A Secure Authentication Infrastructure for Mobile Communication Services over the Internet

Thesis by

Irénée Dupré la Tour

In Partial Fulfillment of the Requirements
for the Degree of
Master in Applied Science
In Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa
Ontario, Canada

March 2001

I hereby declare that I am the soul author of this engineering report. I authorize the University of Ottawa to lend this report to other institutions or individuals for the purpose of scholarly research.

Irénée Dupré la Tour

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

Irénée Dupré la Tour

Abstract

Mobile communication on the Internet sets more security concerns than traditional mobile networks such as GSM. The network infrastructure registration process should give credentials to the user to let him or her being identified by any service provider in order to prevent fraudulent use. In addition, a user should be able to communicate with privacy and to sign a message (e.g. a payment order) so that billing is possible. Users should be able to connect from everywhere, with various types of terminals, possibly mobile. In this thesis, we propose to secure an infrastructure providing telecommunication services on the Internet for mobile users. We establish a trust relationship between any pair of the parties with a password-based user access. As for user-to-user communication, both signaling and media data can be secured. We illustrate the use of this infrastructure to provide secure IP-Telephony.

Acknowledgements

First I would like to thank my co-supervisors, Dr. Gregor von Bochmann and Dr. Jean-Yves Chouinard, whose comments and support have been a great help during the preparation of this work. I learned many things from them, even though, mainly implicitly rather than explicitly. I would like to thank my colleagues in the Distributed Systems Research Group for their cooperation and friendship. I think I should specially mention Khalil El-Khatib and Ronggong Song for their valuable comments. I would also like to express my gratitude to my family especially my parents and my brothers, without whose help and support, this work would have been impossible.

Table of Content

ABSTRACT 3

ACKNOWLEDGEMENTS 4

TABLE OF CONTENT 5

LIST OF FIGURES 9

LIST OF TABLES 11

CHAPTER 1 INTRODUCTION 12

1.1 TELECOMMUNICATION SERVICES TREND 12

1.2 MOTIVATING EXAMPLE 14

1.3 THESIS OVERVIEW 15

CHAPTER 2 NETWORK SECURITY FUNDAMENTALS 16

2.1 DISTRIBUTED SYSTEM SECURITY SERVICES 16

2.1.1 Authentication 16

2.1.2 Authorization 17

2.1.3 Privacy 17

2.1.4 Integrity 18

2.1.5 Non Repudiation 18

2.1.6 Availability 19

2.1.7 Access control 19

2.2 CRYPTOGRAPHY OVERVIEW 20

2.2.1 Symmetric Ciphers 21

2.2.2 Asymmetric Ciphers 21

2.2.3 Key escrow and perfect forward secrecy 22

2.2.4 One-way hash functions 23

2.2.5 Digital signatures 24

2.2.6 Key management 25

CHAPTER 3 NETWORK SECURITY PRACTICE 27

3.1 AUTHENTICATION APPLICATIONS 28

3.1.1 X.509 authentication architecture 28

3.1.2 PGP Architecture 31

3.1.3 Kerberos 32

3.1.4 Password-based authentication mechanisms *35*
3.1.5 User identification schemes *39*
3.2 SECURITY PROTOCOLS 42
3.2.1 ISAKMP *42*
3.1.2 IPsec *44*
3.1.3 SSL/TLS *46*
3.3.4 SET *47*
3.3 MAIN ATTACKS AGAINST COMPUTER SYSTEMS 49
3.3.1 Attacks overview *49*
3.3.2 Attacks on ciphers and hash functions *50*
3.3.3 Attacks on protocols and operating systems *51*
3.3.4 Attacks on information in the network *52*
3.3.5 Sophisticated listening attacks *54*
3.5 CONCLUSION 55

CHAPTER 4 TELEPHONY APPLICATIONS AND THEIR SECURITY FEATURES 56

4.1 GSM ARCHITECTURE 56
4.1.1 GSM architecture overview *56*
4.1.2 GSM architecture *58*
4.1.3 GSM Security features *60*
4.2 PGPFONE 62
4.2.1 PGPfone overview *62*
4.2.2 PGPfone encryption *63*
4.2.3 User key exchange *63*
4.3 SESSION INITIATION PROTOCOL 64
4.3.1 SIP overview *64*
4.3.2 SIP architecture *66*
4.3.3 SIP message examples *66*
4.3.4 Security in SIP *70*
4.4 REAL TIME PROTOCOL 73
4.4.1 RTP/RTCP overview *73*
4.4.2 RTP security *73*

CHAPTER 5 MOBILITY ARCHITECTURES AND THEIR SECURITY FEATURES 74

5.1 PROTOCOLS FOR MOBILITY IN THE INTERNET 75
5.1.1 Mobile IP protocol *75*
5.1.2 Internet Mobile Host Protocol *77*

5.1.3 Security considerations ... *79*
5.2 OTHER ISSUES ON MOBILITY SUPPORT ... 80
5.2.1 Cellular IP ... *80*
5.2.2 Authentication with Mobile-IP ... *81*
5.2.3 Proposed add-on on Mobile-IP for session-based mobility ... *82*
5.3 A PROPOSED COMMUNICATION SERVICES INFRASTRUCTURE ... 84
5.3.1 Global mobility ... *84*
5.3.2 MobInTel architecture overview ... *85*

CHAPTER 6 A PROPOSED SECURE ARCHITECTURE FOR MOBILE INTERNET TELEPHONY ... 89

6.1 MOBINTEL TRUST AND SECURITY REQUIREMENTS ... 89
6.1.1 Authentication phase security requirements ... *90*
6.1.2 IP telephony security requirements ... *92*
6.2 REVIEW OF EXISTING APPROACHES ... 95
6.2.1 Mobile Communication Infrastructure ... *95*
6.2.2 Secure telephony on the Internet ... *97*
6.3 SECURITY SCHEME PROPOSAL ... 98
6.3.1 User Logon ... *98*
6.3.2 IP Telephony ... *99*

CHAPTER 7 PROTOCOL PROPOSAL ... 102

7.1 PROTOCOL SPECIFICATION ... 102
7.1.1 Authentication abstract specification ... *102*
7.1.2 Authentication message coding ... *109*
7.1.3 Phone call protocol ... *123*
7.2 SECURITY ANALYSIS ... 130
7.2.1 General remarks ... *130*
7.2.2 Attacks on the protocol ... *131*
7.2.3 Attacks on the phone call protocol ... *132*
7.3 Conclusion ... *133*

CHAPTER 8 IMPLEMENTATION AND ANALYSIS ... 134

8.1 IMPLEMENTATION DESIGN ... 134
8.1.1 Implementation choices ... *134*
8.1.2 Main blocks ... *135*
8.1.3 Protocols and languages ... *136*
8.1.4 Difficulties on the implementation stage ... *140*

8.2 TESTS SCENARIOS 142
8.2.1 Test architecture *142*
8.2.2 Test scenario *143*
CHAPTER 9 CONCLUSIONS 149
9.1 CONTRIBUTIONS 149
9.2 FURTHER WORK 150
BIBLIOGRAPHY 152
APPENDIX: 157
A - TABLE OF CRYPTOGRAPHIC ALGORITHMS 157
B - RELATIONSHIPS BETWEEN SOME PROTOCOLS IN THE OSI LAYER MODEL 158
C – ACRONYMS 159

List of Figures

Chapter 3

Figure 3A: Radius typical use ……..……35
Figure 3B: RADIUS-PAP message exchanges ……...……..….36
Figure 3C: RADIUS-CHAP message exchanges ……..…37
Figure 3D: RADIUS-OTP message exchanges ……...38
Figure 3E: SSH exchanges ……..39

Chapter 4

Figure 4A: Schematic presentation of the GSM network ……58
Figure 4B: Security architecture in GSM ……60
Figure 4C: Authentication message exchange in GSM ……61
Figure 4D: PGPfone "voice signature" mechanism ……64
Figure 4E: SIP message exchanges ……..66
Figure 4F: SIP scenario ……...67

Chapter 5

Figure 5A: Mobile IP architecture ……..….75
Figure 5B: Communication between mobile hosts ……..….77
Figure 5C: IMHP smooth handoff ……..…….78
Figure 5D: Session-based mobility protocol ……...…….83
Figure 5E: Service Agent architecture ……86
Figure 5F: MobInTel global architecture ……87
Figure 5G: Authentication message exchanges ……....87

Chapter 7

Figure 7A: Authentication protocol overview ……...……....102
Figure 7B: Authentication message exchanges overview – Ack ….……103
Figure 7C: Authentication message exchanges – Ack ……104

Figure 7D: Authentication message exchanges – Nack ……..106
Figure 7E: Home domain message exchanges overview – Ack ……..…107
Figure 7F: Home domain message exchanges – Ack ……..…108
Figure 7G: Home domain message exchanges – Nack ……..……..108
Figure 7H: MobInTel DTD – 1 ……111
Figure 7I: MobInTel DTD-2 ……111
Figure 7J: MobInTel DTD-3 ...……..113
Figure 7K: Phone call protocol overview ……..123
Figure 7L: Phone call message exchanges ……..124

Chapter 8

Figure 8A: MobInTel authentication module GUI ……...……..134
Figure 8B: MobInTel implementation main blocks ……...……135
Figure 8C: class diagram with package dependence ……..…..138
Figure 8D: Testing architecture ……142

List of Tables

Chapter 3

Table 3A: Cryptographic notations ……..28

Table 3-B. IPsec types of services ……...…45

Chapter 6

Table 6A: IP-telephony security requirements ……94

Chapter 7

Table 7A: Cryptographic notations ……...……103

Table 7B: Authentication phase implementation choices ……..109

Chapter 8

Table 8A: Bandwidth consumption comparison ……...…….147

Appendix

Table App-A: Cryptographic algorithms ……...……157

Table App-B: Protocol stack ……..158

Chapter 1 Introduction

1.1 Telecommunication services trend

Improvements in telecommunication network infrastructures, evolution of digital technologies and standardization have together made it viable to use existing data networks for multimedia applications. This clearly reverses the trend of the past where networks were dedicated to a specific application. Integration of all types of information - voice, data, video, and image - into a single network infrastructure is known as *network convergence*. The end result of this convergence is commonly referred to as the next-generation network (NGN), which, conceptually anyway, combines high-speed with the best features of each network.

During the last years, the Internet has emerged as the major network infrastructure, which will form the core of the NGN. As the amount of data traffic over the Internet is becoming larger than the amount of information sent over the traditional telephone network, the latter looses much of its importance. Following this trend, the major telecommunication players ask themselves today the question how the existing and future telecommunication services could best be provided over a network infrastructure, which is modeled after the Internet, that is, based on packet switching.

The issue is not only how to provide the same services over the Internet, but rather how to easily provide more advanced and integrated services, giving end-users flexibility. These new services include more and more multimedia content. Moreover, mobility support for these services is crucial for their future success. In the telephony context, IP-Telephony should not be only a reengineering of existing public switching telephony network (PSTN) services - like the plain old telephone service (POTS) - in the Internet but should rather implement new value added services such as teleconferencing and call filtering. This is getting more and more important because the cost reduction argument for investigating in IP-Telephony will not be a strong argument in the future [PPYSSMS99].

It seems that in this context where the Internet plays a key role, interest in the Intelligent Network standardization has diminished. Similar standardization efforts are underway for defining standard network middleware services over the Internet, which provide an efficient and flexible basis for developing new communication services and applications over the Internet. Several major efforts from universities and industrial companies have been contributing to the research in this network middleware field. The work has been focusing on creating a set of middleware services that match the requirements of advanced applications with respect to the resources provided by the NGN.

One of the main technical challenges to deal with is Quality of Service (QoS). Most existing multimedia applications are generally fixed concerning the media types they can process and present and they require the user to select certain QoS parameters [HEB-00-1]. But as the range of hardware, software, and media types is getting wider, the need for an automated system to do the selection process is increasing. Work done in the Distributed Systems Research Group at University of Ottawa (Ontario, Canada) shows how a QoS aware middleware can be used to merge the device profile and user profile and make, on behalf of its user, an automated selection of these QoS parameters. A scalable architecture has also been proposed to deal with user mobility [HEB00-2]. This thesis deals with the security aspect of this architecture. No architecture built on a QoS-aware middleware, supporting user global mobility and a full range of security services has been defined yet.

