*, recover the message "Hello" from Alice.

If both coordinators have agreed on one secret key, tuples on the Shared Tuple Space such as *(3)* can be encrypted completely or partially using this secret key:

Partial encryption:

(a) Tuple{("sender", "C01@Mitel"), ("receiver", "C02@NRC"), ("content", E(aTuple))}

(b) Tuple{(E("sender"), "C01@Mitel"), (E("receiver"), "C02@NRC"), (E("content"), aTuple)}

Full encryption:

(c) Tuple{(E("sender"), E("C01@Mitel")), (E("receiver"), E("C02@NRC")), (E("content"), E(aTuple))}

Notation: *E("string")* means to encrypt a string, which could be a key or a value. *E(Tuple)* means to encrypt all the keys and values of the Tuple.

The first expression *(a)* only encrypts the value of the content field, so Mallory may still be able to remove this Tuple before Bob does or use the same format sending tons of messages to Bob. On the other hand, the last expression *(c)* is a little bit tedious and time consuming. From the security point of view, if we could protect the physical memory of the MicMac server from being scanned by unauthorized agents (normally we can), full encryption *(c)* doesn't offer obvious improvements than *(b)* does.

However, the last option *(c)* enables an agent to hide sensitive information inside the content of the embedded tuple so that other unexpected agents as well as the Coordinators that established the session could not be able to browse it. Moreover, the coordinators of the session could utilize the outer fields to include some information for other network services like routing or QoS. This control information should be encrypted in order to counter *Traffic Analysis* attacks. The question is should it be encrypted using the same key as the one for the embedded tuple? Of course NOT, otherwise, the coordinator

(system) would be able to perform decryption on the embedded tuple and see the original message. So, in this case, we need two different session keys, one (**outer key**) for control information, the other (**inner key**) for embedded tuple (or real data).

Fact: Most of the authentication and key exchange protocols generate only one session key in the end. We can take advantage of these traditional yet reliable protocols to establish the **inner key**, while the **outer key** could be computed out of the inner key using one way Hash function (see chapter 2 for Hash function). One advantage of this solution is that the coordinators can only en/decrypt the control information, while the end users (caller and callee) are able to en/decrypt the entire tuple.

Now, let's look back to the previous example, this time, we assume that the **inner key** has been created. Please note that the two coordinators of the session only have the knowledge about the **outer key**, while Alice and Bob have the knowledge about both keys since they can compute out the outer key using hash function.

Note: In this example, *Eo("M")* means to encrypt string M using the **outer key**, *Ei("M")* means to encrypt M with the **inner key**.

First, Alice **posts** the following encrypted tuple to the Private Tuple Space.

(1) Tuple{(Eo("sender"), Eo("Aclice@Mitel")), (Eo("receiver"), Eo("C01@Mitel")), (Eo("forwardTo"), Eo("Bob@NRC")), (Eo("content"), Ei("Hello"))}

The coordinator "C01@Mitel" will **pick** up this tuple by listening the tuples of the following format on the Private Tuple Space:

(2) Tuple{(Eo("sender"), any)), (Eo("receiver"), "C01@Mitel")), (Eo("forwardTo"), any)), (Eo("content"), any))}

Then, the coordinator will convert this tuple, and **post** it to the *Shared Tuple Space*:

(3) Tuple{(Eo("sender"), Eo("C01@Mitel")), (Eo("receiver"), Eo("C02@NRC")), (Eo("content"), aTuple)}

Where aTuple = Tuple{(Eo("sender"), Eo("Aclice@Mitel")), (Eo("receiver"), Eo("Bob@NRC")), (Eo("content"), Ei("Hello"))}

Next, the coordinator called "C02@NRC" will **pick** up this tuple by registering the following request at the Shared Tuple Space:

(4) Tuple{(Eo("sender"), Eo("C01@Mitel")), (Eo("receiver"), Eo("C02@NRC")), (Eo("content"), any)}

After stripping off the unwanted headers, C02@NRC forwards the content to Bob via the Private Tuple Space at NRC in form of the Tuple shown below:

(5) Tuple{(Eo("sender"), Eo("C02@NRC")), (Eo("receiver"), Eo("Bob@NRC")), (Eo("content"), aTuple)}

Finally, Bob could retrieve the message form the Private Tuple Space by sending a **pick** request like:

(6) Tuple{(Eo("sender"), Eo("C02@NRC")), (Eo("receiver"), Eo("Bob@NRC")), (Eo("content"), any)}

Now, Bob can parse the Tuple and extract the value of *content*, decrypt then recover the message "Hello" from Alice using the session key. We will discuss how to establish the session key in the following section.

3.4.2 User Authentication Framework

Since one random key needs to be created prior to the establishment of the secure communication session, proper authentication and key exchange protocol should be applied to the system. We developed an authentication framework (figure 3.4-3) that is capable of supporting various kinds of authentication and key exchange protocols.

As shown in the figure, the caller, say Alice, initiates the request (1) and sends it to the visiting site CM (Mitel_CM) via the private tuple space. Mitel_CM picks up the request (2), locates the receiver's (Bob's) home site (NRC), and then transmits the request to NRC (3). Upon receiving the calling request (4), the site CM at NRC (NRC_CM) transfers the request to the callee Bob via the private tuple space at NRC ((5) and (6)). Before establishing the connection, NRC_CM should activate the Authentication Agent to authenticate the caller. The agent proposes an authentication and key exchange protocol and sends it back to Mitel_CM together with some parameters (optional)

through the dotted virtual path. Mitel_CM decides whether or not to activate the visiting site Authentication Agent according to the local site security policy. Let's assume that the proposed authentication and key exchange protocol is allowed and supported at Mitel site. Then, the two Authentication Agents will follow the schema to mutually authenticate the caller and the callee by verifying their identity against the site's private data repository (D. R.). Note: The Alice and Bob could be either system users or agents.

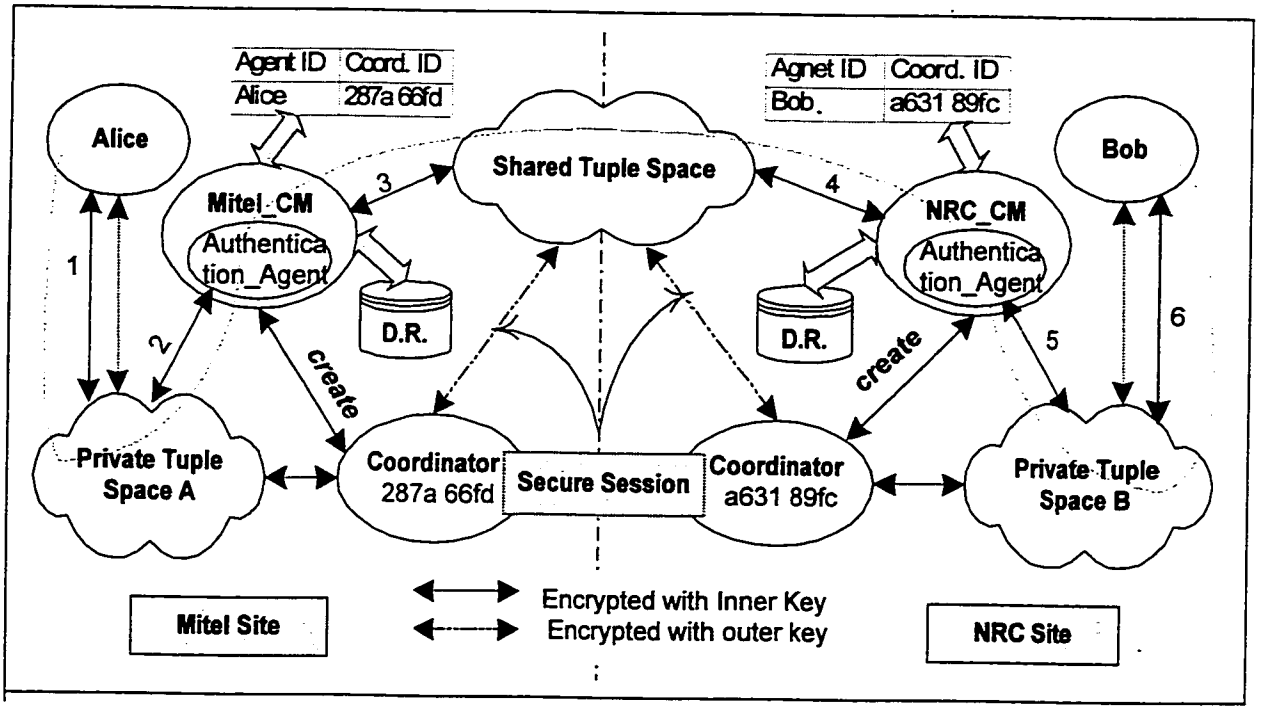


Figure 3.4-3 Mobile User Authentication Framework

After successful authentication and key exchange, the two Authentication Agents will agree on three pieces of secret information including one session key and two dynamic identifiers. Among them, the session key (**inner key**) is passed to the caller (Alice) and the callee (Bob), while the hashed value of the session key (**outer key**) and the two

identifiers are passed to the two Coordinator Managers, which, in turn, will use these information to create two coordinators. The creation of the two coordinators initiates the secure communication session between Alice and Bob. The hashed value (**outer key**) assigned to the coordinators is used to en/decrypt all the **keys** of the tuples as well as the **values** of outer fields used for control purpose. Alice and Bob then use the **inner key** to en/decrypt the embedded tuple.

Another important aspect of the framework is that each site CM also keeps a dynamic table containing the IDs of all the on-line (active) agents or users and the IDs of their corresponding coordinators. Using this table, the site CM is able to detect possible *Impersonation* attacks. If we define a unique ID for each agent and user, it is easy for the system to make sure that at any given time, there is no more than one session established for a particular agent or user.

Furthermore, without authentication or without positive result of the authentication procedure, communication session will not be initialized for anyone since no coordinator will be created. So, the only possible way to break an established session is to guess out the two random IDs (64 bit each) of the coordinators and the session key (112 bit for Triple-DES [ANS 85] encryption). This is considered extremely difficult and computationally infeasible.

Due to the characteristics of agents, the framework shown in figure 3.4-3 can be used for virtually all kinds of mutual authentication situations including user -- server/system

authentication, server -- server authentication and user -- user authentication. Consider the PMMS as an example, the authentication procedure is between a particular nomadic user and his/her home site system. In this case, one of the agents will be or stand for the user, the other will be or stand for the system.

3.4.3 Authentication Protocol in Use

In our current implementation of the PMMS, we developed an authentication protocol based on One Time Password (OTP) system (see chapter 2). Figure 3.4-4 depicts the authentication protocol used in our implementation. It is named Mobile Authentication Protocol (MAuth for short) and based on the original OTP system. Several improvements have been made to overcome the security weaknesses discussed previously in chapter 2.

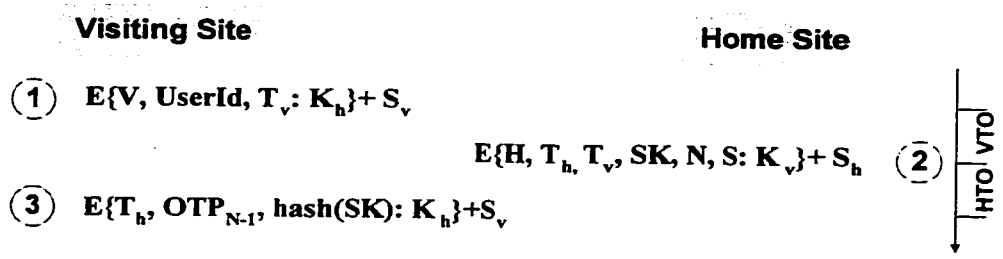


Figure 3.4-4 The Mobile Authentication Protocol

The MAuth protocol assumes that the communicating sites share with each other their encryption keys. The encryption key could be either the public key in asymmetric cryptosystem or secret key in symmetric cryptosystem. The corresponding decryption key (either private key or same secret key) is used to sign and decrypt the message.

According to the protocol, (1) a mobile user first sends the name of the visiting site V, user ID and visiting site timestamp T_v to the home site. This message is first encrypted

using home site's encryption key K_h, then signed by the visiting site (S_v). (2) The home site decrypts the message, retrieves the OTP sequence number N and relating seed S of the acclaiming user from the local database. Then, it generates a random session key SK, and sends T_v, N, S, SK back to the visiting site together with the home site's name H and timestamp T_h. The entire message is encrypted using the visiting site's encryption key K_v, and signed by the home site (S_h). (3) After decrypting and verifying the second message, visiting site will ask the user for his/her SPP, then compute the next one time password OTP_{N-1}. Next, the visiting site will send this OTP_{N-1}, T_h, and the hashed value of SK back to the home site. Again, this message is encrypted and signed as well. If the home site is able to decrypt the message and verify OTP_{N-1} and SK, the user's identity is verified and the session key is initiated.

This protocol overcomes the security problems in the original OTP system. The use of encryption and digital signature is to protect the communication secrecy and detect information modification. Timestamps are also used to introduce uniqueness and some randomness to the exchanged messages. In the same time, both sites setup their Timeouts (VTO for Visiting site Timeout, and HTO for Home site Timeout), if either of them can not receive the other's reply within the specified timeout, the whole authentication procedure will stop immediately. Actually we can use the built in Timeout mechanism of MicMac Agora (see section 3.2) to achieve this functionality. This time monitoring mechanism is useful for detecting network failures, which might be caused by *denial of service* attack.

The MAuth protocol does not specify the encryption (E) and signature (S) algorithms. So the protocol could be implemented using different E-S combinations. The following table gives some examples of E-S combinations. This protocol could also support Public Key Infrastructure (PKI) [SIM 92] [SCH 95] such as RSA [RIV 79]. Our current system implements the last E-S combination shown in the table with 112-bit encryption key and 160-bit signature.

Table 3.4-1 E-S Combination for Mauth Protocol

Encryption Algorithm E	Signature Function S
DES-CBC [ANS 81]	MD5 [RIV 92c]
DES-CBC	SHA [NAT 94]
Triple-DES [ANS 85]	SHA

3.4.4 Access Control Framework

Once the identity of a mobile user has been verified, the system will determine exactly what services and resources are authorized to the user and can be accessed by this user over the network. Furthermore, while presenting the authorized services, the system needs to verify the availability of the minimum resources required executing these services. This procedure is the access control mechanism of the system.

Access control requires a well-protected access policy that specifies the access right for each user and the agents. Some access policies can be reflected by profiles such as service profile and resource profile. In the PMMS, each profile contains three basic types of information, they are descriptive information, mobility information and security information. A must-have component of the security information is the Access Requirement Table (ART). The ART may have different forms in different situations.

Table 3.4-2 depicts two ARTs, where the latter (concrete) one is actually the transformation of the former (abstract) one at the run time.

