Delivering mobile services to mobile users in open networks: Quality of service, authentication and trust-based access control

By

Zhen Eric Zhang

A thesis submitted to the Faculty of Graduate Studies
and Postdoctoral Studies in partial fulfillment of the
requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa

April 2008

Approved by ______________________________
Chairperson of Supervisory Committee

____

____

____

Program Authorized
to Offer Degree______________________________

Date ______________________________

Abstract

This thesis describes scenarios in which a mobile user needs various services, such as Internet telephony, secure printing and online data services in a number of places. From these scenarios we summarize the requirements for quality of service control, service discovery, user authentication and authorization, service access control, and user privacy. In order to implement a prototype to support service discovery, we studied different technologies including Bluetooth, Jini, and Web Services. SDP from BlueZ was chosen to limit the search range within the user's local area while using minimal power consumption. Also included in the implementation, the SIP protocol is used to initiate the session and exchange messages while JMF is used to capture and deliver multimedia data. In the process of adapting Dupre's authentication protocol for user authentication, we found that it is possible for a third party to intercept the messages exchanged between a user and a Foreign Agent, which may lead to denial of service attack and weakens the strength of the user's password. The protocol is then improved by introducing additional message segments and altering the way to verify the server's response. The thesis also deals with trust relationships, which are needed as a basis for communication between the two parties. Shi's probability distribution model is introduced to integrate recommendations from different domains so that a service provider could make better decisions whether a given user should be assigned certain access rights. In the other hand, a user also depends on a trust relationship to make sure that his or her sensitive data will be handled properly. Finally, based on all of the above, a trust-based access control framework for mobile users and services is proposed and choices of implementation are briefly discussed.

Table of Contents

List of figures

Acknowledgments

I would like to thank my supervising professor Dr. Gregor v. Bochmann for his support and guidance in my research. He has given me not only the freedom to think creatively but also the insights to software engineering and valuable opinions on how to conduct proper academic research. I would also like to thank other professors in my research group, in particular Dr. Carlisle Adams, Dr. Morad Benyoucef and Dr. Jennifer Chandler from the University of Ottawa, for their insights to the domain of network security, and user privacy.

In addition, I would like to thank my colleagues in my associated research groups, namely Khalil El-Khatib, Ken Chan, Jianqiang Shi and others, for listening to my ideas and complaints. Of course, I would like to thank my wife Wei Xie, my parents CongFa Zhang and Guoju Yu, and my sister Li Zhang for their support during my study.

Also, this project would not be possible without the support of Communications and Information Technology Ontario (CITO) and The Ontario Research Network for Electronic Commerce (ORNEC).

1. Introduction

With the growing popularity of Internet access, more and more users are used to require services through the Internet. Many of these services are not provided by local vendors. On the other hand, when a service provider deals with its users, it is common that some of them are not living in the same geographic area as the service provider. This becomes a challenge to both users and service providers. To a user, the problem is to find a suitable service while visiting a new place. Once found when he or she arranges payment for the acquired service, whether or not to trust the service provider with his or her financial information is a risky decision given the fact that the user may never have dealt with the provider before. To service providers, how to advertise their services over the Internet to draw as many user attention as possible is also a problem. When a user is inquiring for a service, it is also difficult for the service provider to make sure the user would not abuse the service and to collect payment. Initially, the motivation of this thesis was to support mobile users and services so that while users are traveling across different domains, service providers can continuously deliver services to them. In order to solve this problem, we first identified the differences between mobile users, mobile services and mobile devices (see section 1.1). Then we studied various kinds of services that are offered on the Internet. As well, challenges for delivering such services are also recognized. To further investigate the requirements of supporting user and service mobility in these applications, a list of practical scenarios were examined, in which three categories of requirement were identified (see section 1.2). These requirements are: (1) quality of service control; (2) user and service provider authentication; and (3) service access control. After a general discussion about security issues for distributed applications in section 1.3, we present in section 1.4 a summary of the research problem addressed by this thesis and the contributions obtained. They are further elaborated in the subsequent chapters of this thesis.

1.1. General discussion on mobile users, services and devices

1.1.1. The definition of mobile users, services and devices

To understand the requirements and challenges of supporting mobile users and services , it is necessary to study the definition and the scope of these terms. (a) Mobile services: group of services which are able to continuously provide access while users or services themselves are on the move. A conventional example is a cellular phone system in which users can dial or receive calls while they are walking on streets or riding buses. Note that no matter whether the users are moving, they are using the same cellular phones which are always connected into the same network through different access points. Since users are known to service providers, the challenges of supporting mobile services are focused on service delivery, resource management and Quality of Service (QoS) management. (b) Mobile users: On the other hand, mobile users literally mean users who have demands for services while moving. It is similar to mobile services in terms of the user mobility. However, users now are not using their own devices to access services. Instead, they will pick up whichever devices are available on site and use them. Extended from the example above, instead of using their own cellular phones while travelings, they now rent phones at the place where they visit and start using the same services as if they were in their hometown using their own devices. In this case, the service providers have no knowledge of the users when they start requesting services. To support the user mobility, in addition to the issues brought by service mobility, the challenges are then to understand the users' requirements on requested services and to provide preferred services according to their personal profiles while providing protection of privacy. (c) Mobile devices: Since services are delivered to users and also received by users through various devices, a mobile device is a device that either is on the move in order to offer a service to a moving user or is moving with a user to receive services. In the previous example where a user is using his or her own cellular phone while traveling, the phone is a mobile device used by the user to receive mobile service delivered by a tele-communication company. Another example of a mobile device is a GPS device installed in a car, in which case the same device is used to both offering allocation service and receiving location information updates.

1.1.2. Overview of distributed multimedia applications

Since the rapidly advancing technologies in computers, high-speed networks, data compression and multimedia resource digitalization, more and more service providers are able to delievers a rich set of distributed multimedia applications over the Internet to users through distributed multimedia applications, whose main task is dealing with multimedia content, in most cases, including audio and video data. Example are Netmeeting from Microsoft, video on demand from major cable television companies, and IP-telephony from various long-distance telephone service providers. The keyword *distributed* refers to the character of an application that is performed based on a group of processes which run on different machines connected over a network. These processes process, manage, deliver and synchronize multimedia data while ensuring the quality of service. Based on different characters of applications, there are various kinds of distributed multimedia applications, namely platform-based applications, content-based applications, point-to-point applications, and broadcasting applications. Note that platform-based applications are often used by users and service providers to exchange information, and are therefore also called interactive applications. Similarly, since content-based applications are often used to present specific information to users, they are also called presentational. In the following we explains each of these types, of the applications and discuss their differences.

1.1.2.1. Platform-based (interactive) applications vs. content-based (presentational) applications

The term "Platform-based application" literally means that goal of the application is to provide a working environment where users are able to communicate orally or visually or both. The content of the conversation is not provided or managed by application providers. Tele-conferencing and Internet telephony are typical platform-based applications. As mentioned above, Netmeeting from Microsoft is a point-to-point teleconferencing system which allows users to invite each other and accept audio and video communications. Users are grouped in a virtual meeting room and discuss interesting topics with enabled multi-point data conferencing, text chat, whiteboard and file transfer service. To connect others, users have to install and run the application. Once running, an underlying protocol is then used to communicate with others to build the connection of the conversation. According to the definition, the content of the conversation is not managed by the Netmeeting application itself.

Interactive applications are applications in which two or more parties are exchanging data with each other during a communication session. All participants in the application may send and receive multimedia data at the same time. Applications such as IP telephony and teleconferencing belong to this class. Due to their interactive character, there is a higher level of QoS requirement concerning end-to-end delays than for presentational applications.

Different from platform-based applications, content-based applications respond to users by delivering requested multimedia content. The content is now managed by the application. Hence, to use the services, instead of connecting to another user, users are connected to a group of content servers. A platform is still needed for users to enter and browse managed content, to be charged and to pay for the requested content. However, users tend to focus more on the quality of the content than on the performance of the platform. A typical example would be a digital cable user, who has subscribed to a video-on-demand service, and orders a particular TV program from home. He would be better served and feel more satisfied seeing a high quality movie of his choice, compared to the use of a traditional cable service where he has to watch whatever program is available at the time.

To implement content-based applications, a fundermental requirement is that they should be capable of presenting multimedia content to users, which is the unique character of a presentational application. For a presentational application, the content to present, including audio, video and text, is digitalized and stored on a group of application servers. The data exchange of the applications is a one-way movement from application servers to users. Before and during the reception of data, users are often capable of specifying the content they want and the quality of the delivery. The applications can be triggered by internal conditions, such as a predefined schedule or the status of the servers, or initiated by external requests. For instance, Internet-TV are in the former category, where the program broadcasting process is initiated by the multimedia servers, and not by the receiver as with video-on-demand applications. In contrast, video-on-demand and e-learning applications falls in the latter.

1.1.2.2. Point-to-point vs. broadcasting applications

In addition to the classifications above, distributed multimedia applications can also be grouped into point-to-point applications and multicasting ones. The difference of these applications is the relationship between senders and receivers. A point-to-point application, e.g. short message service provided by wireless communication companies, connects and exchanges data between two participants. The relationship between these two is one-to-one. Multicasting applications, however, has a one-to-many relationship. For instance, an online TV channel provides digitalized TV programs. The programs are sent from one location to as many users as connected to the server. Note the data server is considered one party while, in reality, it may be made up of a group of servers dedicated to the process.

1.1.3. Challenges to support mobile users and services

In order to support mobile users and services, there are three groups of challenges in terms of the targets of the challenges: (1) to deliver services in a distributed network, (2) to search and discover available services in the network and (3) to manage the quality of the services delivered.

To enable services that deliver multimedia content over a distributed network, the Digital Audio-Visual Council (DAVIC) [1] had worked for five years since 1994 to internationally standardize specifications of open interfaces, protocols and architectures for digital audio-visual applications and services. Later, DAVIC formed the TV-Anytime Forum [2]. To support interaction with remote objects, the Object Management Group's (OMG) [3] Common Object Request Broker Architecture (CORBA) [4] provides an underlying object infrastructure. ActiveX [5] from Microsoft is another Internet and multimedia technology. The term *ActiveX* does not refer to a single technology. Instead, it is a set of open technologies which bring the power of the personal computer to the ubiquitous connectivity of the Internet.

Given the context of mobile users and services, a user on the move may enter a local network where he or she has never been. This introduces the problem that one has no knowledge about available services in the environment. In order to provide services to users that are new to a domain, mechnisms are needed either for service providers to advertise available services or for a user to search and discover a particular service he or she wants to use.

To make multimedia applications acceptable in real life, QoS management is a very important aspect of distributed multimedia application. Vogel [6] defines the Quality of Service (QoS) as follows:

> *"Quality of service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application".*

QoS defines a framework in which the requirements of multimedia applications, their users and network infrastructures are described. According to different components in a multimedia system, the framework is made up of three layers: user layer, application layer and network layer. To measure whether and how well the requirements are matched at each layer, the QoS framework also introduced many performance parameters, such as perceived user's satisfaction at the user layer, audio and video quality at the application layer, the transmission delay, jitter, packet loss, and service availability at the network layer. The goal of QoS management is to find the right configurations and parameters for all layers to maintain the quality of the content as well as the delivery process at an acceptable or better level. At each layer, there are many existing research works addressing a number of issues.

1.2. Practical scenarios and requirements to support user and service mobility

In this subsection, three practical scenarios are considered and these requirements are discussed in details.

1.2.1. Case 1: Audio/Video conversation

While Alice is in a conference room in Paris and attending a meeting, she wants to have a VoIP conversation with her friend Alice, who is working in his hometown. After her PDA received the command, it will first try to look for an available Internet connection. The PDA will pick up several candidates, check the service evaluation data for each service provider, and compare it with Alice's preferences, which are stored at her home domain. The preference could for instance be: pick up the fastest available connection (which means maybe Alice does not want to wait more time for a connection with higher bandwidth). After selecting the Internet service provider, it will send a request to the chosen ISP asking for initiating a VoIP conversation with Bob. As soon as the request from Alice's PDA arrives, the ISP will evaluate Alice against its own policy, such like, any person from University of Ottawa with one valid reference could be assigned a *general user* role. Alice's PDA will use this role to initiate the call and the VoIP Server will check if the *general user* role has right to do so. If that is the case, the service is then provided to Alice. After the conversation is over, Alice's PDA will update the service evaluation data. In later studies, we decided to store this data in an ad-hoc device, called Reputation Server (see Section 5.2.4).

This is the most common case of distributed multimedia applications in an open network environment where a client uses his or her own device in a foreign domain to exchange data with friends. In addition to the requirements on QoS for both Alice and her friend, the extra requirements on user and service mobility in this case are: 1) clients should be able to identify themselves in a foreign domain; 2) a mechanism is needed to assign certain privileges to clients, e.g. a prearranged agreement between Alice's employer and the foreign ISP; and 3) with assigned privilege, clients can search and select desired services based on their demands and preferences.

Note that it is also possible for Alice to use a computer in the hotel to call her friend through VoIP. In this case, Alice needs to rely on other mechanisms to authenticate herself. An example of such a mechanism is the authentication protocol proposed by Dupre [7], which is studied in details in Section 4.

1.2.2. Case 2: Secure printing

While Alice is on a business trip in Paris, she wants to print her bank statement in the business center of her hotel. She gives the command to her PDA and the PDA will look for available printers. Alice has her own policy as 'only chose laser printers' and 'the service should be considered as secure and reliable by at least half of all its references'. The hotel has its own policy like 'only provide secure print service to VIP customers', in which the 'VIP customer' should be the role to access the service. Prior to printing, the printer will check if the user has a valid role as 'VIP customer'. Since Alice was assigned the "VIP customer" role, she can now use the printing service. When Alice finishes the work, she will evaluate the print service on 'whether it works in secure mode' and the print server will evaluate Alice on whether she paid for the service. If there was a problem with payment, the server will add Alice on a 'not paid' list and on Alice next visit, she will not be assigned the VIP status because of the remark that the print server provided.

In addition to the requirements summarized in the first scenario, Alice uses reputation data of the services and trust mechanism to help her to select the most preferred instance of a service among available candidates. Once an available service is found, Alice's PDA will send a request for using the service. Service providers will then decide whether to grant her the access, depending on Alice's reputation data. The introduction and dependence on reputation and trust management is due to the fact that Alice and the service providers may have zero knowledge about each other at the time when Alice enters the foreign domain and no agreement about user privileges was made in advance.

1.2.3. Case 3: Anonymous service

While Alice is on a business trip in Paris, her prescription needs to be refilled. Because she cares about her own privacy, she does not want to reveal her identity to the pharmacist online. The request is fulfilled through several steps. First of all, her PDA has to find and chose the most reliable and secure pharmacy website, which enables anonymous access. When the chosen site receives Alice's request, it has to make sure that the request is from a person to whom the prescription really belongs. Then, a payment and shipping mechanism is needed for Alice to pay for and receive the medicine with the option to hide her identity.

User privacy is introduced in this scenario. The most essential privacy information is Alice's identity. When Alice requests some service, she should have the option to hide or release her identity throughout the whole session, from request initiation to disconnection at the end. As a consequence for concealing identifications of clients, it should not compromise the accessibility of the system. The normal payment and shipping system also needs to be modified since it usually requires the client's identity.

1.3. General discussion of security issues

Many studies have focused on security threads and challenges in mobile service systems. Instead of considering the issues as a whole, many of them are only focused on one particular aspect. To summarize the issues and also give a better explanation, this subsection studies different participants in typical mobile service transactions. The study of the behaviors and characters of all of the participants will then logically introduce specific requirements for the system. Complementing these specifics, general requirements for all mobile service systems are also discussed at the end of this subsection.

1.3.1. Roles in mobile services

There are four fundamental roles in a typical mobile service scenario: user, service provider, network access provider and third parities. The division is mainly based on the difference of functionalities of the participants.

1.3.1.1. User

The term "user" refers to the participant that initiates the transaction. It could be a person or an agent that acts on behalf of him or her. In mobile service applications, the user is usually on the move and the transaction is processed through a terminal, mobile or fixed. For instance, a PDA or cellular phone carried with the user or a printer located in a hall, publicly accessible. In our scenarios, Alice is the user.

1.3.1.2. Service provider

Service provider is the participant that provides the service that maybe interesting to the User. The provider could be either an individual person or an organization offering services through their products or customer service personnel. In the context of mobile service, a computer is often needed to provide services, some of which may involve real goods. Depending on the nature of the business, services provided may be free of charge or come with a fee to be paid by users. For instance, clients could register for free at an online store and pay for delivering what they ordered. Examples of services provided in our scenarios are: the ISPs who provide VoIP services, the hotel who provides the printer, and the pharmacy store which provides online shopping for refilling prescriptions.

1.3.1.3. Network access provider

Similar to a service provider, the network access provider is an organization that provides a specific service, network access, to mobile users. The difference between both providers is that the network access provider plays a special role in mobile service environment. Since all other services depend on some network infrastructures, it is essential for users to obtain network access before requesting other services. In real life, this service is provided either free for all users, such as a wireless Internet service provided in a hotel, or at least free for limited access so that new users could establish contracts with the provider to start using the network with full privileges.

1.3.1.4. Third parties

All the other persons or organizations that participate in a transaction are considered third parties. In our first scenario, Alice, as a user, initiated the conversation over the Internet with Bob. Since Bob could not fall in any of the three categories mentioned above, he played the role of a third party.

Note that there are some participants that are not directly involved in a transaction. Instead, they act as references for users or service providers to query some information about the other parties. The following reference roles can be identified and distinguished by the type of the content to which they provide references:

- Credit reference: This role refers to a credit card association, a bank whose debit card is in use, or an organization that offers an electronic version of a monetary instrument. It provides monetary information about credit limitation of users to service providers. Based on the information, the providers will decide whether the users could afford the services they requested.

- Reputation reference: Different from credit reference, this role provides reputation data of users and services to both sides. The data shows the evaluation of a particular user or service. And in most cases, it is based on information provided by other parties that were involved in transactions with the user or service in the past. The reputation reference is often checked to build up a trust relationship or to make a decision in service selection.

- Authentication authority: this is an authority that is involved in an authentication protocol. In order to build a trust relationship between two parties in the context of mobile service, we realize that there is such a need as a third party, which is already trusted by the two and can be referred to attest the information of one party to another. This role can be played by University of Ottawa which confirms that Alice is a professor in the first scenario, while in the third one, the government of Canada plays the role by issuing Alice a passport with which Alice can visit Paris.

In brief, there are four different roles participating in a typical transaction. Although their functionalities and characters of behaviors are different, two or more roles may be located on one machine. For instance, the network access provider, say Bell Canada, is also offering services other than Internet access, which include email and web hosting services. In this case, the company plays not only the role of network access provider but also the role of service provider. These additional services are referred to as add-on values and are very cost-efficient practices for the company to enrich the variety of its service products. In turn, it will improve the loyalty of existing customers and attracting new customers.

1.3.2. Specific security requirements for mobile service

Having studied different functionalities and the character of behaviors of each participant role in a typical mobile service scenario, it is very logical to identify certain security requirements. There are three fundamental issues that need to be addressed in order to successfully process a transaction: authentication between all participants, access control over available services and payment validation to make sure that the charge is in good standing.

1.3.2.1. Authentication

The first specific requirement of a mobile service system is to authenticate the identity of each participant. Before being allowed to access desired services, users often need to be authenticated by service providers. The authentication practice is essential partially due to prevent services being abused because once authenticated, a user cannot deny he or she used the service. In contrary if a user can act as if he or she is another person, it is reasonable to assume that the user may try to take advantage of the service provider, knowing that all consequences will fall on the other person. In addition, the other two requirements, access control and payment validation, are often linked to the identity, which should be examined before any process. More often than not, users prefer to authenticate service providers to make sure that they are dealing with trustworthy organizations or individuals before they provide additional personal information such as birth date, social insurance number or credit card information. In addition to the authentication between users and service providers, mutual authentication is often required between users and any third party that is involved. This is the case in the first scenario when Alice and the chosen ISP are authenticating each other, both of them are also authenticating Alice's H.A. Lacking this procedure may result in attacks from a third party that masks with Alice's identification and authenticates itself with the help of a fake HA employed, which makes the ISP to believe that the third party is in fact Alice.

1.3.2.2. Access control

The issue of access control can be further divided into two challenges: assign users proper access rights and control the access based on the access rights of the users. Note that throughout the thesis, the two phrases "access right" and "privileges" are used interchangeably. Both of them define which subject usually a user, can perform what actions on a particular object, usually a resource offered by the service provider.

Once authenticated, service providers usually assign users certain rights based on predefined policies. The challenge is that in the context of mobile services, a user may be completely new to a service provider. Hence there is no proper base for service providers to decide what kind of rights a user can be assigned. A very limited privilege set may upset a user and potentially result in losing business while an overly generous privilege often puts the service provider in the risk of failing to collect payment.

After a user is assigned certain rights, the challenge in turn is for the service providers to control the access based on the user's rights. In the first scenario, Alice's right of being allowed to initiate conversations over IP through the ISP is associated with the role of *developer* and stored in Alice's PA. The role is presented to the ISP and validated when Alice wants to have a VoIP communication with Alice. In the context of a mobile service, a public key authentication certificate may be used to include such information. Note that many services are provided to anonymous users, that is, the service provider does not need to authenticate the real user. Instead, an account that is designed in the way that no personal information will be exposed is created when the user registered and assigned certain rights. Whoever uses the account to request the service will be served.

1.3.2.3. Payment validation

Excluding services that are provided free of charge, services offered in a mobile environment are paid based on usage or a one-time fee. For instance, users that browse the Internet using their cell phone will be charged based on the volume of the data traffic during the browsing, while a one-time charge may be applied when a user is registering with an authentication authority. The service charge is paid by users through two methods: cash-based or credit-based. The former requires verifying the cash instrument or the withdraw limit of a debit card, while the latter involves a credit institution which makes sure that the due amount is within the user's credit limit and will be paid to the service provider. Given the difficulty to cancel the payment made through the first method, credit-based payment is regularly used in the environment.

1.3.3. General security requirements for mobile service

In addition to specific requirements to support mobile users and services, there are also various generic requirements for every mobile service system, including privacy, message integrity, denial of service, verifiable signature and, optionally, anonymity. Each of these is explained in the following subsections.

1.3.3.1. Interception and Violation of user privacy

Interception defines the process performed by an unauthorized third party to monitor data or traffic on networks or systems without obtaining permission from the owner of the data, as shown in Figure 1. The third party that exists in the middle has access to all the data that is exchanged between the Client and the Server. Since it is not necessary to substitute the content of the traffic, both sides of the link often do not realize the violation of private communication. The attacker could then take the valuable content and conduct actions which may be potentially harmful.

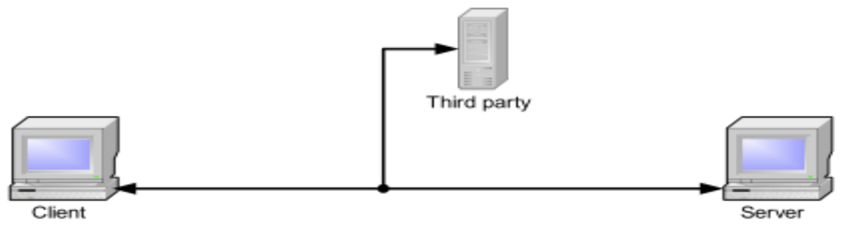


Figure 1: Interception

Giving the difficulty of detecting and preventing this type of attack, common solutions focus on encrypting the content so that in such case, the third party faces only non-sense data rather than some meaningful plain-text.

1.3.3.2. Integrity of message exchange

Message integrity means messages should not be modified when they are exchanged between the parties involved in a transaction. There are two common causes which violate message integrity, transmission errors or intruders. The former case normally happens due to poor network quality while the later is a well-known thread, as shown in Figure 2.

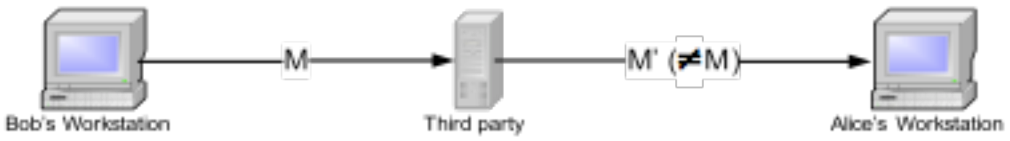


Figure 2: Message Integrity

In order to protect messages against intruders, digital signature and hash function are most common practices.