We should bear in mind that the work related to IP-telephony presented here could easily be extended to provide secure multimedia communications. Indeed, in [SR99] Henning Schulzrinne explains that while using the term of *Internet telephony*, "it should be understood that the addition of other media, such as video or shared applications, does not fundamentally change the problem. Also, unlike in traditional telecommunications networks where distribution applications (radio, television) and communications applications (telephone, fax) are quite distinct in terms of technology, user interface, communications devices and even regulatory environments, this is not the case for the Internet. The delivery of stored (*streaming*) media and telephone-style applications can share almost all of the underlying protocol infrastructure."

1.2 Motivating example

Let us consider the following motivating example that makes use of the infrastructure we deal with in this thesis.

A user named Alice living in Ottawa (Canada) is traveling to Paris (France) for business. She cares about using the Internet access and the terminal provided on the plane to visit her homepage to check the accounting information and pay the bill for using all kinds of services once a month. She can also do some shopping on the web and to check the change of stocks.

Arriving in Paris, she rents a cellular phone. She first reads short news and then she opens her address book stored in a server in Ottawa, click the name on the screen of her friend Bob who works in Montreal to make a confidential phone call to him. Bob is currently in Berlin (Germany). Even without knowing the current location of her friend, she is able to speak to him. The network middleware finds the location of Bob, establishes the session automatically with acceptable QoS for both parties. The security level used for this phone call is not the default one ("low-security") defined by Alice but rather the "high security" one specified for this particular call. After the call she can even join a private multimedia conference using her mobile terminal in a taxi which can access a wireless network. At this time, the bandwidth is lower and the white board application is "receive-only".

She suspends the session when she arrives at her hotel in Paris, and then uses her password to log into the network service from the workstation terminal in her hotel room. She restores the suspending session and continues to attend the conference with high-resolution video quality, CD quality audio and shared white board. She can also listen to music or see a movie in her spare time.

This example raises many security issues. How Alice is going to be authenticated to pay her bills in the plane or to access her address book stored in Ottawa? How can she trust the stocks information she receives in the plane? How can Alice and Bob can make a secure phone call? How can Alice can be authenticated by the local bandwidth provider to be billed for the resources she will use or by any

application service provider on the Internet? We deal with these issues in this thesis using a scalable infrastructure that provides IP-telephony for mobile users.

1.3 Thesis Overview

Security is concerned about ensuring that a system (networks and devices) resists to potential attacks that can compromise the secrecy, integrity, or availability of data and services. In Chapter 2 and 3, we present a brief literature review on network security. Some security infrastructures are presented in Chapter 3. In chapter 4, we study existing architectures providing telephony and current IP-telephony standards and their security features. Mobility issues are presented in Chapter 5. We detail the middleware infrastructure proposed to provide QoS-aware mobile multimedia personal communications. Chapter 6 studies the security requirements of the latter infrastructure to provide mobile Internet telephony. We then propose a scheme to secure the infrastructure. The specification and security analysis of the proposal are presented in Chapter 7. Chapter 8 explains the implementation choices and gives some test scenarios. We finally draw conclusions on the effectiveness of our proposition. Chapter 9 concludes this thesis and proposes some further work. Additional information such as tables, bibliography and acronyms is provided in the appendix.

Chapter 2 Network Security Fundamentals

We need a secure way to communicate in distributed systems. Thorough this thesis, we call *Alice (A)* and *Bob (B)* two honorable users and *Eve (E)* a malicious user. To prevent malicious acts in such systems, users can make use of cryptographic algorithms. In this chapter, we present security features described in term of security services. We also present the main types of cryptographic algorithms and their properties.

2.1 Distributed System Security Services

The Internet Engineering Task Force (IETF) has defined six security services [HA94]. These services are Integrity, Confidentiality, Authentication, Non-Repudiation, Access Control and Availability. The International Telecommunication Union (ITU) also defined about the same security services [ITU91]. Authorization is not explicitly part of these security services, but it is still considered as a very important service. In the next section, we define these services that are examined when discussing distributed systems security.

2.1.1 Authentication

We can make a distinction between two different types of authentication: message origin authentication and peer authentication. Message origin authentication means that we can be sure of the identity of the originator of a message. The message must also be unchanged if we want to be sure that the message is received as it was sent (this is integrity presented in Section 2.1.4). When Bob gets a message from Alice, it should be possible for him to be certain of the origin of the message. Nobody should be able to send a message that looks like having been sent by someone else. Peer authentication is a way of identifying a peer at the other end of a secure connection. It ensures that the person is who he claims to be and we can be sure of it. Typically, systems rely on

user authentication. If the system has an account for "Alice", and Eve wants to use it, the system challenges Eve for a token. This token is based on something that the valid user should know (PIN, password), should have (seal, smart card, credit card), should be (fingerprint, retina pattern, iris, voiceprint, handprint) or should be able to do (signature, handwriting, rhythm typing). Suppose the system asks for a user password. If Eve replies with the correct password, then the system let her use Alice's account. Of course, if Alice has given her password to Bob, Bob can use her account. All you know is that the person at the other end has access to the password. This is a general problem with computer authentication based on something that a user should know or should have. The main threat related to authentication is spoofing. There are several electronic solutions to provide authentication studied below, including password-based mechanisms, digital signatures (Section 2.2.5) and digital certificates (Section 3.1.1).

2.1.2 Authorization

When two parties conclude a contract, both of them have to sign it. The signers have to know that the other party is authorized to sign contracts. If neither party has authorization, the contract is invalid. Authorization in a computer system means that someone, a user or a program, gets rights to do something. Alice may have different privileges depending on what she's trying to do. She may be able to read everything in one database, not be able to access anything in a second database, but be able to read or write data in a third. Today users usually have to identify themselves to get access rights. This is often unnecessary, because the computer system does not need to know the identity of the user; it only needs to know if the user has the right to access to information or to perform an operation. A proxy is an entity that allows user to operate with the rights of the principal that granted the proxy [Neum93]. For instance, Alice can have a certificate that allows her the right to use some service. Solutions that provide authorization are often specific to a system.

2.1.3 Privacy

When people talk about computer security, privacy (or confidentiality) is usually the first thing that comes to mind. According to Edward Amoroso, the majority of research and development in computer security has specifically focused on disclosure threats [Amor94]. Disclosure is a threat that

involves someone accessing information that should not be seen. Respectively, confidentiality means that only legitimate users have access to information. In a company, for example, only the president and members of the board have access to all information concerning future decisions; chiefs of departments may have some information concerning decisions that affect their department. The chiefs must not tell their subordinates about the secrets. Typically, data are transmitted in clear text, so if your name goes across the net, a snooper could read it. Identity privacy is called anonymity. Encryption scrambles the data in some way so that the snooper cannot read it. Presumably, the intended receiver knows the way to decrypt it to get back the original data. Usually information must be protected from both disclosure and integrity threats (Section 3.3). The real-life equivalent for encryption is an envelope in which you put the message. Network listening is one of the main threats on privacy. An electronic solution is data ciphering (Section 2.2).

2.1.4 Integrity

It is important for any system that sensitive information is correct and has not been modified. In hospitals, for example, the patient's medical history has to be correct when a doctor chooses a medicine for the patient who is, e.g., allergic to something. In computer security, integrity means that only those people who have the right to change information can do that. The information must not change during storage or transmission. If the representation of some information has been changed even by a mistake, the integrity of this piece of information has been compromised. For transmissions, this means "Is what you got the same as what I sent?" Eve could have intercepted the email Alice sent and changed it; or it could have been mangled in transit. A main threat is data alteration. Electronic solutions are based on hash-algorithms, MAC (Message Authentication Codes) values (Section 2.2.4) and digital signatures (Section 2.2.5).

2.1.5 Non Repudiation

A contract is usually accepted by signing it. Every party gets its own copy of the contract. If the content becomes disputable, nobody can deny that the contract was signed since everybody has an identical copy (this assumes, of course, that the authenticity of the signatures can be verified). In a distributed system, non-repudiation means that the sender should not be able to deny later that he

has sent a message or that the receiver cannot deny that he has received the message. Typically, in electronic commerce, a client should not be able to deny that he has ordered a product. In telecommunication services, a client should not be able to deny that he has ordered to use a service like video-on-demand or to use network resources to make a phone call. There is no particular threat against non-repudiation apart from denial. In the computer world, non-repudiation is carried out with digital signatures (2.2.5) conceptually similar to ones in the real world.

2.1.6 Availability

Working outside an office is becoming more popular. Many companies also have several offices around the world. If services of a company such as a database or a web server are centralized in one or a few locations, service availability becomes crucial. System availability means that information and computer resources are available when a legitimate user needs them. The threat of Denial of Service (DoS) (Section 3.3) occurs when access to some computer resource is blocked. The blocking may be permanent or long enough that the use of the resource is no longer useful. Disclosure and integrity have been considered to be the primary threats. Less attention has been paid to availability and denial of service attacks until recently. An example of electronic attack is TCP-SYN flooding (Section 3.3). Electronic solutions include redundancy and the use of firewalls.

2.1.7 Access control

Access control is the combination of authentication and authorization. It is a mechanism to control access to a computer system and information. There are two main types of access control: discretionary access control (DAC) and mandatory access control (MAC). A discretionary access control allows users to affect how their files and computer resources are protected. Of course, the security policy limits user power to allow access. A mandatory access control does not allow users to influence access control. The system administration makes decisions according to the security policy. Typically, both discretionary and mandatory access controls may be used together.

2.2 Cryptography overview

Cryptography is the science of encryption and decryption. Modern encryption also includes the concept of a key, which is used by an algorithm to encrypt or decrypt a message. Security in cryptography comes from both the algorithm and the key. If the algorithm allows easy attacks, the system will be weak. Small keys are vulnerable to *brute force attacks* (trying all possible keys) (Section 3.3) and weak keys combined with an algorithm are also vulnerable. A system is secure if it is computationally infeasible to recover the key or the *plaintext* from the *ciphertext*. As time progresses, processes that were computationally infeasible become feasible with increased computing power. Some cryptographic algorithms may therefore become outdated extremely quickly. The remainder of this section is a review of cryptography algorithms and methods that are commonly used.

When Alice and Bob want to communicate securely through an unsecured network, they have to use cryptography to protect their communications. A *cryptosystem* consists of an algorithm that is used to secure communications and the keys that are used for encryption and decryption. All plaintexts and ciphertexts belong also to the *cryptosystem*. First Alice and Bob choose an algorithm for protecting communication, then they agree on a key. Alice uses the chosen key to encrypt the message and sends the *ciphertext* to Bob. Then Bob can decrypt the *ciphertext* and he gets Alice's original message. Alice can use two different types of methods to encrypt a message: she can use symmetric cryptography or public key cryptography. A cryptographic algorithm is a mathematical function that is used for encryption and decryption. A synonym for cryptographic algorithm is a *cipher* [Schn96]. We will not focus on algorithms in themselves but rather on the different types of algorithms. A table of existing algorithms is presented in the Appendix (Table App-A).

A *key* is a series of data, a string of numbers and/or characters. It has a certain length, which is usually given in bits. Typically, a key length can range from 56 bits up to several kilo bytes. It can be stored in a file or in a chip. A key can be sent to somebody through the network. A key can have a lifetime depending of the *cryptosystem* and the agreement for the use of the key. The main threat against the concept of key is the brut force attack i.e. trying all possible keys.

2.2.1 Symmetric Ciphers

Symmetric encryption (or private-key encryption) uses the same key to encrypt and decrypt a message. The length of the key is exponentially proportional to the strength of the encryption. Symmetric encryption usually uses short keys (less or equal to 128 bits). To ensure the best security, the key should be as random as possible. A totally random key that is only used once is the ideal form of symmetric encryption, and such a scheme is called a One-Time Pad. The common standard for symmetric encryption is DES (Data Encryption Standard) which uses a 56-bit key. It is being phased out in favor of AES (Advanced Encryption Standard) [AES00] recently defined by the NIST (National Institute of Standards and Technology). DES security can be expanded through the repeated encryption of a message with two or three different keys. This process is called Triple-DES. Symmetric encryption provides confidentiality. Note that the strength of such ciphers cannot generally be proved mathematically. They make use of a few cryptographic functions (permutation, substitution, XOR, addition and multiplication modulo a number) that are combined together to form the algorithm. Private-key cryptosystems enable to cipher roughly around 1000 times faster than the public-key ones. There are numerous symmetric algorithms. The main standard algorithms are DES, 3-DES, Blowfish, IDEA, CAST and AES.