Table 3.4-2 Examples of Access Requirement Table

Requirement ID	Policy ID	Description
Rqr0001	Policy_0001	Verify Terminal
Rqr0002	Policy_0004	Verify Time Slot
Rqr0003	Polivy_0037	Verify Credit

Requirement ID	Attribute	Value
Rqr0001	Teminal_IP	138.100.23.39
Rqr0002	Timestamp	<1999.05.12
Rqr0003	Credit	>=500

As an example, we consider the scenario of Remote Service Invocation (RSI). In this scenario, an authenticated user is going to use the devices/resources dynamically assigned at the visiting location to access remote services authorized at the home site. After authentication, a pair of coordinators is created with the secret session key, then the user's preference will be negotiated according to the user's profile, and all the authorized services will be displayed to the authenticated user.

The RSI scenario (as shown in figure 3.4-5) proceeds as follow (process may vary with different case):

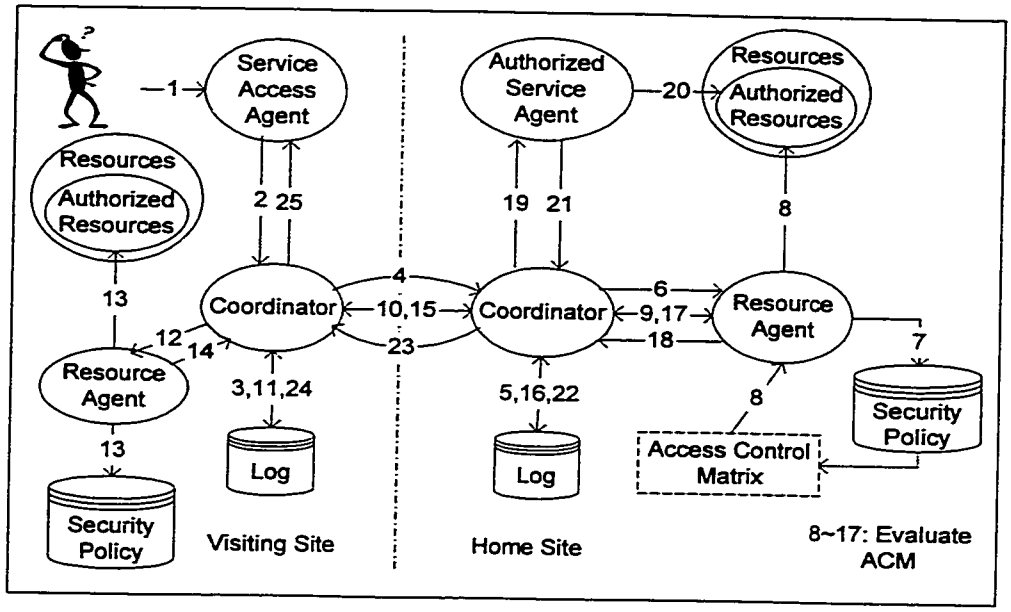


Figure 3.4-5 Remote Service Invocation Framework

(1) The user selects a service from the user interface. (2) Service Access Agent prepares the request for the user and sends it to the site coordinator. (3) The site coordinator updates the log file for the user. (4) The coordinator encrypts the request and sends it to the user's home site. (5) The home site coordinator decrypts the request and updates the user's log file at home. (6) The home site coordinator initiates a Resource Agent to handle the service request. (7) The Resource Agent constructs the dynamic Access Control Matrix (ACM) for this particular request according to the access policies. (8) The Resource Agent evaluates the ACM by checking the availability of the required resources and satisfaction of the associated ARTs for this request. (*How the Resource Agent evaluates the ACM is another interesting topic of the system, which exceeds the scope of this thesis.*) (9) Resource Agent informs the Coordinator about the evaluation result. If the requested service requires devices/resources at the visiting location, analogous ACM evaluation will be performed at the visiting site (10)→(18). (19) The home site

coordinator activates corresponding Service Agent for this request based upon positive evaluation result. (20) The Authorized Service Agent executes. (21) The Service Agent generates unique Reference Number for this service and sends it to the coordinator for tracking purpose. (22)→(25) The reference number is delivered back to the user together with a service report. The user accesses the request service via the interface.

An example of ACM is given in table 3.4-3, where V (x) means to verify the value of attribute(s) x according to the corresponding ART. E, R, W and D stand for Execute, Read, Write and Delete respectively. This table illustrates that in order to provide the requested service, three components will be used, namely, ServiceAgent_03, Resource_01 and Resource_12. The table also outlines the requirements to access each of the three components, the right of the user and the ownership of the user to each component.

Table 3.4-3 An Example of Access Control Matrix

	ServiceAgent_03	Resource_01	Resource_12
Action	V(Time, Space, Bandwidth)	V(Location, Credit)	
Rights	E	R	R,W,D
Ownership	Share	Share	Own

3.4.5 Communication Privacy

To protect the confidentiality of the messages (especially the user sensitive data) exchanged on the MicMac Agora, data encryption is a very effective method. It is a compulsory requirement in our system that all the enabled sites in the virtual environment must support at least 56-bit DES [ANS 81] encryption. However, longer bit encryption

algorithm is strongly recommended. In fact, our system framework is algorithm independent, which means that the framework can support any encryption function. We have chosen 112 bit Triple-DES [ANS 85] encryption in our implementation.

Integrity checking is another important technique to ensure secure communications by means of detecting possible modifications of information. To achieve this, a special type of information called Digital Signature (see chapter 2), is attached to the original message. One can verifies the integrity of the message by performing the same signature function on the message and compares the result with the attached digital signature. In symmetric cryptosystem, digital signature can be computed using one way hash function such as MD4 [RIV 92b] MD5 [RIV 92c] and SHA [NAT 94]. Our system framework supports all standard signature functions.

We suggest that the tuples used for inter-site communication follow a certain format, which should not be changed frequently. Figure 3.4-6 shows a graphical representation of the tuple format.

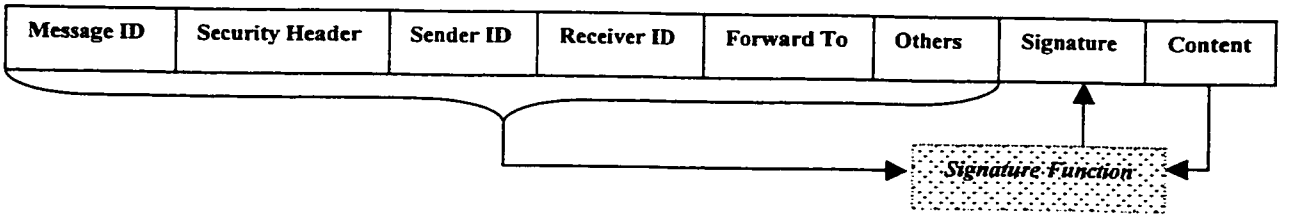


Figure 3.4-6 Inter-Site Communication Tuple Format

The *Message ID* specifies unique sequence number as well as the type of the message. *Security Header* contains information about the security settings of the current

communication session, including the names of the encryption algorithm, the signature function and some parameters. An example of security header is "Triple_DES-112 SHA-80". *Sender ID* and *Receiver ID* specify the expected sender and receiver of the tuple (could be the names of the site CMs during the initialization period or the dynamic IDs of the coordinators constituting the secure session). *Forward To* is used when an agent sends a message to another one in different site via coordinators. *Signature* field contains the digital signature of the entire tuple. *Content* field is used to carry real communication data or embedded tuple. For additional control information, *Others* field could be used. Moreover, there could be more than one *Others* field if needed.

During the initialization (authentication and key exchange) phase, attributes are not encrypted, and some of them might not be used such as *Security Header*, *Others* and *Signature*. After secure session has been established, all the attributes are used and encrypted

3.4.6 Complete Message Flow for a Secure Communication Session

Figure 3.4-7 illustrates the entire process of establishing a secure communication session for Alice and Bob, and exchanging of messages between them. In this example, we assume that both waiting time and response time for all the exchanged tuples are neglectable. This means that a posted tuple will be removed from the tuple space immediately, which indicates that the corresponding pickup request should be registered at the tuple space prior to the arrival of the posted tuple.

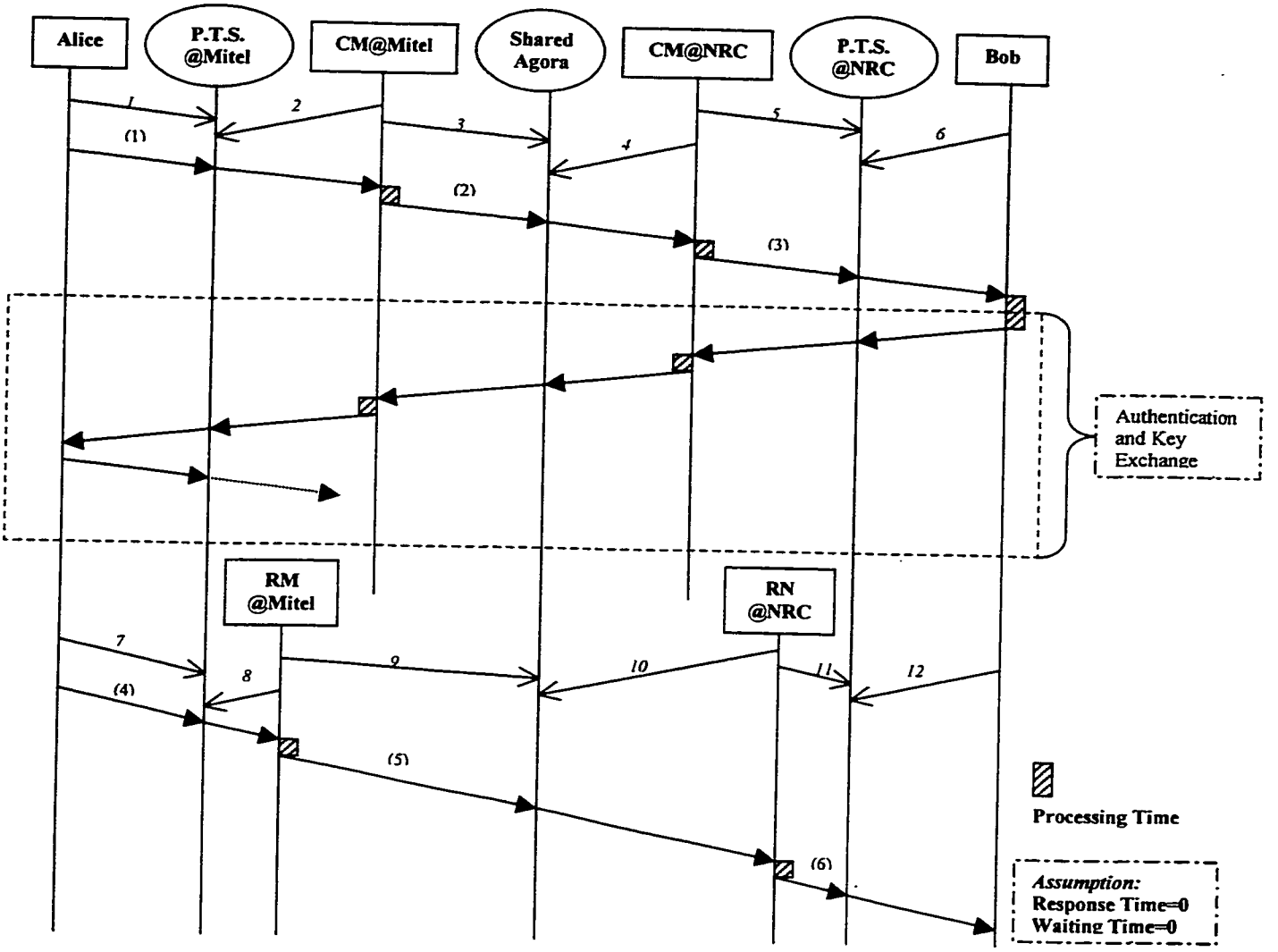


Figure 3.4-7 Complete Message Flow of a Communication Session

According to the assumption, there are six pre-registered pickup requests for all the three tuple spaces: (Please refer to figure 3.4-6 for the format of the inter-site tuples, and we use the same format for intra-site tuples for simplicity.)

1.

any	basic	CM@Mitel	Alice@Mitel	null	any	any	any

2.

any	basic	any	CM@Mitel	any	any	any	any

3.

any	basic	any	CM@Mitel	null	any	any	any

4.

any	basic	any	CM@NRC	null	any	any	any

5.

any	basic	any	CM@NRC	any	any	any	any

6.

any	basic	CM@NRC	Bob@NRC	null	any	any	any

A text version of a given tuple (the first, for instance) is:

Tuple {("Message ID", any), ("Security Header", "basic"), ("Sender ID", "CM@Mitel"), ("Receiver ID", "Alice@Mitel"), ("Forward To", null), ("Others", any), ("Signature", any), ("content", any)}

The value "basic" can be defined as "null-0-sha-80", which means the tuple is not encrypted but digitally signed with Secure Hash Algorithm by default, and the signature contained in the tuple is the first 80 bits of the SHA output/digest.

Moreover, in order to carry binary data in a tuple, we need to convert them into text stream. Base64 Encoding (see chapter 2) as defined in MIME is employed in our system as the only one encoding algorithm.

Tuple (1) through (3) are the initial tuples requesting the authentication process between Alice and Bob:

(1)

0001	basic	Alice@Mitel	CM@Mitel	Bob@NRC	null	sig1	Hello

Where sig1 stands for the actual encoded signature stream of this tuple.

(2)

0002	basic	CM@Mitel	CM@NRC	Bob@NRC	null	sig2	Tuple 1

Where sig2 is the signature of this tuple, and Tuple 1 is an embedded tuple defined as: *Tuple {("Message ID", "0001"), ("Sender ID", "Alice@Mitel"), ("Receiver ID", "Bob@NRC"), ("content", "Hello")}*

(3)

0003	basic	CM@NRC	Bob@NRC	null	null	sig3	Tuple 1

Where sig3 is the signature for tuple (3), and Tuple 1 is the same one as descibed above in tuple (2).

Before posting any tuple to the tuple space, the creator/sender of the tuple is responsible for calculating the digital signature for the entire tuple. On the other hand, right after removing a tuple from the tuple space, the receiver should verify the integrity of the tuple by checking the value of the signature field. Had the tuple been changed during transmission, the receiver would discard the message or ask the sender to re-send.

Once Bob recovers the Hello message, the mutual authentication process begins. Actually, the authentication agent inside the site CM (see figure 4) would be responsible for the task. Here, we skip this authentication process, since the system is able to support most kinds of authentication and key exchange protocols.

After authentication and key exchange, the two parties will agree on a security setting (**SecSet**) for the coming session, and the following random numbers should be ready:

a. An **inner key (Ki)**, which is the result of the key exchange.

b. Two dynamic IDs (**RM** and **RN**), which will be used to create dynamic coordinators for the session.

c. An **outer key** (**Ko**), which is the hashed value of the **inner key**.