1.3.3.3. Denial-of-service attack prevention

A denial-of-service attack (also, DoS attack) is usually conducted by overloading a computer system or the network of the system so that the victim system cannot perform properly its intended behavior. For example, through generating and sending enormous number of requests, all available resources are quickly drawn off and nothing is offered to normal users. Given the fact that it is not possible for any service provider to provide un-limited resources, this kind of attack is most difficult to prevent. Instead of preventing, most solutions are to dedicate the requests to different places and average the payload throughout the system so that no bottleneck is formed.

Although the limited network bandwidth is often the target for the DoS attack, it is worth mentioning that there are other places in a transaction that could also be the target of a DoS attack. For instance, in section 4.4, the thesis demonstrates how the violation of message integrity results in the fact that service requests from normal users could be refused by service providers.

1.3.3.4. Verifiable signature

In some important transactions, messages or documents are required to attach a signature of the participant that created, sent or received them. The signature of a participant only makes sense when it can be verified by others. The verification of the signature makes sure that the content is not substituted once sent or upon reception. In the third scenario, a doctor's signature is required on the prescription of Alice. The signature should be verifiable in the sense that: 1) the prescription is not modified or replaced after the doctor signed it; and 2) the doctor who signed it cannot deny (repudiate) thereafter the fact that the prescription was issued by him or her.

1.3.3.5. Anonymity

Many services provide the option of anonymous usage. Moreover, anonymity gains priority on the list of user preferences or requirements in certain situations. Users tend to be more sensitive when the transactions are related to their health conditions or foibles. For example, users may not want others to know what special pills they are buying or they may not want to be recognized while visiting certain places. While allowing anonymous access to mobile service applications, service providers still need to verify access rights of an anonymous user and validate the payment he or she made. Hence it is essential that such references are provided without interfering with the user's anonymity.

1.4. Problems addressed in this thesis

Throughout the study, we first identified requirements to support mobile users and services. They are (1) to continuously deliver services to a moving user, (2) to secure the transactions between users and services, which includes the authentication of users and services to each other, and control service access against abuse. This thesis provides solutions to satisfy each of these requirements, which are described in the following chapters. In summary, the main contributions of this thesis are the following:

(1) Implementation of a prototype system to support service discovery, QoS adaptation, and continuous service delivery while the user is moving and changing devices.
(2) Improvement of Dupre's authentication protocol [7] to eliminate the thread of interception and to enable user anonymity.
(3) Proposal of a trust-based access control framework for mobile users and services to support online transactions, including service discovery, authentication, authorization, and access control.

The thesis is organized as follows: after this Introduction, Section 2 presents an implementation of the prototype system. It starts with explaining the purpose of the system and the challenges of the implementation; it introduces the concepts used to support mobile users and services, such as Home Agent, Personal Agent, and the proposed network infrastructure. The section then gives an overview of different technologies for service discovery, including Bluetooth [8], Jini [9], and Web Services [10]. The SIP protocol is used to initiate the transaction session and the Java Media Framework (JMF) [11] is used to capture and deliver multimedia data. SDP [12] from BlueZ is used on the Personal Agent to discover services around a user. The system was tested and the performance analyzed. Section 3 provides a general discussion about existing authentication schemas and protocols, access control schemas and trust models. Section 4 discusses the requirements of authentication in the context of mobile services. Dupre's secure authentication protocol is introduced and several weaknesses of the protocol are identified. We then propose some improvements for the protocol and comment on the detailed design of the protocol and how it protects against typical security attacks.

Combined with all the above studies, Section 5 presents a trust-based access control framework for mobile services. It explains the proposed system structure which includes several components such as Home Agent, Foreign Agent, Personal Agent, Reputation Server, Policy Decision Point, Service Server, and Service Directory. We explain the proposed system behavior with the help of several sequence diagrams. At the end, we briefly discuss several technical choices to implement a prototype system. This includes: (1) the choices of presenting security and trust-related information digitally to make it exchangeable and understandable by computers; and (2) the choice of service discovery tools. Finally, the thesis ends with a conclusion and discussion of future work in Section 6.

2. A prototype system implementation to support mobile users and services

With the advance in technologies, including embedded systems, short-range wireless networks and personal computing technologies, there is growing interest in providing personal and service mobility for persons roaming in ubiquitous computing environments. Although a number of architectures [13, 14, 15, 16, 17] have been proposed to support user mobility through the concept of a Directory Service using multiple telecommunication devices, such as PDAs, laptops, and cellular phones, they all failed to satisfy the requirement of support personal mobility in ubiquitous computing environment because:

- The information in the Directory Service is static,
- there is no support for dynamic service discovery,
- they lack support for service mobility,
- they lack support for QoS management for complex (combined) services.

To overcome these difficulties, we propose an architecture that dynamically triggers service discovery and service selection that controls service mobility and the user's policies and preferences through a Personal Agent (PA). The proposed architecture uses Bluetooth to construct the wireless personal area network (WPAN); leverages the service discovery protocol (SDP) to discover services available just in the WPAN of the roaming user; and uses a QoS negotiation and selection algorithm to select the services that best suit the context and preferences of the user. In this section, we present an implementation of a prototype system that applies the proposed architecture and describe the usage scenario which was used to determine its performance.

This section is organized as the following: First, we review a number of technologies that are involved in building the system, such as concepts of Home Agent (HA), Personal Agent (PA), service discovery mechanisms, communication protocol, and multimedia content delivery tools. We then explain how these technologies are applied or adapted to build the prototype. Following that, we present the structure of the prototype system, including the hardware platform and software components. Finally, a typical usage scenario is described and used as the test case to carry out system analysis in terms of performance.

2.1. Overview of recent technologies on related challenges

2.1.1. *Concepts to support mobile users and services*

In our research work on quality of service management for distributed multimedia applications and mobile users [18], we introduced the concepts of the Home Directory, which store the user profile and preferences, and the Personal Agent, whose responsibility is to detect devices in the vicinity of the user as well as to manage the user's communication sessions. Bluetooth, a short-range technology, is adapted to construct a WPAN (wireless personal area networking) around the user.

2.1.1.1. *Home Directory*

The home directory could have three main functionalities: a) it could act as a proxy agent for a user, that is, it could re-direct communication requests since the user profile would have the information about how the user could accept such a request at the given time; b) it could also be the place where the user stores preferences about desired services. For instance, he may mark the calls from clients as highest priority and prefer better quality for the connections while accepting higher charges for services he requested; in contrast, for the calls from friends, a cheaper connection option with fair quality may be satisfactory; c) it could be sensible, with a few modifications, to play the role of an authentication authority in a proposed access control framework, which will be discussed in detail in Section 6.

Additional information such as resource information can also be stored by the user's home agent. However in the context of mobile user and services, the HA is not able to keep track of available resources around the user while he or she is moving around. In this case, a Personal Agent is used.

2.1.1.2. *Personal Agent*

To support user and service mobility, a Personal Agent is introduced to address several issues. Since users may continuously change their locations, the set of reachable services changes accordingly. Manual updates of the information about currently reachable services are clearly not practical. To keep this information up to date, a service discovery program can run on a Personal Agent to periodically search for surrounding services. The PA often runs on a handheld device, such as a PDA (personal digital assistant), so that it can be carried by a user while he or she is on the move. The PA should also satisfy some communication requirements, such as Internet connection and capability to construct a WPAN (wireless personal area networking, see Section 2.2.1.3 for more details).

Corresponding to the different functionalities it must perform, a Personal Agent could be logically divided into five major components: a Communication Agent, a Service Discovery Agent, a User Activity Monitor Agent (UAMA), a QoS Selection and Negotiation Agent (QSNA), and a Service Registry.

The Personal Agent automatically performs service selection on behalf of the user according to the user's preferences and does periodical searches for available service, in order to maintain an up-to-date set of services within the user's vicinity.

2.1.1.3. Network infrastructure

A wireless personal area network (WPAN) is a personal, short-distance area wireless network for interconnecting devices centered around an individual person's workspace. The range of the network is typically a few meters. WPANs address wireless networking and mobile computing devices such as PCs, PDAs, peripherals, cell phones, pagers and consumer electronics. WPANs are also called short-distance wireless networks.

The proposed architecture uses the short-range Bluetooth technology to construct the user's WPAN. This decision is made because of two considerations: a) According to the specification of Bluetooth technology, the distance to connect two parties ranges from zero to ten meters. Within this range, users should have a good chance of audio and visual access to services. In contrast, a monitor that is one hundred meters away from user should be considered unreachable, even though it could be detected through a WiFi wireless network; b) Bluetooth technology is designed to run with minimum power consumption. This makes it more practical when it is used to run on a PDA.

2.1.2. Service discovery

To implement the prototype system, another challenge is to find and make use of available services in our case, these are multimedia receivers to accept multimedia content for presentation. The challenge is to discovers available services in network environments. In addition to service discovery, there are other considerations, for instance, choosing services among rivals and accessing services on the user side, and advertising services, managing connection sessions, and billing for service usage on the server side. There are various technologies being developed to automatically search available services in an open network, including the Service Discovery Protocol (SDP) for Bluetooth, Jini by Sun Microsystems, and most recently, Web Services. Different from other technologies discovering services in a traditional network-based environment, SDP provides an application capability of searching available services and query characteristics of the services in the Bluetooth network environment, where service availability is dynamically changing based on network range and users motion. Compared to SDP, the Jini technology introduces a system to create, discover, and access services while ignoring the infrastructure of the network. This means that it does not matter whether the network is based on Bluetooth, GSM, IEEE 802.11 or others. The Java-based characteristic brings the advantages in code security and system compatibility. Based on XML (eXtensible Markup Language), Web Services defines a standardized way of integrating Web-based applications over the Internet. Its component includes communication protocols (HTTP, SMTP, etc.), SOAP (Simple Object Access Protocol), XML Schema, WSDL (Web Services Description Language) and UDDI (Universal Description, Discovery and Integration). The following discuss in details each of these technologies.

2.1.2.1. Bluetooth Service Discovery Protocol

Bluetooth technology introduced low cost, low power consumption, and a short range wireless network into real life. The nature of the network technology determines that the set of available services changes dynamically. This unique characteristic makes service discovery processes different from those in conventional networks. SDP is developed to standardize the processes and is optimized for Bluetooth communications. Unlike Jini and Web Services, SDP only addresses the method to discover while excluding methods for accessing services. Other service discovery protocols, such as Jini, or protocols defined over Bluetooth can be used to access services in a Bluetooth network environment, after being discovered through SDP.

As shown in Figure 3, an SDP enabled system contains two functional components: An SDP server and an SDP client. A SDP server contains all information about server applications while a SDP client is used by client applications to query existing services. The messages exchanged between the SDP server and client are SDP requests and SDP responses.

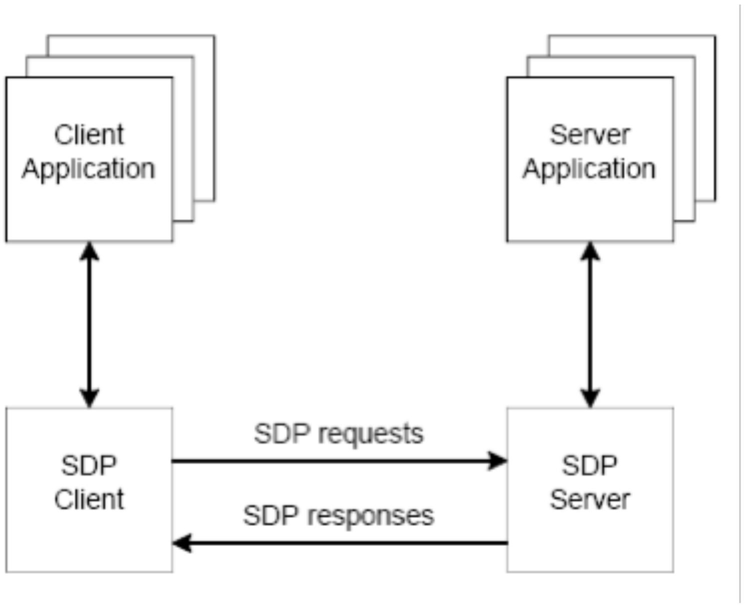


Figure 3: SDP overview [19]

An SDP server manages a list of services running on a Bluetooth device. All the information about available services is kept in service records. Its functions also include responding to queries from SDP clients. On the other hand, an SDP client can query services managed by SDP severs. There are two ways to query services, to search or to browse, which will be discussed later. According to the specification, an SDP server and client can both run on a single Bluetooth device, having different functionalities, as which the Bluetooth device acts depends on whether it requests a service or it serves clients' requests. However, only one instance of the SDP server or client can run on one device at a time. In the case that more than one service application is running on a device, the SDP server instance will act on behalf of all theses service providers to respond to client requests. In the same way, an SDP client will be used to query multiple service applications.

- *Service class, service record and service record handle*

A service class is an abstract class which defines the attribute set of a service, the format of these attributes and how to use them. Each service is an instance of a service class. Every class is assigned a unique identifier which is represented as a UUID (a universally unique identifier that is identical across all platforms and all times.) It is very important to assure the correct value of the class identifier of a service instance, since all service attributes of the instance depend on this value to be queried or updated. Given the importance of the class identifier, it is recommended that the identifier of a class instance should be examined or verified before using any class-specific attribute. When a new service class is defined, it is always a subclass of another service class, excluding the most general one. This gives a new class the capability to inherit common attributes from the parent class.

For an SDP server to manage running services, a service record is created and maintained. The service record contains all the information about the service in the form of a list of attributes. Different from a service class, a service record is more like an instance of a service class, except that a service record does not only hold attributes that are specific for the service class, but also common ones that every service may have.

To query multiple service records within an SDP server, the specification introduces the concept of service record handle. It is defined in a form of a 32-bit number so that a service record can be uniquely identified. Note that even though a handle represents a unique service, the handle does not have to be unique in a multiple-SDP-server environment. If it happens to have two or more SDP servers, a handle from other servers will not be meaningful to the local SDP server. As an exception, there is one service record handle which is understandable by all SDP servers. This service record handle represents the SDP server itself, which means it specifies attributes for the SDP server and the protocol it supports. For instance, it has one attribute to list all the SDP protocol versions supported by the server.

- *Service update*

The service update process is different on the server side, compared to the client side. In the former case, all services are running locally. When a service is started or stopped, it can simply notify the SDP server so that a service record can be created or deleted to reflect the change. The same procedure can also be used to update attributes for a particular service.

On the client side, there are additional issues to be considered in order to update services. First of all, because the set of available servers changes dynamically in a Bluetooth environment, an SDP client needs not only to update service information but also information of servers on which these services are running. When an SDP server is created, all of the SDP clients should be aware of the changes. However, the protocol for the notification is not covered by the SDP specification. Instead, it suggests that a third party mechanism is needed for this purpose. Similarly, if a server becomes unavailable, there is no broadcasting message defined in the SDP to notify all the SDP clients. However, differently from the case of creating a service, there is a passive way for the clients to learn the changes through the fact that no response from the server is received when they query services on it.

Once a set of available SDP servers have been learned by SDP clients, they then could construct SDP requests to query desired services. There are two predefined formats that can be used, one for searching and another for browsing.

An SDP client can query particular attributes of a desired service through the service handle of the service record that contains all information of the service. In order to retrieve the service handle, the SDP defined a so-called *Service Search Transaction* for SDP clients to search the handle of a specific service record based on the values of attributes that can well represent the service. Note that instead of being capable to search based on any attribute, an SDP client can only search the attributes whose values are UUID. The Service Search Transaction defines two messages, an SDP_ServiceSearchRequest to specify values of attributes that clients are expecting and an SDP_ServiceSearchResponse to return the service record handles of service records that match the given parameters.

Usually, SDP clients search services based on some desired attributes, which are represented by UUIDs. There are also other cases when instead of searching for particular services, SDP clients want to find out what type of services are offered in a network without any related information. This kind of information can be found through service browsing. The hierarchy structure of service classes benefits the implementation of the browsing process. Shared by all service classes, BrowseGroupList is an attribute to hold a list of UUIDs in a hierarchy system. Each UUID represents a browse group which is associated with a service for the purpose of browsing. To initiate a browsing process, a client first creates an SDP_ServiceSearchRequest message containing a UUID which is associated with the least specific service class. The message is sent to the SDP server to match all the managed services. The rest of the browsing is similar to the searching process which is explained above.

2.1.2.2. Jini

Developed by Sun Microsystems, the Jini technology is an open architecture that enables the dynamic creation of network services. It extends the Java application environment from a single virtual machine to a network, which may include a number of machines. The nature of the Java-based technology brings the major advantages such as code mobility, build-in security, and strong typing. As a drawback, it brings the dependence on the Java application environment. Although the Jini technology does require some memory and processing power on devices, those without processing power and memory may be connected into a Jini system under the control of another device.

A Jini system is a system that is made up of Jini technology-enabled services and/or devices. These services can be found and accessed by users within a network. Therefore it is possible to create a flexible administration tool to manage all available resources that are provided by the Jini-enabled services. Resources may be hardware, software programs or a combination of both. Therefore a Jini system consists of three parts: an infrastructure provides mechanisms for devices, services and users to join and detach from a network; a programming model that supports the creation of distributed services; and a search mechanism through which resources can be searched and accessed by other members of the network.

The heart of the Jini system is a set of protocols called discovery, join, and lookup. The discovery process occurs when a new service provider or client enters a network. A service provider is the container of the service, which is either a device or software or combination of both. The service is presented in the form of a Java object with attributes and an interface, which will be used by other users and applications to invoke the service. During the discovery process, the service provider or client multicasts a request on the local network for any lookup service to register. Once found, the service performs the join operation. After that the lookup service has an instance of the service object (proxy) with its defined attributes and the service is ready to be looked up and used. Using the lookup operation, a client locates the service by its interface (written in Java), along with the service description and attributes. To access the service, the client will then use the local service object interface that is loaded during the lookup operation.

Unlike Bluetooth technology, the nature of Java technology brings unique qualities to the Jini technology, such as code mobility, protocol agnostic, and leasing. It can be implemented on top of any hardware or software as long as there is a compiler understanding Java.

2.1.2.3. Web Services

Web Services is a framework made up of a set of XML technologies, which can be used to expose applications to the Internet and to merge web-based applications for distributed computing. Aaron Skonnard [20] gave the following description:

> *"... a more precise definition for the term Web Service might be an application component that:*
>
> - *Communicates via open protocols (HTTP, SMTP, etc.)*
> - *Processes XML messages framed using SOAP*
> - *Describes its messages using XML Schema*
> - *Provides an endpoint description using WSDL*
> - *Can be discovered using UDDI"*

This definition outlines all important components involved in implementing web services. As for Jini, an important benefit brought together by these components is the capability for applications to interoperate. Through publishing and advertising themselves on the Internet, applications can be accessed or reused by other applications or end users. For instance, a stock trading company may build a web application for its clients to query the performance of a given stock. If the application is built on web services, a third party application can be developed to access the application automatically and save the information on the client's computer for reviewing. With the help of the two applications, clients can set up a schedule at a desired frequency to update the information about their concerned stocks. Furthermore, the company who owns the application can also reuse it as a building block to develop a client-side application, which allows users to subscribe background information and analysis about some specific stocks.

Among the five components of web services, the communication protocols, HTTP and SMTP are widely used over the Internet; and message formatting, e.g. XML schema, are one of the fundamental standards of XML technologies. The other three, in contrast, are more specific and devoted to web services. Hence, detailed discussion will be given in the following.

- *SOAP*

SOAP stands for Simple Object Access Protocol. Although at the time of creation it is used to move objects over the Internet, the focus of SOAP has soon been moved towards building a generalized framework for XML messaging. Hence, the SOAP 1.2 specification describes itself as:

> "*SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics."[21]*

There are three major characteristics outlined in the definition above: lightweight, extensible and independent of any programming model.

Lightweight means simplicity. It is a key benefit for SOAP to be adopted in web services, and "if there's one thing we've learned from the Web, it's that simplicity always wins over efficiency or technical purity" [22]. Given the fact that it does not provide security, routing, and reliability, its requirements are rather simple and clean. As long as you have formatted messages and enclosed them in SOAP elements, you got a SOAP message. Similar to HTTP, a SOAP message has distinct parts for control information and message payload, named header and body. The former is optional while the latter mandatory.

SOAP is extensible in two directions, its functionality and the underling transport protocols. Although the SOAP specification only defines an XML messaging framework, extra functions can be added later as layered extension on top of an existing implementation. For instance, Web Services Enhancements (WSE) 1.0 Service Pack 1 from Microsoft provides advanced capability for web services to support security features such as encryption and digital signature. In addition to its extensible functionalities, SOAP can run on nearly any transport protocol, with the help of its envelope element. When a SOAP message is traveling from a SOAP sender to a SOAP receiver, several intermediary nodes that run different protocols may sit between the sender and the receiver and intercept the message. To maintain high level of interoperability, however, concrete protocol binding for each protocol is desirable. Since HTTP is so popular, the SOAP specification defines an explicit HTTP binding since version 1.1.

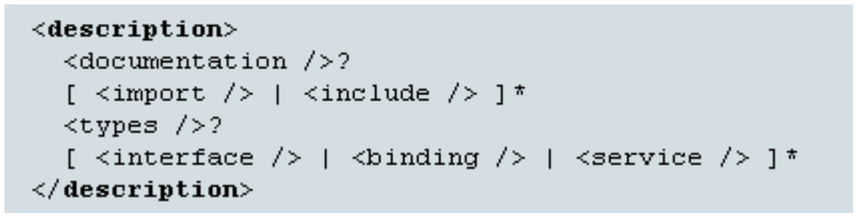
SOAP is also independent of any programming model. It defines a one-way SOAP message-processing model and this processing model defines rules about how a SOAP message should be processed as it is transferred from a source to a destination. To implement the message exchange pattern of a particular programming model, several SOAP messages can be sent between the source and the destination. For example to implement a request/response message exchange pattern, a SOAP message can be built as a request message at a sender, while at a receiver, another SOAP message can be made as a response message and sent back to the sender. Similar to this, other message exchange patterns include solicit/response (the reverse of request/response), notifications, and long-running peer-to-peer conversations can also be implemented. These patterns can then be used to build different programming models.

Armed with these advantages, SOAP becomes largely accepted for building web services as an XML messaging framework.

- *WSDL*

WSDL stands for Web Services Description Language. It is a language in XML format that is used in combination with an XML schema [10] definition to describe web services. The description covers all information about how to interact with a web service. Although it can describe the structure of XML messages, an XML schema definition currently cannot tell the relationship among the messages. For instance, an XML schema can define what a request and a response message should be. However, it cannot specify the condition in what case, a response message is required when a request message is received. WSDL fills this lack of capacity by introducing concepts of operations and interfaces to operations, each of which is a combination of multiple messages.

Each WSDL document includes an optional *documentation* element, zero or more *include* and *import* elements, an optional *types* element and zero or more groups of *interface*, *binding* and service element. Figure 4 shows a pseudo-content model of service description.



```
<description>
  <documentation />?
  [ <import /> | <include /> ]*
  <types />?
  [ <interface /> | <binding /> | <service /> ]*
</description>
```

Figure 4: WSDL structure [23]

The *description* element is the top-most element of a WSDL 2.0 document. This is the container of the rest and is used to declare namespaces that will be used throughout the document. The optional *documentation* element is used to explain the service in human-readable text. To reuse existing XML schema as message types, zero or more *include* and *import* elements can be included. The optional *types* element is used to define the message types directly within the WSDL document. *Interface* elements are declared directly inside the *description* element. A WSDL 2.0 *interface* defines the abstract interface of a Web service as a set of abstract operations, each operation representing a simple interaction between the client and the service. A *binding* element specifies concrete message format and transmission protocol details for an *interface*. The *service* element specifies at which URI the service can be accessed.

Having all information about services, including messages, operations, interfaces, and bindings, a software tool is then capable to automatically generate code that implements the given interface. With the help of WSDL, services can now be accessible by any participant who follows the specifications.