2.2.2 Asymmetric Ciphers

Asymmetric (or public-key) encryption uses two different keys during the encryption and decryption processes. The keys have certain mathematical qualities, which allow one key to be used to decrypt what the other key has encrypted. The keys have to be large enough in order to prevent one key being calculated from the other key. Because of these factors one key can be publicly distributed (the public key usually noted KU).

Alice knowing Bob's public key can send an encrypted message to the Bob who owns the private key (usually noted KR). In this use, asymmetric encryption provides confidentiality. If Bob encrypts a message with his private key, asymmetric encryption provides both authentication and confidentiality. Public keys are made available to applications, hosts and services. The *public key* authenticity can be certified by a *Certificate Authority* (Section 3.1.1) in order for a community of users

to trust that a public key really belongs to a principal. Another approach is to keep public keys in a public repository managed by a trusted party or to let each user decide the keys he trusts. A *private key* belongs to an entity and is never revealed to anyone. It is used by the entity to decrypt incoming messages that are encrypted with the principal's public key. It is also used to sign an outgoing message sent by the principal to anyone else. This provides non-repudiation and authentication, as anyone can use the principal's public key to verify the signature, to be sure that the message originated from that principal.

Many security mechanisms make use of public-key cryptography especially to provide authentication. The main ones are PGP (Section 3.1.2), SSL/TLS (Section 3.1.3) and IPsec (Section 3.1.2). Asymmetric ciphers are very slow compared to symmetric ones, but they are more secure. Public key technology is commonly used to secure short messages or very important messages where real-time encryption and decryption is not an issue. The main public-key algorithm standards are RSA (Rivest-Shamir-Adelman) and ECC (Elliptic Curve Cryptography).

2.2.3 Key escrow and perfect forward secrecy

A *key escrow* system uses public key cryptography to encrypt and decrypt messages. The difference between the standard public key implementation and a key escrow system is that with key escrow, copies of the private key are split into pieces and stored by a trusted third party. In the case of the Clipper Chip (developed by the US government) [Clipper94], a 80-bit key was to be split into two 40-bit keys that were to be stored with two independent agencies. The benefit of a key escrow system is that if the private key is ever lost, it can be recovered from the independent agencies. The downside of this mechanism, from the perspective of privacy advocates that the government can also recover the private keys with a justice court order. The fact that key recovery can give government access to a corporation's or foreign government's private messages has prevented the wide acceptance of key escrow systems. However this capability has kept key escrow at the top of the US government's list for exported encryption technology, and has also kept it one of the most hotly debated subjects in the cryptography field today. Perfect forward secrecy (PFS) in a key establishment protocol is the condition in which the compromise of a session key or long-term private key after a given session does not cause the compromise of any earlier session.

2.2.4 One-way hash functions

Many situations arise when a *fingerprint* of a file or message must be generated. Such a *fingerprint* is called *Hash-value, Hash-code* or *Digest* and is produced by a cryptographic hash function (noted H). The latter can be applied to a block of data of any size and produces a fixed-length output (often 16 or 20 bytes long).

The one-way property means that it is virtually impossible to generate a message that fits a given a code. Technically, H being a hash function, h = H(x) is relatively easy to compute for any given input x but given any given code h, it is computationally infeasible to find x such that H(x) = h. The effort to break H is roughly 2^m (where *m* is the output vector bit length). This property is important if the input of the hash-function involves the use of a secret value. Ideally, even the smallest change to the input data will change about half of the bits in the result. The weak collision resistance property implies that an alternative message hashing to the same value as a given message is virtually impossible to be found as explained in the following. Technically [Stallings99], for any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x). This prevents forgery when an encrypted hash code is used. The strong collision resistance property stands that it is computationally infeasible to find any pair (x,y) such that H(x) = H(y). That refers to how resistant is the hash function is to the birthday attack (Section 3.3). The number of random inputs to try is around $2^{m/2}$ (with an *m*-bit output) for the probability to generate at least a message y given h such that H(y) = h is equal to 0.5.

Typically, an authenticated digest is appended to the message at the source. The receiver authenticates the message by re-computing the digest. Because the hash function itself is not considered to be secret, some means is required to protect the hash value (see digital signature Section 2.2.5). A widespread hashing algorithm is called MD5 (Message Digest version 5). It generates a 128-bit (16-byte) hash, and is considered reasonably secure. Other common used standard algorithms are SHA-1 and RIPEMD-160 (20-byte output). An added digest (or hash-value) provides integrity.

Message authentication code (MAC) techniques enable two parties (A and B) that share a common key to authenticate their transmissions (assuming only A and B know the key). When A has a message to send to B, it calculates the MAC as a function of the message and the key: MAC = $f_K(M)$. Both the MAC and the message are transmitted to B. B compares the received MAC with the MAC he generates using the same secret key. If the two MACs match, that means B is assured that the message has not been altered and it comes from A. A MAC function is similar to hashing but the algorithm need to be reversible, as it must for decryption. Also it turns out that because of the mathematical properties of the authentication function, a MAC is less vulnerable to be broken than encryption [Stallings99]. The process provides authentication but not confidentiality. The difference with a digest is in the use of a shared key. Different solutions have been found to construct such MACs. HMAC [KBC97] is a hash function-based MAC that was designed to meet the requirements of the IPsec working group in the IETF, and is now a standard and it has received widespread use. Some other common MACs are CBC-MAC (Cipher Block Chaining Message Authentication Code [BKR94]) which specifies that a message $x=x_1 \ldots x_m$ is authenticated among parties who share a secret key *a* by tagging *x* with a prefix of $f_a^m(x) = f_a(f_a(\ldots f_a(f_a(x_1) \text{ xor } x_2) \text{ xor } \ldots \text{ xor } x_{m-1}) \text{ xor } x_m)$ where f is some underlying block cipher (e.g. DES) and *a* is its key. This method is pervasively used internationally and is a U.S. standard. Other methods such as XOR MAC (based on the use of a finite pseudorandom function [BGR95]) have been defined. The effort to break MACs is roughly $2^{size\_of\_the\_key}$ for a brute-force attack on the key.

2.2.5 Digital signatures

Digital signature is a combination of several of the above technologies (public key and hash algorithms). A digital signature is the digest of a document encrypted with a private key. It provides integrity and authentication. Assume Alice sends a signed message to Bob. The signature can be attached, for example, to an email message that itself is sent in clear text. In order to authenticate the sender, Bob decrypts the signature with Alice's public key to get the digest. To check the integrity of the message Bob ensures that the digest matches the digest computed from the message received. The main signature digital signatures standards are the Digital Signature Standard (DSS) and the Rivest-Shamir-Adleman (RSA) signature. A digital signature has all the characteristics of a real signature and it provides authenticity and integrity.

2.2.6 Key management

For conventional encryption to work two parties of an exchange must share the same key and that key must be protected from access to others. For two parties A and B, key distribution can be achieved in a number of ways, as follows. First, a key can be selected by A and physically delivered to B. Secondly, a third party can select the key and physically deliver it to A and B. Thirdly, if A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key. Fourthly, if A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B. For link encryption, manual key delivery (such as previous scenarios one and two) is not scalable since a key for each pair of host is needed that is N(N-1)/2 pairs (for N hosts). If encryption is done at the application level, then a key is needed for every pair of users and the same scalability problem applies. Key distribution using conventional encryption often uses the last scenario (number four). It assumes that each user shares a unique master key with the Key Distribution Center (KDC). This master key is used to deliver session keys. Session-keys have a lifetime and are renewed regularly. An example of KDC-architecture is Kerberos (Section 3.1.3).

One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key encryption in this regard: the use of public key encryption to distribute secret keys and the distribution of the public key itself.

For privacy of communication, data have to be encrypted. One could use public-key encryption based on the keys provided by digital certificates. This could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption. In order to get the benefit of both speed and easy security, a hybrid system is used. If Alice wishes to establish a secure connection with Bob, she generates a session key K_S (symmetric key) and encrypts it with Bob's public key with an asymmetric algorithm. K_S is then used to secure all subsequent messages. The encrypted session key is called a *digital envelope* because this data (envelope) must be opened (decrypted) before the data can be read. The session key is then sent safely across the network. Bob decrypts K_S with his private key. At this point both sides have the same session key, although the latter was created based on the input from only one side of the communications. The

benefit of that key exchange scheme is that it has less computational overhead than the *Diffie-Hellman* (DH) algorithm (Section 3.2.1). In the latter algorithm, the key exchanged between the two parties is based on information held by both users. Another advantage of DH algorithm is long-term or perfect forward secrecy (Section 2.2.3). If Eve records all communication between Alice and Bob and gains access to Bob's computer, no session keys are disclosed. In the previous public key delivery scheme, access to Bob's private key allows to decrypt all the digital envelopes and therefore all data.

There are many applications, especially transaction-oriented applications, in which the session keys change frequently. These changes can use the previous K_S to transmit the new session key. Public-key encryption can be used occasionally to update K_S.

Several techniques have been proposed for the distribution of public keys and they can be grouped into the following general schemes: public announcement, publicly available directory, public key authority and public key certificate. Public announcement of public keys is used by PGP (Section 3.1.2). The major weakness of this scheme is that anyone can forge such a public announcement. It is up to the user to decide whether he trusts the key or not. In the second scheme, a trusted entity maintains a publicly available directory with a {name, public key} entry for each participant. It is a more secure way to provide keys than the first scheme but it still has vulnerabilities. An opponent can tamper with the records kept by the authority. Moreover, information given in the directory is not authenticated and thus can be modified on the wire. In the public key authority scheme, a public authority provides tighter control over the distribution of public keys from the directory. Each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. That way, any participant *A* could request the authority to provide the public key of any receiver *B*, encrypted with the private key of the authority in order to authenticate *B*'s public key. This scheme is attractive but still has some drawbacks: the public-key authority could be the bottleneck of the system due its central role. Public-key certificates are widely used and are explained in Section 3.1.1.

Chapter 3 Network Security Practice

In this chapter we give a quick overview of network security from a practical point of view. Current security architectures evolve in a way such that they rely strongly on a public-key infrastructure (PKI). We present such architectures to figure out whether they can be integrated in an IP-telephony infrastructure for mobile users or not. The X.509 standard authentication scheme is the core of a PKI. PGP (Pretty Good Privacy) is noteworthy for its original approach of public-key management. Kerberos is a famous authentication system that relies on symmetric encryption only. Password-based authentication mechanisms are the most popular authentication schemes and are very simple for mobile users. We study a few of them. The Radius protocol is presented as an example of AAA (Authentication Authorization and Accounting) protocol. AAAs are used by many companies and ISPs (Internet Service Providers) to offer user authentication and authorization.

After this overview, some important security protocols currently used on the Internet are presented. ISAKMP (Internet Security Association and Key Management Protocol) and IKE (Internet Key Exchange) are used as building blocks in security applications to handle key management and security association (i.e. fully defined secure connection) management. IPsec (IP security protocol) with IKE is an often-cited solution to provide security services at the IP level. IPsec is used to build VPNs (virtual private networks). SSL (Secure Socket Layer) is the most popular solution to protect web transactions and web access. SET (Secure Transaction Protocol) is used especially to secure web transactions.

We finally briefly study the main attacks against computer systems. We check in Chapter 8 that our proposed architecture answers to these kinds of attacks. Thorough this chapter and the rest of the thesis, the following notations will be used to describe exchanges that use cryptographic tools:

ID	userID (ex: alice@ottawa.ca)
KU_X/KR_X	public-key of X/private-key of X
Ks_X	session key (secret key)
N	nonce value
TS	timestamp
HV	hash-value
SecCx	secure connexion
E_K[data]	means data are encrypted using key K
pwd	password
$K_{X,Y}$	key shared by X and Y
Ack	Acknowledgement
Nak	Negative acknowledgement

Table 3A: cryptographic notations.