Ki will be passed to Alice and Bob by the corresponding authentication agent respectively. **RM**, **RN** and **Ko** are used as parameters when creating the coordinator instances: **RM** as the identifier for the one at Mitel and **RN** for the one at NRC.

After the establishment of the communication session between Alice and Bob, Alice, Bob and the two coordinators will immediately register to the tuple spaces for any incoming messages. There are six (7 ~ 12) such requests in total (see figure 3.4-7 for detail).

7.

any	SecSet	RM@Mitel	Alice@Mitel	null	any	any	any

The grey background means that the keys/attributes of the tuple are encrypted using **Ko**. As an example, the text version for the above tuple is:

Tuple {(Eo("Message ID"), any), (Eo("Security Header"), "SecSet"), (Eo("Sender ID"), "RM@Mitel"), (Eo("Receiver ID"), "Alice@Mitel"), (Eo("Forward To"), null), (Eo("Others"), any), (Eo("Signature"), any), (Eo("content"), any)}

8.

any	SecSet	Alice@Mitel	RM@Mitel	Bob@NRC	any	any	any

Here, the value for the field "Forward To" (Bob@NRC) is optional, cause RM@Mitel will only communicate with RN@NRC, which will forward the message to Bob only.

9.

any	SecSet	RN@NRC	RM@Mitel	Alice@Mitel	any	any	any

10.

any	SecSet	RM@Mitel	RN@NRC	Bob@NRC	any	any	any

11.

any	SecSet	Bob@NRC	RN@NRC	Alice@Mitel	any	any	any

12.

any	SecSet	RN@NRC	Bob@NRC	null	any	any	any

Because of the randomness of the **outer key (Ko)**, which is used to encrypt the attributes, it's infeasible for other agents to re-generate any of the above requests.

Now, Alice can send her message to Bob in a secret way. First, she encrypts the message **(Msg)** using the **inner key (Ki)**, and assigns the result of the encryption **(Ei(Msg))** as the value of the content (actually it's **Eo("content")**) field.

(4)

0004	SecSet	Alice@Mitel	RM@Mitel	Bob@NRC	null	sig4	Ei(Msg)

Then this tuple will be picked up by RM@Mitel, then forwarded to RN@NRC in the following tuple:

(5)

0005	SecSet	RM@Mitel	RN@NRC	Bob@NRC	null	sig5	Ei(Msg)

Note: Here, although embedded tuple could be used in the content field, we don't really need to use it, since Bob will know this message comes from Alice once he receives the following one from RN@NRC.

(6)

0006	SecSet	RN@NRC	Bob@NRC	null	null	sig6	Ei(Msg)

After Bob gets this tuple, he can decrypt the value of the content field **(Ei(Msg))** using the **inner key (Ki)** then recover the message **Msg** from Alice.

3.4.7 Key Management

In the real world, key management [DEN 82] [SIM 92] [SCH 95], which covers key generation, key distribution, key storage, key deletion and etc., is the hardest part of cryptography. Designing secure cryptographic algorithms and protocols isn't easy, keeping the keys secret is much harder. Keys must be protected to the same degree as the data they encrypt, and they should be changed regularly. Using a strong encryption algorithm (such as Triple-DES) only is not enough, especially when the key is chosen poorly.

In the PMMS, we employ Java's Secure Random Number Generator [JAV 99], which generates a secure random object that implements the specified Pseudo Random Number Generator (PRNG) algorithm. By default, the PRNG algorithm is SHA1PRNG, a pseudo-random number generation (PRNG) algorithm supplied by the SUN Microsystems. This implementation follows the IEEE P1363 [IEE 99] standard, Appendix G.7: "Expansion of source bits", and uses SHA1 as the foundation of the PRNG. It computes the SHA1 [NAT 94] hash over a true-random seed value concatenated with a 64-bit counter, which is incremented by 1 for each operation. From the 160-bit SHA1 output, only 64 bits are used. This type of random number is used for coordinator IDs, agent IDs and session keys. Furthermore, these IDs and keys are only valid for a single logon, which is called a "session" in the PMMS.

The choice of a user's secure pass phrase, which is used for user authentication in our implementation, is another important issue of key management. We setup the following policy on the user's choice of pass phrase:

1) The length of the pass phrase must be no less than 12 characters
2) The length of the pass phrase should not exceed 64 characters
3) Each character is selected from the set of 95 Printable Characters
4) The pass phrase must contain at least one capital character and one non-alphabetic character
5) The pass phrase should contain numeric character(s)
6) The initial sequence number should be no less than 256 in decimal

If properly selected, the chance of a pass phrase being cracked is less than 95^{-12}, which is approximately equivalent to 2^{-79}. Assume we have 1 million super computers, each of which is capable of checking 1 million phrases per second, working in parallel, it still needs to take about 10 thousand years in average to break a 12 byte long phrase. And the time needed to break the phrase increases approximately by a factor of 100 for every one-byte increment in the key length.

A user's secure pass phrase will not be stored in the D.R. on the server, instead, the server only keeps the most recent OTP, the associated sequence number N and random seed M of the user (see chapter 2). The OTP, N and M are only accessible to the authorized user (owner) and the administrator of the system. The user's secure pass phrase is not recoverable, since it is infeasible to invert the one way hash function. However, it is changeable, if a user forgets his/her pass phrase, he/she can contact the administrator directly to change it. Moreover, a pass phrase has its lifetime, when it is going to expire,

the system will ask the user to update the pass phrase automatically. In our implementation, we initiate the sequence number to 256, this limits the use of same pass phrase for no more than 256 times/logons.

3.4.8 Profile Management using LDAP

One important component of the PMMS is so called Data Repository, which keeps all kinds of data needed for the system including the data that specifies the personal information, data that specifies the services and resources and data specifies various types of policies of the system. In the PMMS, at each enabled location, all the profiles, namely user profile, resource profile and service profile, are stored in an LDAP X.500 [NET 97] directory information base in a hierarchical fashion (as shown in figure 3.4-8).

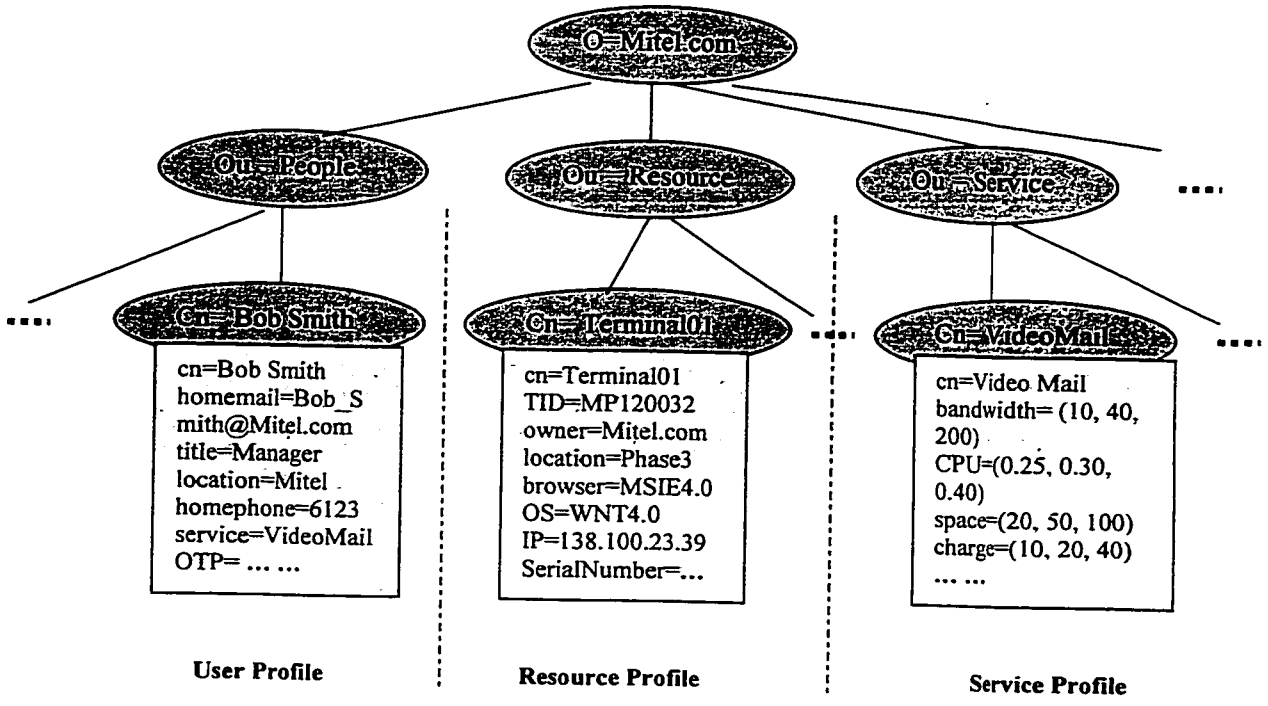


Figure 3.4-8 Profile Management using LDAP

Developed at the University of Michigan at Ann Arbor in conjunction with the Internet Engineering Task Force (IETF), LDAP (Lightweight Directory Access Protocol) is a protocol for accessing and managing directory services. It is a protocol defining a directory service and access to that service. LDAP is based on a client-server model. LDAP servers provide the directory service, and LDAP clients use the directory service to access entries and attributes.

A **directory** consists of entries containing descriptive information. For example, a directory may contain entries describing people or network resources, such as printers or fax machines. The descriptive information is stored in the attributes of the entry. Each attribute describes a specific type of information. For example, attributes describing a person might include the person's name (common name, or cn), telephone number, and email address.

```
cn: Zheng Cui
mail: zcui@sol.genie.uottawa.ca
telephoneNumber: 562-5800 x. 6313
roomNumber: CBY B502
```

A **directory service** is a distributed database application designed to manage the entries and attributes in a directory. A directory service also makes the entries and attributes available to users and other applications. A user might use the directory service to look up someone's telephone number. Another application might use the directory service to retrieve a list of email addresses.

Because LDAP is intended to be a global directory service, data is organized hierarchically, starting at a root and branching down into individual entries. At the top level of the hierarchy, entries represent larger organizations. Under these larger organizations in the hierarchy might be entries for smaller organizations. The hierarchy might end with entries for individual people or resources.

In the PMMS, the system profiles (as shown in figure 3.4-8) contain essential information for user authentication, terminal authentication and access control. For instance, a user's *homemail* in the User Profile is used to locate the user globally, while a terminal's *SerialNumber* in the Resource Profile is used to authenticate the mobile terminal, etc. The Resource Agent in the system prototype will construct ACM according to the information specified by these profiles. The management of these profiles is a very interesting topic, some discussion on this can be found in [HOO 98a][HOO 98b].

Chapter 4

Implementation

4.1 Introduction to Java Security

The Java ™ programming language [JAV 99] [FLA 99] has been warmly accepted by the community of software developers and Internet content providers. Internet users benefit from access to secure, platform independent applications that can come from anywhere on the network. Software developers who create applications in Java benefit by developing code only once, with no need to modify applications to every software and hardware platform.

We've chosen Java as the developing language for the PMMS not only because of the feature of "write once, run anywhere" [FLA 99] [HAR 97], but also because of the concept of Java Security, which mainly includes two aspects:

- Provide the Java platform (primarily through JDK) as a secure, ready-built platform on which to run Java-enabled applications in a secure fashion.

- Provide security tools and services implemented in the Java programming language that enable a wider range of security-sensitive applications.

4.1.1 Java Sandbox Model

The original Java security model [OAK 98] provided by the Java platform is known as the sandbox model, which exists in order to provide a very restricted environment in which to run untrusted code obtained from the open network. The essence of the sandbox model is that local code is trusted to have full access to vital system resources (such as the file system), while downloaded remote code (an applet) is not trusted and can access only the limited resources provided inside the sandbox. This sandbox model is illustrated in the figure below.

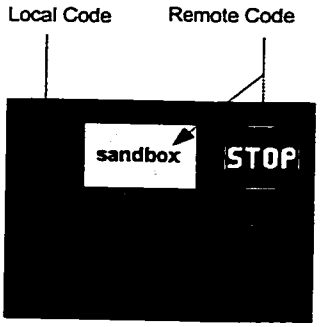


Figure 4.1-1: JVM Sandbox Security Model

Overall security is enforced through a number of mechanisms. First of all, the language is designed to be type-safe and easy to use. Java language features such as automatic memory management, garbage collection, and range checking on strings and arrays are examples of how the language helps the programmer to write safe code. Second, compilers and a bytecode verifier ensure that only legitimate Java bytecodes are executed. The bytecode verifier, together with the Java Virtual Machine (JVM),

guarantees language safety at run time. Moreover, a classloader defines a local name space, which can be used to ensure that an untrusted applet cannot interfere with the running of other programs. Finally, access to crucial system resources is mediated by the Java Virtual Machine and is checked in advance by a SecurityManager class that restricts the actions of a piece of untrusted code to the bare minimum.

4.1.2 Applet and Signed Applet

JDK 1.1 introduced the concept of a "signed applet" [FLA 99] [OAK 98], as illustrated by the figure below. A correctly digitally signed applet is treated as if it is trusted local code if the signature key is recognized as trusted by the end system that receives the applet. Signed applets, together with their signatures, are delivered in the JAR (Java Archive) format. In JDK 1.1, unsigned applets still run in the sandbox.

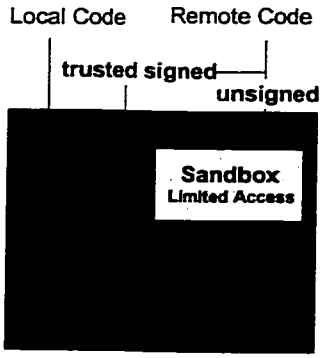


Figure 4.1-2: Trusted and unsigned code in JVM

4.1.3 Improved Sandbox Model

In the new security architecture in JDK 1.2 (illustrated in the figure 4.1-3), there is no longer a built-in concept that all local code is trusted. Instead, local code (e.g., non-system code, application packages installed on the local file system) is subjected to the

same security control as applets, although it is possible, if desired, to declare that the policy on local code (or remote code) be the most liberal, thus enabling such code to effectively run as totally trusted. The same principle applies to signed applets and any Java application.

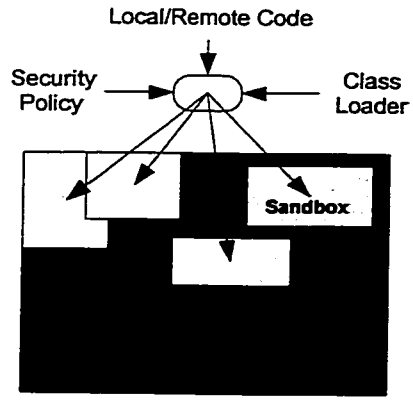


Figure 4.1-3: Improved JVM Sandbox Model

4.2 System Configuration for the PMMS

Currently, the PMMS contains three enabled locations, each of which is represented by a personal computer, namely Mitel Corporation, National Research Council of Canada and University of Ottawa respectively. On each of these computers, the PMMS prototype (see chapter 3) has been installed and configured. In order to simplify the system, each of these three sites maintains their own LDAP directory for all the system profiles. In our current implementation, we just use one separate PC to maintain all the profiles for all the three enabled sites. Since we are using LDAP X.500 directory, it really doesn't matter how the directory information bases are physically distributed. Meanwhile, each site may have several services available to their users (normally employees). The services currently implemented in our system include E-Mail Forwarder, Video Mail and Key-Frame Abstractor.