- *UDDI*

Implemented by SOAP and WSDL, available web services can be accessed by clients through third party applications. "How to find these services on the Internet?" in turn becomes the next question to ask. A UDDI (Universal Discovery Description and Integration) registry is then proposed for each service to publish its information including the kind of service it offers and how to access it. UDDI provides an interoperable infrastructure, on which both public and internal services can be published on the Web and accessed by desired parties.

The UDDI specification [24] by OASIS (Organization for the Advancement of Structured Information Standards) defines the data structure of a UDDI registry shown in Figure 5.

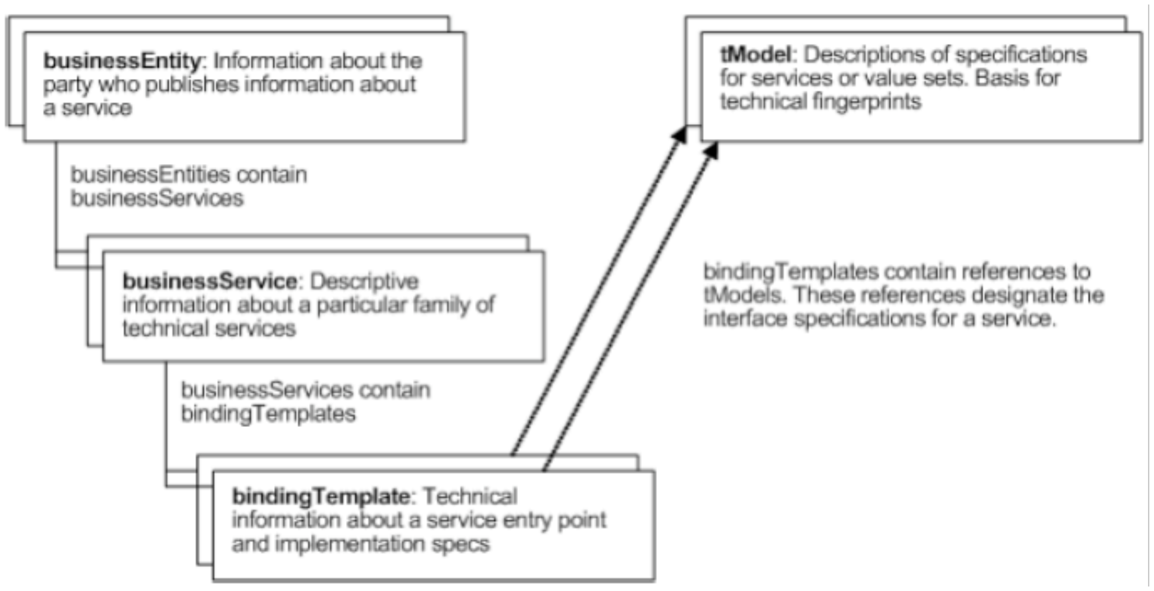

businessEntity: Information about the party who publishes information about a service
businessEntities contain businessServices
businessService: Descriptive information about a particular family of technical services
businessServices contain bindingTemplates
bindingTemplate: Technical information about a service entry point and implementation specs
tModel: Descriptions of specifications for services or value sets. Basis for technical fingerprints
bindingTemplates contain references to tModels. These references designate the interface specifications for a service.

Figure 5: UDDI core data structure (from [24])

From this point of view, the top-level data structure is a businessEntity, which holds the description of a business or organization. Each businessEntity contains a set of businessService entries, which describe logical services offered by the business or organization. Similarly, each businessService contains many entries of bindingTemplate. Each instance of the bindingTemplate provides the technical description of a web service, e.g. the access point of the service in the form of URL. Each bindingTemplate contains references to tModels. These references describe interface specifications for all registered web services.

The UDDI protocol is designed to be one of the fundamental components to create a standard platform, through which clients and applications can dynamically search and access published services over the Internet. However in practice, because the specification of a web service implementation is usually known to users, they can chose not to use the protocol. Instead, hard coding the location of the service is often used.

2.1.3. The Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) is a signaling protocol used for establishing sessions in an IP network. Similar to HTTP, SIP works in request-to-response mode and the messages are human readable. Consequently, Internet applications could easily adapt SIP to build a superset of the telephony call processing functions and features. That is why the most common applications of SIP currently are Internet conferencing, events notification and instant messaging.

Because the main interest of SIP is to setup a communication session over IP networks, it works in concert with several other protocols and is only involved in the signaling portion of the session. For instance in order to provide high quality telephone services, SIP works with RTP (Real-Time Transport Protocol for voice and video data transfer), the authentication protocol RADIUS, and possibly with the service directories access protocol LDAP or the resource reservation protocol RSVP.

2.1.3.1. SIP protocol overview

There are two basic components within SIP: the SIP user agent and the SIP server (Figure 6). The SIP user agents are the end point of a SIP call and the SIP network servers are the network devices that provide additional features to the top of basic SIP calls. An example of these add-on features is to proxy and redirect requests.

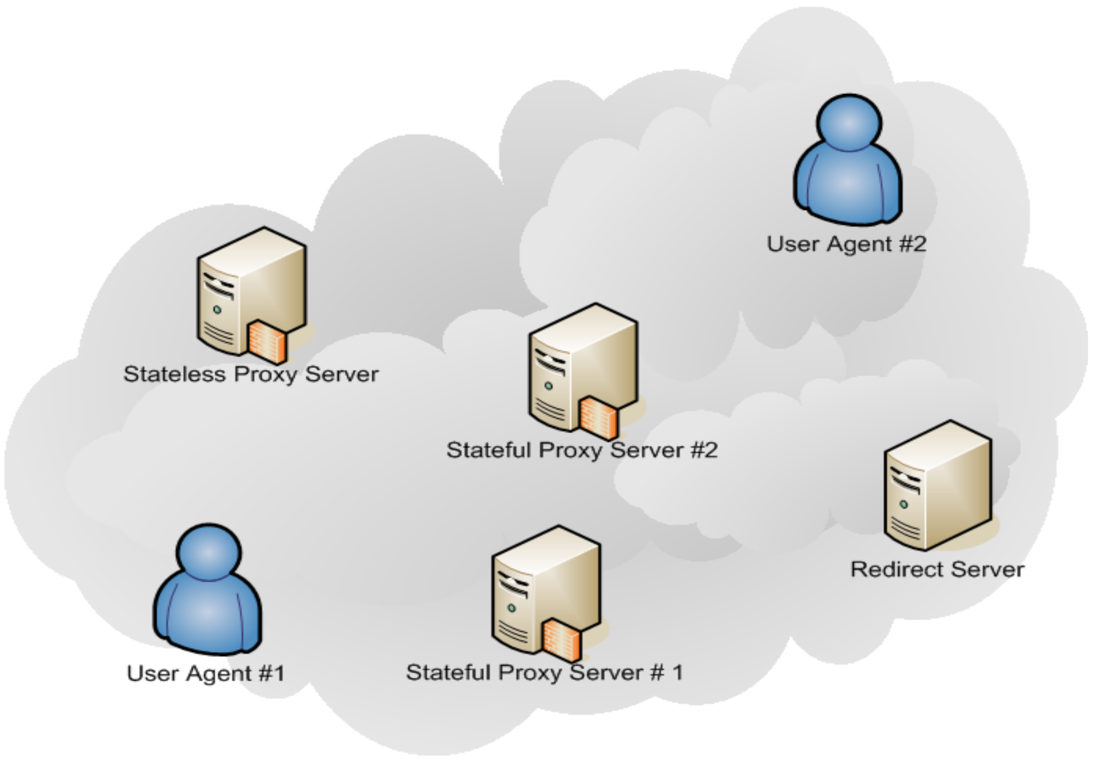


Figure 6: SIP protocol architecture

The SIP user agent itself often acts as two roles: the User Agent Client (UAC) and the User Agent Server (UAS). The UAC starts calls while the UAS answers the calls. When a user agent is idle, it acts as a UAS which can receive calls. Before the agent initiates a call, it switches to the UAC and remains as that until the call is finished. This makes it possible to use a client-server protocol to implement peer-to-peer calls. Depending on the resources of the hosting device, a light-weighted SIP user agent can be used on hand-held equipments such as PDAs or conventional desktop while more sophisticated user agents can be used to provide addition features, like conference call or call forwarding.

The SIP network servers are devices existing on networks between user agents. These devices provides name resolution so that it is not necessary for a SIP user agent to know the IP address or URL of another user agent before a call to the user agent is initiated, which is likely the case in an open network environment. A SIP network server can operates in either stateful or stateless mode. The difference between the two modes is that the SIP network server in the former mode remembers all the requests processed along with the subsequent responses while the latter forgets all the information once processed. An example of a stateful server is a proxy server that connects a local area network LAN to the Internet and has knowledge about user agents in the LAN. On the other hand, the SIP infrastructure is made up of stateless servers, which pass SIP calls from one user agent to another like routers on the Internet.
Other function that provided by the SIP servers is redirection. When a redirect server receives requests, it sends a response indicating the location of the destination user agent instead of passing the call onto the next server.

Additional features that are significant to SIP are: (1) To split or "fork" an incoming call so that several extensions may be rung at once. The first extension to answer takes the call. This feature is handy if a user works at two locations and wants to receive SIP calls at both places; (2) To return different media types within a single session. For example, a customer could call a merchandise retailer, view video ads of a interesting product, complete an on-line ordering form and pay the price - all within the same communication session.

2.1.3.2. SIP protocol in detail

The SIP protocol works in request-response mode. The requests can be used to invoke predefined SIP methods. A basic set of methods proposed by [RFC 3261] are the following:

- INVITE: is used to send an invitation from a user to another one in order to create a session between the two parties.
- ACK: is used to acknowledge the reception of the answer to an INVITE. It ensures reliability of the message exchange process.

- BYE: is used to disconnect a session and release occupied resources when the conversation ends. It is also used to refuse a call from a caller.
- CANCEL: is used to terminate a request that was sent out previously, which may have been a call invitation, a search for a user or other actions.
- OPTIONS: is used to exchange the information about the capacities of a SIP server.
- REGISTER: Registers the address listed in the "To" header field with a SIP server.

Upon receiving a SIP request, a SIP server or user agent will process the message and execute certain actions. The result of the execution is then returned to the sender in the form of SIP responses. The following are SIP response codes:

- 1xx Informational (e.g. 100 Trying, 180 Ringing)
- 2xx Successful (e.g. 200 OK, 202 Accepted)
- 3xx Redirection (e.g. 302 Moved Temporarily)
- 4xx Request Failure (e.g. 404 Not Found, 482 Loop Detected)
- 5xx Server Failure (e.g. 501 Not Implemented)
- 6xx Global Failure (e.g. 603 Decline)

Figure 7 shows an execution sequence of the SIP protocol in order to create a simple communication session between user agent #1 and user agent #2. In the following we give an explanation of the messages:

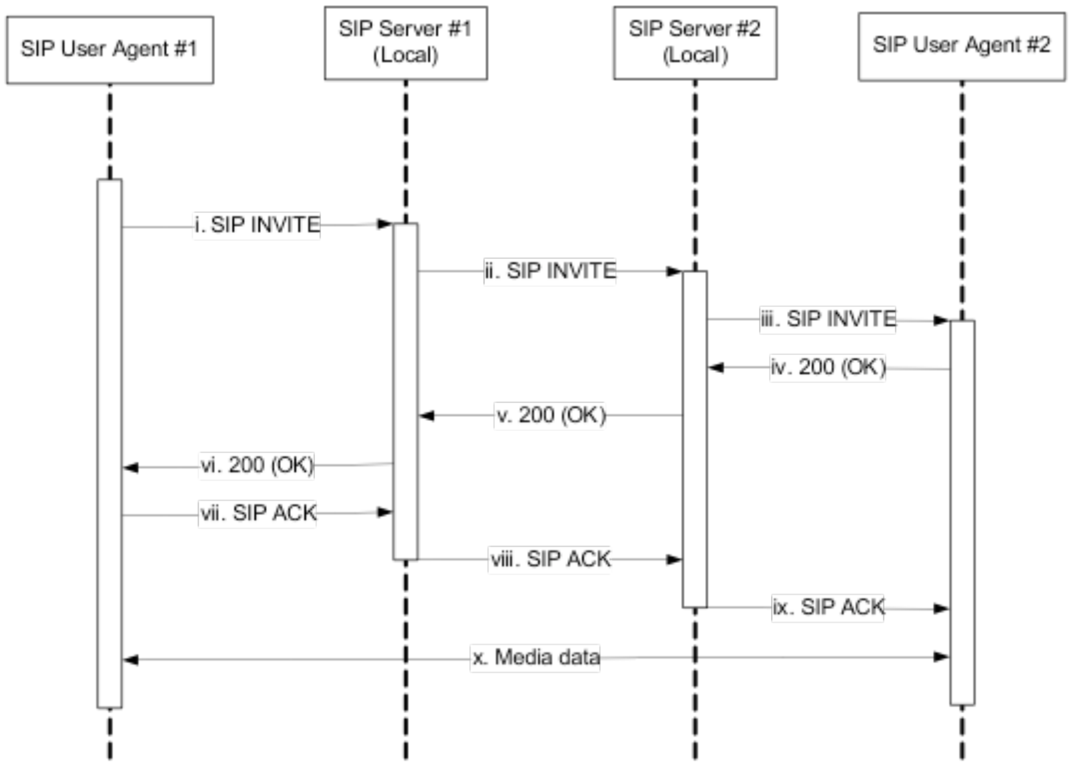


Figure 7: Message exchange sequence in a simple SIP call

a) Message i: To initiate a session, the caller (or User Agent #1) sends a SIP INVITE request with the SIP URL of the called party.

b) Message ii - iii: The request is sent to a locally configured SIP server,called SIP server #1, at which the agent is registered. The server will attempt to resolve the called user's location and send the request to them. There are many ways this may be done, such as searching the DNS or accessing databases. Alternatively, the server may be a redirect server that may return the called user location to the calling client for it to try directly. During the course of locating a user, one SIP server may proxy or redirect the call to other servers until it arrives at a server (here called SIP network server #2) that definitely knows the IP address where the called user can be found. This server then will forward the SIP INVITE to the user agent of the called user. Note if the client knows the location of the other party it can send the request directly to its IP address.

c) Message iv - vi: If the user agent #2 decides to accept the SIP INVITE request, it sends back a SIP 200 OK response message, which is then forwarded through SIP server #2 and SIP server #1 to the user agent #1. Note that if the user agent #2 refused the call, the session may be redirected to a voice mail server or another user following the policies that are pre-defined and stored in the SIP server.

d) Message vii - ix: To ensure the quality of the message exchange process, an SIP ACK message is sent from the origin agent to the destination agent. Upon the reception of the message, both agents now have confidence that the agent at the other side will answer the call when they try to establish the voice data transfer in the next step.

e) Message x: a connection is now established. The media data can now be transmitted between the two user agents.

2.1.4. Java Media Framework (JMF)

The Java^{TM} Media Framework API (JMF) [11] enables audio, video and other time-based media to be added to Java applications and applets. This package, which can capture, playback, stream and trans-code time-based media data in multiple media formats, such as AIFF, AU, AVI, GSM, MIDI, MPEG, QuickTime, RMF, and WAV, extends the multimedia capabilities on the J2SE platform, and gives multimedia developers a powerful toolkit to develop scalable, cross-platform technology. With the extensibility provided by the framework, advanced developers and technology providers now can perform custom processing of raw media data and seamlessly extend JMF to support additional content types and formats, optimize handling of supported formats, and create new presentation mechanisms.

The Java Media Framework consists of four basic hardware components: capturing device, transmitting and storage media, data processor and presenting device (Figure 8).

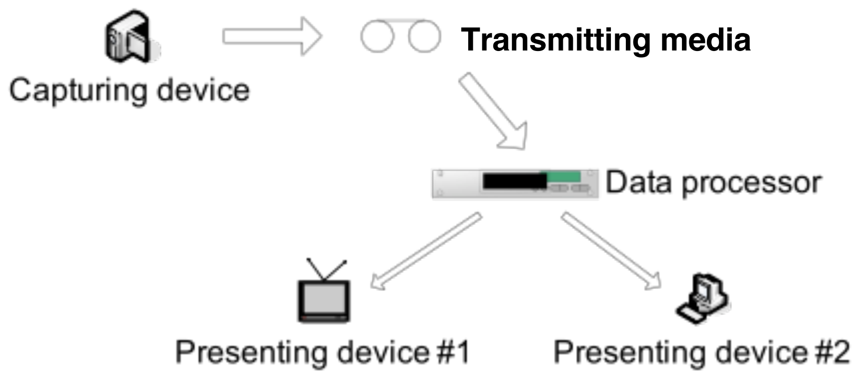


Figure 8: JMF high level architecture

- Capturing devices can capture raw multimedia input and either deliver to a data processor or save on some form of media, such as tapes or DVDs for future usage. Some capture devices deliver a single data stream, for example, a microphone only captures and delivers raw audio input, while others may have multiple data streams like a digital camera often records and sends both audio and video data. Different algorithms used to digitalize raw multimedia data and later on to retrieve it from data stream are called codec.

- Transmitting and storage media is a special form of object that is used to keep raw digitalized multimedia data and to pass it between capturing devices and data processors.

- Data Processors takes transmitted and stored media as input, perform some user-defined processing, and then deliver the result in realtime. The pre-defined processes selected by the user are carried out by separate processing components, called JMF plug-ins. There are five types of processing components: 1) *De-multiplexer* to extract separate media tracks from a multiplexed data stream, 2) *Effect* to perform special effects processing on a track of media data, 3) *Codec* to change encodings, 4) *Multiplexer* to combine multiple tracks of input data into one data stream, and 5) *Renderer* to process media data and deliver it to a presenting device such as a TV or speaker. The output data from one Data Processor can then be sent to a presentation device, saved on storage media or forwarded to another data processor.

- Presenting devices, often called Players, take media data and play it back to users. It does not provide any control over rendering and processing. Note that a Player is a simplest data processor that processes an input stream of media data and renders it at a precise time. Beside presenting media data, a more sophisticated processor will provide control over what processing is performed on the input media stream. A Processor supports all of the same presentation controls as a Player.

In the JMF implementation, four utility data models are developed to facilitate the features of the framework: time model, managers, event model, data models. A brief description of these models is following:

- Time model: It is used to represent a particular point in time. By implementing the time model, processors and players can keep track of time for a particular media stream.

- Managers: By using intermediary objects called managers, JMF makes it easy to integrate new implementations of key interfaces that can be used seamlessly with existing classes. JMF uses four managers: 1) *Manager* to handle the construction of Players, Processors, or other objects, 2) *PackageManager* to maintain a registry of packages that contain JMF classes, such as custom Players and Processors, 3) *CaptureDeviceManager* to maintain a registry of available capture devices, and 4) *PlugInManager* to maintains a registry of available JMF plug-in processing components, such as Multiplexers, Demultiplexers, Codecs, Effects, and Renderers.

- Event model: JMF uses a structured event reporting mechanism to keep JMF-based programs informed of the current state of the media system and enable JMF-based programs to respond to media-driven error conditions, such as out-of data and resource unavailable conditions.

- Data model: JMF media players usually use data model to manage the transfer of media-content. A data model encapsulates both the location of media and the protocol and software used to deliver the media. Once obtained, the model cannot be reused to deliver other media.

2.2. Applying existing technologies for construction of the prototype

We built a prototype system that consists of two play-back stations for receiving and playing back multimedia content, one multimedia data source to be sent to the user, and one PDA running the user's Personal Agent. The detail of these hardwares as well as the software components of the prototype are described in Section 2.3. There are a number of technologies are used in the prototype, such as SDP from BlueZ for service discovery, SIP protocol for establishing sessions, and JMF for delivering multimedia content. The following explains how we used these technologies in detail.

2.2.1. SDP from BlueZ

The overall goal of the BlueZ is to make an implementation of the Bluetooth™ wireless standards specifications for Linux. One of the main modules of the BlueZ implementation is the service discovery module, called sdptool. It allows clients to discover the services provided by hosts as well as their attributes. When searching for services, clients need to specify the location of the SDP server, a service name and/or particular service attributes. These inputs will be used to match service records that are registered on the specified SDP server. All the service records have an identifier, called handle, that uniquely identifies each service record within an SDP server. When a matching service record is found, its handle will be returned to the client who then uses the handle to retrieve the content of the record. The BlueZ implementation of these SDP servers is a daemon program, called sdpd.

The sdpd daemon keeps track of the Bluetooth services registered on the host. Bluetooth applications, running on the same host, register services with the local sdpd daemon. Operation like service registration, service removal and service change are performed over the control socket. In addition, the sdpd daemon responds to Service Discovery inquiries from Bluetooth devices. When a Bluetooth device needs to find a particular service, it sends Service Search and Service Attribute or Service Search Attribute request to the sdpd. The sdpd daemon will then try to find matching Service Record in its Service Registry and will send appropriate response back. The Bluetooth device then will process the response, extract all required information and will make a separate connection in order to use the service. It is also possible to query entire content of the sdpd Service Registry with by issuing browse command on the control socket.

The sdptool module can be used to check which services are made available on a specific device. sdptool provides the interface for performing SDP queries on Bluetooth devices, and administering a local sdpd.

The following commands are available.

a. Search: to search for a specific service. This command scans all accessible devices for the requested service. If one of the devices offers the service, the program prints the (full) service name returned by the device together with a brief description. Examples of known service names are: vcp for Video Conference, bpp for Printing, and dun for Dial-Up Networking
b. Browse: to browse all available services on the device specified by a Bluetooth address, such as sdptool browse 00:80:98:24:15:6D.
c. Add: to register a service to the local sdpd, for instance command "sdptool add DUN" will register a dial-up network in the local sdpd.
d. Del: to remove a service from the local sdpd, for example command "sdptool del bpp" will delete all printing services from the local sdpd.
e. Get: to retrieve a particular service from the local sdpd by its record_handle.
f. Setattr: to set or add an attribute to an SDP record.
g. Setseq: to set or add an attribute sequence to an SDP record.

An SDP server and tools from BlueZ [12] are used for service discovery. Each multimedia server runs a SDP server to register and manage local services on the machine. SDP tools are used on the iPAQ to query or browse for the desired services on SDP servers. Also on the iPAQ, a java class *SDPserviceHandler* was developed to periodically discover available services in a personal area network (PAN) and also query service parameters for each discovered services. Once found, the *SDPserviceHandler* will write the service alone with its parameters to a flat file called *serviceList.temp*. An example of this file is shown below:

```
Inquiring ...
Browsing 00:80:37:14:1E:05 ...
Service Name: SDP Server
Service Description: Bluetooth service discovery server
Service Provider: BlueZ
Service RecHandle: 0x0
Service Class ID List:
"SDP Server" (0x1000)
Protocol Descriptor List:
"L2CAP" (0x0100)
Channel/Port: 1
```

```
Version: 0x0001

Service Name: Public Browse Group Root
Service Description: Root of public group hierarchy
Service Provider: BlueZ
Service RecHandle: 0x804cb88
Service Class ID List:
"Browse Group Descriptor" (0x1001)

Service Name: LAN Access over PPP
Service Description: 137.122.95.82:42050
Service RecHandle: 0x804f498
Service Class ID List:
"LAN Access Using PPP" (0x1102)
Protocol Descriptor List:
"L2CAP" (0x0100)
"RFCOMM" (0x0003)
Channel/Port: 2
Profile Descriptor List:
"LAN Access Using PPP" (0x1102)
Version: 0x0100
```

The *SDPserviceHandler* then parse the file and find service candidates that are likely to fulfill the user's request. These candidate services then were matched to the user's requirement and the closest matching service becomes the chosen one. The information about this service is then to write to another plain file called `serviceReadyList.temp.` Now this file will look like:

```
Video
0x804f498
137.122.95.82
42050
```

The Multimedia data sender will then take this file as input and start sending media data to this address.

2.2.2. SIP

A simple version of a SIP Agent is programmed to run on the laptop and the iPAQ. It is used to initiate the communication session of exchanging SIP messages, to which service parameters, information about switching service provider are attached and sent from one multimedia server to another. After the session is created, these control messages are sent to the other side as attachments of SIP messages. This simple implementation understands two types of SIP messages: SIP-INVITE request and SIP response 200 (OK).