3.1 Authentication applications

3.1.1 X.509 authentication architecture

3.1.1.1 X.509 Digital certificate

A *Digital Certificate* is data about an entity (e.g. a person or a company) that establishes a trusted link between a public key and its holder. That data is stored in a particular format that allows many different programs to identify the contents of that information. The principal format standard is the X.509 certificate [X509-93] defined by the ITU in the X.500 framework recommendations. Version 1 (1988) defines base criteria; Version 2 (1992) provides flexibility of names and Version 3 (1993) allows extensions (adding some information in a certificate e.g. role and authorization).

An entity's certificate (say Alice's one) contains three sections:

- Information about Alice such as her name, e-mail address (alice@site.uottawa.ca) and location. It could also contain complementary information such as her home address, work address and telephone numbers.

- The Public Key section stores Alice's public key. This not only ensures that the certificate is bound to a particular key, but it allows the recipient of the certificate to engage in secure communications with the holder of the certificate (and private key).
- The *Certification Authority* (CA) (see below) section contains a signature of the certificate. The CA is a trusted authority, which has checked or which states that the information in the certificate was correct at the time of signing.

In practice, Alice sends her private key to the trusted authority (CA). The latter generates a certificate for the user signed with the private key of the CA. A digital certificate helps to provide authenticity. If Alice wants to send Bob an email with privacy, she first needs to get Bob's certificate and then she uses his public key to encrypt the email. Alice, or rather her security software can use the *Certificate Authority's* public key to check that the certificate is valid. To check a certificate completely, you need both the CA certificate and the last CRL (certificate revocation list). The latter is often unavailable if the application is not connected to a complete PKI infrastructure (see Section 3.1.1.4). Web based applications often rely upon public key and X.509. Built into current web servers and browsers, certificates scale well in the hierarchical trust model. On the other hand, they require a substantial infrastructure: CA, parent CA, directory server, and unified namespace (see Section 3.1.1.4). They pose significant problems with mobile users. Indeed, they must move certificates from system to system, or install multiple certifications. Their use on public workstations is very problematic and can be costly. Today, browsers allow ciphered storage of the private key but it is not securely portable from one computer to another. You also need to protect the private key on the local machine (with a ciphered storage on the disk) or in a secure card.

3.1.1.3 Certification Authorities

Behind the certificate is the concept of trust. There is no reason for you to believe anything you see on the Internet without some form of trust. So the idea of a chain of trust was created. Say you believe that a particular company (or organization) will not allow a lie to be placed in a certificate. When you see the signature of that company on a certificate, you would therefore believe the contents of that certificate. This company is called a *Certificate Authority (*CA*)*. Well-known CA players include Verisign, Entrust, CertPlus and Thawte. This trust in a company can be extended. If

a company that your application trusts (say Verisign) states that another organization (say "SITE, U. of Ottawa") can be trusted, your application will trust certificates signed by "SITE, U. of Ottawa". "SITE, U. of Ottawa" becomes a certified certification authority (CA). Relations of trust between organizations form a trust tree. The organization at the top of the trust tree is called a root certification authority (CA). A root certificate authority generates a self-signed certificate whose fingerprints (or digest) are made publicly available. Verisign is such a root certificate authority.

There is no world root certificate authority because of the many dangers that could arise. The disclosure of the hypothetical root CA private key would break up the whole authentication tree. Every certificates would have to be revoked and re-issued. Moreover that root CA would have full control to authenticate or revoke any certificate. That would give too much power in a single entity. In order to connect the different trust trees, cross certification authorizes trust transfer between hierarchies. A root certificate may produce a certificate for another root certificate such that the public key of the latter is certified by the former. Thus users certified by CAs in different trees could find a path in the tree to check a certificate. Each CA has a certification policy (CP) that defines the security policy (certificates lifetime, emission period of Certificate Revocation List (CRL), responsibility, assurance level). The Certification Practice Statement (CPS) states precisely operational procedures, standards used in the infrastructure to fulfill identified functions in a CP. Both CPS and CP need juridical advisee to be written. The set of certificates of a CA is stored in a certificate repository (X500, LDAP,…) that is accessible according the CP. A CA also publishes regularly a CRL to revoke certificates when a CA is compromised, a private key is compromised, a user status changes or a user is suspended. Verisign, Entrust and most CAs support the traditional model of CRL where the whole revocation list is published. Delta CRLs (showing the difference between two successive CRLs) have been standardized to be distributed on CRLs distribution points (CDPs) using the online certificate status protocol (OCSP). There are three types of CA for organizations. In the first type you administrate your own CA (and have the full responsibility). In-source CA uses a rented skilled team to manage the infrastructure of an organization. This method is fast but expensive. Out-source CA manages registrations on your own but not certificate management. Today in-sourcing is growing in popularity.

3.1.1.4 Public Key Infrastructure

Public Key Infrastructure (PKI) is the combination of software, encryption technologies, and services that enable organizations to protect the security of their communications, especially business transactions on the Internet. PKIs integrate digital certificates, public-key cryptography, and certificate authorities into a company network security architecture. It is built to minimize user intervention. PKI services include key generation, key life cycle management (generation, registration, certification, authority key distribution, key usage, certificate validation/revocation, key expiry/key archival, key update) and key use (authentication, integrity, confidentiality, non-repudiation, communication, notary). Applications make calls (through an API) for PKI services but they are not directly involved in providing those services. A PKI is application independent. Tomorrow, PKI will have full non-repudiation support, archival and authorization management (through a PMI: privilege management infrastructure). PKIs appear to be part of the long-term future of the Internet. SPKIs (Simple PKIs) have been proposed to make PKI implementation easier. A PKI includes many entities including a root CA, CA, Registration authority (check identity), register repository, a certification practice statement (CPS), certificate management protocol, personal security environments, user API (Application Programming Interface). Note that the Personal Security Environment (PSE) is also part of the PKI infrastructure.

3.1.2 PGP Architecture

Pretty Good Privacy (PGP) [Stallings99] is a software that uses cryptography to provide data security for electronic mail and other applications on the Internet. PGP has grown explosively and is now widely used. It is freely available on a variety of platforms. It is based on algorithms considered as extremely secure (RSA, DSS, DH, CAST-128, IDEA…). Its source code is freely available and it was not developed by any governmental or standards organization. PGP provides authentication (through digital signatures), confidentiality (through message encryption), compression, e-mail compatibility (Radix64-conversion) and message segmentation.

For each message sent with PGP by Alice to Bob, a session key chosen by Alice is associated and is used for encrypting and decrypting that message. An encrypted form of the session key that was used accompanies Alice's encrypted message. The session key itself is encrypted with Bob's public key (i.e. digital envelope). Bob can recover the session key using his private key.

PGP has a particular approach of public-key management i.e. to establish ownership of public keys. No rigid public-key management scheme is set up. In particular, the way of establishing trust and establishing certifying authorities is not specified but several suggested options can be used. Suppose Alice wants to get Bob's public-key. A can physically or by telephone get the key from Bob. A could obtain Bob's public key from a trusted certifying authority. Finally, the main scheme uses a *web of trust*. Alice could obtain Bob's public key through a mutual trusted individual D. D would issue a signed certificate for Bob. Bob's key legitimacy is related to the one of D from Alice point of view. This *web of trust* technique is used for building a file of validated public keys by making personal judgments about being able to trust certain people to be holding properly certified keys of other people. Each key of this file contains a *key legitimacy field*, a *signature trust field* and an *owner trust field* that indicate the degree to which a particular public-key is trusted as valid, trusted to sign or trusted to sign other certificates. This file constitutes the personal public key ring. Using this file people trust each other's public key when they communicate. The owner of a public key can issue a key revocation certificate to revoke the use of a public key.

3.1.3 Kerberos

3.1.3.1 Kerberos Overview

Kerberos is an elaborated authentication service developed at MIT (Massachusetts Institute of Technology, USA). The Kerberos system is a widely used implementation of secure communication channels, based on the DES encryption scheme. Integrated into the DCE (Distributed Computing Environment), Kerberos is currently a *de facto* standard in the UNIX community. Several Kerberos-like systems have appeared (KryptoKnight of IBM; SESAME of Bull and Siemens). That approach offers a major improvement in security over that which is traditionally available within UNIX. Its primary limitation is that applications using Kerberos must be modified to create communication channels using the Kerberos secure channel facilities.

It is intended to authenticate users to servers and servers to users. Unlike most authentication schemes, Kerberos relies exclusively on conventional encryption, making no use of public-key encryption. Version 5, the latest, has been issued as a RFC [RFC1510] in 1993. Kerberos System

follows the KDC scheme (Section 2.2.5) and has a centralized structure. Each realm contains one master KDC, which acts as a trusted third party between the client (user) and the server or resource for which authentication is being requested. KDCs may be replicated for reliability and load balancing. A KDC is comprised of three functional components: *Authentication server (AS),* responsible for authenticating the user, *Ticket granting server (TGS),* used by authenticated users to gain access to specific servers and a *secret key database*. The authentication server shares a unique secret key with each server in its realm. Therefore, All servers of the realm are registered with the Kerberos server.

The basic Kerberos protocols revolve around the use of a trusted authentication server, which creates session keys between clients and servers upon demand. The basic scheme is as follows. At the time the user logs in, he or she presents a name and password to a login agent, which runs in a trusted mode on the user's machine. The user can now create sessions with the various servers that he or she accesses. The user requests that the authentication server creates a new unique session key and send it back in two forms: one for use by the user's machine and one for use by the file server. The authentication server, which has a copy of the user's password and also the secret key of the server itself, creates a new DES session key and encrypts it using the user's password. A copy of the session key encrypted with the server's secret key is also included. The resulting information is sent back to the user, where it is decrypted.

The user now sends a message to the remote server asking it to open a session. The server can easily validate that the session key is legitimate, since it has been encrypted with its own secret key, which could only have been done by the authentication server. The session key also contains trustworthy information concerning the userID, workstationID, and the expiration time of the key itself. Thus, the server knows with certainty who is using it, where the user is working, and how long the session can remain open without a refreshed session key.

3.1.3.2 Kerberos Message Exchanges

Six messages are exchanged to obtain a service. This scenario takes place in three parts. First, the user identifies himself to the authentication server to obtain a ticket-granting ticket. A ticket is a series of pieces of information (encrypted or not) that especially identifies the beneficiary and the

issuer of the ticket. This "ticket-granting ticket" is presented to the "ticket-granting service exchange" to obtain a "service-granting ticket". The latter that authorizes the client to use a service is sent to the actual server that provides the service. The details of the message exchanges are presented in [Stallings99].

3.1.3.3 Inter-realm authentication

For a user in his home realm wishing service on a server in another realm, Kerberos provides inter-realm authentication if the two Kerberos servers are registered with each other. If there are N realms, then there must be N(N-1)/2 secure key exchanges so that each realm can interoperate with all other realms. For inter-realm authentication, Authentication Servers have to trust each other. Note that a user outside his realm must first connect to his authentication server.

3.1.3.4 Kerberos limitation

Perhaps the most serious exposure of the technology is the one associated with operation during a network partitioning. If a portion of the network is cut off from the authentication server for its part of the network, Kerberos session keys will begin to expire, and it will be impossible to refresh them with new keys. Gradually, such a component of the network will lose the ability to operate, even between applications and servers residing entirely within the partitioned component. In future applications requiring support for mobility, with links forming and being cut very dynamically, the Kerberos design would require additional development.

A less-obvious weakness of the Kerberos approach is the one associated with active attacks on its Authentication and Ticket-Granting Server. The server is a software system operating on standard computing platforms, and those platforms are often subject to attack over the network. A knowledgeable user might be able to concoct a message, which will look sufficiently legitimate, to be passed to a standard service on the node; this message will then provoke the node into crashing by exploiting some known intolerance to incorrect input. Thus, one could imagine an attack on Kerberos or a similar system aimed not at breaking through its security architecture, but rather at repeatedly crashing the authentication server, with the effect of denying service to legitimate users.

3.1.4 Password-based authentication mechanisms

The following protocols are authentication protocols. They do not provide any encryption key to the final user for future communications during the authentication process.

3.1.4.1 Radius and AAA protocols

Radius (Remote Authentication Dial-In User Service) [RFC2138, RFC2139] is an access control protocol that is designed to function as a management protocol enabling communications between clients and servers for authentication, authorization, and accounting of various services. It is typically used by Internet Service Providers (ISPs) to authenticate users that connect through a dial-up service. Three actors are involved: the remote user, the Network Access Server (NAS) and the Authentication, Authorization, and Accounting (AAA) server (with the user profile database).