The current system has been implemented on Windows environment, specifically Windows NT 4.0, using Java technology including Java Applet, Servlet and Java Web Server. The LDAP directory server we adopted is Netscape Directory Server 3.1.

4.3 Implementation Issues of User Authentication

4.3.1 Authentication Client

The Authentication client can be decomposed to 6 parts as shown in the following figure.

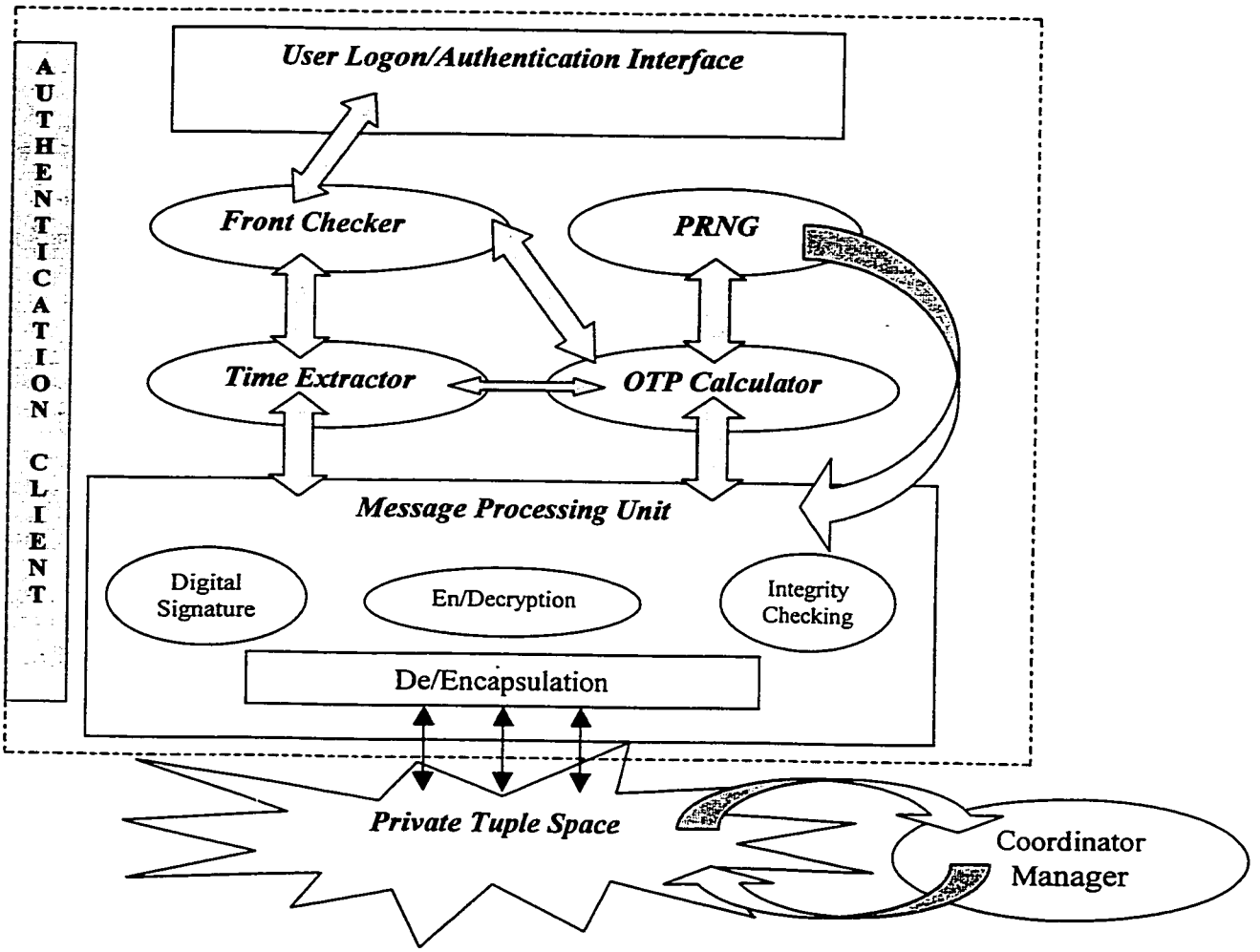


Figure 4.3-1: Authentication Client Architecture

- User Logon/Authentication Interface

This interface will be displayed every time when a user is trying to access the system. The user is asked to type in his/her home site email address and his/her secure pass phrase. These two pieces of information is used for user authentication.

- Front Checker

This is the component that performs front-end (client side) form validation, whose primary objective is to reduce unnecessary network traffic. The formats of the email address and the secure pass phrase are verified solely by the Front Checker component in the client, the user will be asked to re-type/correct the information if the verification comes negative.

- Time Extractor

This component provides client-side system clock (timestamp) upon request.

- Pseudo Random Number Generator (PRNG)

Used to generate coordinator IDs and session keys, PRNG is one of the most important parts of the authentication client. The randomness of a number will effect directly on the security performance of the protocol. Using a strong encryption algorithm doesn't necessarily provide strong protection. In the PMMS, the PRNG algorithm is SHA1PRNG, a pseudo-random number generation algorithm supplied by the SUN

provider (see chapter 3) and implemented inside Java core API (see JDK documentation for SecureRandom class).

- OTP Calculator

It is the functional unit that computes the user's One Time Password (see chapter 2) according to the following formula:

$$OTP_N = hash^N(SPP + Seed)$$

where SPP stands for Secure Pass Phrase, and Seed is the random seed initialized for the user, N is the OTP sequence number.

The kernel of this component is the one way hash function such as SHA and MD5. We support both of them in our implementation.

- Message Processing Unit (MPU)

As the heart of the authentication client, MPU employs data encryption and digital signature in order to meet the security requirements of the Mauth (see chapter 3) protocol. The following capabilities should be accomplished by this component:

1) Verification of Sender's Identity
2) Data Integrity Checking
3) Data Encryption and Decryption
4) Timestamp Monitoring
5) Digital Signature Calculation
6) Message De/Encapsulation
7) Exception Handling

4.3.2 Authentication Server

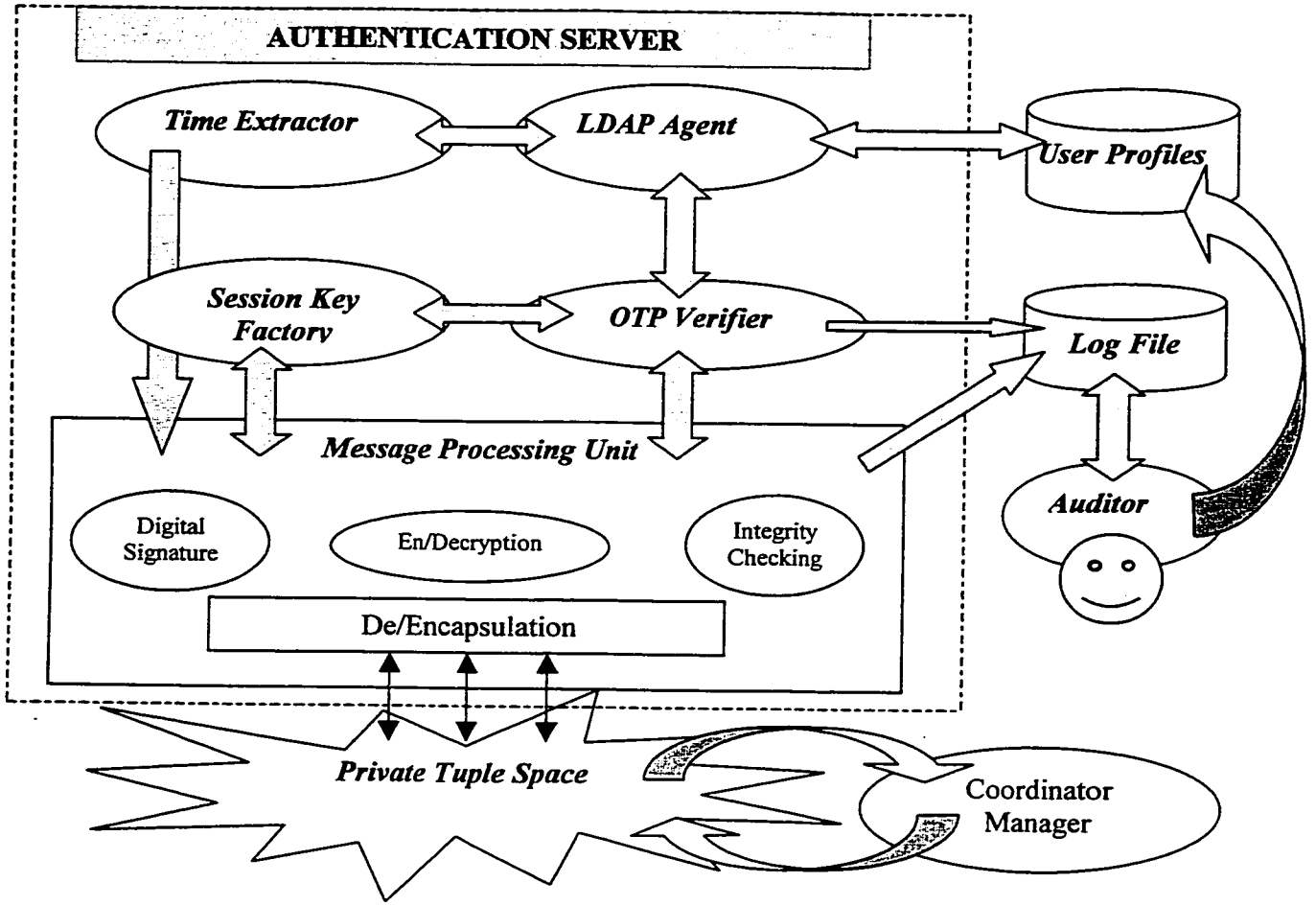


Figure 4.3-2: Authentication Server Architecture

The authentication server contains some components similar with those in the authentication client, such as MPU and Time Extractor. Besides, it maintains other important components essential for the authentication mechanism.

- LDAP Agent

This component is the interface with the back-end LDAP directory information base, which keeps all the profiles of the system. In the authentication process, this agent

verifies the existence of a user against the DIB, retrieves the user's information needed for authentication and updates the user's information after successful authentication.

- OTP Verifier

Responsible for verifying the user's authenticity, this component calculates the message digest of the user's current OTP (sent from the client) and compares the result with the previous one stored in the user's profile. If the comparison comes positive, OTP Verifier will notify the LDAP agent to update the user's profile, it will also send a request to the Session Key Factory to generate a random session key for the user. It is authorized to update the log file stored on the server of the user's home site.

- Session Key Factory

Like the PRNG of the authentication client, the SKF generates session key (112 bits long) for the communication session of the authenticated user. In addition, it generates another shorter (64 bits long) random number used as the ID of the home site coordinator instance for this communication session.

4.3.3 Message Flows of User Authentication

In this section, we will discuss three most important scenarios of the user authentication process: successful authentication, non-existent user and authentication failure. All the scenarios here are represented using *Sequence Diagram* in UML [BOO 98] notation.

4.3.3.1 Successful Authentication

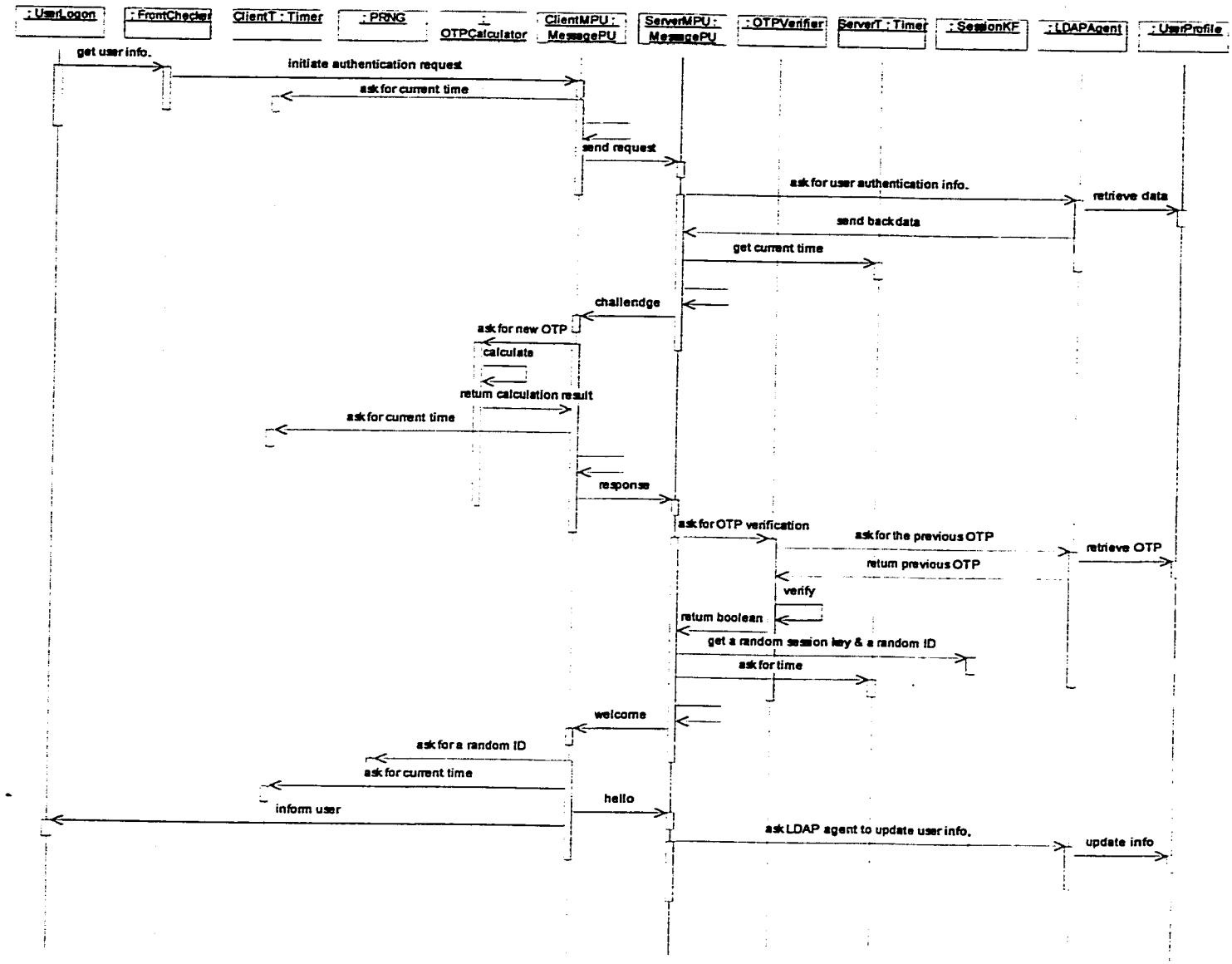


Figure 4.3-3: Sequence diagram of successful user authentication

This sequence diagram shows the message flow of an entire successful user authentication procedure. As you can see in this diagram, a secret session key and two random Ids are generated at the end of the authentication. The two Ids will be used as the identifiers of the sender and the receiver of the dynamic session, while the session key

will be used by the sender and the receiver to encrypt/decrypt the communication messages for this session.