We developed this program using java.net, java.io, and java.util packages of the J2SE (Java 2 Standard Edition) library. It runs as a standalone multi-threading application. During initiation of the agent, it creates two thread instances: a UAC and a UAS. The UAS is continuously listening on a particular port for incoming calls while the UAC is inactive until triggered by a user. The message processor of the UAC and UAS are also programmed to run as a separate thread that is initiated at runtime so that the SIP agent can accept more calls while processing an ongoing call.

Two Java objects, called SipInvite and SipOk, were created to prepare and parse the SIP-INVITE and SIP-OK messages.

2.2.3. JMF

All audio and video play-back services are instances of JMStudio, which is a standalone Java application built by SUN Microsystems using JMF. These services run on different play-back stations and play back media content they receive from multimedia data sources. These multimedia sources are captured from input devices by using JMF and then send to the play-back services. The data capturing process is running on the laptop and communicates with the SIP Agent about the information of the surrounding play-back services.

2.3. The prototype system structure

Figure 9 shows a snapshot of the experimental system. A laptop connected with a microphone and a digital camera captures and sends out audio and video data. Two multimedia servers will be used to receive the data from the sender. A PDA-centered circle shows the WPAN constructed by the PDA through Bluetooth, within which services are reachable. An arrow between the two positions represents the movement of the user. The distance between two multimedia servers is designed to be greater than ten meters but less than twenty meters. In this case, the PDA could detect both servers when it moves in Zone 3. Meanwhile, the PDA could only reach Server 1 in the rest of Zone 1, and Server 2 in the rest of Zone 2. All of the devices are connected within one network, either LAN or WAN.

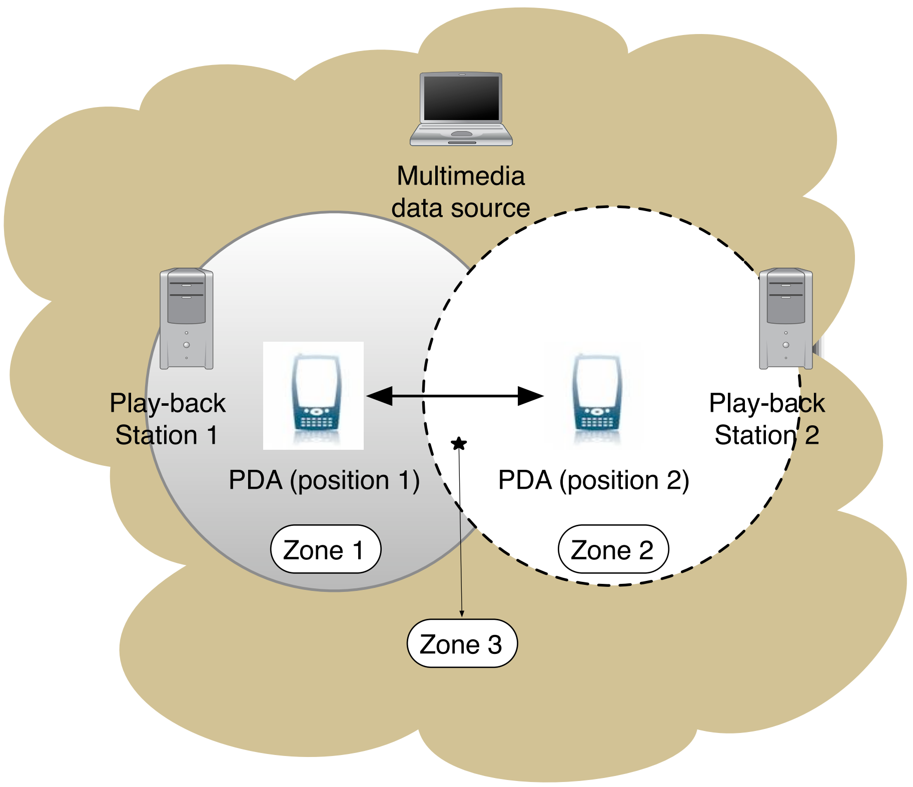


Figure 9: the system layout

2.3.1. Hardware platform:

Several devices are used in the prototype, including:

a. Network infrastructure: a local area network (LAN) is built to send multimedia content data and SIP messages to all devices. The LAN is constructed by using a combination of an Ethernet hub and a wireless access point. The sender and the two servers are connected directly to the Ethernet hub, while the PDA is connected through the access point. As well, a bluetooth network is also built to create a wireless personal area network (WPAN). This bluetooth network is used to search for available services around the PDA.

b. Play-back station: two Desktops (IBM Pentium III 900 MHz) with 128 MB RAM are adapted to serve as the servers. Both PCs are equipped with a Bluetooth device connected on the USB port. Both multimedia servers run Linux with the BlueZ [12] module for service discovery. JRE (Java Runtime Environment) version 1.3.1 from SUN is installed on the two multimedia servers to support JMF. Both server connect to the LAN through their built-in ethernet network card.

c. Multimedia data source: a laptop (IBM ThinkPad T40) with Windows 2000 is used to capture and send video and audio data. The laptop is connected with a digital camera and a microphone. It also has a ethernet network card to connect to the LAN.

d. The PDA: the Personal Agent is running on a Pocket PC (Compaq iPAQ 3870). The iPAQ has also a wireless 802.11b card that connects the iPAQ into the LAN which connects to other devices. The iPAQ is running Linux in order to use the Bluetooth stack from BlueZ. Blackdown Java [25] is installed on the iPAQ to run the SIP Agent and SDP tools, all of which are Java-based.

2.3.2. Software used in the system and software component

Different software modules are used to run the experiments, some of which are developed by ourselves while the others are adapted from existing software.

a. Discovery Agent: The discovery agent instantiated on the PocketPC runs periodically every 30 seconds. It uses the SDP tools to browse for services in the vicinity of the handheld PocketPC. The SDPTool takes 10 seconds to browse for services and up to another 15 seconds to clean up. Note that the discovery agent periodically discovers available services and register them in the Service Registry. This helps to reduce the setup delay for a session.

b. Service Registry: This database contains all the services that are currently available in the vicinity of the PocketPC held by the user.

c. Quality of Service Negotiation and Selection Agent: The PA running on the PocketPC continuously checks for changes in the available service database and select the best available services for the user. The controller then passes this information to the SIP client that sends an update to the sink for the media to the other party. This update is carried with a SIP INVITE message.

d. Play-back station: These stations play back the media content if a connection is established and at least one data stream is received. These servers are instances of JMStudio, which is a standalone Java application built by SUN Microsystems.

e. Multimedia data sender: The Media Sender Server (MSS) has a server listening for requests messages sent by the application controller. These request messages carry the IP address and the port number where the receiving application will be listening for media data. The MSS initiates separate threads to send audio and video streams. The media sender uses also JMF to capture the media data from the input devices and to send it to the play-back station.

f. SIP Agent: the SIP agent is installed and running on both the PA and the Multimedia data sender. Once the PA finds the server that can play back multimedia content, the information about how to reach the server is transmitted back to the sender by the SIP agents.

2.4. Test scenario

In the experimented scenario, service selection among the two multimedia servers is done according to service availability and user's preferences. To show the differences between the two servers, different parameters are assigned to them and registered at local SDP servers. The setting is as follows: multimedia server 1 has audio service that can playback with stereo quality and the best video playback rate as twenty-five frames per second, while server 2 could only play mono sound and has lower video playback rate at fifteen frames per second. A detailed description of the test case is as follows:

User Bob starts receiving multimedia content at Zone 1, and moves to Zone 3 and then to Zone 2, and back to Zone 3.

When Bob is in Zone 1, he orders audio and video content to be delivered by the sender. Since his PA could only reach the server 1 in Zone 1, the PA will send back the information about the server 1 to the sender and the sender will start sending the multimedia content to the server. The server then will automatically play back the content it receives. When he moves to Zone 3, his PA now will find two servers, both of which could receive and playback multimedia content. Since the information about each multimedia server from the SDP servers shows that Server 1 offers better service, Bob's PA will chose to not switch but still use the server 1 to receive the content. When Bob enters Zone 2, his PA could only find Server 2 and will send the information about the new server to the sender so that the sender could switch and deliver the content to the new destination. And Server 2 will start receiving the multimedia data and play it back. Finally when Bob moves back to Zone 3, his PA could find two servers and again, choose Server 1 to receive the content. Having received the information of Server 1 from Bob's PA, the sender will stop delivering data to Server 2 and start sending to Server 1 again.

Note that this case covers other possible cases such as when user Bob starts receiving multi-media content in Zone 3 and Server 1 becomes offline and later, it becomes online again. When a server becomes offline, it would not respond to any request. Hence, it appears to Bob's PA as if the server is out of range and unreachable. In contrast, when the server becomes online again, it would respond to requests from Bob's PA so that the PA understands that the server is within its reachable range again.

The test case is tested five times in order to get the average value of each measurement.

2.5. Performance analysis

The performance of the system is analyzed in terms of functionality and quality of service. As proposed, the prototype has met the functionality requirements in the experiment. The PA detected changes about services around the user while he was moving and sent update information to the sender.

Several measurement are collected during the experiment: 1) *set up time* is the time period from the moment the PA sends a request to the media sender to the time the multimedia data is displayed on the multimedia server. The set up time is around 9.5 seconds in each of the five runs.; 2) *transfer time* is the time period from the time when services around Bob changed to the moment when the new multimedia server starts displaying the data. It varies from 5 to 55 seconds. Note the PA periodically searches and updates information of available services every 30 seconds. Hence, the shortest time happens when a new service become available just before the PA executes the service update process. On the contrary, the longest time happens when a new service becomes available right after the PA finishes its periodic process. The new service will be found and selected among other similar servers only after the next service update.

In general, the sdptool from BlueZ is relatively poor in service browsing especially when the user is moving. It is frequent that the sdptool cannot pickup the new server in the first run when the PA is moving towards a new multimedia server and comes to the place around the edge of the area where it can reach the server. In these cases, it takes two runs to update the available service set, one for cleaning up services that are no longer available and another for adding new services. We suspect the cause of this is that the PA caches all services from the last service update and always tries to find them in next service update unless they were found unavailable and removed. Therefore, it takes the first service update (after moving into a new area) to remove all previously available services from the PA's profile and in the second service update, newly available services are found and registered in the PA's profile.

Note that during the switch from one multimedia server to another, since the media content is recorded beforehand, it is possible for the sender to remember at which precise moment the content is stopped and restart sending it again from the same point to avoid data loss. In the case of sending live data, certain changes need to be made at the sender to prevent data loss. A typical modification is to create a buffer to temporarily save the captured data while switching. However, this was not implemented.

3. Review of security technologies

In order to support mobile user and services, it is fundamental to address complex security issues based on a dynamic network infrastructure. These issues include authentication, authorization and access control over users as well as services. Lacking knowledge of each other, users and services usually depend on reputation and trust management mechanisms to provide references. In this section, we discuss existing technologies that provide solutions to threats and challenges for mobile services and compare their advantages and weaknesses. These solutions include authentication schemes, authorization and access control schemes, and trust models.

3.1. Existing authentication schemes:

There are many authentication methods and protocols that already exist in the practice of mobile applications. Before going into the discussion about the advantages and weakness of each mechanism, general approaches are briefly presented. The division of these approaches is based on what information is used in the process of authentication.

3.1.1. General models

In general, authentication between a user and a server is accomplished through the confirmation of acknowledgment of a piece of shared information. For example, a password that a user uses to access a web site is the information shared between the site and the user. In PKI [26], a site identifies a user by using his or her public key to decrypt the message encrypted by the user's private key. This key pair, if treated as a whole, is shared between the site and the user, each of whom has a part of the key pair. Depending on the size of the system, the server varies from being a stand-along workstation, whose sole responsibility is to authenticate users, to a logical module integrated into a multi-functional server machine. In terms of types of shared information, the three general authentication models include: symmetric authentication, asymmetric authentication and biometrics authentication.

3.1.1.1. Authentication based on a shared secret

This model employs a secret key (password) to authenticate two parties between each other. The secret key is agreed by both parties beforehand. Having an established shared key, authentication is achieved through exchanging either the key directly or a question that is encrypted by a symmetric encryption algorithm using the shared key. In the former case, the receiver simply verifies the key with its own record and decides whether the sender is whom he or she claims to be. In the latter case, upon the reception of the encrypted message, the receiver will try to decrypt it using the shared key. If the message is decrypted, the receiver should be able to understand the question. At this point, the authentication of the sender is assured since only the particular party has the key. The receiver then sends back a proper response. When the sender gets the response and verifies it, the authentication of the receiver is successfully finished. Note that instead of performing one-way authentication in the former case (the receiver authenticates the sender, not the opposite way), the latter case actually achieved mutual authentication. Since the same encryption algorithm and the key are used to authenticate both sides, it is therefore called as symmetric authentication. For instance, the password authentication schemes used by most web servers are the implementation of this model which is based on a shared secret.

Since the shared secret is used as a key in the authentication protocol to encrypt and decrypt the message, it is very importance to address the issues of key management, key distribution and in particular the strength of the key. In a centralized system where a central authentication server manages all users, the key distribution is accomplished when a user registers at the server. A shared key is created and sent to the user upon the registration. Since repeatedly using a same key to encrypt messages and sending them in a network increases the possibility of breaking the key, it is strongly recommended that passwords be changed from time to time to keep the strength of the key.

3.1.1.2. Authentication based on public keys

Instead of sharing one key between two parties, the Public Key Infrastructure (PKI) [26] proposed a public/private key pair to authenticate one party to the other. The private key of a key pair is kept by the party that generated them while the public key is distributed to others. Since a different key pair is used for each party, the authentication model is also referred to as asymmetric authentication in contrast to the symmetric model. With the purpose of accomplishing the authenticate protocol, the public/private key pair is designed in such way that once a message is encrypted by a private key, it is almost impossible to decrypt it unless the corresponding public key is used, and vice versa. This property of the key pair leads to the logical conclusion that once a message is successfully decrypted by a public key, the party that encrypted the message must be the one that possesses the matching private key.

In addition to authenticate two parties, the PKI technology also provides the capability to verify signatures of a participant. To create a signature, a standard hash function is applied to the message to be signed and the result of the hash value is then encrypted with the private key of the signing party. Attached with the signature, the original message may be sent to various destinations. Upon reception, the recipient verifies the signature by decrypting it with the corresponding public key, computing the hash value of the original message using the same hash function and comparing the results of the two calculations. In the case of matching, the sender must admit that he signed the message since only he is in possession of the private key that was used for the signature. For the same reason, the recipient may be confident that no modification has been done to the message after the sender signed it.

Although the PKI technology is more suitable for mobile service application in terms of capabilities of fulfilling requirements, several issues still need to be discussed: 1) how to distribute reliably public keys of various users and organizations to other parties; 2) where and how to store a private key and make it only accessible by the owner who is on the move, giving that it is too long to be remembered and entered manually; 3) the de/encryption algorithms are less efficient than those for symmetric encryption while a mobile device usually has limited computing power, memory and power supply.

3.1.1.3. Authentication based on biometric information

In addition to the two general models, authentication based on biometric information characteristic of the user, is a most recent technology in practice. Compared to random numbers that are created and used as shared or public/private keys, the biometric information of a user, including fingerprints, eyeball scans and DNA recognitions, is more difficult to manipulate but no key needs to be memorized. Similar to symmetric authentication, the biometric information is stored in a central server, which serves as an authentication authority. To authenticate a user, the information stored in the server is compared with the reading directly from him or her. Unlike symmetric authentication, this approach is asymmetric because the biometric information provided by one of the two communicating parties can only be used to verify against his or her own information that was previously stored in the central server. In the case of matching, only this party gets authenticated while the other party still remains un-authenticated. Apparently, this authentication approach cannot be applied to organizations.

3.1.2. Existing authentication protocols

There are various authentication protocols that are already in practice. This section briefly discusses the advantage and weakness of each protocol when it is applied in the context of mobile e-commerce applications.

3.1.2.1. Radius (Remote Authentication Dial-In User Service)

Using CHAP (challenge handshake authentication protocol) [27], Radius is a password-based mechanism that is widely used by Internet service providers to give point-to-point access with mobility [28]. Figure 10 presents the sequence of message exchanges in Radius, where CV is used as a challenge value and K is a key shared by the network access server (NAS) and the authentication authority, called Radius Server.

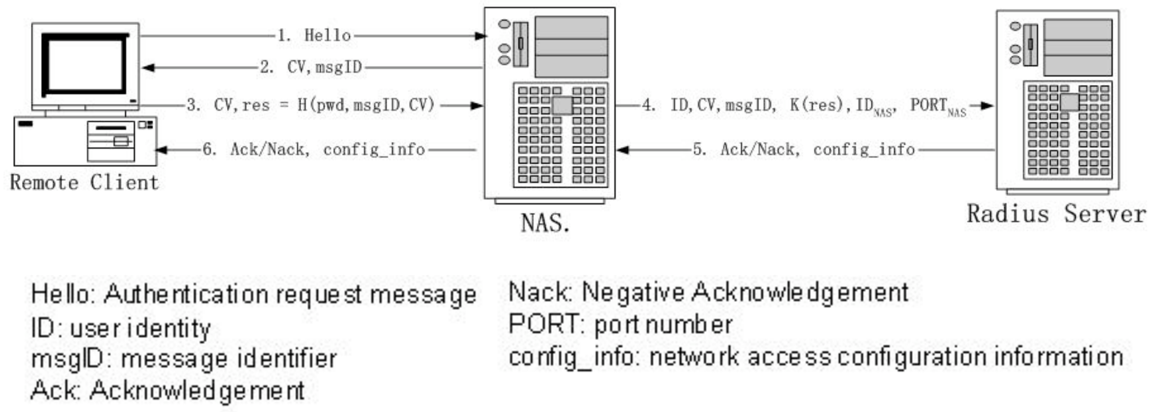

1. Hello
2. CV, msgID
3. CV, res = H(pwd, msgID, CV)
4. ID, CV, msgID, K(res), ID_{NAS}, $PORT_{NAS}$
5. Ack/Nack, config_info
6. Ack/Nack, config_info
Remote Client
NAS.
Radius Server
Hello: Authentication request message
ID: user identity
msgID: message identifier
Ack: Acknowledgement
Nack: Negative Acknowledgement
PORT: port number
config_info: network access configuration information

Figure 10: Message exchange in Radius (from [28])

In brief, the user first requires a challenge value (CV) from the NAS and creates a message (*res*) based on the CV and the password that the client and Radius Server shares. The message (*res*) along with CV is then forwarded by the NAS to the Radius server. The latter checks the validity of *res* and sends back an acknowledgement according to the result of the validation.

Radius is applicable in systems with a centralized network infrastructure and fails to meet the requirement of supporting mobile users because of the following:

(1) The NAS and the Radius server are supposed to know and trust each other. And the link between them is supposed to be secure. However both of these assumptions may not be true in a mobile e-commerce environment.

(2) Anonymity cannot be provided with this scheme, giving that the protocol sends a personal identifier in plain text.
(3) The random challenge value along with a CHAP identifier (called 'msgID' in the figure) CV is also sent in plaintext in step 2. It is relatively easy for attackers who listen to the traffic over the network to capture CV, msgID and res. The attackers could then perform a chosen plain-text attack by guessing the password to calculate the hash value and comparing the result with the value *res* included in the message.

3.1.2.2. Kerberos

Similar to the case of Radius, Kerberos uses a centralized key distribution center (KDC) to assist in key management [29]. A ticket or authenticator is issued by the KDC to a client for service access control at servers as shown in Figure 11, where Kc is the key shared between the client and the KDC, Ks is between the server and the KDC, and Kc.s is between the client and the server.

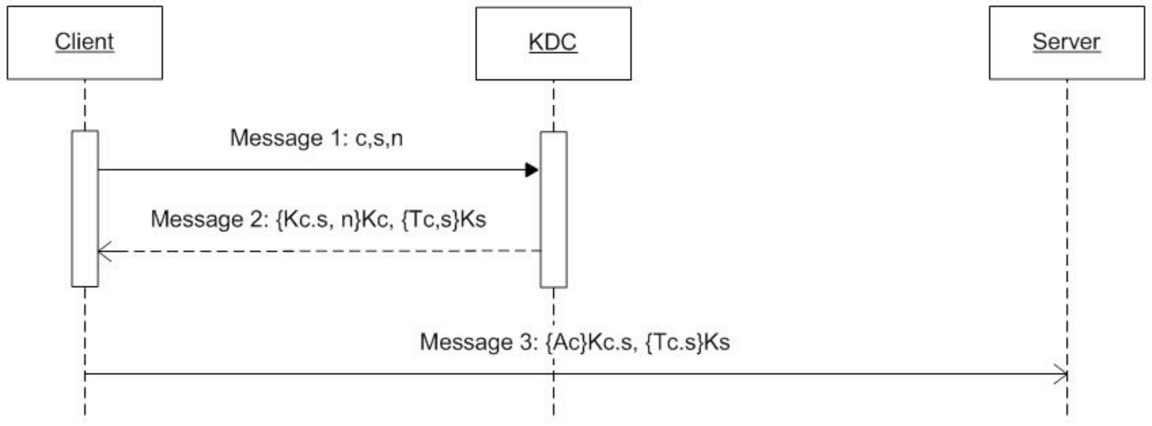

Client
KDC
Server
Message 1: c,s,n
Message 2: {Kc.s, n}Kc, {Tc,s}Ks
Message 3: {Ac}Kc.s, {Tc.s}Ks

Figure 11: Ticket generation and usage in Kerberos (from [30])

A ticket (Tc.s) will be used to authenticate a client at servers that are providing services and a session key (Kc.s) is employed to keep the strength of the key as well as avoid over-using the master keys (Kc and Ks). In addition to authentication, the ticket can also be used to control access to servers.

Anonymity and privacy are not supported in Kerberos since a client has to send out his or her identity as well as the required services in clear text to a KDC. This information is not protected from third parties that may listen on the communication path. Another challenge of the Kerberos is that the first message is sent from a client to the KDC while the client may have no idea about where the KDC is and whether it is trustworthy. Moreover how a secret key is distributed between a client and a KDC located in foreign domain also needs to be addressed. We conclude that Kerberos is impractical especially when the user's visit to a foreign domain is unpredictable.

3.1.2.3. SSL

SSL stands for Secure Sockets Layer and is renamed by IETF as TLS [31] (Transport Layer Security). Originally developed by Netscape, SSL is especially used by Web browsers to provide authentication and privacy for sensitive Web applications. SSL contains various options for authentication including several versions of public/private key authentication. The protocol also provides for a fresh shared session key that can be used for encrypting the messages exchanged over the session.

3.1.2.4. XML utility

Security Assertion Markup Language (SAML) [32], is the first industry standard for enabling secure e-commerce transactions through the eXtensible Markup Language (XML). Independent of any particular platform, SAML enables companies to securely exchange authentication and authorization information with customers, vendors and suppliers, while the XML Key Management Specification (XKMS) [33] efficiently manages digital signatures and encryption. As a supplement, XMLPay [34] provides further facilities for payment transactions to build trust-supported B2B and B2C e-commerce.

3.1.2.5. Smart and SIM card

Many types of smart cards and the SIM-card used with mobile phones contain an authentication certificate including the public key of the user (owner of the card) and some attributes (e.g. user name) and the associated private key. For security reasons, the private key will never be communicated through the card reader interface. Instead, any message to be encrypted or decrypted with the private key is transferred to the card and the result of the operation is returned to the card reader. Thus, any device that can interface with the card could perform an authentication handshake with a remote party through which the owner of the card would be identified as the user.

3.1.2.6. Others: (SSH, SHTTP)

SSH [35] is a protocol that provides secure access over insecure channels to remote server computers, including file transfer and a command line interpreter. Two version of the protocol are available. SSH1 provides both server and user authentication, while SSH2 only provides user authentication, but it is more secure. The Diffie-Hellman Algorithm [36] is used to negotiate a shared secret key.