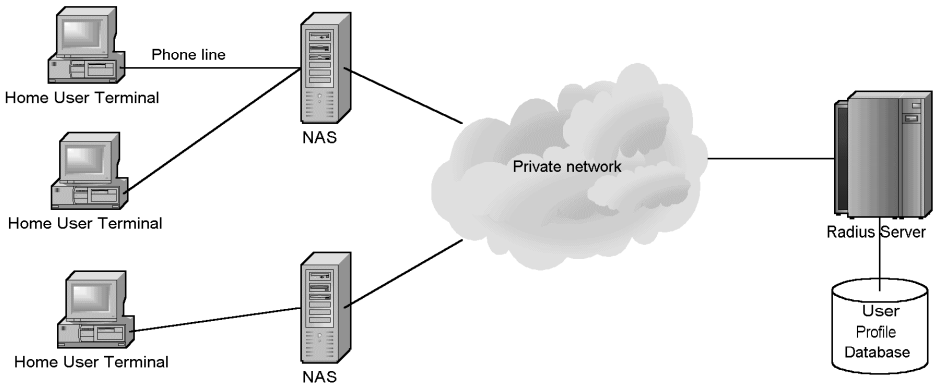

Figure 3A: Radius typical use.

Radius describes communication between a NAS (the Radius client) and an AAA server (the Radius Server). It supports multiple authentication mechanisms and it is extensible to cope with new architectures such as IPsec, tunneling and dial-in roaming. The communication between the NAS and the AAA server is protected by authentication and encryption services. Many NASs are deployed for a given network (this is typically the case for an ISP network), which usually has a centralized AAA server. Communication between Radius-Client and Radius-Server uses the scheme "MD5 (nonce+shared secret) XOR *data* (128 bits)" to encrypt *data.*

Other major AAA protocols include Diameter, COPS and TACACS. Diameter enhances Radius and provides many extensions. Common Open Policy Service Protocol (COPS) [RFC2748] is also an admission control protocol under development by the IETF RSVP working group. COPS has been originally specified to allow authorization of RSVP resource requests in IntServ (Integrated Services). However, it is applicable to authorize access to generic resources in the network. Terminal Access Controller Access Control System (TACACS) [RFC1492] and developed by Cisco Systems Inc. It is a password-based authentication protocol working over TCP/IP. TACACS+ (the latest version) has the three AAA functionalities. Authentication is provided with the PAP/CHAP support (see next sections) and the encrypted transmission of the password between the NAS and the TACACS server. Authorization is provided with the transmission of the services access list according the user profile in the authorization message of the TACACS server. Accounting comes from the accounting attributes: UserID, user protocol, used service, time, number of packets and commands. TACACS+ is especially adapted to control the access to routers. Password-information is encrypted the same way as in RADIUS.

3.1.4.2 Password Authentication Protocol

The *Password Authentication Protocol* (PAP) is a simple authentication mechanism in PPP (Point-to-Point Protocol). PAP is the main authentication protocol used on Internet access through PPP. In PAP, a user identifier and password are transmitted in clear text. [RFC1334] The client sends the login and the password; the server sends back the acknowledgement or not. PAP is also supported by Radius and the message exchanges in that scenario are presented in Figure 3B.

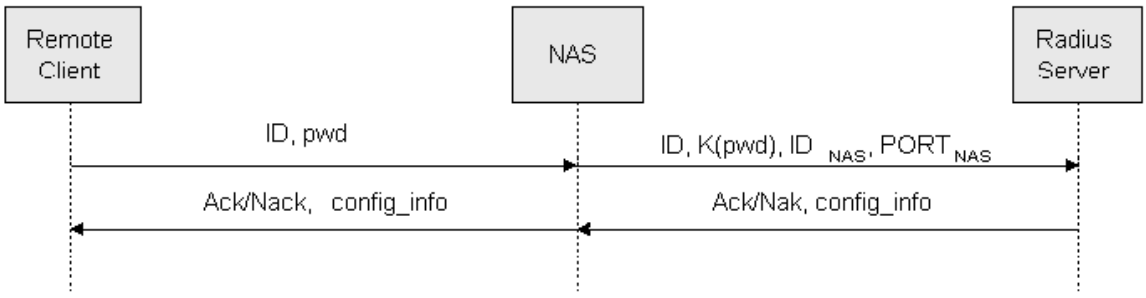

Figure 3B: RADIUS-PAP message exchanges.

There are many PAP-variants: SPAP (Shiva Password Authentication Protocol), ARAP (Appletalk Remote Access Protocol). Note that PAP is not secured against eavesdropping.

3.1.4.3 Challenge Handshake Authentication Protocol

The *Challenge Handshake Authentication Protocol* (CHAP) [RFC1994] is a peer entity authentication method for PPP, using a randomly generated challenge and requiring a matching response that depends on a cryptographic hash of the challenge and a secret key. CHAP is also supported by Radius and the series of exchanges in that scenario are presented in Figure 3C.

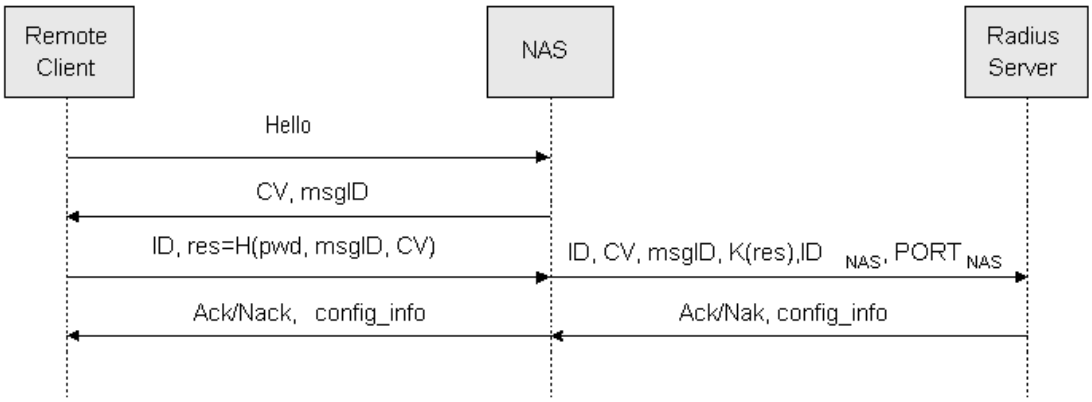

Figure 3C: RADIUS-CHAP message exchanges.

Typically, the client sends an Hello message, the server replies with a message identifier and a challenge value. The client sends back his name in clear and the hash-value of the password, the messageID (msgId) and the challenge value. Finally, the server sends back its acknowledgement or not. MD5 is usually used as a cryptographic hash function and msgID is used to prevent replay attacks. One should note that the name of the client is transmitted in clear text.

3.1.4.4 One Time Password

The *One-Time Password* (OTP) [RFC1938] is an Internet protocol that uses a cryptographic hash function to generate one-time passwords for use as authentication information in system login and in other processes that need protection against replay attacks. OTP is supported by the *Extensible Authentication Protocol* (EAP) [RFC2284]. The client (say Alice) sends first her name and receives from

the server a challenge value (CV) for example: "Otp-md5 487 xyz7". She concatenates the password with the unpredictable seed (xyz7) and pass the result as an input to a hash function (MD5) for a certain number of iterations (487). This is the first OTP-answer. The authenticator server keeps this initial value in memory. Finally, the server sends back the acknowledgement or not. When a new authentication is to be done, Alice's generator does the same as above but now the number of iterations is reduced by one (486 iterations here). The authenticator server receiving this new OTP has only to pass it through the hash function once and compare the result with the memorized OTP to perform the client authentication. With this method, you only need to check the password once (for the first OTP message) per iteration cycle. OTP is also supported by Radius and the series of exchanges in that scenario are presented in Figure 3D.

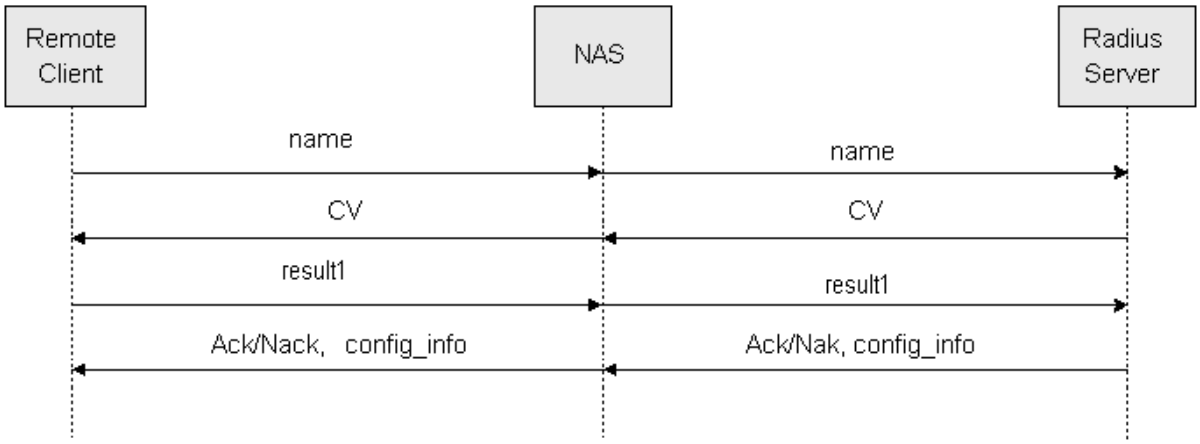

Figure 3D: RADIUS-OTP message exchanges.

3.1.4.5 Secure Shell

The *Secure Shell Protocol* (SSH) [SSH00] is a protocol for secure remote login and other secure network services (e.g. secure file transfer) over an insecure network. Secure Shell is mainly used in Unix systems to establish a secure channel between a client and a server. It consists of three major components: - Transport layer protocol: Provides server authentication, confidentiality, and integrity. It may optionally provide compression. The transport layer will typically be run over a TCP/IP connection, but might also be used on top of any other reliable data stream. - User authentication protocol: Authenticates the client-side user to the server. It runs over the transport

layer protocol - Connection protocol: Multiplexes the encrypted tunnel into several logical channels. It runs over the user authentication protocol. The message exchanges is presented in Figure 3E.

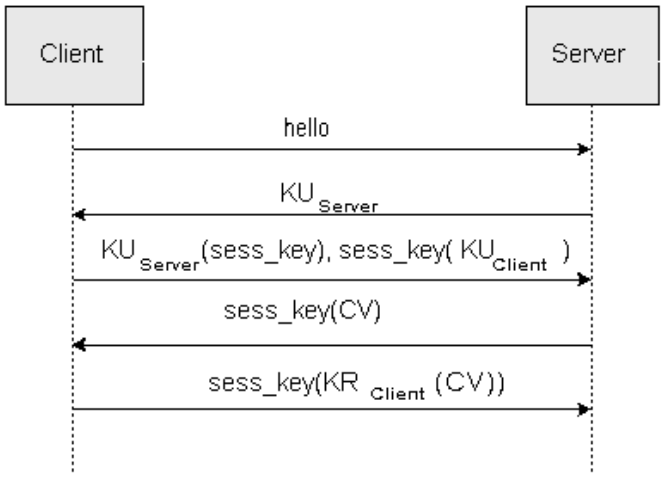

Figure 3E: SSH message exchanges.

SSH can also be used with a password instead of using public-key cryptography). There is a trust issue for the second message (public key sent by the server). A third party could intercept the communication between the two entities and send back its own public key. It is up to the user to decide whether he wants to trust the key or not.

3.1.5 User identification schemes

Let us study authentication methods from the user point of view. Generally, user identification schemes use one or more of the following objects:

- Method 1: User's knowledge: PIN, password
- Method 2: User's possessions: seal, physical key, ID card, credit card.
- Method 3: User's biometrics: fingerprint, retina pattern, iris, voiceprint, handprint, face pattern.
- Method 4: User's action: signature, handwriting, typing rhythm.

Method 4 uses generally a tablet and a digitizer with pen, but they are not popular devices yet. In [SOM98], a user identification system using a mouse to check signature is proposed. It uses geometric average means of the figure to match the signature written on the screen with the mouse. Using a signature learning-process, they have achieved a successful verification rate of 93%.