4.3.3.2 Non-existent User

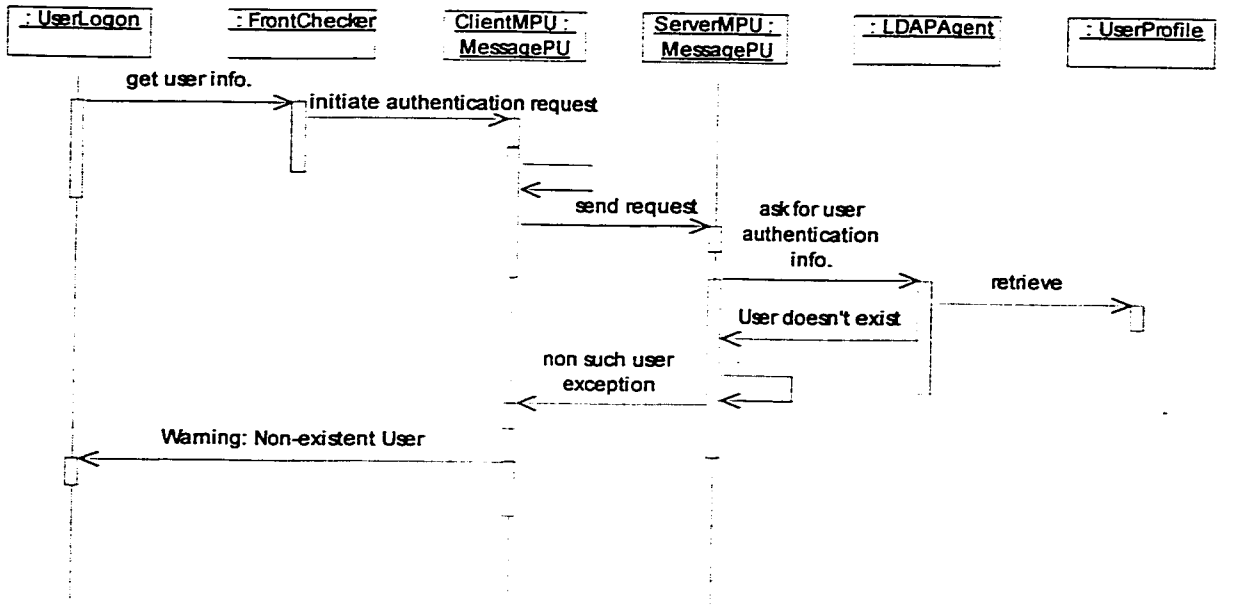


Figure 4.3-4: Sequence diagram if the user doesn't exist

In this scenario, a user may types in wrong information (ID/home email) or an attacker tries to access the system using a random chosen email address. If the home site of the claimed user can not find such a user, the system will refuse the user to access the system and give a warning message to the user. The user may modify the ID/home email and try again if necessary. Now, what about the claimed user does exist in the system, but is impersonated by someone else using a wrong password? Authentication failure will be reported (see the following section).

4.3.3.3 Authentication Failure

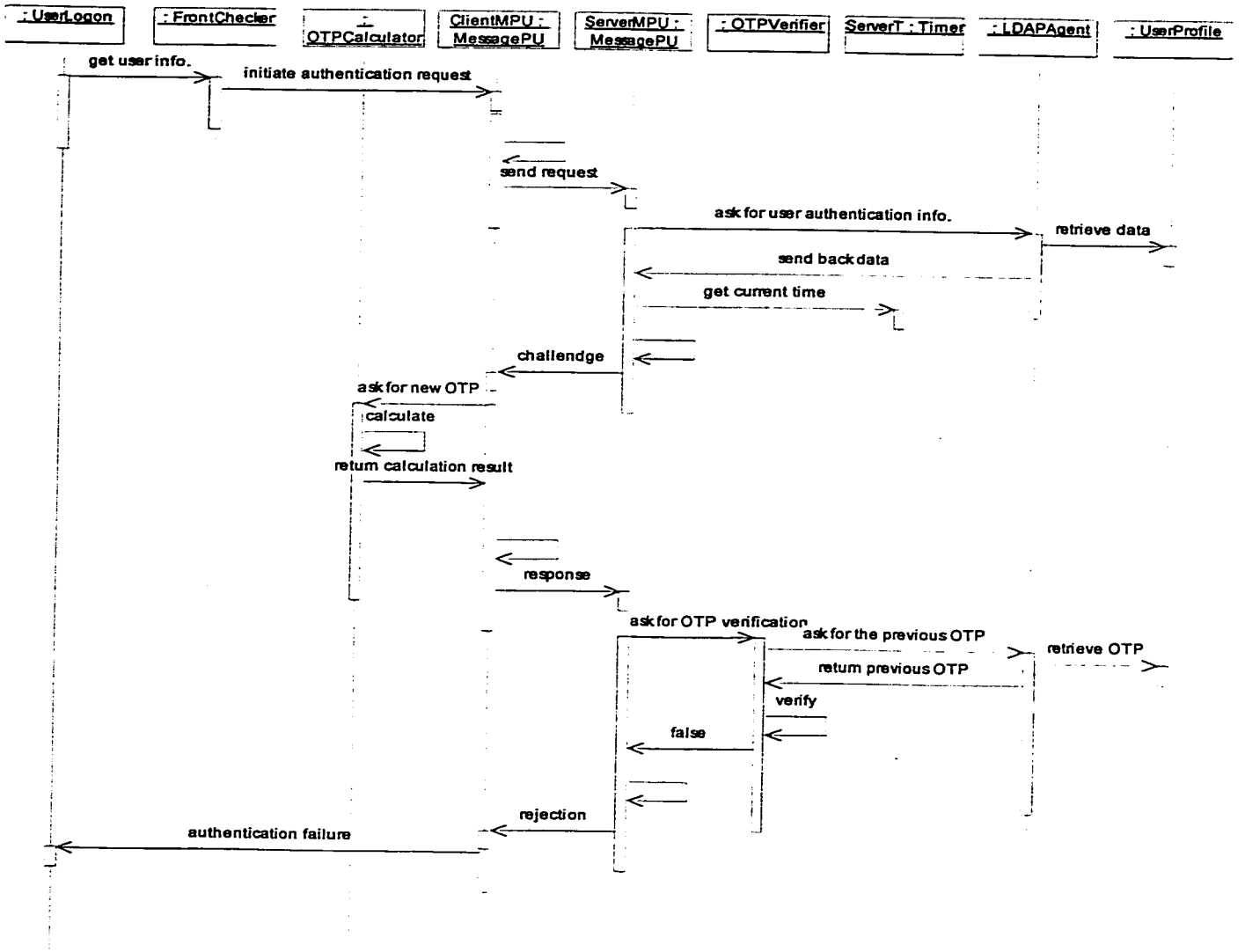


Figure 4.3-5: Sequence diagram of a failed authentication procedure

User authentication fails only due to the invalid secure pass phrase. This happens when a user accidentally types in a wrong phrase or an attacker tries to impersonate someone he knows by trying different secure pass phrase. In either case, the system will refuse the user to access the services. There is a 3-time access limit for a single logon, which means

if the same user has been rejected 3 times (3 wrong phrases) successively in a single logon, his/her account will be locked for a certain period of time (24 hours for instance).

4.3.4 Diagrams of Coordinator Creation

After user authentication, two instances of coordinator will be created. These coordinators will then establish a secure communication channel for the authenticated user between the visiting site and the home site. Four pieces of information are needed to create each coordinator instance, they are the user's identity (home site email), two random IDs for the coordinator instances and the secret session key.

```
SAMPLE CODE #1 --- Coordinator Class
public class Coordinator extends Thread
{
  private String Caller  = null;
  private String Callee = null;
  private byte[] sessionKey = null;
  private String userID = null;

  public Coordinator(String VisitingID, String HomeID, byte[] key, String UID)
  {
    this.Caller = VisitingID;
    this.Callee = HomeID;
    this.sessionKey = key;
    this.userID = UID;
  }
.....
}
```

The site CM (Coordinator Manager) is responsible for creating the local coordinator instances. In the same time, the site CM maintains a lookup table containing all the active instances created by it. This lookup table also keeps relating user information, since the user can not physically visit two different site simultaneously, this user information is

unique in the table at any given time. When a user tries to log in the system, the CM will check if there is such a user active in the system by searching the lookup table, thus, to detect possible impersonation attacks. This procedure can be shown in the following sequence diagram.

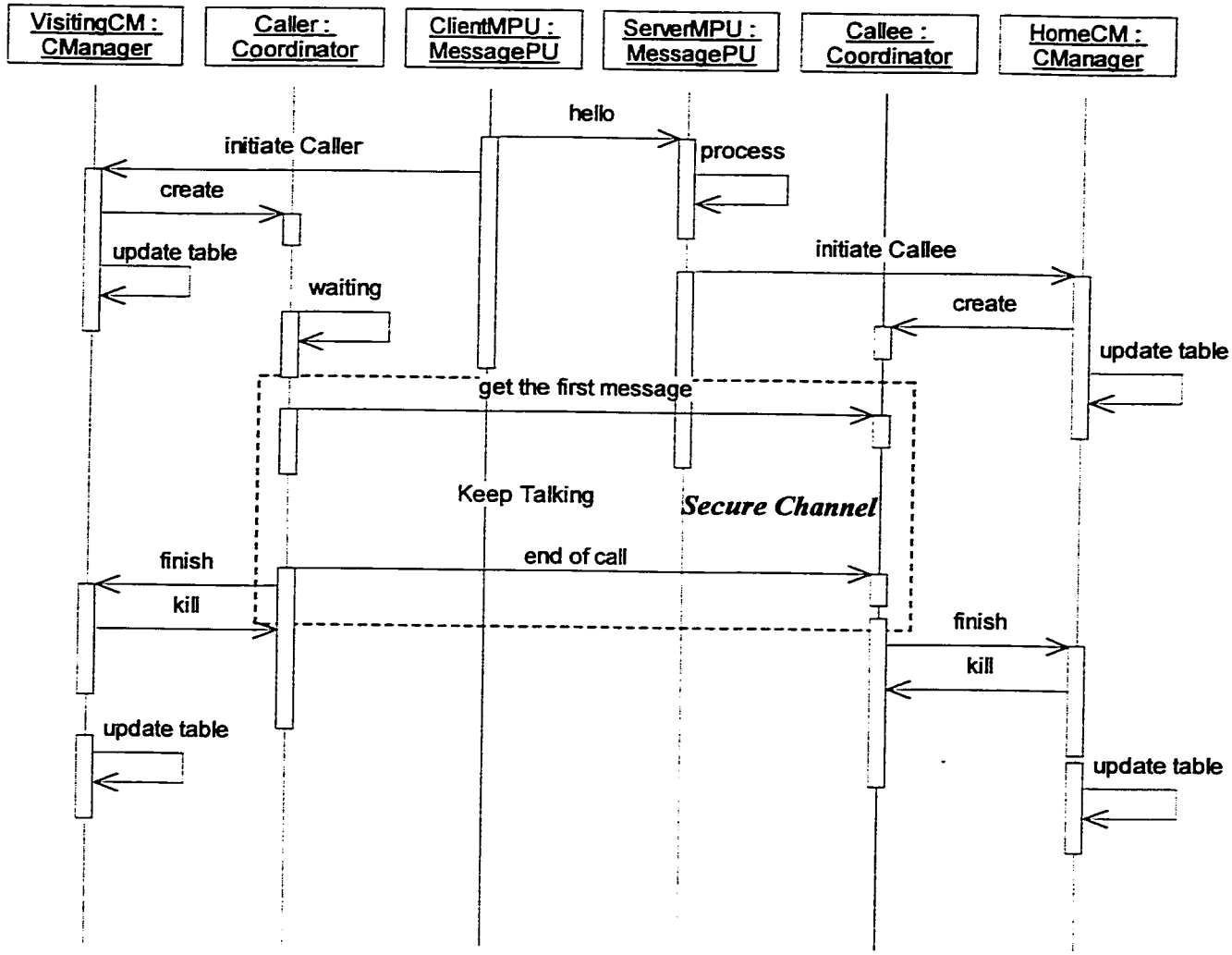


Figure 4.3-6: Sequence diagram of creating coordinator instances

This is the scenario only when everything goes smooth. The coordinator instance (Caller or Callee) is able to destroy itself and inform the coordinator manager (CManager) if

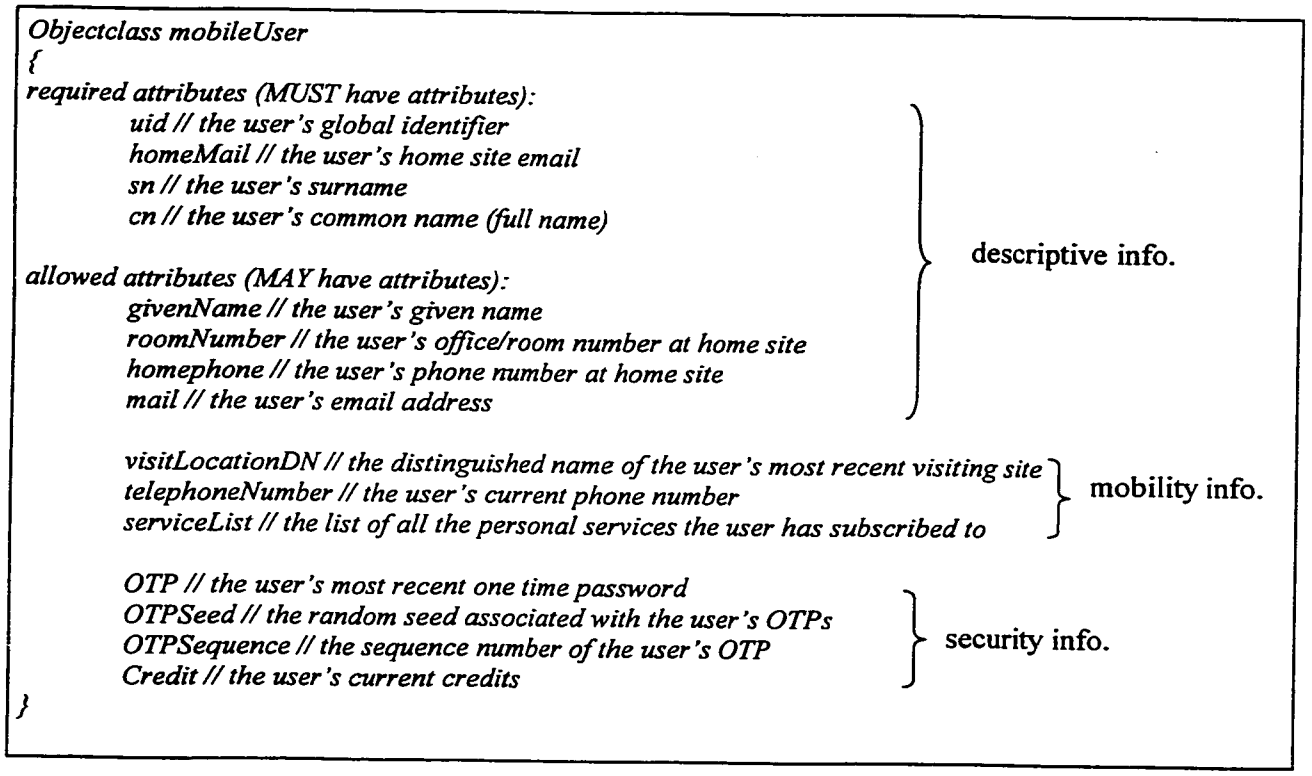
there is some error, for example, Caller could not get any response from Callee after time limit T, or the Callee finds out that the message sent from Caller has been modified.