SHTTP (Secure HyperText Transfer Protocol) was designed to secure only HTML (Hypertext Markup Language) web pages. Server and client preferences and security constraint are negotiated for each web page or set of pages. The client-side public key certificates are optional, "as it supports symmetric key-only operation mode" [37]. In this case, password-style manually shared secrets is prearranged and used to encrypt and decrypt data

There are also extensions of the IP protocol for mobility [38] and security [39], however, the security framework at the IP level is not very useful for authentication in mobile commerce applications.

3.2. Existing access control schemes:

In this section, three general access control models including MAC (Mandatory Access Control), DAC (Discretionary Access Control) and RBAC (Role Base Access Control) [40], are discussed in terms of advantage and weakness. Other than these, ADAM (Autonomic Distributed Authorization Middleware) [41] is also studied because of the benefits it brings when the model is applied in a mobile service environment.

3.2.1. Mandatory Access Control

Mandatory Access Control literarily means the enforcement of the access control policy is solely operated by the system once setup. The policies are set by the administrator and entirely determine whether the access to a particular resource is granted and users cannot grant less restrictive access to resources they own than the administrator specified. In other words, it is the system rather than users that have full control of the access to all of resources. Within this scheme, security levels are organized as a tree-like structure. A service being assigned to lower security level means to use it a user needs less security clearance. In the other hand, a service being assigned to higher level of security requires a user to have higher security clearance to access it. In addition, a user will have access to all services whose required security level is lower than the security clearance that is assigned to him or her. An example system, which employed Mandatory Access Control, is shown in Figure 12.

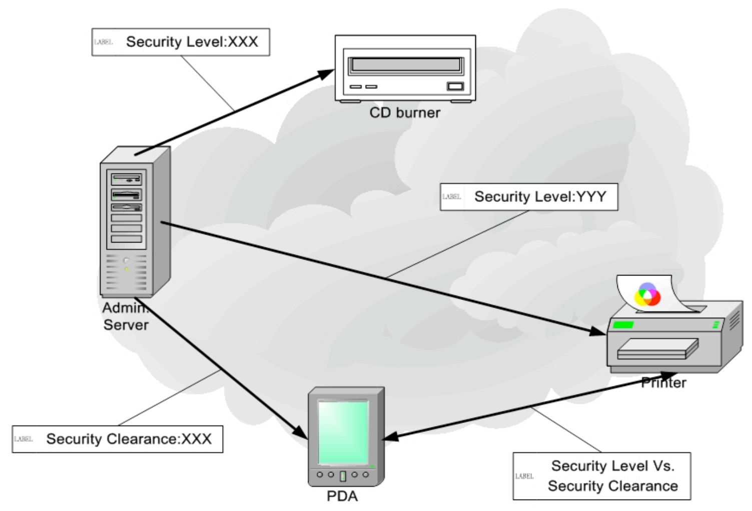


Figure 12: MAC model

As shown, an administrative server is previously set up to control the access over all of resources in a system, including a CD burner and a printer. The server assigns each resource a security level according to its sensitivity, confidentiality and protection requirement, while a security clearance is given to each user, including the owner of the PDA, say Bob. When Bob wants to use the printer, the system will grant him the access by comparing the security level of the printer to the security clearance of Bob. In the case of Bob possessing higher or equivalent clearance than the security level the resource requires, the access is granted.

Given the fact that only administrators, and not resource owners, manage the security level of a resource, MAC effectively prevents resource owners from granting access to the resource for their personal usage. It also improves the strength of a system's security by relieving owners from the complicated task of setting up access control policies for resources they own since the complexity of the task frequently results in security holes. At the cost of users convenience, MAC provides stronger security protection than DAC and is usually appropriate for extremely secure systems including military applications or critical data applications, such as financing or banking systems.

3.2.2. Discretionary Access Control

In contrast to MAC, a Discretionary Access Control model gives users full control of the access rights for the resources they own. The controls are discretionary in the sense that the resource owner is capable of determining the type of access given to different users and even transferring the ownership of a resource to others. Once set up by the owner, the access restrictions are based on the identity of a user and/or membership in certain groups, which are typically verified based on the credential the user presented at the time of authentication. Typical credential includes user name, password, hardware platform and installed software. These restrictions are called access control policies and usually in the form of a list containing identities of users and groups that users may belong to. Figure 13 shows an example system, which uses the Discretionary Access Control model.

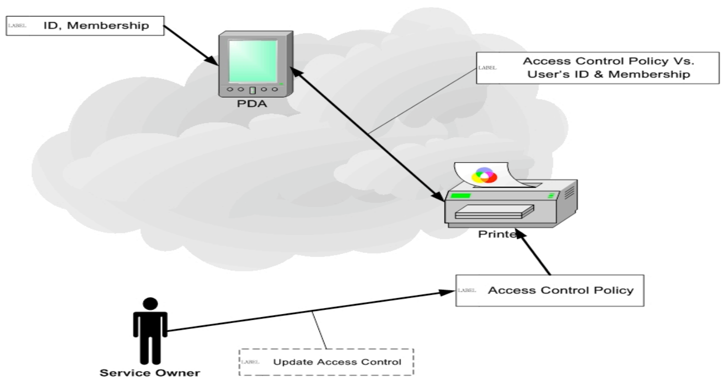


Figure 13: DAC model

As shown in Figure 13, the access control policies of a publicly accessible printer are set up and maintained by the owner of the printer. A user, say Bob, is authenticated upon entering the domain which includes the printer by presenting his name, password and other necessary information through his PDA. During the authentication, the information of his identity and memberships to certain groups are also verified. When Bob requires the access to the printer, the decision is made by matching the policies and his identity and/or the memberships he possesses. In the case that the list allowing access contains Bob's identity or one of the groups he belong to, the access is granted.

In typical DAC models, the resource owners are in charge of setting up permissions for accessing resources based on their own judgments. Therefore they may accidentally or maliciously give access to unauthorized users. Despite the flexibility of resource management it provides, DAC has the drawback that administrators can not centrally manage all resources to ensure a certain security level for the system as a whole, as well as maintain consistency and compatibility throughout various environments.

3.2.3. Role Based Access Control

The Role Based Access Control model is a newer and alternative approach to DAC and MAC. Based on three fundamental sets of entities including users, roles and permissions, the family of RBAC models has been extending among the growing demands of various applications. For instance, role hierarchies, constraints and a management model are also introduced into RBAC architectures. In a typical RBAC system, access right is defined by role instead of directly by users, that is, access decisions are based on the roles the user is authorized to assume. The separation of the association between permissions and users simplifies common management tasks such as adding a user, or changing a user's privilege. For instance, when an administrator want to grant a new user the access to a system, a number of permissions is needed to assigned to the user. Conventionally, the administrator needs to assign each one of the permissions to the user manually. This process is lengthy and highly repetitive. Therefore it normally introduces human mistakes. In a RBAC system, the administrator can pre-define a role and associate it with all the permissions that are required for using the particular system. After that whenever a new user want to use the system, the administrator can simply associate the role with the new user and the latter will automatically has all the permissions to access the system. Moreover, the RBAC model can also be applied to distributed systems, where administrative tasks are carried out locally without compromising the flexibility as well as the security level of the system. Figure 14 shows a family of RBAC models starting from the basic model $RBAC_0$ to the advanced $RBAC_3$ not including the management model.

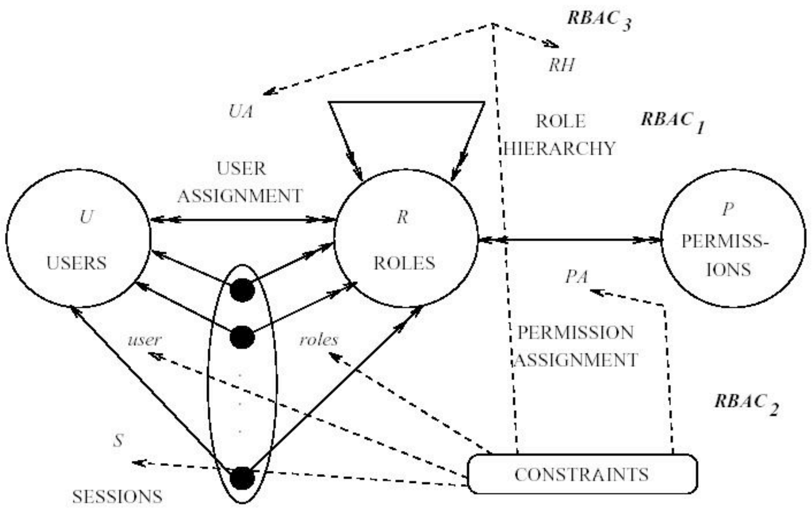


Figure 14: RBAC models [40]

As shown, the basic model RBAC$_0$ contains three entities: Users, Roles and Permissions. A *user* is a human being or an agent of a person acting on behalf of him or her. A *role* is a semantic term associated with a group of responsibilities. For instance, a developer role is in charge of low-level design and programming according to design documents while an architecture role is responsible for requirement analysis and high-level design. Permission is an approval of a particular operation such as accessing a patient's health records. The RBAC$_0$ also includes two processes: user assignment and permission assignment. The former is to assign roles to a user while the latter assigns permissions to a role. Based on each user's responsibility, system administrators decide which roles can be assigned to the user. The user should only be assigned to the necessary roles, and no more than that, in order to carry out his or her duties. Likewise, permissions are assigned to roles based on the principle of least privilege, that is, a role cannot be assigned more privilege than is necessary to perform its duties. For example, within a hospital the role of doctor can access a patient's information including name, address and health records while the role of researcher can access health records of all patients but not the information that may identify a particular person. Ultimately, the two assignments enable a user to exercise permissions. The last component included in the RBAC$_0$ is called *session*. A *session* is a mapping of a single user to many active roles he or she was assigned to. Each session is under the control of the user and combines any active roles that are suitable for the tasks to be accomplished in a transaction. The concept of session is concordant with the principle of least privilege since a user can dynamically activate and deactivate any role that is needed or no longer needed for a task. Note that the principle of least privilege is hard to achieve and in less precisely controlled system this difficulty often leads to unintended access.

On top of $RBAC_0$, the model $RBAC_1$ introduces role hierarchies, that is, each role has unique attributes and may contain other roles in terms of permissions that are associated with it. A role hierarchy system can significantly avoid the inefficient and repeated operations in case of different roles perform common tasks. For instance, within a hospital both the role of specialist and the role of doctor can access health records. In an enterprise, role hierarchy can be established to reflect the structure of the organization in terms of authority and responsibility.

In $RBAC_2$, the concept of constraint is introduced. Being used to lay out higher-level policies within the scope of an organization, constraints can be applied to user assignments, permission assignments as well as sessions. Based on a well-known principle called separation of duties, a common example of constraints on user assignments is that within an organization a user possessing the role of purchasing manager can not be granted the role of accounts payable manager. Constraints on permission assignments are often used to ensure the implementation of the principle of separation of duties. An example is the constraint that, the permission of signing checks cannot be assigned to both purchasing and accounts payable managers. The third type of constraints are those applied on sessions, like the number of sessions a user can activate at the same time. In a distributed system, the constraint mechanism could prevent overall security from being compromised while administrative tasks are decentralized and delegated to different sites. Note that a role hierarchy could be considered as a constraint since permissions assigned to a role must also be assigned to its child roles. On the other hand, a user that is assigned to a role must also be assigned to its parent roles.

$RBAC_3$ combines both $RBAC_1$ and $RBAC_2$ to provide role hierarchies and constraints. Along with the advantages from both models, subtle interactions between constraints and hierarchies also arise from the combination. In the case that the role of purchasing manager and the role of account payable manager are prevented from being assigned to a single user, the role of store manager, as a parent role for both, violates the constraint through the hierarchy. To properly adapt the $RBAC_3$, such interactions need to be carefully solved by introducing an algorithm to solve the contradiction.

3.2.4. Autonomic Distributed Authorization Middleware

A. Seleznyove and S. Hailes proposed a new architecture for access control management, called Autonomic Distributed Authorization Middleware (ADAM) [41], within which each action on a service are authorized based on the trust relationship between the client and the service. This trust relationship can be built from distributed knowledge throughout the network, that is, the knowledge is collected through direct observations, recommendations from trusted parties or reputations of the user or service. The authorization decision in the ADAM is made by negotiation performed between two groups of agents, user agent and authorization agent. The user agent contains information about its legal user such as secret keys and certificates that may be required for later use so that it could automate certain tasks without interaction with the user. The authorization agent, however, is aimed at protecting network services by authorizing access request, enforcing access restrictions and monitoring usage of services. Numbered by the order of executions, Figure 15 shows the sequence of exchanges of an authorization scenario in the ADAM.

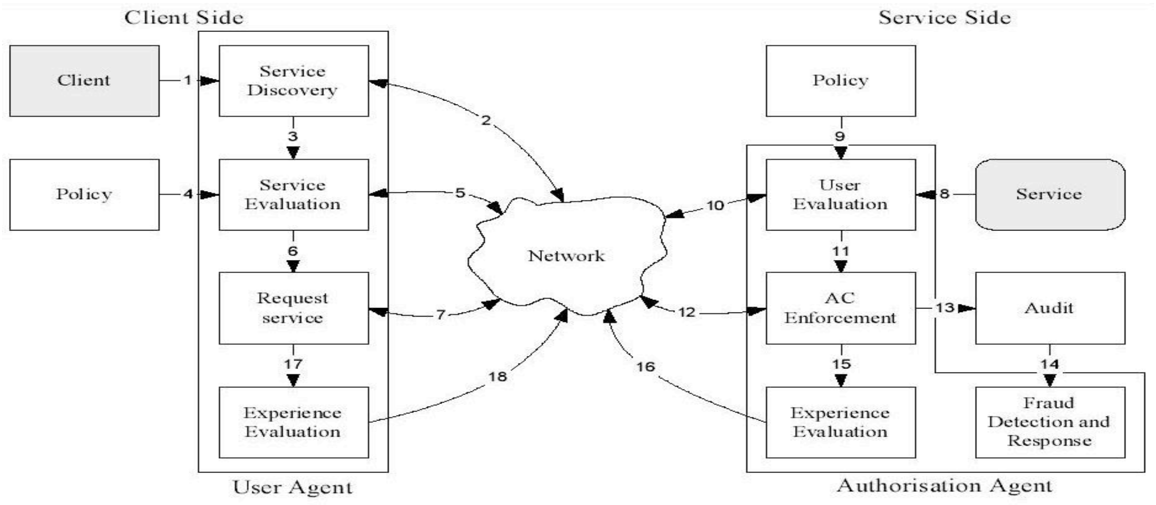


Figure 15: Authorization and access control in ADAM [41]

Initiated by a client's request (1), the user agent first conducts a service discovery process (2) that searches throughout the network looking for available services that could fulfill the request. Having a list of available services, a service evaluation process (3) assesses all the services against the local policy of the user (4) and opinions from other users (5). Once a particular service is chosen (6), a service request is sent out (7). Upon the reception of the request message, a user evaluation process is executed against the local policy (9) of the service provider, based on all information provided by both the service and the user. While the information from the service is verified locally (8), the verification of the user's information is delegated to third parties which have experience with the user (10). Based on the result of the user evaluation, an authorization decision is made (11). In the case of approval, permission to access is sent back to the user (12). During the time of the access, the authorization agent performs continuous auditing (13) as well as fraud detection (14). When the access is over, the authorization agent will evaluate the experience with the user (15) and update its opinion accordingly (16). At the client side, the user will also evaluate his or her experience with the service (17) and modify his or her judgment.

The ADAM provides flexibility to users to access services even though zero knowledge may be shared between both users and service providers. This advantage is achieved with the involvement of reputation data throughout the network. While the authorization decision is based on opinions from previous users or services, it is unreasonable to assume that the opinions cannot be maliciously given to mislead the deciding procedure. A trust management model, which takes the inaccurate feedbacks into count, is therefore needed.

3.3. Existing trust models

With the growth of e-commerce, users and servers are often required to deal with unknown entities. The concept of trust is therefore introduced to help users and servers making decisions when authorizing strangers. Among many definitions, Elofson defines trust as: "*trust is the outcome of observations leading to the belief that the actions of another maybe relied upon, without explicit guarantee, to achieve a goal in a risky situation*" [42]. Figure 16 shows the data sources of trust: direct experience from previous interactions and recommendations from other entities. Based on the combination of the two sources, authorization decisions are made for each entity. After an interaction, the experience of the interaction is then evaluated and converted into the form of trust to modify the information about the trustworthiness of the entity.

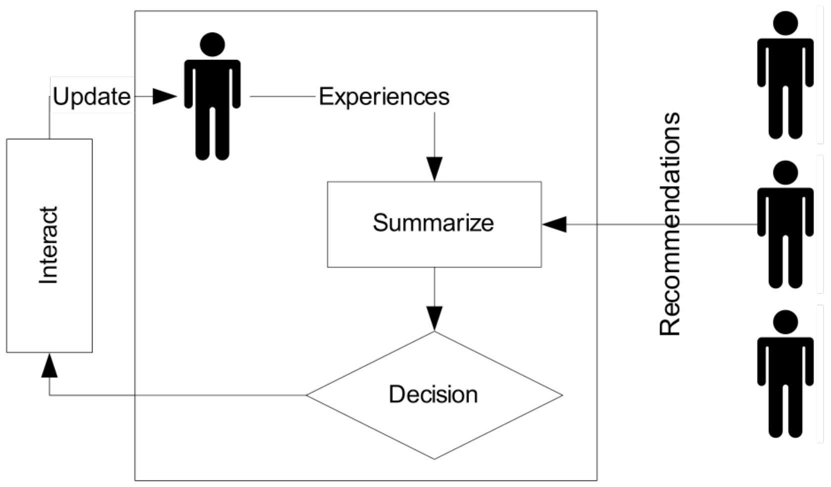


Figure 16: Data source of trust

One of the fundamental issues in trust models is that they all need a trust management mechanism to represent the trust in the way that it can be exchanged without introducing ambiguity. While some models proposed qualitative representation such as *very trustworthy, trustworthy* and *not trustworthy*, they are not sufficient to differentiate specific characteristics of an entity. Quantitative representation, on the other hand, can measure the trust more precisely by a continuous real value. However, it cannot answer the questions such as "What does a user with a trust value of 3.5 mean? And how much trustworthy he or she is compared to a user with 3.0?" In order to address such issues, J. Shi proposed a probability distribution of the outcomes that an entity may bring through his or her practice [43]. An example of such distribution is shown in Figure 17.

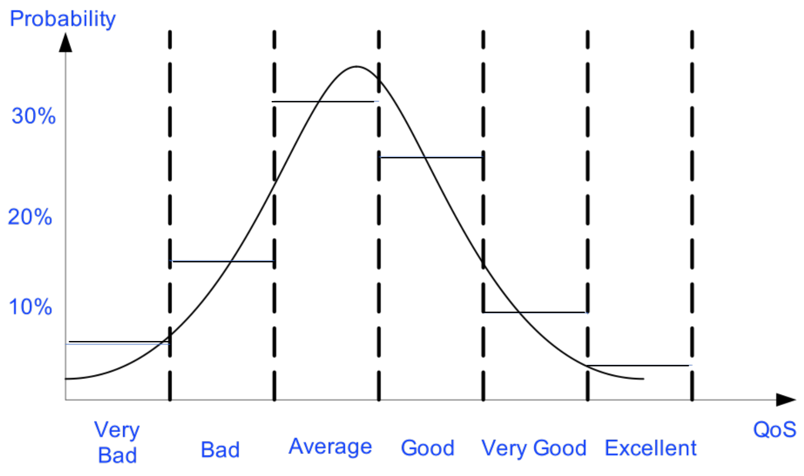


Figure 17: Outcome distribution of an entity in the stochastic model

Consequently, the trust is represented as an estimated distribution of outcomes based on previous direct experience and recommendations. Instead of making decision based on the best or worst case as in other trust models, Shi's model tends to select the most probable case that may happen. For instance, when a user, say Bob, wants to print a document, his personal agent (PA) finds there are two available printers, printer 1 and printer 2. Having no direct experience with neither of them, the PA then turned to other parties for recommendations. The calculation of the feedbacks from recommenders showed that printer 1 has 50 percent probability of providing good printing quality while printer 2 may provide best quality printing in 10 percent of the cases. Rather than choose printer 2 because of the best printing it may provide, printer 1 is chosen if Bob is satisfied with the document printed in good quality.

Another issue is to combine recommendations from different entities. Given the fact that recommendations collected from the network are not necessarily accurate, the extended work [44] mainly focuses on a recommendation management model, which summarizes all recommendations with mechanisms to detect malicious recommendations. In addition, recommendations collected from one particular network may not be valid in other networks. In the effort of globalizing an online reputation system, Li's work [45] introduced a personalized aggregation method for local and global reputation data based on a discrete statistical model.

4. Improvement of Dupre's proposed secure infrastructure

In order to design a secure service framework for mobile users, one of the fundamental challenges to be answered is to authenticate participants, giving that they may have no previous knowledge about each other. Dupre proposed an authentication protocol [7], which is based on a secret key that is shared between a user and his or her home agent and has limited usage of public key infrastructure. The protocol was originally designed to provide a scalable secure infrastructure as well as a key distribution mechanism for multimedia communications. While the protocol satisfied most requirements when applied in the context of mobile service, further study shows that it has certain shortcomings and can be improved. Starting with recalling the scenarios studied in the Section 1.2, this section analyzes detailed requirements for authentication in particular cases and how Dupre's authentication protocol can be applied to fulfill these requirements. We identify certain shortcomings of Dupre's authentication protocol, and present improvements. Finally, security analysis is conducted for the revised system.

4.1. Authentication requirements

The first scenario described in Section 1.2.1 states that, when Alice is in a foreign domain, she wants to have VoIP conversation with her friend Bob. In order to connect to Bob, she uses her PDA to automatically find, select and connect to one ISP (Internet Service Provider). The PDA then sends an invitation to Bob requesting VoIP conversation. Once Bob answered the request, the connection between Alice and Bob is successfully built and the conversation can take place. Although this scenario implies various requirements such as quality of service, access control and service selection, the following discussion focuses only on authentications.

Because the authentication process between Bob and his home agent is similar to the process between Alice and her home agent except with foreign agent interpretations, it is reasonable to focus the discussion only on Alice's side, that is, authentications among Alice's PA, the foreign agent and her home agent. Note that foreign agents and home agents of users are generally connected in the network and rarely moved once setup. Therefore, it is sensible to assume that a secure connection between the two can be set up using conventional mechanism such as verifying the digital signature of each other so that no extra authentications are needed. We note that this scenario contains six transactions: Alice's PA authenticates the FA, the FA authenticates Alice's PA, Alice's PA authenticates Alice's HA, Alice's HA authenticates Alice's PA, and the HA and the FA authenticates each other.

In addition, the third scenario (see Section 1.2.3) requires that users should have the option of hiding their identity. This means that anonymity is an optional requirement for the authentication process.

4.2. Dupre's authentication protocol

Dupre proposed a password-based authentication protocol [7] that establishes a trust relationship between any pair of participants involved. The protocol was originally developed for mobile communication over the Internet but is applicable to any mobile applications.

4.2.1. Protocol overview

The main design objectives for the authentication protocol are the following:

a. The user's authentication is based on a secret password that is shared between the user and the Home Agent.

b. The protocol leads to the creation of a new public/private key pair that can be used for the authentication of the user. The private key will reside on the device that the user is currently using and an authentication certificate signed by the Home Agent is provided for the new public key.

c. A trust relationship is established between the Home Agent and the Foreign Agent based on reciprocal authentication.

Note that the use of a secret password for authentication has the advantage that it is easily implemented with a relatively short password (of a length of approximately 6 to 10 characters) that the user can remember. The authentication based on public key technology requires a much longer private key that must be stored in some device or card carried by the user. This makes it difficult for the mobile user to use any device that may be locally available. On the other hand, public key technology is essential for authentication to third parties and for the generation and verification of signatures. This is the reason for the second design objective. The main characteristic of this new authentication protocol is therefore to combine the use of a password with public key authentication. The new public/private key pair generated by the authentication protocol may be used for authentication to third parties, for instance for Alice's telephone conversation with Bob, and allows the user to generate verifiable signatures.

Let us compare this protocol with Radius (see Section 3.1.2.1). Radius also uses password-based authentication, but it does not provide the creation of a public key certificate for authentication to third parties. Also, it assumes that the Network Access Server (NAC), which corresponds to the Foreign Agent in our architecture, is associated with a single Radius Server, while our protocol foresees inter-working with a variety of different Home Agents throughout the world.