Method 3 uses a measurable physical characteristic or personal trait to recognize the identity or verify the claimed identity of a person through automated means. The measurable physical characteristic can be fingerprints, retinal scan, iris scan, voice print. On the one hand, it is easy to use, and difficult to deceive, unless the system is already compromised. On the other hand most implementations require relatively costly hardware at each workstation and can give false negative results.

The fingerprint scanners are recognized as a tough security system to crack, so they represent a high level of security. The cost per user is more than a hundred dollars so that it may be a disadvantage if this technology has to be deployed on a large user base. Face recognition systems offer many perspectives as authentication of every user who approaches the machine.

Voice authentication systems are perhaps the least expensive to implement, and they require no special hardware, and some centralized systems are ideal for granting mobile users remote access. But the most robust of these centralized systems can be pricey.

Hand Recognition (Hand Geometry) systems analyze and measure the shape of the hand. There are three different technologies that look at the shape of the hands or fingers: hand geometry, single-finger geometry and two-finger geometry. The finger/hand geometry systems do not raise many privacy issues and the technology is easy to use but expensive (a couple of thousand dollars per unit, depending on quantity and configuration).

Iris recognition: Iris recognition technology involves the use of a camera to capture an image of the iris (the colored portion of the eye). The iris is an excellent choice for identification: it is stable throughout one's life, it is not very susceptible to wear and injury and it contains a pattern unique to the individual. Indeed, an individual's right and left iris patterns are completely different. Iris scan is

the most expensive biometrics technology, costing tens of thousands of dollars. This technology has only been commercially available for a few years and is not being mass-produced.

Retina analysis is a physical biometrics method that analyzes the layer of blood vessels situated at the back of the eye. The retina is the surface inside the back of the eye, upon which images that have passed through the pupil are focused. This technology is generally used for physical security applications rather than for data security applications.

In method 2, the traditional passwords are translated (and improved) in a physical way into security-access cards, also called "smart cards". Smart Cards are already widely used in Europe. In order to explain the role Smart Cards play in security, it is important to provide an overview of what Smart Cards actually are. Smart Cards are typically credit cards, which contain a small amount of memory and sometimes a processor. Since smart cards contain more memory than a typical magnetic stripe and can process information they are being used in security situations where these features are necessary. They can be used to hold system logon information along with other personal information on the user, including passwords. They allow users to store passwords for applications that they use frequently on the smart card, so they don't have to remember and re-enter information each time they use the application. However, the applications of Smart Cards are not reduced to identification. There are other uses as the data capture and collection. Smart Cards can also be used to capture and collect information about people to make their daily experiences more enjoyable. We may draw a parallel between this technologies and the cookies, which give many information's about the user (such as nationality, interests) to the different visited sites with intent to answer the best possible needs of the customer. The possible applications of Smart Card technology are still being discovered. Smart Cards so-called "security cards" are relatively easy and inexpensive to deploy to a large user base. But they are inconvenient if users lose them or forget them at home, and a found card can let an unauthorized user into the system. The latter can be avoided if one adds a PIN number mechanism necessary to use the card. A smart card can replace many passwords and it is a solution to avoid people from having to remember a lot of passwords.

The degree of security of method 1 is low since it is very easy to imitate knowledge like password. However, password is the simplest and the most popular user identification scheme, and does not need exclusive devices.

3.2 Security protocols

A key concept that appears in security protocols concerning both authentication and confidentiality is the security association (SA). An SA is a one-way relationship between a sender and a receiver that affords security services to the traffic carried on it. An SA includes all needed information to execute network security services. The kind of information that SA specifies are an authentication method, an encryption algorithm, encryption and authentication keys, the lifetime of the encryption key, the lifetime of the SA and a sequence number for replay prevention. Secure associations are generally unidirectional and can be set manually or automatically. If a peer relationship is needed, for two-way secure exchange, then two SAs are required.

3.2.1 ISAKMP

3.2.1.1 ISAKMP Overview

"ISAKMP [Internet Security Association and Key Management Protocol] defines payloads for exchanging key generation and authentication data. These formats provide a consistent framework for transferring key and authentication data which is independent of the key generation technique, encryption algorithm and authentication mechanism." [RFC2408]. There may be many different key exchange protocols, each with different security properties. However, a common framework is required for agreeing on the format of Security associations (SA) attributes, and for negotiating, modifying, and deleting SAs. ISAKMP serves as this common framework. It defines things like packet formats, retransmission timers and exchange state requirements. Authentication relies on digital certificates owned by the entities or users to establish a secure connection. Password-based identity establishments are not considered in this context because of the inherent weakness in the use of a password. A digital signature algorithm (Section 2.2.4) must be used within ISAKMP's

authentication component. However, ISAKMP does not mandate a specific signature algorithm or certificate authority (CA).

Requirements that should be evaluated when choosing a key establishment algorithm include: establishment method (generation vs. transport), perfect forward secrecy (Section 2.2.3), computational overhead, key escrow (Section 2.2.3), and key strength. Based on user requirements, ISAKMP allows an entity initiating communications to indicate which key exchanges it supports. After selection of a key exchange, the protocol provides the messages required to support the actual key establishment. For key negotiation and exchange, Internet Key Exchange must be supported by ISAKMP.

3.2.1.3 Internet Key Exchange

The Internet Key Exchange (IKE) [RFC2409] protocol operates within the ISAKMP framework. IKE is also used by other protocols such as IP security (IPsec). This protocol establishes a bi-directional security association between communicating systems that is policy negotiation (agreeing on encryption algorithms and hash functions) and establishment of authenticated keys. IKE describes a specific key exchange based on the Oakley [RFC2412] key exchange. IKE specifies that authentication can use Digital Signature, public-key encryption or a pre-shared key. Several Key exchange methods can be used (RSA, fixed Diffie-Hellman, ephemeral Diffie-Hellman, anonymous DH, Fortezza, etc.). Diffie-Hellman is very commonly used to establish a shared-secret key between two peers.

3.2.1.4 Diffie Hellman key exchange

The Diffie-Hellman (DH) key exchange allows two users to exchange a secret key over an insecure medium without any prior secrets. The DH algorithm is begun by two users exchanging public information: both parties agree upon a generator and a prime number. The Diffie-Hellman protocol relies on the difficulty of extracting discrete logarithms modulo a prime. This is difficult for almost all primes, but a deliberate search can produce some weak primes for which the problem is much easier.

Each user mathematically combines the other's public information along with their own secret information to compute a shared secret value. This secret value can be used as a session key or as an encryption key for encrypting a randomly generated session key. This method generates a session key based on public and secret information held by both users. The protocol depends on the discrete logarithm problem for its security. It has been demonstrated that breaking the DH protocol computationally difficult, if keys are of sufficient length. The benefit of the DH algorithm is that the key used for encrypting messages is based on information held by both users and the independence of keys from one key exchange to another provides perfect forward secrecy (Section 2.2.3).

3.1.2 IPsec

3.1.2.1 Architecture and services

IPsec, a short form for IP Security is both a set of protocols and an architecture being developed by the IETF to support secure exchange of packets at the IP layer. The base architecture for IPsec compliant systems is described in [RFC2401]. The overall IPsec proposed standard is described in many other RFCs ([RFC1826-1829]) and drafts. IPsec is expected to be deployed widely to implement secure branch office connectivity (Virtual Private Networks), secure remote access over the Internet (secure call to an ISP) and extranet and intranet connectivity with partners. IPsec is optional for IPv4 (IP version 4) and it is required for IPv6 (IP version 6).

IPsec can encrypt and/or authenticate all traffic at the IP level. Thus, all distributed applications can be secured. When IPsec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter.

There are currently two specific headers [RFC2402-2406] that can be attached to IP packets to achieve security: the IP Authentication header (AH) and the IP Encapsulating Security Payload (ESP) header. AH provides integrity and authentication (using HMAC-MD5 HMAC-SHA-1) and ESP provides integrity and confidentiality (using 3-DES, RC5, IDEA, 3-IDEA, CAST). Table 3-B shows a brief summary of services that IPsec can provide:

	AH	ESP (encryption only)	ESP (with authentication)
Data integrity	✓		✓
Origin Authentication	✓		✓
Replay protection	✓	✓	✓
Confidentiality		✓	✓
Prevent traffic analysis		✓	✓

Table 3-B. IPsec types of services.

3.1.2.2 Transport and tunnel mode

Both AH and ESP support two modes of use: transport and tunnel mode. Transport mode protection extends to the payload of an IP packet (TCP or UDP packet for instance). Typically, transport mode is used for end-to-end communication between two hosts. This mode leaves the header untouched so that anybody can see the origin and destination of the packet. ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload and selected portions of the IP header. The more secure Tunnel mode encrypts both the header and the payload. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet is treated as the payload of new "outer" IP packet with a new outer IP header. The entire original packet travels through a "tunnel" from one point to another. A number of hosts on networks behind firewalls may engage in secure communications without implementing IPsec. The unprotected packets generated by such hosts are tunneled through external networks by tunnel mode set up in a firewall or a secure router at the boundary of the local network. ESP in tunnel mode encrypts and optionally authenticates the entire inner IP packet, including the inner IP header. AH in tunnel mode authenticates the entire inner IP packet and selected portions of the outer IP header.

3.1.2.3 Key management

The key management portion of IPsec involves the determination and distribution of secret keys. The IPsec architecture mandates support for two types of key management: manual and automated.

The first mode where an administrator manually configures systems with some keys is useful for small environments. The automated mode enables on-demand creation of keys for SAs and facilitates the use of keys in large distributed environments. IKE (Section 3.2.1.3) is the default automated key management protocol for IPsec. IKE is used for SA management and it is meant for establishing, negotiating, modifying, and deleting SAs in the IPsec context.

3.1.3 SSL/TLS

Secure Socket Layer (SSL) [SSL98] was initiated by Netscape and version 3 has been designed with public and industry review. Transport Layer Security (TLS) [RFC2246] is the IETF version of SSLv3 with a few added enhancements and is backward compatible with SSLv3. Most recent web browsers support SSL/TLS.

SSL is actually two layers of protocols. The first one above TCP includes *SSL Record Protocol*. Three higher-level protocols are also defined: *SSL Handshake Protocol*, *SSL Change Cipher Spec Protocol* and *SSL Alert Protocol*. *SSL Record Protocol* provides Confidentiality and Integrity for SSL connections. It fragments, optionally compresses, adds MAC, encrypts, appends SSL record header to the result and provides the result to TCP. Two protocols following their names take care about the alert messages, and a possibly change of cipher specification. *The Handshake protocol* is the most complex part of SSL and consists of a series of messages exchanged by client and server. In the first phase, the two entities establish security capabilities. Then server authentication, optionally client authentication (if it has a digital certificate) and key exchange take place. Finally, final messages verify that the key exchange and authentication processes were successful.

The "key exchange" process occurs in several steps. First, a pre-master secret is exchanged using RSA (encrypted with the public key of the server) or Diffie-Hellman (both exchanges DH public keys to create it). From the pre-master secret, a master secret is calculated (with a pseudo-random function using the pre-master key as a seed value). The real key used for the secure connection is generated with the same pseudo-random function using the master secret as a seed value. TLS differs from SSLv3 on a few details including the pseudo-random function design, additional error codes and padding rules.

3.3.4 SET

3.3.4.1 Overview of SET

SET (Secure Electronic Transactions) [SET00] is a set of protocols and formats that enables users to employ the existing credit card payment infrastructure on the Internet, in a secure way. It provides a secure communication channel among all parties involved in a transaction, provides trust by the use of X509 certificates, ensures privacy because the information is only available to parties in a transaction when and where necessary. Secure Electronic Transaction (SET) was developed by Visa, MasterCard and many other companies such as Microsoft and IBM.