4.4 System Profiles Management

4.4.1 Objectclasses Needed for Profile Management

In PMMS, we make use of LDAP to manage all the profiles needed for mobile computing, including user profile, resource profile, service profile, etc.

User profile contains information about every mobile user registered to the system. Since some attributes such as home-site-email-address are not standard LDAP attributes, a new objectclass [NET 97] called *mobileUser* has been defined in order to hold such information:



Actually, some of the attributes listed above have already been defined by other LDAP standard objectclasses, for example *sn* and *cn* can be found in objectclass *Person*, while *mail* and *homephone* can be found in objectclass *interOrgPerson*.

The information in a user's profile can be further divided into three categories, namely, descriptive information, mobility information and security information. Descriptive information describes the user's personal attributes and establishes the user's identity. Mobility information can be used to locate the user in the mobile computing system. Security information is used to authenticate the user and to control the user's access to the system services and resources.

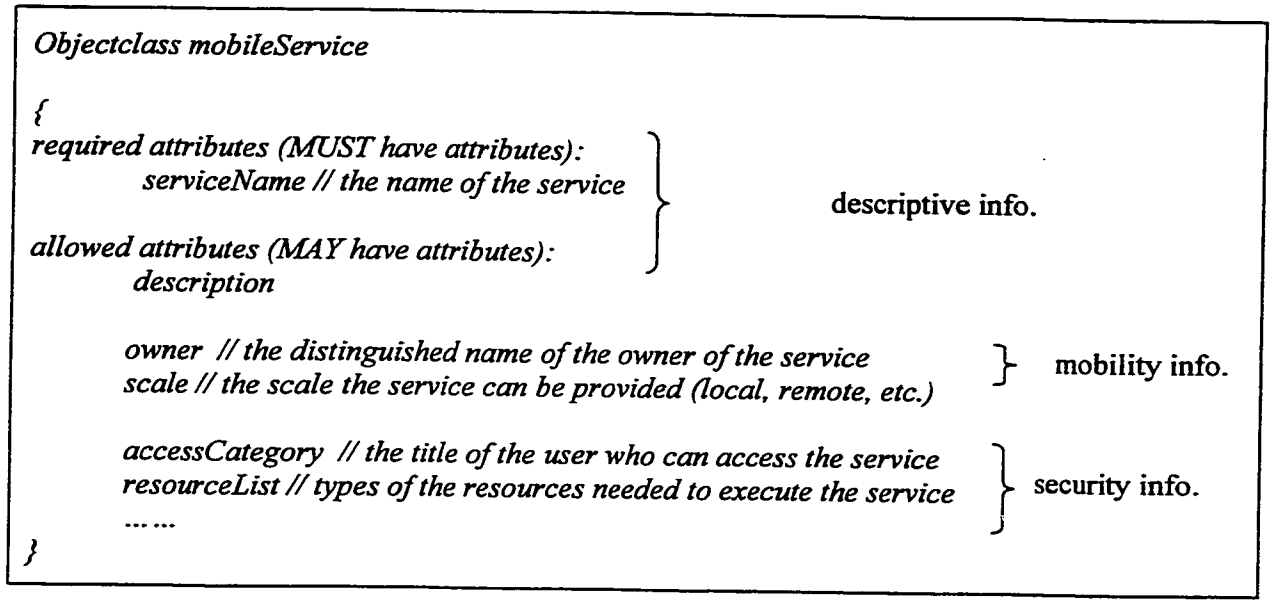
Same as the user profile, the resource profile and the service profile also contains these three types of information. The definitions for *mobilityResource* and *mobileService objectclasses* are given below:

```
Objectclass mobilityResource
{
required attributes (MUST have attributes):
        resourceID // an unique identifier for the resource

allowed attributes (MAY have attributes):
        resouceType  // the type of the resource(what kind of resource)
        browser  // the name of the browser of the device/terminal
        OS // the operating system of the device/terminal (if it is a computer)

        owner  // the distinguished name of the owner of the resource
        location // the physical location of the resource

        networkAddress // the network address of the resource
        status // the current status of the resource (idle, busy, etc.)
        serialNumber // the global unique serial number of the resource
        ……
}
```

descriptive info.

mobility info.

security info.



4.4.2 Software Agents for Profile Operations

In order to manage the data stored in the directory server upon request, various kinds of agents are designed for different purpose.

4.4.2.1 User Finding Agent

The objective of this type of agent is to figure out the user's home site based on the information (i.e. user's home site email) provided by the user, and to verify the existence of that user against the directory of the user's home site.

4.4.2.2 General Query Agent

General query agents are only able to perform operations that won't reveal sensitive information about the user or the system. Operations like extracting a user's email address or telephone number are considered general query. More precisely, general queries should not conflict with the system's access control policy, which is specified by the Access Control Lists of the LDAP directory server.

4.4.2.3 Authorized Query Agent

This type of agents is authorized by the system, they have the ability to retrieve information which is prohibited by the ACL of the directory server. Private information such as the user's one time password, current credits could be accessible to these authorized agents. However, these agents are not allowed to make any changes to the data in the directory server. Furthermore, each agent has its own privilege that is defined by the system, it is also assigned a pair of ID and password used to authenticate itself when accessing the private data in the directory.

4.4.2.4 Updating Agent

This type of agent is designed mainly to modify the profiles stored in the LDAP server. Of course, launching such task requires appropriate authority. We can specify the authority using the ACL and/or the security policy of the site, and give proper access ID and password combination to the agent. We can either limit the privilege of the agent to a specific branch of the hierarchical tree of the LDAP DIB or a particular node.

To implement these agents listed above, we need to utilize proper APIs for LDAP access. Since the entire system is written in Java, we choose LDAP API for Java form Netscape. Figure 4.4-1 shows the architecture of the implementation.

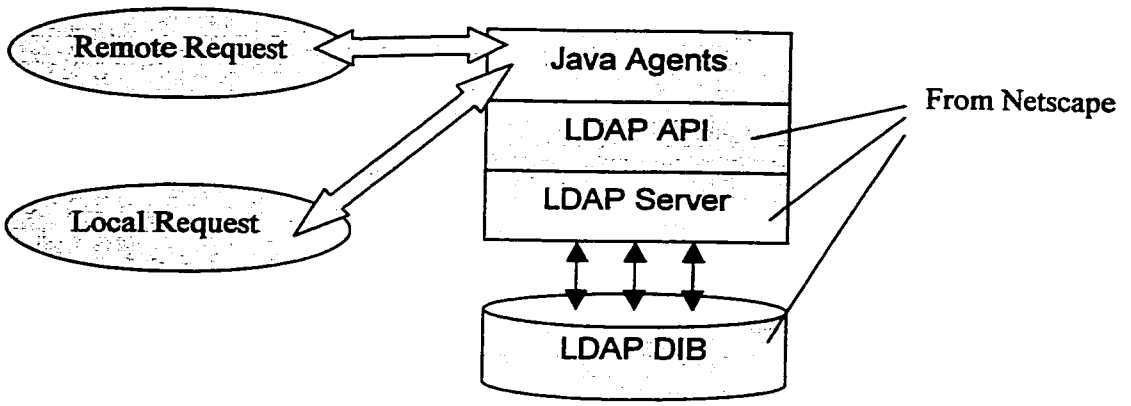


Figure 4.4-1. LDAP Access Architecture

4.5 User and Key Management

4.5.1 Use Cases

The user management and the key management are two major responsibilities of the system administrator. There will be new users subscribing to the system from time to time, and sometime we may need to remove a user's profile from the directory. Occasionally, some users may forget their secure pass phrase, etc. All of these require human intervention. In the PMMS, two standalone applications are developed to facilitate these processes.

In the current PMMS, user management facility is only applicable to the site administrator locally. A user can establish his/her profile by filling a form, and pass the form to the site admin. The site admin is responsible for the verifying the authenticity of the information provided by the user, and putting the information into the directory. Or, the site admin may establish the profile on behalf of the user and ask the user for his/her

confirmation. The following figure illustrates the use cases [LAR 97] of the user profile management.

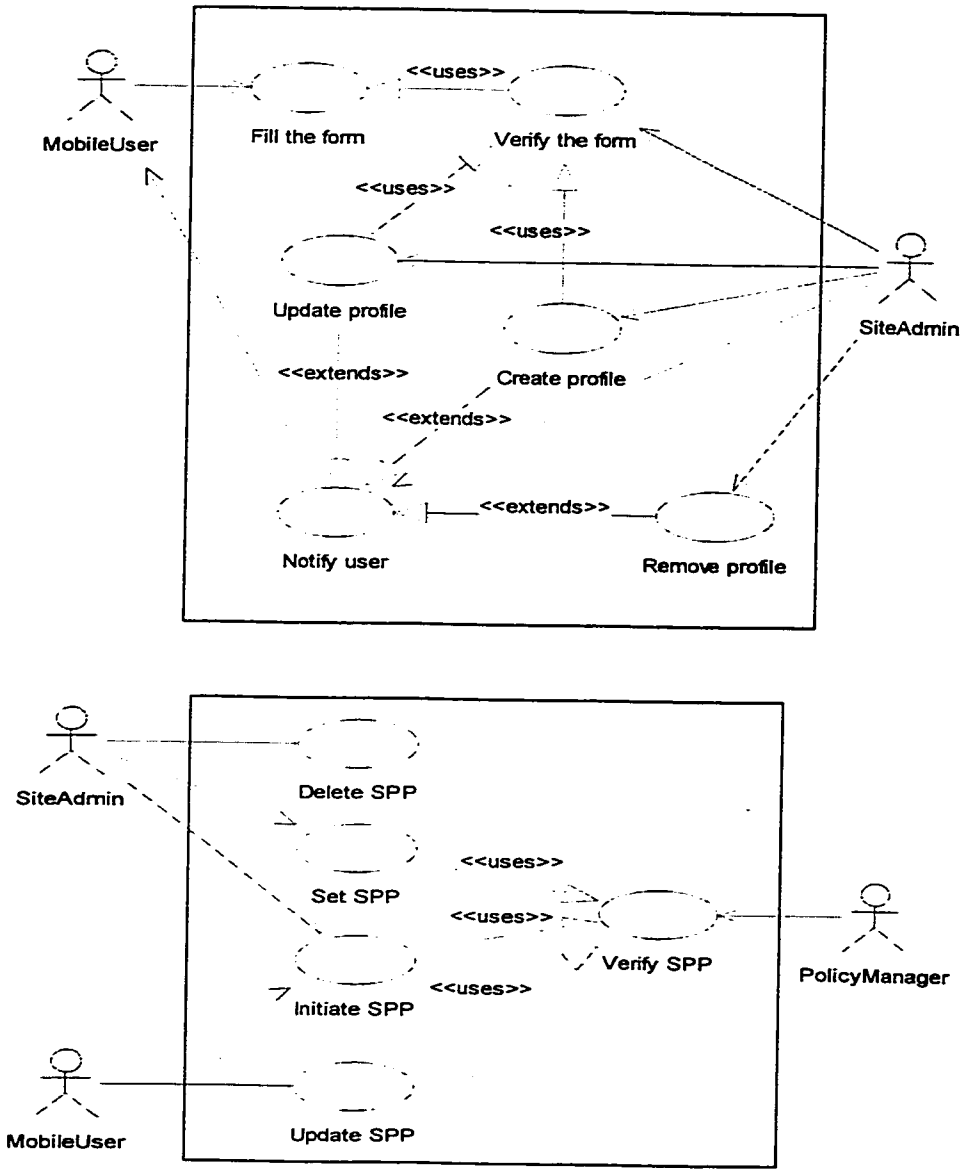


Figure 4.5-1: Use Cases for User and Key Management

4.5.2 User Management Interfaces

We are more interested in "create profile". This process normally contains three steps: First, the user fills the registration form. Then, the admin verifies the form for the

authenticity. Last, the admin types in the verified information about the user to the system directory via the user management interface. At this time, the user may specify his/her initial secure pass phrase or the system will generate one for the user and the user can change it at the first time he/she logs on the system. The interface for user management is shown in the following figures.

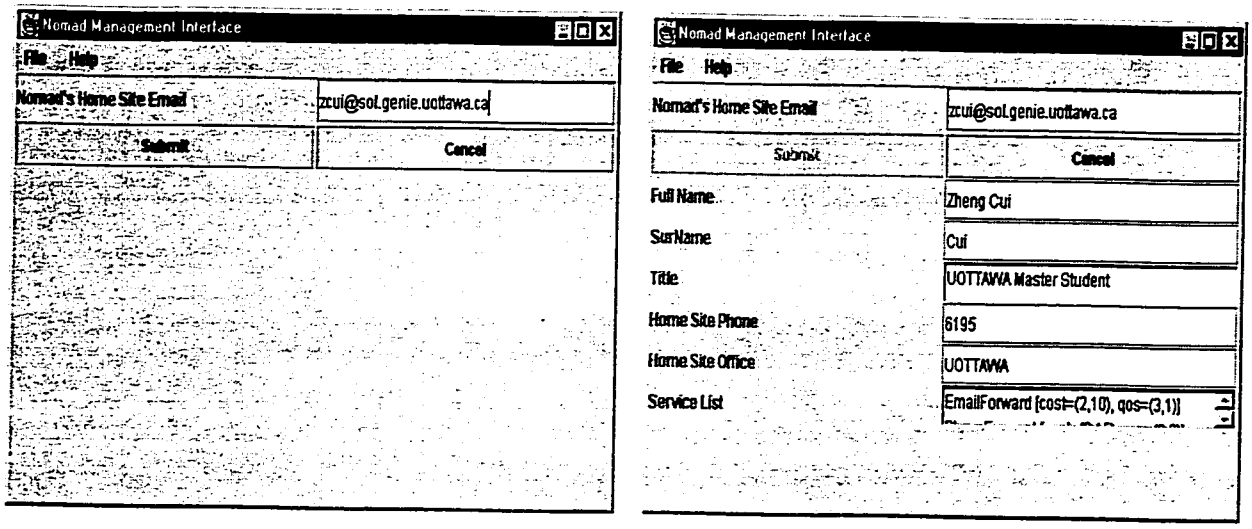


Figure 4.5-2: User Management Interface

If a user by that email address already exists in the directory of the system, this interface will only display some general information of this user, and as we can see, the ***Submit*** button of this form will be disabled. The content of the user profile can not be changed at this moment. If it's a new user (see figure), the system will ask the user/admin to fill an empty form so that appropriate user profile could be established.