Objective (c) is also important. In fact, no initial trust relationship is assumed between the user and the Foreign Agent. However, when the authentication protocol completes successfully, the Home Agent will have authenticated the Foreign Agent, and the resulting trust is indirectly available to the user. This trust relationship may also be used by the user to obtain services from other service providers within the foreign domain.

It is important to note that the protocol is structured in such a way that the user side of the protocol, also called Personal Agent (PA) is realized by software that runs on the device that the user happens to use within the foreign domain. This device may be his/her own PDA, but it may also be any device that happens to be available. The user has to trust the integrity of the software that represents the PA, but it does not have to trust the Foreign Agent.

The protocol can be broken down into three phases:

i. **Foreign agent advertisement:** A locally broadcast message (Message 1) provides information about the FA, e.g. the FA's IP address and its public key. This kind of message is necessary for any mobile user who is connected to a foreign domain to find the foreign agent's server. A user can start the subsequent authentication exchange after receiving this message.

ii. **Authentication request:** Alice sends the authentication request to the FA, which then forwards the request to Alice's HA through some secured connection (Message 2 and 3).

iii. **Authentication reply:** Alice's HA verifies the request based on the password shared with Alice and sends back either ACK or NACK according to the verification result (Message 4 and 5). In the case of positive acknowledgment, Alice shares Ks_2 with the

FA and Ks_3 with her HA. These secrete keys can then be used in the future as session keys. Alice also receives $CERT_A$ from the HA. Since her KR_A is kept in her PA, Alice can then use $CERT_A$ and KR_A to establish security associations with new parties or sign documents using. In the case of negative acknowledgment, Alice will learn that her request is rejected and the reason why it is rejected.

4.2.2. Detailed Protocol description

Figure 18 shows the message sequence of the protocol: a broadcast message and a 2-way handshake. The following notations are used:

"," denotes concatenation	$CERT_X$: Certificate of X
ID: the unique identifier of the user	pwd: password of the user
KU_X: public-key of X	ACK: acknowledgement
KR_X: private-key of X	NACK: negative acknowledgement
Ks: a session key (symmetric key)	$HV_1 = H(ID_A, CSR_A, N_1, pwd)$; H is a hash function
K(M): M is encrypted using key K	$HV_2 = H(HV_1, N_2, NACK, pwd)$
Nx: a nonce with integer subscript x	Ks_1: session key randomly chosen by Alice's PA
SecCx: secure connection	$Ks_2 = H(N_1, N_2, pwd)$
CSR: Certificate Signing Request	$Ks_3 = H(N_1, N_3, pwd)$

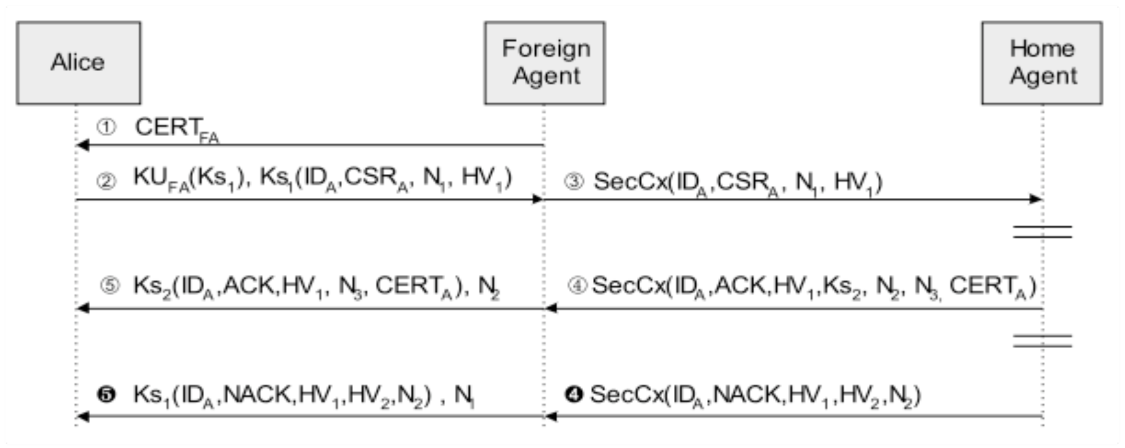

Alice
Foreign Agent
Home Agent
① CERT_FA
② KU_FA(Ks_1), Ks_1(ID_A,CSR_A, N_1, HV_1)
③ SecCx(ID_A,CSR_A, N_1, HV_1)
⑤ Ks_2(ID_A,ACK,HV_1, N_3, CERT_A), N_2
④ SecCx(ID_A,ACK,HV_1,Ks_2, N_2, N_3, CERT_A)
❺ Ks_1(ID_A,NACK,HV_1,HV_2,N_2) , N_1
❹ SecCx(ID_A,NACK,HV_1,HV_2,N_2)

Figure 18: Dupre's authentication protocol [7]

Phase i: **Foreign agent advertisement:** A broadcast message in the local domain informs Alice about the location and the digital certificate of the FA. This could be realized through existing protocols, such as the Dynamic Host Configuration Protocol (DHCP) or Jini.

Phase ii: **Authentication request:** Alice's device executes the following steps in order to prepare Message 2:

(1) Generation of a random number N_1 and a certificate-signing request (CSR) according to the PKCS#10 standard [46].

(2) Generation of a digest, called HV_1, of the above information, the user's identifier ID_A and the password pwd.

(3) Selection of a random session key Ks1 that is used to encrypt all the above information HV_1 as well as ID_A, CSR_A, and N_1. This will allow the FA to forward HV_1 to the HA. Ks_1 is then encrypted with FA's public key, which was obtained from the FA's certificate included in Message 1.

Upon reception of Message 2, the FA performs the following actions in order to prepare Message 3:

(1) Decryption of the KU_{FA} (Ks_1) using KR_{FA}, and then decrypts Ks1 (ID_A, CSR_A, N_1, HV_1).

(2) Establishment of a secure connection with Alice's HA and sends ID_A, CSR_A, N_1, and HV_1.

Phase iii: Authentication reply: Upon reception of Message 3, the HA computes its own digest HV_1' and compares it with HV_1. If they are equal, authentication of the user succeeds and an "ACK" (acknowledgment) is returned; otherwise a "NACK" (negative acknowledgement) is returned.

In the case of ACK: authentication reply - positive:

(1) HA signs Alice's CSR and generates two random numbers N_2, N_3.

(2) Using the current secure connection established with the FA, the HA sends back an ACK message (Message 4) including the answer of the authentication process (ACK), the hash value HV_1 sent by Alice that uniquely identifies the request, and security material for Alice (ID_A, Ks_2, N_2, N_3, $CERT_A$).

(3) Upon reception of Message 4, the FA calculates Ks_2, encrypts ID_A, ACK, HV_1, N_3 and $CERT_A$ with Ks_2 and transmits it together with the nonce N_2 in clear as Message 5.

(4) Alice's device receives Message 5. It computes Ks_2=H (N_1, N_2, Ks_1) and decrypts Ks_2 (ID_A, ACK, HV_1, N_3 and $CERT_A$). It also computes Ks_3=H (N_1, N_3, pwd). It now shares a security association with FA based on the shared key Ks_2, and with HA based on the shared key Ks_3. Alice can now establish a security association with a new party or sign a document using $CERT_A$.

In the case of NACK: authentication reply - negative:

(1) HA generates a random number N_2.

(2) HA prepares a "NACK" message (Message 4) that includes a rejection reason (e.g. "revoked user" or "password expired") and the hash value HV_1 that identifies Alice's request. It then computes HV_2=H (HV_1, N_2, NACK, pwd).

(3) Using the current secure connection established with the FA, the HA sends back Message 4, including N_2, the answer of the authentication process (NACK), HV_1 sent by Alice, and HV_2, which serves as a proof of answer and ID_A.

(4) Upon reception of the message, the FA computes Ks_2=H (N_1, N_2, pwd) and encrypts ID_A, NACK, HV_1, HV_2 and N_2 with Ks_1 and transmits it together with N_1 in clear text as Message 5.

(5) Alice's device receives this message, calculates Ks_2 and decrypts Ks_1 (ID_A, NACK, HV_1, HV_2, N_2). It then computes HV'_2=H (HV_1, N_2, NACK, pwd) with HV_2 to check that authentication answer actually comes from the HA. Alice now knows that her request has been rejected and she has received the reason.

4.2.3. Protocol analysis

Two cases are covered in the protocol shown in Figure 18: a positive acknowledgement for a successfully authenticated user and a negative acknowledgement for a user whose authentication request is denied (corresponding to the third and fourth row in the sequence diagram, respectively).

There are six authentication transactions in the protocol. They are the following: the HA and the FA authenticates each other with certificates, Alice and the FA authenticates each other through the home agent that is trusted by both, and Alice and the HA authenticates each other through a shared secret. At the end of the authentication process, the user gets a certificate that allows him or her to sign documents and also to establish security association with another user. The home agent proposes the session keys for use between the user and the agents. These keys can be used in subsequent communication exchanges between the user and the home or foreign agents. Sensitive information such as session keys and authentication information are always encrypted during the exchanges.

If all of these transactions return positive result, the authentication phase is complete and Alice now can make business transactions with the FA. On the other hand, any negative result from one of the six authentication transactions will prevent Alice making business transactions with the FA. For example, in the case that the HA fails to authenticate the FA, Alice can not trust the FA to be a legitimate agent therefore she will not send her service request to the FA.

Note that Alice relies on her HA to authenticate the FA. Since Alice does not have a root certificate, her PA could not verify the FA's certificate sent in Message 1. Instead, the PA will send an encrypted request to the FA which should then be forwarded to the HA. If the FA could not be authenticated by the HA, the secure connection between these two parties could not be established. Without the secure connection, the request would not be sent. Therefore the PA would time-out after waiting for a reply message from the FA. Such a time-out indicates that the FA may have failed to get authentication.

We also note that this protocol has minimal usage of public key technology at Alice's side which satisfies the limitation of computing capability and battery power of mobile devices. Through the authentication protocol, public key encryption is used only once in Message 2. After successful authentication, there is a session key shared between Alice and the FA (Ks_2), as well as between Alice and her HA (Ks_3). Further negotiation will be based on these session keys using symmetric key operations.

4.3. Weaknesses of Dupre's protocol

We discovered two weaknesses of Dupre's protocol: a flaw in the protocol that facilitates denial-of-service attacks and the lack of support of anonymity.

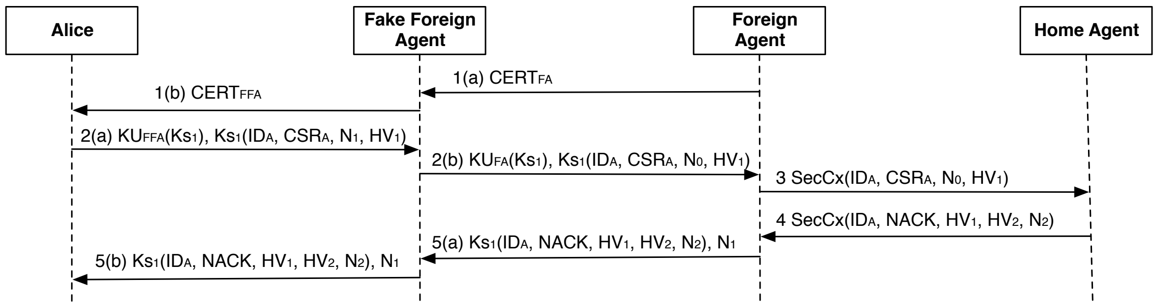

Alice
Fake Foreign Agent
Foreign Agent
Home Agent
1(a) CERTFA
1(b) CERTFFA
2(a) KUFFA(Ks1), Ks1(IDA, CSRA, N1, HV1)
2(b) KUFA(Ks1), Ks1(IDA, CSRA, N0, HV1)
3 SecCx(IDA, CSRA, N0, HV1)
4 SecCx(IDA, NACK, HV1, HV2, N2)
5(a) Ks1(IDA, NACK, HV1, HV2, N2), N1
5(b) Ks1(IDA, NACK, HV1, HV2, N2), N1

Figure 19: Scenario of abuse

Figure 19 shows a scenario of denial-of-service attack by employing a Fake Foreign Agent acting as a gateway between Alice and the real Foreign Agent. In this case, since Alice could not verify the certificate of the Foreign Agent locally, she will still send her authentication request and rely on her Home Agent to authenticate the Fake Foreign Agent when building the secure connection between them. This presents two threads:

Threat one: We may assume that the fake foreign agent, who is sitting between Alice and the real Foreign Agent, will receive the nonce N_1 in Message 2(a). By sending out a different nonce N_0 in Message 2(b), the fake foreign agent can let the Home Agent refuse Alice's authentication request because the value of HV1' is now calculated based on N_0 instead of N_1, and will not be the same as HV1, which was originally generated by Alice. Following the protocol, the Home Agent will then generate a new hash value based on HV1, NACK, pwd and N_2 and send it back to Alice as the reason of the negative acknowledgement. Since none of them have been modified by the fake foreign agent, there is no way for Alice to find out the existence of the fake agent. Having failed the authentication process, she cannot use the service at all even though she sent all valid information to the Foreign Agent and her Home Agent. This is the worst case of the denial-of-service attack.

Threat two: According to this scheme, Alice will send out her own identification in Message 2(a) (Fig 19) without verifying the certification of the Foreign Agent. It gives the chance to the interceptor to get the clear value of HV_1, ID_A, CSR_A, and N_1. Noting that the value of HV_1 is calculated through some hash function based on ID_A, CSR_A, N_1, and pwd ($HV_1 = H(ID_A, CSR_A, N_1, pwd)$). Therefore this increases the possibility of breaking the security of the password. Besides, the interceptor can also get the clear value of HV_2 and N_2 in a later step; this also helps to break the security of the password because the HV_2 is a hash function of ID_A, NACK, N_2, and pwd, in which only pwd is unknown.

Generally speaking, if Alice could not verify the certificate of the Foreign Agent at the first step, the user's privacy (especially anonymity or location privacy) is at risk. And the denial-of-service attack will become relatively easy to implement, along with increasing the possibility of breaking the password.

4.4. Improvements of the protocol

Figure 20 shows the message sequence of the improved protocol. The revised message segments are written in bold and marked with double underlines. The following notations are revised, compared with those given in Section 4.2.2:

- $Ks_2 = H(N_1, N_2, Ks_1)$, which is different from $H(N_1, N_2, pwd)$ in the original protocol
- $HV_1' = H(ID_A, CSR_A, N_1, pwd)$ is calculated by the Home Agent. It is not used in the messages directly. Instead, it is used to calculate HV_2, which is used in reply messages
- $HV_2 = H(HV_1', N_2, NACK, pwd)$ in negative case, and (HV_1', N_2, ACK, pwd) in positive case. The original protocol uses HV_1 instead of HV_1'.

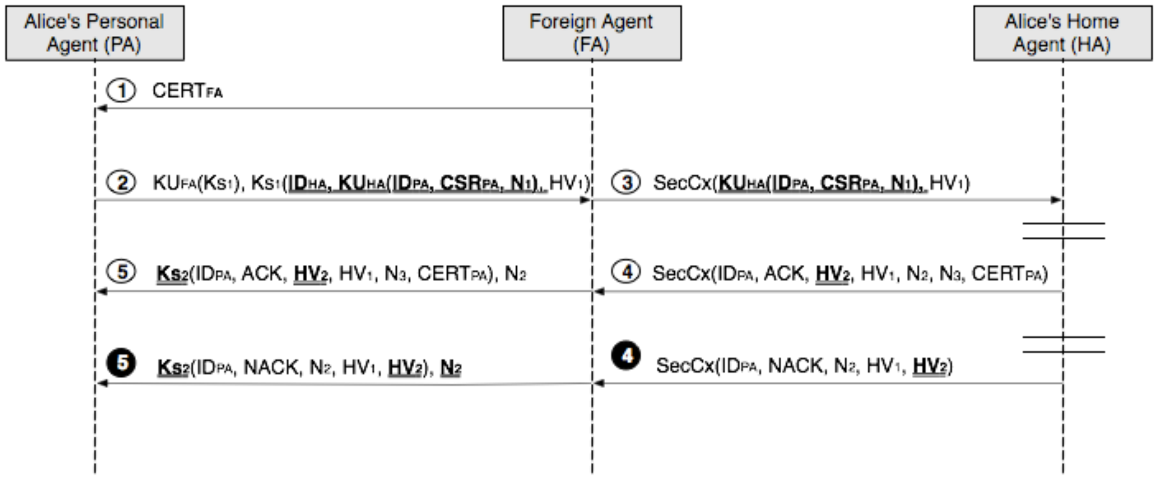

Alice's Personal Agent (PA)
Foreign Agent (FA)
Alice's Home Agent (HA)
① CERTFA
② KUFA(Ks1), Ks1(IDHA, KUHA(IDPA, CSRPA, N1), HV1)
③ SecCx(KUHA(IDPA, CSRPA, N1), HV1)
⑤ Ks2(IDPA, ACK, HV2, HV1, N3, CERTPA), N2
④ SecCx(IDPA, ACK, HV2, HV1, N2, N3, CERTPA)
❺ Ks2(IDPA, NACK, N2, HV1, HV2), N2
❹ SecCx(IDPA, NACK, N2, HV1, HV2)

Figure 20: Improved authentication protocol

As shown in Figure 20, the improvements includes:

i. In Message 2, the information of Alice's Home Agent is now included and encrypted with the session key Ks_1, which is encrypted by the Foreign Agent's public key and sent separately. This way the FA knows where to forward the request. Because the identification of Alice herself is encrypted by her Home Agent's public key, the FA has no knowledge about whom it is dealing with. This enables the option of anonymity and location privacy.

ii. Also in Message 2, instead of sending clear values of the certificate signing request (CSR_{PA}), Alice's ID as well as the nonce N_1, these values are now encrypted by HA's public key to hide from the FA all clear values that are passed between the User and his or her HA. This way, no Foreign Agent could observe who is actually making the request. Therefore the anonymity or location privacy requirements are satisfied. The modification also prevents the breaking of the password by hiding clear text as much as possible. Furthermore, the scenario of denial-of-service described previously is now impossible because any change in the original message will be notified by Alice after getting back the return message from the Home Agent. This encrypted message segment is forwarded by the FA to the HA in message 3.

iii. In Message 4, the HV_2 is now $H(HV_1', N_2, NACK, pwd)$ or $H(HV_1', N_2, ACK, pwd)$, in which HV_1' is the value calculated by the Home Agent. This is forwarded by the FA to Alice. When Alice receives Message 5, she can double check the HV_1 that she sent by comparing it with this value. It ensures that no third party has altered the message. While in the case of positive reply, Alice can also make sure that the HA got her request and responded.

iv. In Message 5, the session key Ks_2 is now $H(N1, N2, Ks1)$, which is different from $H(N1, N2, pwd)$ in the original protocol. This provides Alice additional assurance that the FA actually received the Ks1 in Message 2. As well, it reduces the frequency of using the shared secrete pwd in plain format, which in turn strengthen the password.

v. Anonymity option: The user's anonymity can be guaranteed by hiding the user information from the FA and using tickets provided by the FA to gain access to services within the domain of the FA. The anonymity option implies the following modifications to the protocol. In the case of a positive acknowledgement, Message 4 now becomes $SecCx\ (ACK, HV_1, N_2, N_3, Ks_3\ (CERT_{PA}))$ and Message 5 becomes $Ks_2\ (ACK, HV_1, N_3, T, Ks_3\ (CERT_{PA})), N_2$. We note that ID_{PA} is removed from these two messages. Instead of sending $CERT_{PA}$ in clear, it is now encrypted with a shared session key Ks_3 which is only known by the PA and the HA. A ticket T is created by the FA and sent to the PA to be used for local service access. The service server would validate the ticket and provide service upon validation regardless who presents the ticket. In the case of a negative reply, Message 4 becomes SecCx (NACK, N_2, HV_1, HV_2), and Message 5 becomes Ks_2 (NACK, HV_2, HV_1), N_2. As in the previous case, ID_{PA} is removed from these two messages.

4.5. Security analysis of improved protocol

4.5.1. Verification of the authentication requirements

In case of positive authentication, all six authentications among the three participants take place:

1. Alice authenticates the FA: Alice authenticates the FA by checking its certificate ($CERT_{FA}$) after she receives Message 1. If she cannot verify the certificate, Alice could always depend on her HA to authenticate the FA when they try to establish a secure connection. Later when Alice successfully decrypts Message 5 in cases of both positive and negative reply, she knows that the FA received HV_2 from the HA because that value could only be computed knowing the password. This means that the HA authenticated the FA previously when establishing the secure connection SecCx.

2. Alice authenticates the HA: After decrypting Message 5, Alice knows that the HA computed HV_2, because the HA is the only agent that knows the password.

3. The FA authenticates the HA: The FA checks the certificate of the HA before sending Message 3 when establishing the secure connection.

4. The FA authenticates Alice: After receiving Message 4, the FA knows the authentication answer of the HA and trusts the authentication done by the HA. In addition Alice can decrypt Message 5 if and only if she recovers Ks_2 from N_2. If she does so and uses Ks_2 to communicate later with the FA, the latter knows that she shares some information with the HA.

5. The HA authenticates Alice: After receiving Message 3, the HA compares HV_1 with HV_1' to check Alice's password. When the authentication request is received, the HA will calculate the hash function: H (ID_A, CSR_A, N_1, pwd), in which 'pwd' is supposed to be only known by the HA and Alice. If the result of the hash is identical to the one (HV_1) included in the request, the HA knows that the request was generated by Alice.

6. The HA authenticates the FA: The HA checks the certificate of the FA when FA tries to establish a secure connection.

In the case of a negative response, Alice is sure that the answer was prepared by the HA because of the following reasoning. After receiving Message 5, Alice checks that the negative answer was made by the HA by computing HV_2. The latter value could have been computed only by HA and is related to Alice's initial request because of the presence of HV_1. This verifies that no third party is misbehaving in the middle between Alice and the HA. The presence of ACK/NACK in the HV_2 computation is useful to check that the middle party did not change the reason of acceptance or rejection.

4.5.2. Consideration of typical security attacks

We discuss in the following a few typical security attacks and explain how the protocol copes with them.

1. Spoofing attack of a malicious user: A malicious user, says Eve, may try to usurp Alice's identity. Authentication information is included in HV_1 sent by Alice to the FA in Message 2. Since Eve does not know Alice's password, the HA while calculating HV_1' finds a different value and does not authenticate Eve as Alice. In message 4, the HA sends the authentication result to the FA so that the FA knows that Alice (actually Eve) is not authenticated.

2. Spoofing attack of servers (the FA, the HA) is denied by the systematic use of digital certificates. Alice relies on the HA to authenticate the FA.

3. Replay attacks of an authentication request are impossible owing to the nonce. If an attacker tries to replay messages or a rogue FA tries to replay messages, they will be detected by the HA that keeps all successful login nonce for a given time period (e.g. a few days). Even if the attack is not detected by the HA, a malicious user replaying the request message could not decrypt Message 5 because he/she would require the knowledge of the key Ks_2 which can only be calculated with the knowledge of Ks_1 which is generated by Alice.

4. Denial-of-service (DoS) attacks would consists of sending rogue authentication request that would consume both bandwidth and processing time at the FA and the HA. Such an

attack can be realized more easily by simultaneous mass replay attacks. It would make the HA compute all the key material for each request. Denial of service is a general and open issue for any service on the Internet.

Note that the protocol satisfies all the requirements when executed on a user-owned mobile device. However, when executed on any device that may be locally available to the mobile user, there are two common problems (which are not related to the particular protocol): (a) The user has to trust the integrity of the software, and (b) the private key generated by the protocol may be left on the device and used by other people, if the user does not properly terminate the application.

4.6. Comments on the detailed design of the protocol

The description of the authentication protocol represents, in some sense, an "abstract protocol", that is, only the logical meaning of the message parameters is described, while the coding of these parameters is left undefined. It is important to note, however, that a complete protocol specification (describing all requirements for an implementation) should also include the definition of the parameter encoding and the description of the cryptographic functions that are used. It is clear that the choice of these cryptographic functions has a strong impact on the level of security that can be obtained by the given "abstract protocol". In the following, we give some comments on the possible choices.