3.3.4.2 Actors of SET

- Cardholder: authorized holder of a payment card (e.g. MasterCard, Visa) issued by an issuer.
- Acquirer: financial institution that establishes an account with a merchant and processes payment card authorizations. It provides authorization to the merchant that a given transaction is authorized and provides transfer of payments to the merchant account.
- Merchant: seller of goods, services or information. A merchant must have a relationship with an acquirer.
- Issuer: a financial institution such as a bank that establish an account and a card for the cardholder. It guarantees payment for authorized transactions.
- Payment gateway: It interfaces between SET and the existing bankcard payment networks for authorization and payment functions. The merchant exchanges SET messages with the payment gateway over the Internet, while the payment gateway has some direct or network connection to the acquirer's financial processing system.
- Certification authority (CA): entity that is trusted to issue X509v3 public-key certificates for cardholders, merchants and payment gateways. SET relies strongly on the existence of a reliable CA infrastructure. A hierarchy of CAs is used, such that participants need not be directly certified by a root authority.

3.3.4.3 Dual signature

SET includes an important innovation: the dual signature (DS). The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, the customer wants

to send the OI to the merchant and the PI to the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order. The customer is protected in term of privacy by keeping the OI and the PI separate. However, the two items must be linked in a way that can be used to resolve disputes if necessary. During the transaction, the merchant receives OI and H(PI). The bank receives PI and H(OI). The customer has linked OI and PI by generating DS:

$$DS = E_{KR(C)}[H(H(PI), H(OI))]$$

Both the merchant and the bank can check the dual signature so that OI and PI are linked for every actor. OI and PI contain *transID*, a transaction ID number.

3.3.4.4 Payment processing

First a customer opens an account and get a credit card and a X509v3 digital certificate signed by the bank. It establishes a link between the credit card and the customer's key pair. Merchants have two certificates for two public keys: one for signing messages, one for key exchange. They also have the payment gateway's certificate.

When a customer selects a list of items to be purchased (e.g. on a merchant web site), the merchant sends back an order form with an order number and its characteristics with the merchant certificate. Then, the customer sends both order information (OI) and payment information (PI) to the merchant with the customer certificate. The payment information contains credit card information and is encrypted in such a way that it cannot be read by the merchant. The merchant sends the payment information to the payment gateway, requesting payment authorization and it sends confirmation of the order to the customer. If authorization is OK, the merchant provides goods or service and send the payment request to the payment gateway.

3.3.4.6 Conclusion on SET

SET relies strongly on PKI: every actor has a digital certificate and a CA trust chain is used. The root CA is owned and maintained by SET Corporation. It makes use of a lot of digital envelopes and digital signatures. RSA, DES, SHA-1 and X509 are standards used for encryption, digest and

certification. The most important data in SET can never be disguised or modified nor disclosed to the intruder. Main threats of SET are masquerade, *DoS*, service interruption and modification of information.

3.3 Main attacks against computer systems

Physical security violations (physical access to computer or device) can highly compromise plaintext data, written or spoken information. That security threat is often underestimated. It can be far more easy and cost-saving for hackers to break into computer devices physically than to do it through the network or with a massive cryptanalytic attack. The physical security will be tackled by studying the following technologies: video cameras, smart cards, and biometry. In this Section we deal only with electronic attacks.

3.3.1 Attacks overview

Classically, attacks were neither named nor classified; there was just: "here is a computer system, and here is the attack." And while this gradually developed into named attacks, there is no overall attack taxonomy. Currently, attacks are sometimes classified by the information available to the attacker or constraints on the attack, and then by strategies which use the available information. Here are five general categories of attacks classified in function of the service they attack:

- Interruption: an asset of the system is destroyed or becomes unavailable. This is an attack on *availability*.
- Interception: an unauthorized party gains access to an asset. This is an attack on *confidentiality*.
- Modification: an unauthorized party gains access and tampers with an asset. This is an attack on *integrity*.

- Fabrication: An unauthorized party inserts counterfeit objects into the system. This is an attack on *authenticity*.

- Decryption: An unauthorized party decrypts a ciphertext by breaking a cipher. This is an attack on *confidentiality*.

We give below a quick review of attacks on the different parts of a security system.

3.3.2 Attacks on ciphers and hash functions

The goal of an attack is to reveal some unknown plaintext, or the key (which will reveal the plaintext). An attack, which succeeds with less effort than a brute-force search is called a break. An "academic" ("theoretical") break may involve impractically large amounts of data or resources, yet still be called a "break" if the attack would be easier than brute force. (It is thus possible for a "broken" cipher to be much stronger than a cipher with a short key.) Sometimes the attack strategy is thought to be obvious, given a particular informational constraint, and is not further classified.

Ciphers and also cryptographic hash functions can be attacked, generally with very different strategies:

- Brute Force (also Exhaustive Key Search): Try to decipher ciphertext under every possible key until readable messages are produced. (Also "brute force" any searchable-size part of a cipher.)

- Codebook (the classic "codebreaking" approach): Collect a codebook of transformations between plaintext and ciphertext.

- Differential cryptanalysis: Find a statistical correlation between key values and cipher transformations (typically the Exclusive-OR of text pairs), then use sufficient defined plaintext to develop the key.

- Linear cryptanalysis: find a linear approximation to the keyed S-boxes in a cipher (if any), and use that to reveal the key.

- Key schedule: choose keys, which produce known effects in different rounds.

- Birthday attack (usually a hash attack) [Stallings99]: The birthday paradox part is a form of attack in which it is necessary to obtain two identical values from a large population. The idea is that it is much easier to find two values, which match, than it is to find a match to some particular value. The "birthday" part is the realization that it is far easier to find an arbitrary matching pair than to match any particular hash-value.

- Formal Coding (also Algebraic): From the cipher design, develop equations for the key in terms of known plaintext, then solve those equations.

- Correlation: In a stream cipher, distinguish between data and confusion, or between different confusion streams, from a statistical imbalance in a combiner.

- Dictionary: form a list of the most-likely keys, then try those keys one-by-one (a way to improve brute force).

- Meet-in-the-Middle: Given a two-level multiple encryption, search for the keys by collecting every possible result for enciphering a known plaintext under the first cipher, and deciphering the known ciphertext under the second cipher; then find the match.

- Time attack: Given the time necessary for the cipher to encrypt or decrypt data of a certain length, try to get information on the key.

3.3.3 Attacks on protocols and operating systems

Systems such as SMTP, DNS, SNMP, X-window, NFS, HTTP or FTP are vulnerable to attacks. It is mainly due to the fact that security was not the principal concern of those who have designed the first Internet protocols. Attackers can make use of flaws of lower layer protocols such as IP, ICMP, TCP and UDP: “Ping of death”, land, TCP SYN flooding, teardrop, FTP bounce, TCP sequence number prediction, TCP connection killing, TCP hijacking and spoofing (usually of the source address) with ARP spoofing and IP spoofing. Let us explain one example of attack using a flaw of TCP. The TCP SYN flooding attack consists of sending a high number of TCP connections

requests (TCP-SYN) in order to saturate the target. The latter will not have resources any more to accept valid connection requests from valid users.

Protocols flaws bring up attacks on service availability and authentication:

- The *Denial of Service* (Dos) attack attempts to cause a failure in (especially, in the security of) a computer system or other data processing entity by providing more input than the entity can process properly. It prevents authorized access to a system resource. These kinds of flooding can be used for example through email, DNS requests, UDP packets or TCP connections opening. All systems are potentially vulnerable against *DoS* attacks.

- Spoofing allows to impersonate somebody else. The most common spoofing attacks include email spoofing, DNS spoofing and web spoofing.

The internal design of operating systems can be used to attack a machine: buffer overflow, system resources saturation (cf. DoS attacks), password stealing, access without authorization, virus through network. Operating systems flaws are published on the Internet. These flaws are very serious for a system since they can be used to attack the service availability, authentication or to directly access to sensitive information.

3.3.4 Attacks on information in the network

3.3.4.1 Passive attacks

Passive attacks are difficult to detect because they do not involve any alteration of the data. Preventive measures are available to avoid their success. Passive attacks are done mainly by listening in the network by sniffing, eavesdropping, or using "ICMP redirect" for information or password robbery. The two main passive attacks are: release of message contents (learning the content of transmissions) and traffic analysis that is a more subtle attack. Indeed even if an opponent (say Eve) cannot extract information from a message, she can observe the pattern of these messages. She

could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged.

3.3.4.2 Active attacks

Active attacks (including the man-in-the-middle attack) involve some modification of the data stream or the creation of a false stream.

- Replay: It involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized or useful effect.
- Masquerade: It takes place when an entity pretends to be a different entity.
- Denial of Service: It prevents or inhibits the normal use or management of communication facilities (see Section above).
- Message alteration: Modification of data for masquerading and integrity violation of data.

The Diffie-Hellman key exchange (Section 3.2.1) is vulnerable to a man-in-the-middle attack. In this attack, Eve intercepts Alice's public parameter and sends her own public parameter to Bob. When Alice transmits her public parameter, it is intercepted by Eve, who repeats the substitution of parameter with Bob. So the man (or woman) in the middle (Eve) has set herself up as the bridge between the other two. Any message from Bob to Alice is intercepted by Eve, decrypted using the keys generated with Bob, and then encrypted using the keys generated by Alice. This vulnerability relies on the middle man's ability to reliably intercept the message traffic, and is present because the Diffie-Hellman key exchange does not use authentication. An implementation in conjunction with digital signatures (called the authenticated Diffie-Hellman key exchange) could correct this issue, since users could be authenticated.

3.3.5 Sophisticated listening attacks

3.3.5.1 TEMPEST attacks

Another kind of attack that has been used by well-equipped opponents involves the remote detection of the electromagnetic signals from your computer. This expensive and somewhat labor-intensive attack is probably still cheaper than direct cryptanalytic attacks. An appropriately instrumented van can park near Alice's office and remotely pick up all of her keystrokes and messages displayed on her computer video screen. In some cases it may even detect data as it moves through her computer's bus or I/O cables. This could compromise some of her passwords, messages, or multimedia streams such as digitized voice, etc. This attack can be thwarted by properly shielding all of a computer equipment and network cabling so that it does not emit these signals. This shielding technology is known as TEMPEST (Transient ElectroMagnetic Pulse Emanation Standard), and is used by some government agencies and defense contractors. There are hardware vendors who supply TEMPEST shielding commercially, although it may be subject to some kind of government licensing.

3.3.5.2 Specific sophisticated attacks

Many attacks are specific to a system and are very difficult to classify. Let us just give an example on voice interception since it is our main preoccupation. Some listening devices and other remote sensing can be used to remotely access to a conversation. Obviously no software can protect from someone planting a microphone in the room where Alice is speaking on an encrypted phone call. Acoustic information may leak out of Alice's room by other means, too. For example, from hundreds of feet away, someone may bounce a laser beam off her window, and watch the beam's reflection wiggle as Alice's voice makes the window vibrate. They can use the reflected beam's wiggle to reconstruct the sound. If they use an infrared laser, Alice won't be able to see the beam on her window. This kind of attack is cheaper than cryptanalytic attacks on ciphers.

3.5 Conclusion

To conclude, this part reflects an essential observation: it is impossible to secure a network completely. There is always a hacker who will manage to enter into the system. On the other hand, there are many solutions to minimize damages. The one solution, which is often forgotten, is prevention. System administrators must train users because security relies at the end on people more than on devices or software. Internal security is very important. Half of the attacks on a given system are from system users or former users. By establishing a security policy, a company can evaluate needs about security, and often evaluate financial needs to develop a security level which matches with its policy. Security, more and more, takes an important place on Internet. Bringing security over a computer network isn't something easy to determine but it certainly needs means and important investment (money and time).

Chapter 4 Telephony applications and their security features

In this Chapter, we first study GSM (Global System for Mobile communications) and its security mechanism. GSM is the main second-generation wireless telephony system that provides terminal mobility. It is based on a fixed signaling architecture. We then present two existing solutions that provide secure telephony on the Internet and we underline their limitations. PGPfone is a piece of software that enables to make secure phone calls. SIP (Session Initiation Protocol) is a signaling protocol still under development at the IETF and RTP (Real Time Protocol) is a protocol for data streaming. The combined use of the two latter protocols (SIP and RTP) appears to be a promising solution to provide IP-telephony. We show that this solution currently provides limited security features. SIP relies on other security protocols to provide full security and this could reduce its usefulness in the scenario of a mobile user. In the next chapter, we study user mobility more in detail.