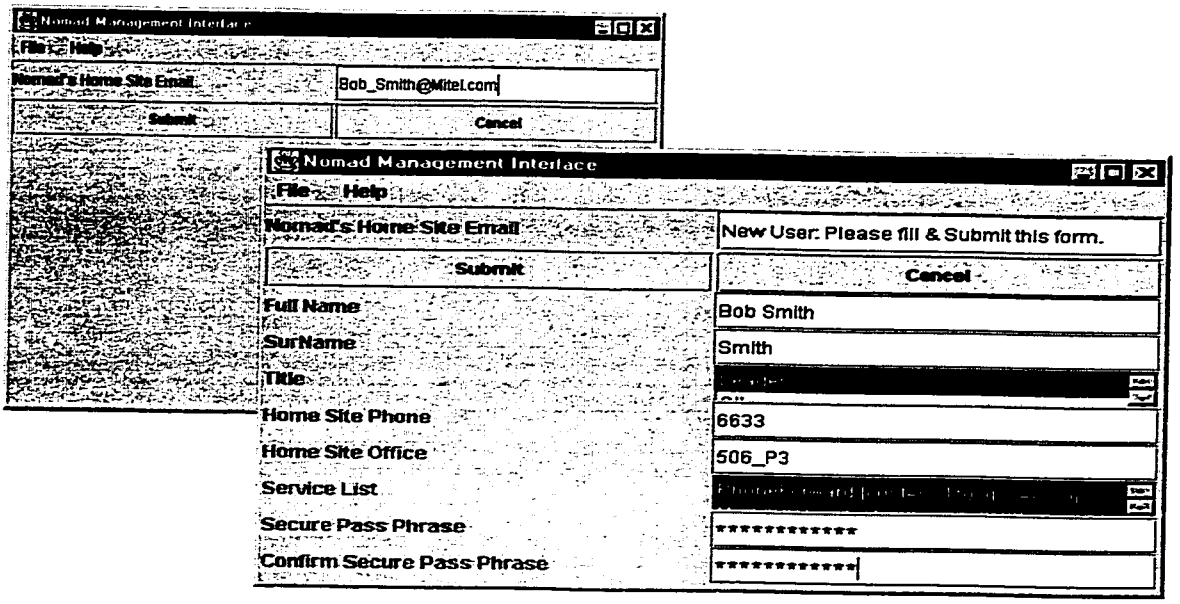


Figure 4.5-3: User Initialization Interface

The form will be checked after submission. More importantly, the user's initial Secure Pass Phrase (SPP) will be examined against the security policy (see previous chapter) of the system. When every item of the form passes the verification, a new user profile will be created and activated. Now the user can access the system with the initial SPP.

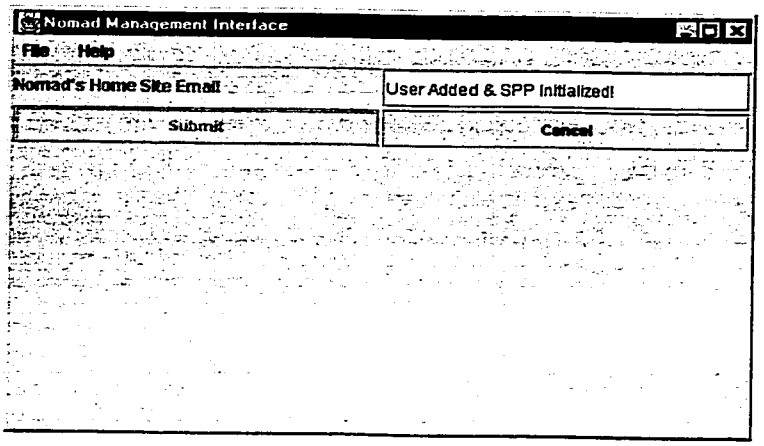


Figure 4.5-4: User Initialization Successful

The sequence diagrams for initializing and changing the secure pass phrase later on are given in the following two figures respectively.

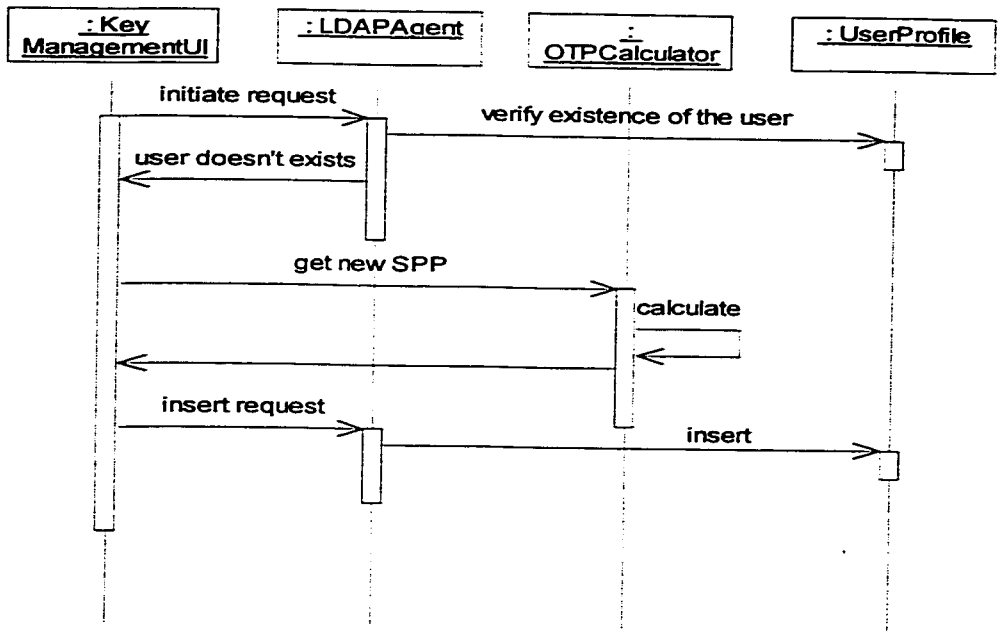


Figure 4.5-5: Sequence diagram of SPP Initialization

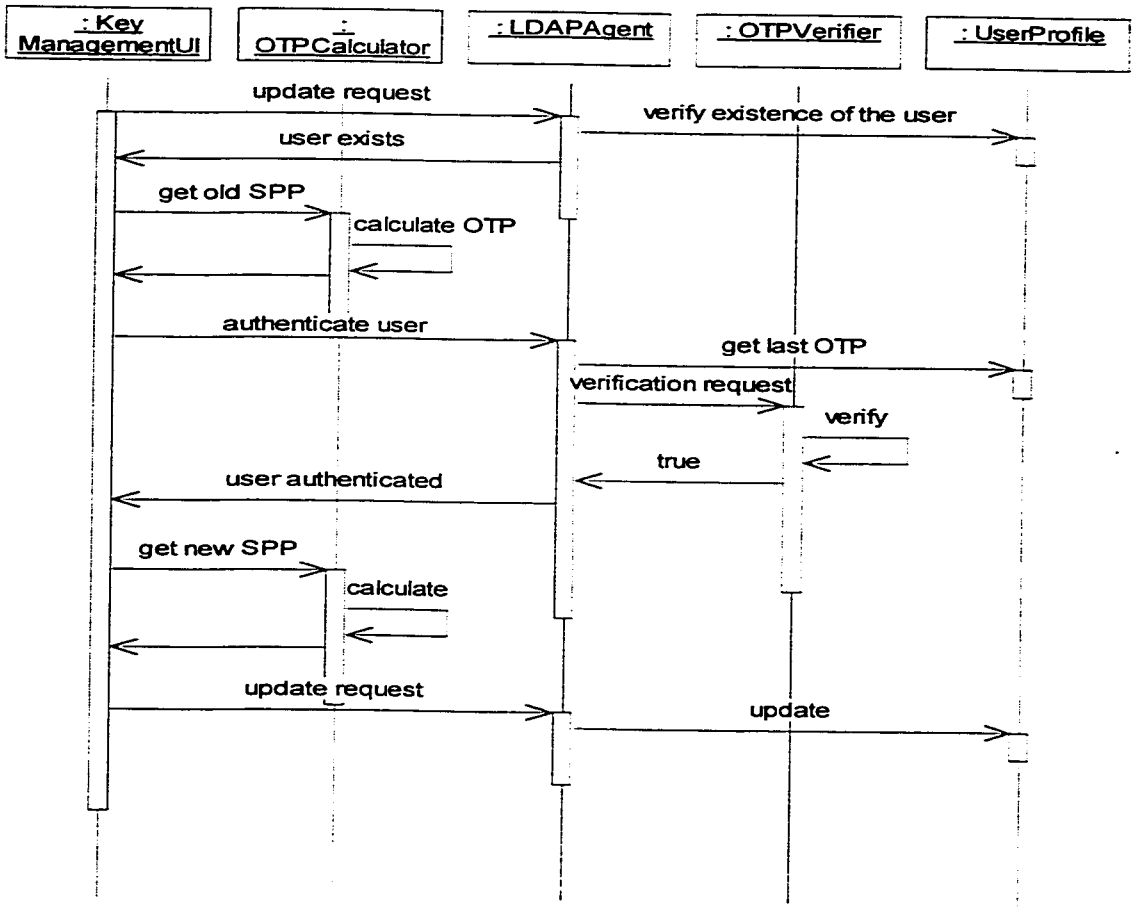


Figure 4.5-6: Sequence diagram of changing SPP

4.5.3 Key Management Interfaces

The following two interfaces are developed for mobile user and system admin respectively. Mobile users can update their secure pass phrase using the first interface at any time. If a user forgets his/her SPP, the system admin can set the SPP to a prefered combination and let the user change it later on within a time limit (say 24 hours). However, even the system admin does not have the capability to recover the old SPP.

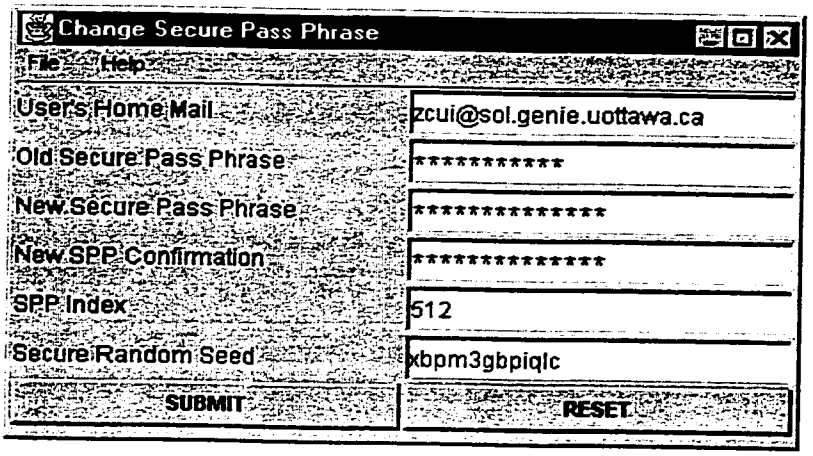


Figure 4.5-7: Interface for users to update SPP

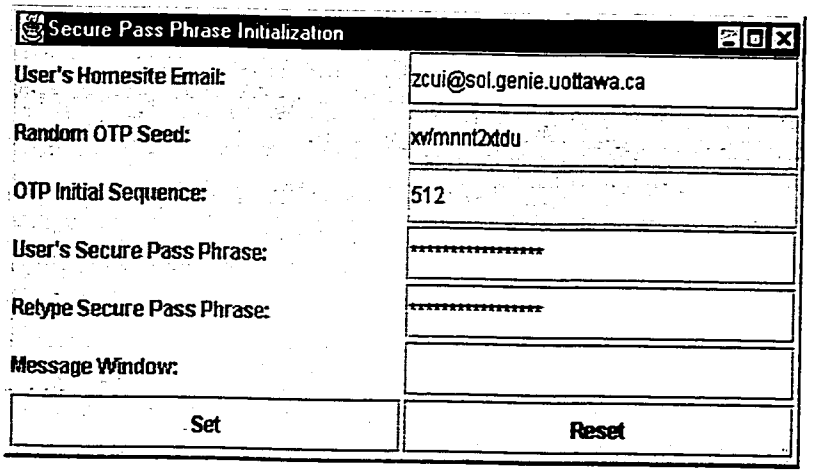


Figure 4.5-8: Interface for Admin to set user's SPP

4.6 Testing and Verification

4.6.1 Validation of Cryptographic Algorithms

All the programs developed for the PMMS should be verified to be error free and capable of performing all the intended functions. As the kernel of the system, cryptographic algorithms are very critical, they must be implemented precisely according to their standards. The most common method to verify the correctness of a given cryptographic algorithm is to apply a set of well-known standard testing vectors to it, and verify the outputs against the standard solutions. We performed this kind of validation on all the cryptographic algorithms implemented by ourselves. Since the current system is symmetric, we also verified the symmetry (see the formula below) of all the encryption algorithms employed by the system including the DES and the Triple_DES.

$$D(E(M:k):k) = M$$

Where *D (M: k)* and *E (M: k)* mean to decrypt and encrypt a given message M using a given key k respectively.

4.6.2 Encryption Strength

As we have discussed previously, the privacy of a given cipher depends on the strength of the encryption algorithm used to generate the cipher, while the strength of a well-designed cryptographic algorithms (for instance DES, RCx) should only depend on the key length [DEN 82] [SCH 95]. In our current implementation of the PMMS, we employ 112-bit Triple_DES encryption. Assuming that we have a billion super computers/machines, each of which is capable of performing 1 billion Triple_DES decryption operations (much faster than the reality) in a single second, the average time

needed to decrypt a given cipher is still around 1 million years (see the following expression):

$$Time \quad in \quad years = \frac{2^{112}}{10^{10} \times 10^{10} \times 60 \times 60 \times 24 \times 365 \times 2} \approx 820{,}000$$

4.6.3 Encryption Performance Testing

We need to make check if the overhead introduced by the encryption/decryption will affect the overall system performance. On an IBM compatible PC with 64-MB ram, PII 200 MHz CPU and Windows NT 4.0 OS, the speed of the 112-bit Triple_DES en/decryption is approximately 8M bps (bit per second) using standard JDK1.1.6 compiler. With PIII 500 MHz CPU, the speed increases to over 16M bps. Considering the fact that the data transfer rate of a typical Internet connection varies from several K bps to several M bps (i.e. T1 line provides 64K bps), this encryption and decryption operation will hardly become the bottleneck of the overall performance.