Private-key algorithms should be chosen such that the length of the key can be adapted to the computational power of the mobile terminal. Triple-DES, Blowfish and AES (Advanced Encryption Standard) are such algorithms. Elliptic Curve Cryptography (ECC) should preferably be used for public-key encryption, rather than RSA, to make use of its shorter key length at equal security level.

Secure connections could be set up in several ways since both FA and HA own a digital certificate. TLS (Transport Layer Security), IPsec (IP security), IKE (Internet Key Exchange) or any secure link establishment protocol could be used between the two agents.

5. A Trust-based access control framework for mobile services

Besides authentication, there are other requirements for mobile service systems, such as authorization, access control and service selection. In this section, the ADAM framework proposed by A. Seleznyove and S. Hailes [41] (described in section 3.2.4) is adapted to meet these requirements. The major extensions made to the original framework are the involvement of a role-based access control mechanism and a trust management model for the reputation data involved in the business transactions.

As discussed in the previous section 3.2.3, role-base access control models provide flexibility for assigning privileges to users while keeping the administrative work relatively simple to avoid unintentional mistakes. The fact that these models only deal with a limited number of roles instead of with potentially a large number of individual mobile users brings significant benefits when the connectivity between users and service servers grows along with the extension of network infrastructures. Hence, we propose to adapt a role-based access control mechanism to the framework. Although the role-based access control model is suited for mobile service systems, a number of issues still need to be addressed, such as defining roles, associating roles and privileges, and where the policy decision points should be located for managing various services.

In the context of mobile services, it is common that when a user enters a network, he or she is unknown to the service provider of the network, and vice versa. Although once the authentication process is finished, the user and the service provider identify each other; however, the information about the party does not include the character of its behavior. For instance, through the identification information of a user, there is no way for a service provider to anticipate how properly he or she will use a service once he or she gained the access to it. To overcome this limitation, the ADAM framework introduced the concepts of reputation and trust. Instead of trying to predict the behavior of each other, users and service providers use reputation data to build a trust relationship between them. Every time a user accesses a service, the service provider evaluates his or her behavior, and the user also evaluates the service. The results of these evaluations are then used to update the reputation of both parties. Based on these reputations, a trust relationship is built between them. Depending on the trust level, the service provider decides whether or not to grant access to the user. In the mean time, the user also makes a decision about whether or not to use the service. Lacking of particular mechanisms to manage reputation data across different domains, Shi's model [43] is therefore included into the ADAM framework.

This section is structured as following: special requirements for role-based access control in mobile e-commerce systems are discussed first; then an overview of the adapted framework is presented, covering components such as client, service server, service registry, authentication authority, reputation server, and access control policy decision and/or enforcement point. Then we explain how to apply the framework with the help of a set of sequence diagrams showing the messages exchanged between each party involved in the scenarios. At the end of this section, we discuss briefly about the choice of representation for security and trust information, and several suggestions for implementation are made.

5.1. Requirements for role-based access control in mobile service environments

In addition to general requirements that are applicable for every role-based access control system, there are several specific requirements to be fulfilled in the context of mobile services:

(1) Since the user maybe completely new to the domain, the decision of assigning a role to the user should be made solely based on previous behavior of the user while he or she dealt with other providers in the past. Further more, it should be possible to transfer the roles that are assigned to a user to another domain while the user is on the move;

(2) Now that roles of a user can be carried from one domain to another, it is also desirable to associate roles and privileges in a manner that is globally understandable;

(3) Since the decision making now depends on the user's previous behavior, the concept of trust relationship between users and services is introduced. Even though previous experiences are to be documented, integrated, presented and evaluated in some form of trust, the management of trust includes various issues, such as how to document previous experience; how to integrate trust data from different services into a single result [45]; And where to store the trust data without violating the privacy of personal information.

5.1.1. Role assignment and transfer

The first challenge of role-based access control in a mobile service environment is to assign a proper role to a user when entering a domain. If the user does not belong to the domain, the latter may have no knowledge about the user. With the help of the reputation data of the user, the role assignment can be processed based on a trust relationship at a particular level between the user and the domain. We note that it is relatively easy to build trust relationships among services located in a single domain. In a domain where all the local services trust each other, a transitive trust relationship can be built between the user and a local service once he or she is trusted by at least one service in the domain. Depending on the level of the trust relationship, the domain should be able to assign to the user a set of roles that have privileges to perform certain tasks. Furthermore, the format of the role set should be designed in such a way that it can be stored and carried with the user and verified by other services within the same domain. A common mechanism is to create a digitally signed certificate.

In addition to role assignment, role transfer is a nice feature that allows users to bring certain access privileges when they enter a different domain. When users are dynamically changing their physical location, they frequently move from one domain to another. If users are already granted a set of privileges to access certain services in their home domain, it is preferred for them to have the same set of privileges to access compatible services in another domain. In other words, users should be able to bring privileges they already possessed in one domain when they enter another domain. This requires several conditions: (1) the privileges should be presented in a standard form to be understood across domains. (2) Among various domains, a trust relationship should be built so that one domain can assign a user a particular set of privileges according to what the user received in another domain. After a user gets assigned a particular set of privileges in one domain, these privileges should be able to be carried over to another domain. The new domain will accept these privileges because of the trust relationship that is built between the two domains. This trust relationship ensures the new domain that the previous domain made proper decision when it assigned privileges to the user. (3) Although the privileges are transferred across domains, the present domain should have the capability to override the decision that another domain has made and assign different privileges according to the user's most up-to-date reputation data.

5.1.2. *Standardized set of privileges and role-privilege assignments*

Prior to the process of role assignment in real scenarios, how to associate a role with a set of privileges is another administrative challenge for role-based access control in the context of mobile services. The task can be further divided into two phases: to identify different privileges required to access various services and to associate a set of privileges with different roles.

When users travel from one domain to another, they often would like to use different services, such like listening to music while waiting for a bus, or checking email at the airport before boarding a flight. To date, most of these needs are fulfilled by continuously evolving technologies. Among all the convenience these services bring, the diversity of the services also raises issues of resources management. Conventionally, a service resource is protected by some mechanism, which will first assign every user a set of privileges and has a policy in place stating and enforcing that only the users who have certain privileges could be allowed to access the service. For instance, a person who makes local phone calls in a hotel should possess the privilege of regular guest or staff while a manager has the privilege to make long distance calls. In a mobile service environment, the variety of the services, as well as the corresponding privileges, becomes much richer and is also dynamically changing. A standard definition that defines all kinds of services, privileges as well as attributes of the services and parameters of the privileges is needed so that a user's request of service can be understood and then served across domains.

Once the set of pairs of services and privileges has been defined, an immediate issue is to associate privileges with difference roles. Although the least-privilege principle states that a user should not be granted more privileges than what he or she needs to access a particular service, a role, to which a user is assigned, should not be too strictly associated with particular privileges that every time a user requests a different service, he or she has to switch between different roles or even re-apply for it. At the mean time, the association between roles and privileges should also not be too lenient to potentially generate significant amount of overlapping or conflict of privileges among different roles, which may lead to abuse of resources, inconsistent results or denial of services. An example of inconsistent results when requesting a service could be a person, who has a role of regular staff that can only make local phone calls and also a role of manager that can make long distance calls, may get different response when he or she try to make a long distance call. If the role of regular staff is chosen, the request will be denied. However, if the role of manager happens to be chosen, the request will be admitted.

5.1.3. Policy decisions and enforcement based on trust

Unlike in the conventional context where the responsibility of each user is clearly stated and proper privileges are already assigned to the user so that the decision making process is simply a find-and-match procedure, mobile users and mobile services may not have knowledge about each other. Whether or not to grant access to a service for a user cannot be verified against the conventional policy. To overcome this issue, the concept of trust is introduced and the decision making process is now able to base on a trust relationship built between the user and the service provider. The level of this trust relationship may vary depending on the past experiences with the user and the service (called direct trust), or another service provider that he or she can trust, called transitive trust.

Note that the policy decision-making point and the policy enforcement point are two logical component of an access control framework. Differentiated by their functionality, they can physically run on single hardware platform.

5.1.4. Trust data management

In order to design or adapt a trust-based access control framework, it is essential to have a trust management mechanism in place. In the context of a mobile service system, two requirements are specially needed: 1) the integration of trust data from different sources; and 2) the physical location to store the reputation data of a user or a service.

In a trust-based access control framework, service providers evaluate a user each time he or she uses a service offered by the service provider. Since the user can move from one domain to another, the number of service providers he or she dealt with in the past could be potentially large. These providers may give different results when evaluating the behavior of the user because of various reasons, such as different measurement against user's behavior, the dishonest recommendations from user-favored service providers and different expectations for different type of services. For instance, for an audio/video conversation over the Internet, it is acceptable for a service provider to send images over the network with a few seconds delay or pause. However, this delay is not fully satisfactory when audio data is played back. Giving the diversity of the results from all sources, a trust data integration model is needed to calculate the overall value of the reputation data of the user [45].

In order to apply a trust-based access control framework in a mobile service environment, we need to decide, among various choices, where to save the reputation data of a user or a service. Three possible places could be used to store the reputation data for a particular party, a user or service: the home agent, a third party server in the foreign domain, the PA or the service server. Each choice has its advantages and disadvantages. If one lets the home agent store all the reputation data, it is easier to manage since the home agent tends to be fairly stable and can be announced to every party connected in the network. The drawback is that it makes the data less trustworthy because a home agent may favor the user or the service that belongs to it when calculating the overall reputation from data collected from various sources. Another possible place to store the reputation data is the user's PA or the server that hosts the service. The shortcoming of this solution is the same as above. However, the reputation data of a user is spread out over several domains that he or she visited. To collect the data from several domains in real time significantly slows down the system performance. In addition, there are also legal issues concerning personal privacy that affect the place where the user's reputation data can be stored [47]. For instance, the privacy law in Europe states that all personal information collected within the geographic region of Europe cannot be moved out to other regions. The regulation eliminates the possibility of using a user's home agent to store reputation data when the user is from Canada because the evaluation data generated by service providers in Europe cannot be shipped back to Canada. No matter how the user used the service, properly or not, it cannot affect the decision that service providers in Canada would make upon receiving a service request.

5.2. Proposed system structure

Before explaining the details of the adapted trust-based access control framework, it would be helpful to identify all components in the framework. As shown in Figure 21, the framework includes a Home Agent, a Foreign Agent, a Personal Agent, one or more Reputation Servers (located in different places), a Policy decision point and enforcement point (PDP/PEP), a Service server with a Policy Store and a Service Directory. These components are logically divided by functionalities that they provide. In terms of hardware configuration, they could be configured to run on one computer. For instance, although the F.A. and the Reputation Server are separated as shown, they could run on a single machine.

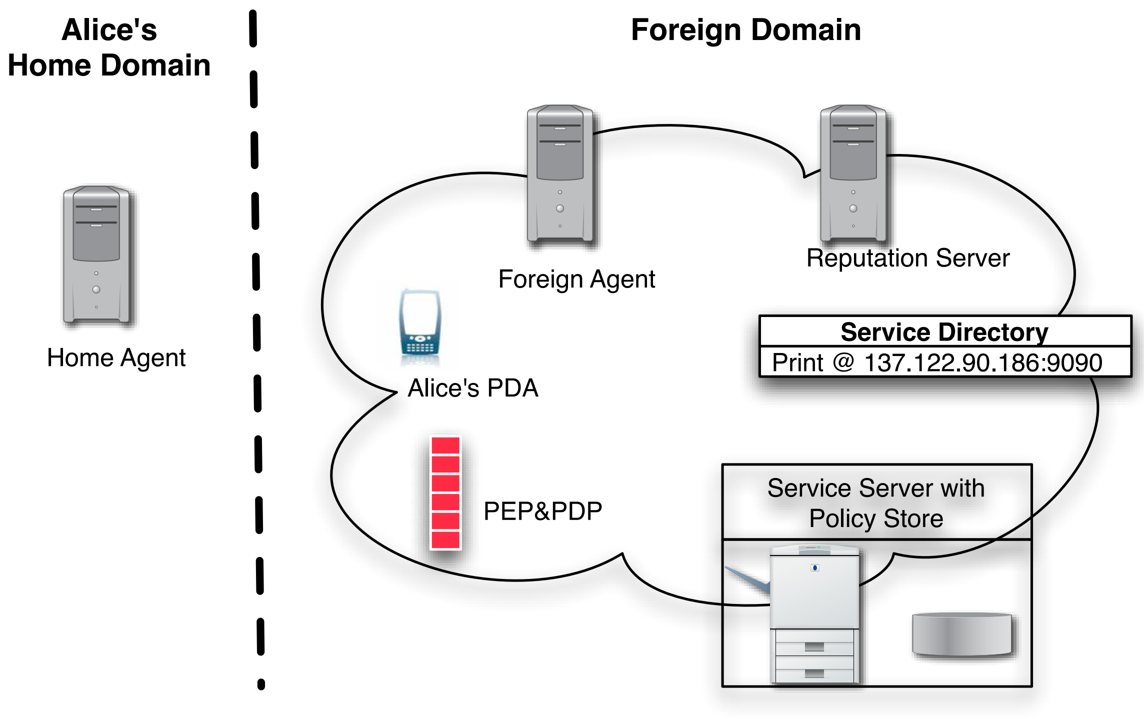


Figure 21: the architecture of the framework

Each component in Figure 21 has different duties in terms of access control as explained in the following sections.

5.2.1. Home Agent

Initially when connecting to the Internet, users or services register themselves at their local network. This local network is called their home domain. The home agent of a users or a service runs in the user or the service's home domain. It stores the profiles and preferences of the user or the service. It also shares a secrete password with the user or service, which will be used in authentication processes. When a user sends an authentication request to his or her home agent, the shared password will be included to ensure the HA that this request is really issued by the user. Once approved, the HA will generate a certificate for the user, which will later be verified in other domains. This way, the user can always be authenticated in foreign domains as long as the latter can verify the certificate generated by the HA. Note that the reputation data of a user or service may also be stored and managed by the home agent under certain circumstances. In that case, the home agent will also be involved in the process of the trust-based access control.

5.2.2. Foreign Agent

The term 'foreign' means the domain in which a user or service currently resides is not the home domain where the user or service was originally created or registered. Therefore when an agent acts as foreign agent for a user, it could also be a home agent at the same time for another user if the latter is registered locally. In contrast, a home agent of a user could also act as a foreign agent for others who were registered in other networks but recently joined the current network.

Due to the inter-changeable nature of the home and foreign agent, both of them offer similar functionalities. These functionalities can be divided into two sets, one for users whose home network is the local network and the other set is for users not originally registered here. The functions for a home user are described in the previous section while for a foreign user; the agent only provides limited access when the user enters the network. Starting from there, the user could get authenticated and then be authorized to perform further actions.

5.2.3. Personal Agent

A personal agent runs on a handheld device and is kept by the user during the movement across domains. It automates the authentication process for the user when it enters a new domain so that no human intervention is needed. It also runs the service selection process to help the user to find and choose available services. With additional modules, a personal agent could provide some services by itself. For instance, it could run an email client to send and receive emails for the user.

5.2.4. Reputation Server

A reputation server literally means a server that stores the reputation data of users or services. As discussed in previous sections, the server could run on a separate machine or be implemented as a part of a H.A., F.A. or P.A. Since the reputation data are from many sources and varieties, a trust management model is included to calculate overall level of trustworthiness.

5.2.5. Policy decision point and enforcement point (PDP/PEP)

In addition to identifying roles and privileges, assigning roles to users, and associating privileges with roles, an access control system would never be complete without the set of access control policies, the decision-making and the policy enforcement mechanism.

A policy decision point is the place where the decision is made about whether to grant the permission to access a service. All access requests will first be transferred to this point, where the privilege of the requester will be verified against the access policies that are kept in a policy store.

The access request will be either permitted or refused. The duty of the policy enforcement point is to make sure that users cannot access the service if their requests are refused. In the mean time, the other users, whose requests are permitted, should be able to access the service.

5.2.6. Service server with policy Store

A service server is the actual place where a service is hosted. The service provider who offers the service manages it. To publish the service, it should register the service in the service directory and include all information needed to access it. It is also recommended to include key parameters of the offering service to help users select most satisfying ones. For instance, the frame rate, at which a digital camera could record, is one of the most important parameters of a video service that is offered to users to capture rapid motions in a car race and send them over the Internet.

A policy store is the repository of access control policies. Once an access request is received, the policy decision point will query the store to find the applicable access policies.

5.2.7. Service directory

A service directory is a place where information about local services is registered so that users or other services could search for desired services within the network. It contains a set of services that are available to users connected to the network. In order to help selecting better or preferred services, each service entry in the directory also includes attributes that values characterize the services. For instance, a printing service may register itself with an attribute stating the paper sizes that are available for printing.

5.3. Proposed system behavior

In this section, the behavior of the framework is explained with the help of a set of sequence diagrams, each presenting all the messages exchanged between the parties in a typical mobile service scenario happening in the context of mobile service. There are four phases throughout each scenario: authentication, service selection, access control and reputation update phase.

These phases are listed in the order of their execution and different tasks are performed during each phase. During the authentication phase, a user's PA will authenticate the FA where the user is visiting with the help of her or her HA, and vice versa. In the following service selection phase, the user's PA will discover all the available services around the user and select most preferred one according to the user's preferences. The access control phase covers from the time when the user starts requesting the access until the user stops using the service and releases occupied resources. Prior to be allowed to access the service, a policy decision point will verify if the user possesses a role which has been assigned sufficient privileges to access the service and a policy enforcement point will impose the judgment, positive or negative, that is made by the decision point. As a final step, the behavior of the user and the quality of the service are evaluated and the reputation data of each is updated in the last phase.

Although different tasks are to be completed in each phase, the execution of the tasks in a given phase relies on the results of those in earlier phases. For instance, all the tasks in the other three phases could not be executed if the authentication task in the first phase returns a negative result. Since only the result of each phase is passed to the next phase, it is possible to replace one or more of the suggested phases as long as the alternative implementation realizes the functions of the corresponding phases and generate result in the designated format.

5.3.1. Authentication phase

As explained in suggested protocol (Section 4.4), the authentication phase includes messages exchanged between a user's PA, FA and the user's HA. Six authentication transactions are completed in this phase. The execution of a business transaction will proceed if and only if this phase returns a positive result. Additional to a positive result, Alice receives a set of Roles from FA, each of which has the form of a certificate $CERT_{FA}(Rx, ID_{Alice})$.

5.3.2. Service selection phase

In this phase, Alice's PA will first search for available services and then negotiate with them according to her preferences, which are stored either with the HA or the PA. Although there are various technologies to discover services in distributed networks, Figure 22 shows a particular case where a centralized server maintains a Service Directory that keeps track of all available services in the network. When a new service is online or an existing service becomes offline, the Service Directory will be updated. For those who want to find a particular service in the network, they can simply send a query to the Directory and receive a set of available services from which they could choose. The server that hosts the Service Directory should have a standard configuration so that every entity in the network knows how to access it. In addition, the scenario shown in Figure 22 also assumes that a separate reputation server is present in the network. Therefore, all the reputation data of a specific user or service server will be stored and managed at the reputation server. Since the reputation server also registers itself as a special service at the Service Directory, the information about how to access the reputation server could be queried and obtained.

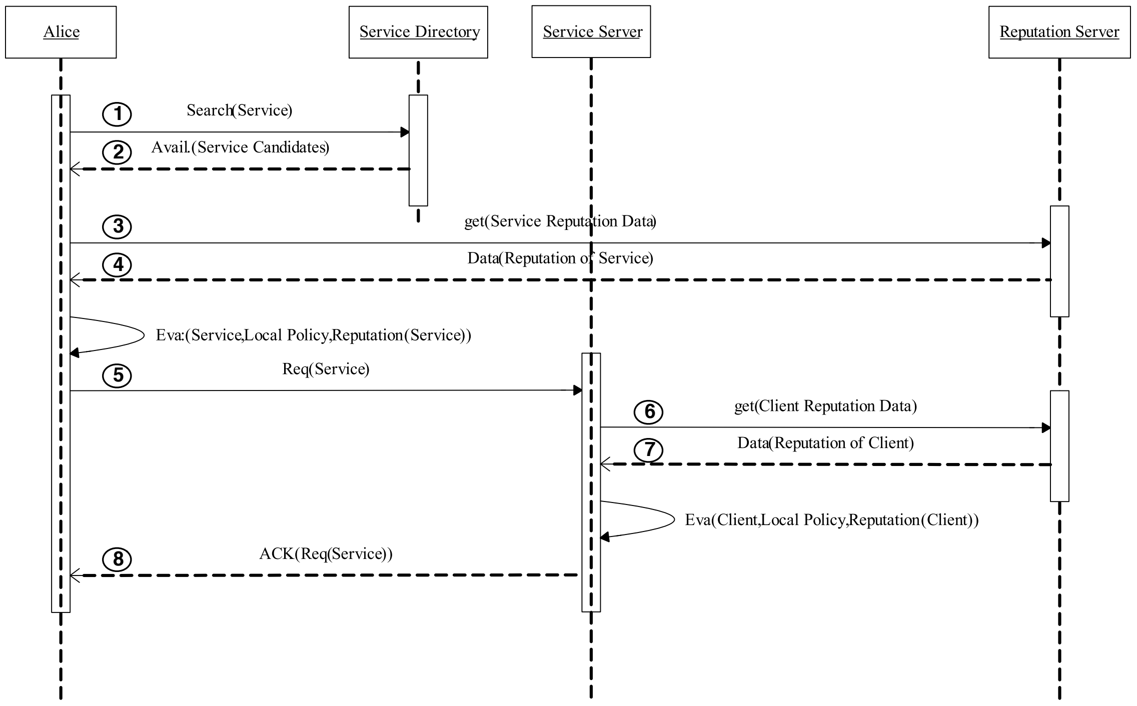

Alice
Service Directory
Service Server
Reputation Server
1 Search(Service)
2 Avail.(Service Candidates)
3 get(Service Reputation Data)
4 Data(Reputation of Service)
Eva:(Service,Local Policy,Reputation(Service))
5 Req(Service)
6 get(Client Reputation Data)
7 Data(Reputation of Client)
Eva(Client,Local Policy,Reputation(Client))
8 ACK(Req(Service))

Figure 22: Service selection phase

The messages shown in Figure 22 are explained in the following:

Message 1: Alice sends a query message to the Service Directory. The parameters included in the query indicate the type of the service and also some related attributes of the desired service. For instance, if Alice wants to search for a service to capture video images, a related attribute could be preferred frame rate.

Message 2: Once the query is received, the server that hosts the Service Directory will look through the directory and try to find a set of available services that satisfies user's preferences, which are specified in the query as a list of attribute values. The result is then sent back to the user.

Message 3: Alice's PA then issues another query message to the reputation server to collect the reputation data for each candidate service obtained in the previous message.

Message 4: To answer the query, the reputation server will assemble a collection of reputation data for all the services included in the query and send it back to the user.

At this point, Alice's PA will make a decision about which service is choose. The decision is based on the reputation data of the services and the policies pre-defined by Alice.

Message 5: As soon as a particular service is chosen, Alice's PA will send a request message asking for permission to access the service.

Message 6, 7: Similar to messages 3 and 4, the service server will issue a query to the reputation server, asking about Alice's reputation data. The reputation server will send back a collection of Alice's reputation data that was collected in previous transactions from different sources.

At this point, the service server will decide whether or not to serve Alice's request. The decision is based on the reputation data of Alice and the policies pre-defined by the service provider who owns the service.

Message 8: Once decided, the result of the decision is sent back to Alice's PA to inform it whether service request is approved.

If the result from the service server is positive, Alice's PA will understand that the service will be provided as long as Alice has adequate privileges to access it. The actual consumption of the service will take place in the next phase: authorization and access control phase.

5.3.3. Access control phase

In this phase, the service is actually provided. The scenario shown in Figure 23 is based on the assumption that the policy decision point (PDP) is located at the same place as the policy enforcement point (PEP). This assumption simplifies the message exchanges during the phase and reduces the total number of messages involved, especially the messages between the PDP and the PEP.

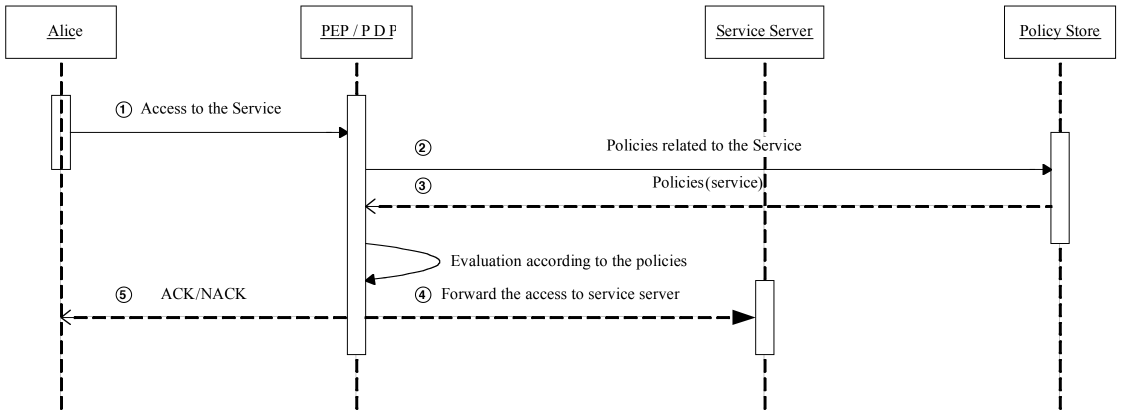

Alice
PEP / P D P
Service Server
Policy Store
① Access to the Service
② Policies related to the Service
③ Policies(service)
Evaluation according to the policies
⑤ ACK/NACK
④ Forward the access to service server

Figure 23: Access control phase

The messages shown in Figure 23 are explained in the following:

Message 1: Alice sends a service request message to the PDP/PEP. Along with the message, Alice's PA includes the identity of Alice and the set of active roles that Alice is currently possessing.

Message 2: The PDP/PEP will send a request message concerning all of the access control policies aimed at the particular service to the corresponding policy store.

Message 3: A collection of applicable policies is sent back to the PDP/PEP.

At this point, the PDP will then make the decision on whether to approve the access request from Alice, given the policies for the service and the active roles of Alice. Once the decision is made, the PEP will enforce the result, that is, will block the access in a negative case or let the request go through in a positive case.

Message 4: Only if Alice's request was approved, the access request will be forwarded to the actual service server, which will then serve Alice upon the reception of the request.

Message 5: No matter the service request was approved or not, a positive or negative acknowledge message is sent to Alice's PA. At this point, Alice's PA will learn that whether the request will be server or not. In the negative case, it has to choose another available service server and start the procedure again.

After the fifth message, Alice is either being served or being refused to be served. The phase will end right away in the latter case while in the former case it will end as soon as Alice stops using the requested service.

Note that although in Figure 23 the PEP and the PDP is located at the same place, it is possible to run these two components separately. In that case, the PEP/PDP becomes the PDP and the Message 4 will be forwarded to the PEP, which is running between Alice's PA and the Service Server.

5.3.4. Reputation data update

When Alice finishes using the service, the occupied resources are released. The final step is then for the service to evaluate the behavior of Alice during the service time and also for Alice to evaluate the quality of the service just received. This update will be combined with previous evaluations to generate the overall value of the reputation data for Alice and the service. This is shown in Figure 24.

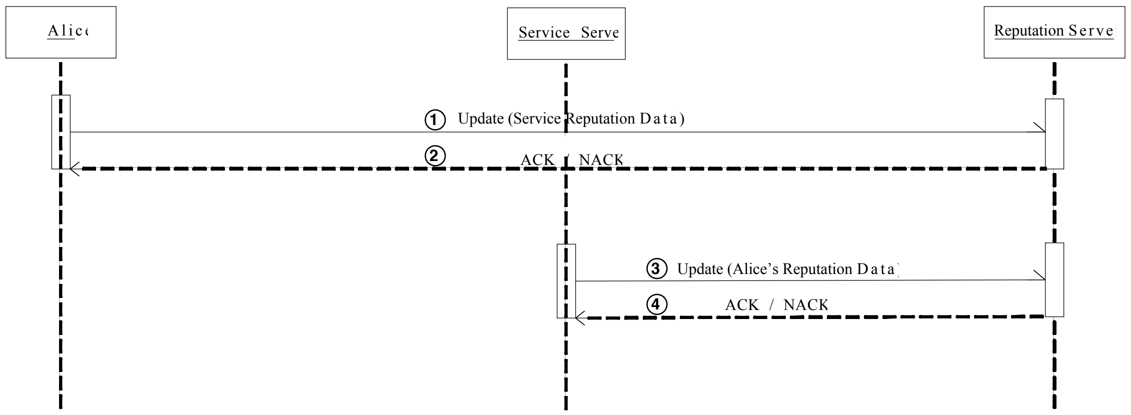

Alice
Service Serve
Reputation Serve
① Update (Service Reputation Data)
② ACK / NACK
③ Update (Alice's Reputation Data)
④ ACK / NACK

Figure 24: Reputation data update phase

The messages shown in Figure 24 are explained in the following:

Message 1: Alice sends an update message, carrying the evaluation result of the service she just received.

Message 2: A positive or negative acknowledge message will be sent back to Alice indicating whether the update was successfully stored.

Message 3: Similar to Message 1, an update message, carrying the evaluation result about Alice's behavior is sent to the reputation server.

Message 4: Similar to Message 2, an acknowledge message is sent to the service server.

Note that these two pairs of messages can be generated and exchanges in parallel.

5.4. Choices of technologies for implementing the proposed framework

Although the complete implementation of the proposed framework is not covered in this thesis, several comparisons have been done and suggestions about which technologies to use are made in preparation of a future implementation. The following includes two comparisons: (1) to compare and choose a language to represent the security and trust related information, and (2) to study and select a service discovery tool to find mobile services for a mobile user.

5.4.1. *The choice of a representation language for security and trust-related information*

There are many vendors who developed XML-formatted tools to represent security related information. A few typical examples are:

(1) Developed by the OASIS Security Services Technical Committee [48], Security Assertion Markup Language (SAML) profile of eXtensible Access Control Markup Language (XACML) from OASIS [49], which promises to standardize policy management and access decisions, provides a markup language to express the service access requests and decisions across different applications;

(2) XML Access Control Language (XACL) [50] from IBM, which is now graduated from IBM research lab and packaged into IBM's popular WebSphere Application Server as an internal component [51].

5.4.1.1. *SAML profile of XACML*

SAML is an XML-based framework for exchanging security information. Assertions about subjects are the key concept of the framework. Presented in XML-format, assertions express the information about acts carried out by subjects, attributes of the subjects, and authorization decisions about whether the resource access request is granted or denied, while a subject is an entity that would be either a human or computer that initiates the request. In the study cases described in section 1.2, Alice is a subject. As standard XML document, assertions may have a nested structure, whereby a single assertion might contain several different elements of authentication, authorization, and attributes. Assertions are issued by SAML authorities, namely, authentication authorities, attribute authorities, and policy decision points. Upon receiving a request, these authorities will use various sources of information, such as external policy stores and assertions that were received as input in requests for making a decision and create a response. To complete the framework, a protocol describing the behavior of clients and SAML authorities about how they exchange assertions is defined. Currently bound to the underlying Web Services protocol SOAP (over HTTP), this protocol consists of XML-based request and response message.

Although the single most important problem that SAML is trying to solve is the *Web Browser Single Sign-On* (SSO) problem, it provides an approach through assertion to state a subject can be trusted within a network, which fulfills the requirement of authentication and authorization to provide mobile services to a mobile user. One drawback of the mechanism is that assertions containing authentication statements merely describe acts of authentication that happened previously. In other words, reputation data of both users and service providers cannot be included in assertions.

5.4.1.2. IBM XML Security suite (XACL)

XML Security Suite (IBM) is a tool that provides security features such as digital signature, encryption, and access control for XML documents. Similar to existing policy languages, XACL describes access control policies in the form of object-subject-action-conditions. Throughout a target XML document, an object can be a single element, a single attribute, a set of elements, or a set of attributes. The notion of subject comprises identity, group, and role. As stated in the XACL documentation, there are four primitive actions (read, write, create and delete), but the structure of the language is not limited to these. Conditions are used to specify provisional actions such that the access is granted if the action gets logged. XACL is a provisional authorization model, that is, the decision of an access request is not simply "grant" or "deny." It tells the user that her or his request will be granted if he (or the system) takes certain actions or that his request is denied but the system must still take certain actions. XACL also supports hierarchical structures for roles and groups. While providing the benefit of propagating access policies downward or upward in the hierarchy, it creates possibilities for conflicts when a system tries to answer a user's access request. For example, the decision may be "grant" based on the user's current role while the policy denies an identical request from the parent role of this user. To solve this, XACL specifies three types of conflict resolution policies for each action namely: *dtp* (denials take precedence), *gtp* (grants take precedence) and *ntp* (nothing takes precedence).

XACL provides a mechanism to exchange access control policies and execute access control decisions across different applications and platforms, but it assumes that all parties are properly identified and authenticated. It is out of the scope of XACL to authenticate the identity of the initiator. As well, XACL does not cover the topic about how to determine which roles are to be assigned to a user after authentication. In other words, XACL need to work with other authentication and authorization mechanism to providing access control. In our case, the protocol described in Section 4.4 solves the authentication problem while Shi's trust model (Section 3.3) can be used for user and service authorization.

The following table summarizes the difference between XACL and SAML/XACML that are related to our framework. A more complete comparison can be found in Yagüe's survey on XML-based policy languages for open environments [52].

Features	XACL	SAML/XACML
Policy specification method	RBAC	Based on identity and credentials
Syntax	DTD	XML Schema
Complexity level	Low	High
Policies can be modified in a dynamic and transparent way	Yes	No
Application Scope	XML documents	Resources identifiable through a URI (any URI)
Access Control Schema	Based on DTDs to express RBAC elements	Based on XML schema to express identifies and conditions about the attribute certificates, the resource and the environment.

We suggest XACL to be used to implement the experimental system for the following reasons:

(1) XACL has just enough features to satisfy the requirement of the system. Given the hardware limitation, usually it has to be able to run on a hand-held device, it is crucial to keep the overhead cost of running an application as low as possible;

(2) While providing sufficient capabilities to carry out common access control tasks, XACL also provides adequate flexibility to describe complex access control policies. Its hierarchical structure, policy propagation and conflict resolution is highly efficient to be used to answer access requests and execute access decisions.

Note that although XACL is designed to be applied on XML document, it can be used for our proposed framework. To do this, we first define the user roles and resources in XML format so that they become a node of a XML document. Thus a request of a service is translated into a request to access a particular node in the XML document. XACL can be used to describe the latter request.

5.4.2. The choice of service discovery tool

Another choice to be made in order to implement a demonstration system is to find a service discovery tool that can be deployed on a hand-held device and run with limited computing resources. A few candidates are Jini Lookup Service, Web Services, and Sdptool from Familiar Linux [53] implementing Bluetooth Service Discovery Protocol as described in Section 2.1.2. Depending on the range of the search, different tool will be chosen. For instance, a user moving around in a conference place is likely to choose the Bluetooth technology to discover surrounding services.

6. Conclusion and future work

In this thesis, we studied how to deliver mobile services to mobile users in an open network. Given the uncertainty of available services and protection level of the services around a mobile user as well as the lack of knowledge about the user from the perspective of the service provider, we identify three major challenges:

1. To discover available service within the range of a user's WPAN and for multimedia services, to negotiate Quality of Service with available service providers;
2. To securely authenticate a user to a service provider and vise versa;
3. To control the access to a service.

To overcome the first challenge, we propose an architecture, of which a Personal Agent is used to maintain a list of services that a user can use while he or she is moving. We then implement a prototype system applying the architecture (see Section 2). This implementation involves various technologies: a simple SIP protocol implementation is developed to carry out the communication between a mobile user and a service provide; Java Media Framework (JMF) is deployed on three machines, one of them acts as a service provider while the rest implements as receivers; a Service Discovery Protocol (SDP) implementation from BlueZ is used to register services and to search available ones. Using performance analysis on several runs of the test case, we have shown it is possible to provide smooth switch from one service provider to another when a user is moving from one area into another. We also conclude that the QoS negotiation can also be done through a user's PA.

While we are trying to adapt Dupre's secure authentication protocol to authenticate a user to a service provider and vise versa in the context of mobile users and services, we identify the weakness of the protocol in terms of vulnerable to Deny of Service attack and exposure of user's data in plain text. To strength the security of the protocol, we proposed an improved version and testify it again several common security attack (see Section 4.4). We have also shown the improvement has not only strengthened the protocol, it also provides additional benefits such as the option of anonymity.

To control as well as protect a publicly available service, we propose a trust-based access control framework for mobile services (see Section 5). With all the components in place, namely a system to support mobile users and services (see Section 2.3), an authentication protocol (see Section 4.4), an existing access control model draft (see Section 3.2.4) and a trust management model (see Section 3.3), we have shown how to integrate them into a trust-based access control framework. In the case of a user requesting a service, the framework enables us to use the authentication protocol to identify the user and a service provider. We then uses a trust model to evaluate the reputation of the two parties and make decisions about whether the user will select the service provider or the service provider will grant the user for the service request. The framework also makes sure the access is controlled so that only users who have been assigned enough privileges can use the service. When the user finishes and releases the resource, the reputation data of both parties will be updated to reflect experience of the most recent usage.

In summary, the main contributions of this thesis are the following:

1. The integration of different aspects, including service discovery, QoS negotiation and continuous service delivery. The success of building a prototype and using it to run test cases proves that it is possible to put together individually developed technologies to achieve certain objective, which in this thesis is to support mobile users and services.

2. Improvements of Dupre's authentication protocol. Through the study, we identified the weakness of the original protocol and suggested several improvements. We then proved that these improvements not only eliminate the thread of interception but also enable user anonymity option. As well, the performance analysis shows the improved protocol can defend against typical security attacks

3. A proposal of a trust-based access control framework. The proposed framework integrates trust and role-based access control mechanism to support online transactions for mobile users and services. It provides four main features: service discovery, authentication, authorization, and access control.

Although the trust-based framework is proposed to support mobile users and services, it can also be applied for conventional services that are provided in a fixed location and for which users and service providers are known to each other. In this case, one or more phases of the framework can be skipped or greatly simplified. For instance, if Alice wants to use a printer located in her home domain, she needs not check the reputation data of the print service provider because she dealt with it before and a trust relationship is built beforehand. As well, the authentication phase may not need to involve PKI technologies if the local network is trustworthy and protected by a firewall from external access. Since the framework is made up of four phases, including authentication, service selection, access control and reputation update phase, and the technologies applied in these phases can vary depending on each particular application context, this framework is flexible enough to be applied in many environments, such as Internet applications and distributed applications, through minor adaptation.

Although a number of studies are made in this thesis to enable mobile services to be delivered to mobile users in open networks, there are still several interesting issues to be explored, such as the following:

1. In typical scenarios, we assume all devices are connected into one network across domains. However, when Alice enters into a new domain, the question how she can find an access point to connect to this network is not covered; as well, once the access point is found, how can Alice make sure it is trustworthy before her PA connects to it?

2. In the discussion of the improved secure authentication protocol, we mention that Alice's PA stores the certificate it received from its HA and uses it for signing documents and building security associations with other parties. However, this thesis does not cover how this certificate can be used when Alice leaves the current domain and enters into another domain: Should the certificate be revoked? When will the certificate be revoked? Can it stay valid forever? Does it mean that Alice does not need to request the certificate at the first place because she can store it in her PA all the time? One may answer yes to this, but there are other scenarios. For instance, what if she finds a computer workstation in the new domain and wants to use it? Can she transfer her certificate into the PC?

3. Also we mentioned that the four phases of the framework can use different technologies as long as the alternative implementation realizes the functions of the corresponding phases and generate result in the designated format, there is no guideline to help developers chose a particular technology under certain circumstances.

References

[1] The Digital Audio-Visual Council (DAVIC), http://www.davic.org/

[2] TV-Anytime Forum, http://www.tv-anytime.org/

[3] The Object Management Group, http://www.omg.org/

[4] Common Object Request Broker Architecture, http://www.omg.org/gettingstarted/corbafaq.htm

[5] Microsoft, *ActiveX*, http://msdn.microsoft.com/library/default.asp?url=/workshop/components/activex/activex_node_entry.asp

[6] A. Vogel, B. Kerherve, G. v. Bochmann, and J. Gecsei, "*Distributed Multimedia and QoS: a Survey*", IEEE Multimedia, vol. 2, no.2, pp.10-18, Summer 1995.

[7] I. Dupré-la-Tour, G. v. Bochmann and J. Y. Chouinard, *A secure authentication infrastructure for mobile communication services over the Internet*, in Communications and Multimedia Security Issues of the New Century (Proc. IFIP Working Conf. CMS'01, Darmstadt), pp.405 – 416, R. Steinmetz et al. (Eds.), Kluwer Academic Publ. 2001

[8] Bluetooth Specification Volume 03 Part B, *Service Discovery Protocol*(SDP), November 2004

[9] SUN, Technical White Paper*: Jini Architectural Overview* 2003, http://www.sun.com/software/jini/specs/jini2_0.pdf

[10] W3C, Working Group Note, Web Services Architecture, February 2004, http://www.w3.org/2002/ws/

[11] Java Media Framework (JMF) home page, http://java.sun.com/products/java-media/jmf/index.jsp

[12] BlueZ, Official Linux Bluetooth protocol stack, http://www.bluez.org/

[13] He, X., El-Khatib, K., Bochmann, G.v.: *A communication services infrastructure including home directory agents.* Technical report. University of Ottawa, Canada. May 2000

[14] Anerousis, N. et. al.: "*TOPS: An architecture for telephony over packet networks*", IEEE Journal of Selected Areas in Communications, Jan. 1999

[15] Roussopoulos, M., Maniatis, P., Swierk, E., Lai, K., Appenzeller, G., Baker,M.: P*erson-level Routing in the Mobile People Architecture.* Proceedings of the USENIX Symposium on Internet Technologies and Systems, October 1999

[16] Kahane, O., Petrack, S.: *Call Management Agent System: Requirements, Function, Architecture and Protocol.* IMTC Voice over IP Forum Submission VOIP97-010, 1997

[17] Wang, H. et. al.: *ICEBERG: An Internet-core Network Architecture for Integrated Communications.* IEEE Personal Communications (2000): Special Issue on IP-based Mobile Telecommunication Networks

[18] K. El-Khatib, Zhen E. Zhang, N. Hadibi and G. v. Bochmann, "*Personal and Service Mobility in Ubiquitous Computing Environments*", Journal of Wireless communications and Mobile Computing, v. 4, pp. 595-607, 2004.

[19] SDP Overview: BLUETOOTH SPECIFICATION Version 2.0 + EDR [vol 3] page 117 of 250

[20] Aaron Skonnard, “The birth of Web Services”, http://msdn.microsoft.com/webservices/webservices/understanding/webservicebasics/default.aspx?pull=/msdnmag/issues/02/10/xmlfiles/default.aspx

[21] SOAP 1.2 specification, http://msdn.microsoft.com/webservices/webservices/understanding/webservicebasics/default.aspx?pull=/library/en-us//dnsoap/html/understandsoap.asp

[22] Aaron Skonnard, “Understanding SOAP”, http://msdn.microsoft.com/webservices/webservices/understanding/webservicebasics/default.aspx?pull=/library/en-us//dnsoap/html/understandsoap.asp#understandsoap_topic4

[23] WSDL specification 2.0, http://www.w3.org/TR/2005/WD-wsdl20-primer-20050510/

[24] OASIS, UDDI specification 3.0.2, http://uddi.org/pubs/uddi_v3.htm#_Toc85908412

[25] Blackdown Java, Java Platform 2 for Linux, http://www.blackdown.org/java-linux/java2-status/index.html

[26] PKI Technical Specifications, http://csrc.nist.gov/pki/twg/baseline/pkicon20b.PDF.

[27] W. Simpson, PPP *Challenge Handshake Authentication Protocol* (CHAP), Request for Comments, RFC 1994, The Internet Engineering Task Force, August 1996

[28] B. Abdel Aziz, *Using IKE and Radius with MobInTel*, Technical report, University of Ottawa, Ottawa, Canada, December 2000.

[29] J. G. Steiner, B. Clifford Neuman, and J.I. Schiller. *Kerberos: An Authentication Service for Open Network Systems. In* Proceedings of the Winter 1988 Usenix Conference. *February, 1988*

[30] John T. Kohl, B. Clifford Neuman, and Theodore Y. T'so, *The Evolution of the Kerberos Authentication System. In* Distributed Open Systems*, pag*es 78-94. IEEE Computer Society Press, 1994.

[31] IETF, *SSL/TLS, RFC*, http://www.ietf.org/rfc/rfc2246.txt

[32] *Draft Security Assertion Markup Language Specification,* http://www.xmltrustcenter.org/saml/docs/draft-sstc-core-12-final.pdf

[33] *XKMS Specification,* http://www.verisign.com/resources/gd/xml/xkms/xkmsv1-1.pdf

[34] *XMLPay Specification,* http://www.verisign.com/resources/gd/xml/xmlpay/xmlpay.pdf

[35] *SSH Overview,* http://www.ietf.org/ids.by.wg/secsh.html

[36] *Introduction of Diffie-Hellman,* http://www.rsasecurity.com/rsalabs/faq/3-6-1.html

[37] *IETF draft of SHTTP,* http://www.ietf.org/proceedings/99jul/I-D/draft-ietf-wts-shttp-06.txt

[38] S. Glass, T. Hiller, S. Jacobs,and C. Perkins, *Mobile IP Authentication, Authorization, and Accounting Requirements*, Request for Comments, RFC 2977, The Internet Engineering Task Force, October 2000

[39] *Secure IP Overview,* http://www.ietf.org/html.charters/ipsec-charter.html

[40] R.Sandhu, E.J.Coyne, H.L.Feinstein and C.E.Youman, *Role-Based access control models,* IEEE Computer 29, 2(Feburary), 38-47, 1996

[41] A. Seleznyov, S. Hailes, *An access control model based on distributed knowledge management,* 18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004

[42] Greg Elofson, *Developing Trust with Intelligent Agents: An Exploratory Study*, Proceeding of the 1st International Workshop on Trust, pp. 125 – 139, 1998

[43] J. Shi, G. Bochmann, and C. Adams, *A Trust Model with Statistical Foundation*, Workshop on Formal Aspects in Security and Trust (FAST '04), Toulouse, France, 18th IFIP World Computer Congress, pp. 169-181, Toulouse, France, August 26-27, 2004

[44] J. Shi, G. v. Bochmann, and C. Adams, *Dealing with Recommendations in a Statistical Trust Model*, Proceedings of the Workshop on Trust in Agent Societies (held in conjunction with the 4th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)), Utrecht, The Netherlands, July 25, 2005

[45] Hui Li, Morad Benyoucef and G. v. Bochmann, *A Personalized Online Reputation Aggregation System,* University of Ottawa Telfer school of Management, Working papers 07-28, 2007

[46] PKCS#10 standard, http://www.rsa.com/rsalabs/node.asp?id=2132

[47] J. A. Chandler, K. El-Khatib, M. Benyoucef, G. v. Bochmann and C. Adams, *Legal challenges of online reputation systems,* Chapter IV in "Trust in E-services: Technologies, Practices and Challenges", R. Song et al. (eds), Idea Group Inc., 2006.

[48] OASIS Security Services (SAML) TC web site, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[49] SAML 2.0 profile of XACML specification, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf

[50] XACL specification, http://www.trl.ibm.com/projects/xml/xacl/xacl-spec.html

[51] IBM XML Security Suite, http://www.alphaworks.ibm.com/tech/xmlsecuritysuite

[52] Mariemma I. Yagüe, *Survey on XML-Based Policy Languages for Open Environments,* in Journal of Information Assurance and Security, Volume 1, Issue 1, 2006, pp.11-20

[53] The Familiar Project, http://familiar.handhelds.org/