4.6.4 Accessing the PMMS Using Invalid User Information

We have made some testing using invalid user information to access the PMMS. Each user will be asked to type in his/her user ID (i.e. home site email address) and the Secure Pass Phrase (SPP). If the user ID does not exist in the home site user profiles, or the home site can not verify the One Time Password (OTP) based on the SPP, a web page containing warning information will be displayed to the user. If the user is rejected three times in a row under the same user ID, the system will lock this account for the following 24 hours to prevent possible impersonation attacks. In the case that a real user forgets his/her SPP, he/she may contact the system administrator to unlock the account by means

of establishing a new SPP. Since the system doesn't contain any information about the SPP, there is no way to recover the user's original SPP.

4.6.5 Accessing System from an Un-enabled Terminal

Another testing we've made is trying to access the PMMS using valid user information but from an un-authorized / un-enabled terminal. Un-authorized terminals can be defined in the system profile. In the current PMMS, users access the system through a Java-enabled web browser, the web page containing the interface for user login anf authentication is generated dynamically on the fly by using Java Servlet techniques. Before spitting out the dynamic page, the Java Servlet will check if the terminal is enabled and authorized. If the terminal is not authorized, the user login and authentication interface will not be displayed to the user, instead, a warning page will be displayed.

4.6.6 Concurrent Accesses

The desired system should be able to handle multiple simultaneous accesses. Since the PMMS is just a prototype, we did not test this issue in a very large scale. However, we tried to access the system from a limited number of terminals simultaneously, the PMMS succeeded in handling these access requests. Because we programmed our code in a Multi-threaded fashion wherever possible, it is reasonable to believe that the system is capable of processing concurrent accesses in a fairly large scale.

Chapter 5

Conclusion and Future Work

5.1 Summary

This thesis introduced Personal Mobility Management System, a distributed mobile computing system based on agent technology and MicMac Agora messaging server. Some security concerns of the PMMS were presented, and the design and implementation of several security frameworks including user authentication, user authorization and secure communication were discussed in details.

This system was implemented in a three-site virtual network, namely, University of Ottawa, Mitel Corporation and National Research Council of Canada. Java was chosen as the primary programming language, not only because Java provides the capability of "write once, run anywhere", but also because Java Security Manager (JSM [OAK 98]), the build-in security mechanism in Java, helps to improve the security performance of the system.

The user authentication and key exchange protocol (MAuth, see chapter 3) employed in our current system is based on One Time Password System. The MAuth protocol supports both symmetric and public key infrastructure and is optimized for the system. Combined with the secure communication framework, it is effective in countering many types of security attacks. This protocol is implemented using symmetric infrastructure (112 bit Triple DES with 160 bit SHA). Although this is more secure than ordinary schema using DES, we open this framework to other authentication protocols and public key cryptosystem such as RSA for higher security pursuit. This places future improvements on authentication agent and authentication server.

5.2 Suggestions on Future Work

5.2.1 Terminal Authentication & Authorization

Terminal Authentication and Authorization mechanisms have been designed but not implemented in our current system. The PMMS needs proper terminal authentication and authorization techniques to identify every mobile device and control its access behaviors to the system. The basic idea of this procedure is to verify the terminal using a certain kind of hardware information physically located or associated with that device. Special information used to identify the terminals is required to be stored at the server in a secret manner. This could be done by extending the standard LDAP directory schema to support attributes needed for terminal authentication and access control.

5.2.2 Remote Access Scenario

In our current PMMS, the remote access scenario, which enables a nomadic user to access services remotely by logging into his home site via an Internet Service Provider, has not been implemented. A typical situation is that a user who goes abroad to attend a meeting stays in a hotel and wants to access network services (like video mail) from the laptop he is carrying. In order to accomplish the task, first he needs to gain access to the Internet by hooking up to a local Internet Service Provider (ISP), then logs into his home site, authenticates himself and eventually enjoys the services.

One question remains here: some services may require certain special setup or configuration at the client side in order to execute. This places several challenges:

(1) Some entity should be available and responsible for setting up the environment dynamically before delivering the services;

(2) After the environment has been properly configured, service agents should be able to migrate from the home site to the client;

(3) After providing the services, service agents should be able to remove themselves from the client or destroy themselves;

(4) Some entity should take care of cleaning up the environment created for delivering the services leaving the client's machine untouched, and

(5) The home site should be able to keep track of the status of those service agents.

In our opinion, mobile agent technology is the best solution to these challenges. Moreover, in the PMMS, mobile agents can also be used to negotiate the user preferences before delivering the services. However, employing mobile agent will introduce more security concerns [VIT 97] in the mean time. Section 5.3 will briefly discuss these security concerns in a typical mobile agent system.

5.2.3 Using Mobile Agent

All the agents in our current PMMS are static. Because of this static feature, there is less concern about verifying the authenticity and functionality of those agents. However, since the agents are separated and distributed across the network, there are more inter-site agent communications involved in a single task like profile negotiation and resource allocation. This means more network traffic, which might become the bottleneck of system overall performance.

One solution to this problem is to employ mobile agent instead of static agent. Consider the Site Profile Agent (see chapter 3 and 4 for more information) as an example, if we make it mobile, the home site SPA would be able to move to the client site once, and communicate with the visiting site SPA locally for the user's preference and services. This can be done easily if the SPA could carry a private knowledge base about the user's preference and part of the home site's policy. Since a mobile agent is able to carry data while its moving, this idea is feasible. However, we haven't gone through yet. We need to consider this issue from security point of view.

5.3 Security Threats in Mobile Agent System

5.3.1 Code Modification

When a mobile agent roams around the network, it faces the possibility of being modified by other agents or hosts it encounters. This modification could be accomplished in many ways. For example, a malicious host may try to change the executable code of the mobile agent in such a way that the agent behaves under the host's control. Or, the host may try to modify the data carried by the agent such as the price list of certain merchandise.

5.3.2 Arbitrary Access

In the mean time, mobile agents can also cause damages to the hosts. For instance, a mobile agent can try to access the local file system of a host, modify the system configuration, or even leave a Trojan. Having the capability to travel across the Internet, mobile agent might be abused by some malicious parties to do something really harmful, like broadcasting a virus or causing traffic jam in a very large scale.

5.3.3 Secrets Releasing

Mobile Agents could make everything possible but nothing secret. Hosts can steal information that mobile agents carry; mobile agents can steal information that hosts posses; and agents can steal information from each other.

5.3.4 Function Hiding & Monitoring

In any mobile agent system, there is always a conflict between function hiding and function monitoring. Hosts are intended to control every detailed behavior of a mobile

agent, on the other hand, mobile agents are not willing to reveal the details of **how** they behave when they execute on a remote host, though they may be happy to tell **what** they will do. How to make sure a mobile agent is really and only doing what it claims to do on a remote host and how to conceal the function of a mobile agent in an un-trusted environment are two of the most challenging issues emerged in the mobile agent technology. And they have received a lot recent attention.

For the future, we should address these security concerns in order to apply mobile agent technology into the prototype of PMMS. A lot research work could be conducted in this area as well.

References

[ANS 85] ANSI X9.17 (Revised), "American National Standard for Financial Institution Key Management (wholesale)," American Bankers Association, 1985.

[ANS 81] ANSI X3.92, "American National Standard for Data Encryption Algorithm (DEA)," American National Standards Institute, 1981

[AWE 95] B. Awerbuch and D. Peleg, "Online Tracking of Mobile Host", JACM, 42(5), Sep 1995, pp.1021-1058.

[BOO 98] G. Booch, J. Rumbaugh and I. Jacobson, *The Unified Modeling Language User Guide*, Reading Mass.: Addison-Wesley, 1998.

[BRA 97] J. Bradsahaw, *Software Agents*, Menlo Park, California: AAAI Press, c1997

[DEN 82] D. E. Denning, *Cryptography and Data Security*, Reading, Mass, Addison-Wesley Publishing Company, 1982

[FAR 98] J. Farley, M. Loukides, *Java Distributed Computing*, Sebastopol CA: O'Reilly & Associates, 1998

[FEI 97] S. Feit, *TCP/IP: Architecture, Protocols, and Implementation with IPv6 and IP Security*, 2nd edition, New York: McGraw-Hill, c1997.

[FLA 99] D. Flanagan, *Java in a Nutshell: a Desktop Quick Reference*, 3rd edition, Sebastopol, CA: O'Reilly, c1999.

[GAS 89] M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson, "The Digital Distributed Systems Security Architecture," Proceedings of the 12th National Computer Security Conference, NIST, 1989, pp.305-319

[HAR 97] E. R. Harold, *Java Network Programming,* Sebastopol CA: O'Reilly & Associates, c1997.

[HOO 98a] A. Hooda, A. Karmouch, S. Abu-Hakima, "Nomadic Support Using Agents," 4th International Symposium on Internetworking, Ottawa, Jul 1998

[HOO 98b] A. Hooda, A. Karmouch, S. Abu-Hakima, "Personal Mobility Management Using LDAP Distributed Directory and Static Agents", to appear in 5th Int. Conference on Intelligence in Networks, ICIN'98, France

[IEE 99] http://grouper.ieee.org/groups/1363/index.html, "IEEE P1363: The Standard Specifications for Public Key Cryptography", Apr 1999

[JAV 99] http://java.sun.com/products/jdk/1.2/docs/index.html, "Java 2 API Documentation", Apr 1999

[LAI 92] X. Lai, "On the Design and Security of Block Ciphers," ETH Series in Information Proceeding, V.1, Konstanz [Germany]: Hartung-Gorre Verlag, c1992

[LAN 94] S.K. Langford and M.E. Hellman, "Cryptanalysis of DES", presented at 1994 RSA Data Security conference, Redwood Shores, CA, 12-14 Jan. 1994.

[MAC 96] R. S. Macgregor, A. Arest and A. Siegert, *WWW Security – How to build a Secure World Wide Web Connection*, Upper Saddle River, N.J.: Prentice Hall, c1996

[MIC 97] Microsoft, "Understanding Point-to-Point Tunneling Protocol (PPTP)," Technical Documentation, Microsoft Corporation, 1997

[MIT 96] http://www.Mitel.com, "The MicMac software Testbed," Technical Report, Mitel-CATA 1996.

[NAT 94] National Institute of Standards and Technology, NIST FIPS PUB 186, "Digital Signature Standard," U.S. Department of Commerce, May 1994

[NET 96] Network Working Group, "SOCKS Protocol Version 5," RFC 1928, Mar 1996

[NET 97] Network Working Group, "Lightweight Directory Access Protocol (v3)," RFC 2251, IETF, Dec 1997

[NET 98] Network Working Group, "A One Time Password System," RFC 2289, February 1998

[NET 99] Network Working Group, "Layer Two Tunneling Protocol (L2TP)," RFC 2661, Aug 1999

[NEU 94] B.C. Neuman, and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," IEEE Communicaitons Magazine, v.32, n.9, Sep 1994, pp. 33-38.

[OAK 98] S. Oaks, *Java Security*, Sebastopol CA: O'Reilly & Associates, c1998

[OPP 96] R. Oppliger, *Authentication Systems for Secure Networks*, Boston: Artech House Inc., c1996

[ORF 98] R. Orfali, D. Harkey, *Client/Server Programming with Java and CORBA, Second Edition*, New York: John Wiley & Sons, Mar 1998

[RES 92] Research and Development in Advanced Communication Technologies in Europe, "RIPE Integrity Primitives: Final Report of RACE Integrity Primitives Evaluation (R1040)," RACE, June 1992

[RIV 78] R. L. Rivest, A. Shamir and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, v.21, n. 2, Feb 1978, pp. 120-126

[RIV 79] R. L. Rivest, A. Shamir and L. M. Adleman, "On Digital Signatures and Public Key Cryptosystems," MIT Laboratory for Computer Science, Technical Report, MIT/LCS/TR-212, Jan 1979

[RIV 92a] R.L. Rivest, "The RC4 Encryption Algorithm," RSA Data Security Inc., Mar 1992

[RIV 92b] R.L. Rivest, "The MD4 Message Digest Algorithm," RFC 1320, Apr 1992

[RIV 92c] R.L. Rivest, "The MD5 Message Digest Algorithm," RFC 1321, Apr 1992

[RIV 95] R.L. Rivest, "The RC5 Encryption Algorithm," Dr. Dobb's Journal, V. 20, n. 1, Jan 1995, pp. 146-148

[ROB 94] M.J.B. Robshaw, "Block Ciphers," Technical Report TR-601, RSA Laboratories, July 1994

[SCH 94] B. Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," Fast Software Encryption, Cambridge Security Workshop Proceedings, Springer-Verlag, 1994, pp. 191-204.

[SCH 95] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, second edition, New York: John Wiley & Sons, c1995

[SIM 92] G. J. Simmons, Contemporary Cryptology – the science of information integrity, Piscataway, NJ: IEEE Press, c1992

[SPA 95] O. Spaniol, A. Fasbender, S. Hoff, J. Kaltwasser and J. Kassubek, "Impacts of Mobility on Telecommunication and Data Communication Networks," IEEE Personal Communications, Oct 1995.

[VIT 97] J. Vitek, "Security and Communication in Mobile Object Systems," *Mobile Object Systems: Towards The Programmable Internet: Second International Workshop, MOS '96, Linz, Austria, July 8-9, 1996: Selected Presentations and Invited Papers,* Berlin; New York: Springer, c1997 pp. 177-200

[ZIM 95] P.R. Zimmermann, *The Official PGP User's Guide*, Boston, MIT Press, 1995

Publications

[CUI 99a] Z. Cui, A. Karmouch, T. Gray, S. Mankovski, and R. Impey, "Ensuring Secure Communication for a Distributed Mobile Computing System based on MicMac," Proceedings of the First International Workshop on Mobile Agents for Telecommunication Applications, October 6-8, 1999, Ottawa, Canada, pp. 375-391

[CUI 99b] Z. Cui, A. Karmouch, "Approaching Secure Communications in a Distributed Mobile Computing Environment," Proceedings of the IASTED International Conference, IMSA '99,October 18-21, 1999, Nassau, Bahamas, pp.. 426-430

[CUI 99c] Z. Cui, A. Karmouch, T. Gray, "Security Mechanism and Architecture for Collaborative Software System using Tuple Space", Patent (pending), October 1999.