4.1 GSM architecture

4.1.1 GSM architecture overview

In the 80's a group of European experts technically defined a digital cellular network capable of accepting several millions of users called Global System for Mobile communication (GSM) . The specifications and technical standards are public and managed by the European Telecom Standard Institute (ETSI). GSM is the leading worldwide standard for digital mobile phones: more than 225 million GSM subscribers around the globe in 133 countries [UWCA97].

In a cellular system, a cellular phone user must be located in an area where the network can route an incoming call to his terminal. Users are located by a grouping of geographical areas (cells). If an incoming call cannot go through, a pager message is sent into the geographical zone where the last

communication took place. User mobility can involve a change of cells. An automatic transfer in this way, from one cell to another, without cut-off in the connection, is called "handover".

The services supplied by GSM can be grouped into voice services in which the information is supplied by voice (digitized speech) and data services for the transmission of text, images, fax documents, files or messages via the GSM network. The voice services are the same as those available on fixed telephone networks. It is possible to communicate through transmission networks ranging from 300 to 9600 Bit/s. The Short Message Service (SMS) makes it possible to send messages of up to 160 characters. These messages can be read on a portable phone or on the screen of a PC using the SMS management software. Additional services enable users to choose, for instance, how incoming and outgoing calls will be processed by the network (call forwarding, call filtering,…). The only GSM sub-system that users need to know is their own mobile terminals, in theory, a portable telephone. For data services, users may wish to use a PC or notebook with a data card.

The main parts of the terminal are the mobile transmission equipment and the customer identification module. Schematically, the terminal can be divided into three separate parts:

- The terminal inputs/outputs module, which is the link with the user (microphone, speaker, and keyboard) and a peripheral (PCMCIA card, loader, etc.).
- The radio modem (modulator/demodulator) is handling the analog-to-digital and channel-specific conversion. The radio frequency module receives and transmits signals over the cellular network through the radio interface.
- The customer identification module: the module is in the form of a chip card called a SIM (Subscriber Identity Module) card. It contains all the information relative to the user. It is used as a memory for messages and telephone numbers.

4.1.2 GSM architecture

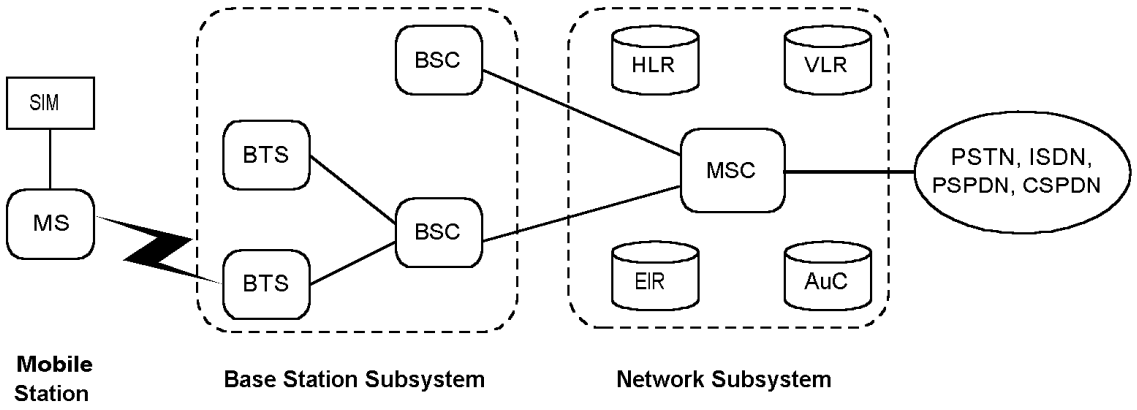

Figure 4A - Schematic presentation of the GSM network

4.1.2.1 The mobile station

The mobile station (MS) consists of the mobile equipment (the terminal) and a smart card called the Subscriber Identity Module (SIM). The SIM provides personal mobility, so that the user can have access to subscribed services irrespective of a specific terminal. By inserting the SIM card into another GSM terminal, the user is able to receive calls at that terminal, make calls from that terminal, and receive other subscribed services. The mobile equipment is uniquely identified by the International Mobile Equipment Identity (IMEI). The SIM card contains the International Mobile Subscriber Identity (IMSI) used to identify the subscriber to the system, a secret key for authentication, and other information. The IMEI and the IMSI are independent, thereby allowing personal mobility. The SIM card may be protected against unauthorized use by a password or personal identity number (PIN). SIM has an EEPROM and ROM. ROM contains some algorithms necessary for encryption. EEPROM contains the International Mobile Subscriber Identity (IMSI) and Individual Subscriber Authentication Key (K_i). The International Mobile Subscriber Identity (IMSI) uniquely identifies the subscriber. Rather than sending IMSI, the Temporary Mobile Subscriber Identity (TMSI) is sent in most instances. This prevents the intruder from gaining

information on the resources the user is using, tracing the location of the user, and matching the user and the transmitted signal. IMSI is sent only when necessary, for example when the SIM is used for the first time when there is a data loss at the VLR (Visitor Location Registar). When the SIM is used for the first time MS reads the default TMSI stored in the card MS and sends the default TMSI to the VLR. Since the VLR does not know this TMSI, it requests the IMSI from the MS. The latter sends the IMSI to the VLR. The latter then assigns a new TMSI to this user.

4.1.2.2 The other equipment

The Base Station Subsystem is composed of two parts, the Base Transceiver Station (BTS) and the Base Station Controller (BSC). The central component of the Network Subsystem is the Mobile services Switching Center (MSC). It acts like a normal switching node of the PSTN or ISDN, and additionally provides all the functionality needed to handle a mobile subscriber, such as registration, authentication, location updating, handovers, and call routing to a roaming subscriber.

The Home Location Register (HLR) and the Visitor Location Registar (VLR), together with the MSC, provide the call-routing and roaming capabilities of GSM. The HLR contains all the administrative information of each subscriber registered in the corresponding GSM network, along with the current location of the mobile. The location of the mobile is typically in the form of the signaling address of the VLR associated with the mobile station. The actual routing procedure will be described later. There is logically one HLR per GSM network, although it may be implemented as a distributed database.

The VLR contains selected administrative information from the HLR, necessary for call control and provision of the subscribed services, for each mobile currently located in the geographical area controlled by the VLR. Although each functional entity can be implemented as an independent unit, all manufacturers of switching equipment to date implement the VLR together with the MSC, so that the geographical area controlled by the MSC corresponds to that controlled by the VLR, thus simplifying the signaling required. Note that the MSC contains no information about particular mobile stations. This information is stored in the location registers.

The other two registers are used for authentication and security purposes. The Equipment Identity Register (EIR) is a database that contains a list of all valid mobile equipment on the network, where each mobile station is identified by its IMEI. An IMEI is marked as invalid if it has been reported stolen or is not type approved. The Authentication Center (AuC) is a protected database that stores a copy of the secret key stored in each subscriber's SIM card, which is used for authentication and encryption over the radio channel.

4.1.3 GSM Security features

We now focus on the security features of the GSM network rather than radio-link aspects, the detailed architecture, speech coding or channel coding. Authentication and security is fully described in [GSMSEC]. The security mechanisms are only defined for the air interface. The fixed network is supposed to be secure and its security is left to network providers.

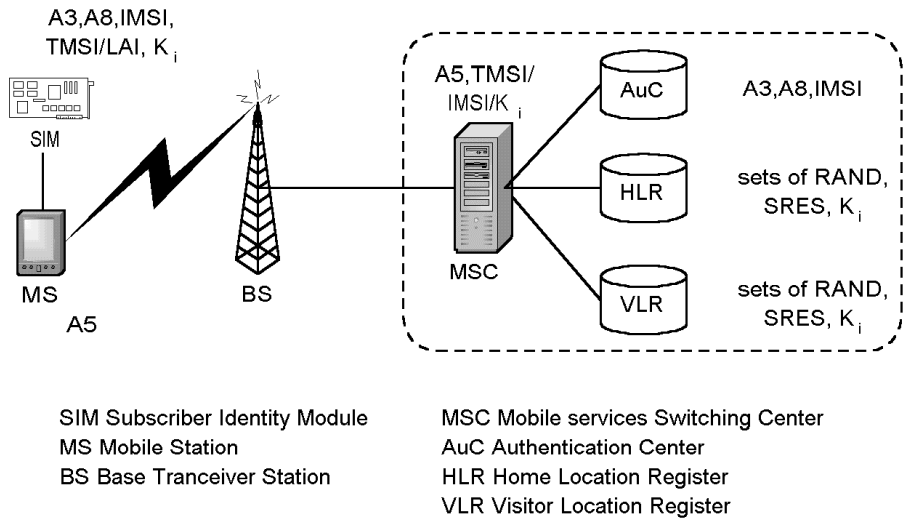

Figure 4B – Security architecture in GSM

Since the radio medium can be accessed by anyone, authentication of users, is a very important element of a mobile network. Authentication involves two functional entities, the SIM card in the mobile, and the Authentication Center (AuC). Each subscriber is given a secret key, one copy of

which is stored in the SIM card and the other in the AuC. During authentication, the AuC generates a random number that it sends to the mobile. Both the mobile and the AuC then use the random number, in conjunction with the subscriber's secret key and a ciphering algorithm called A3, to generate a signed response (SRES) that is sent back to the AuC. If the number sent by the mobile is the same as the one calculated by the AuC, the subscriber is authenticated [MoPa92] The same initial random number and subscriber key are also used to compute the ciphering key using an algorithm called A8. This ciphering key, together with the TDMA frame number, use the A5 algorithm to create a 114 bit sequence that is "XORed" with the 114 bits of a data or voice blocks (pseudo one-time pad technique). The signal is coded, interleaved, and transmitted in a TDMA manner, thus providing protection from all but the most persistent and dedicated eavesdroppers. Several A5 algorithms can be used in different countries. The algorithms A3 and A8 are associated with a given operator and are secret. Most of them use COMP-128.

The message exchanges are illustrated in Figure 4C.

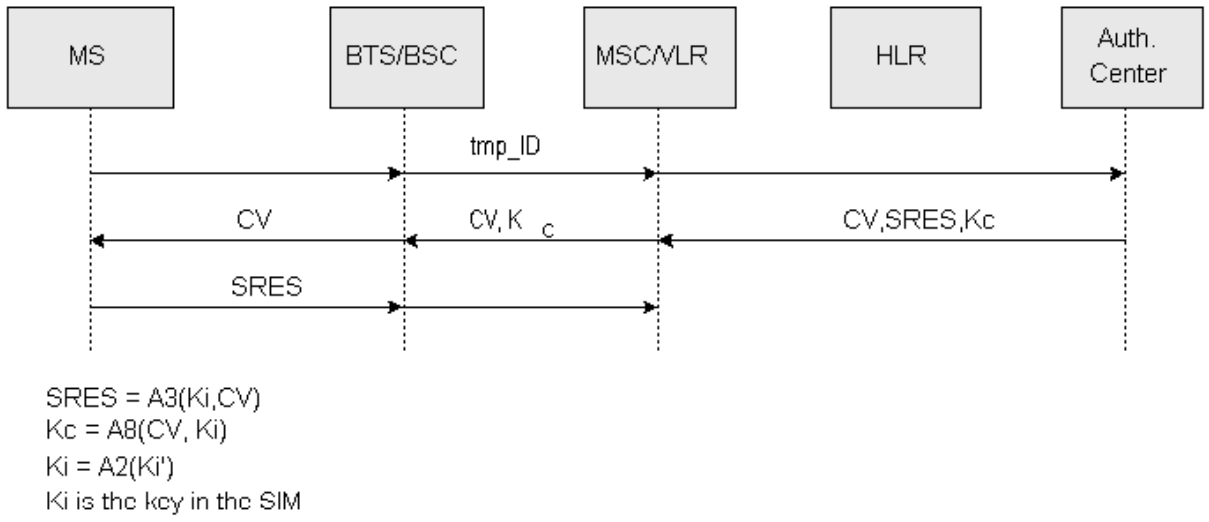

Figure 4C: authentication message exchange in GSM.

Users and providers rely on the GSM security and they have to trust the algorithms A3/A8. Note that there is no end-to-end security (user-to-user voice encryption).

Another level of security is introduced by on the mobile equipment itself, as opposed to the mobile subscriber. As mentioned earlier, each GSM terminal is identified by a unique International Mobile Equipment Identity (IMEI) number. A list of IMEIs in the network is stored in the Equipment Identity Register (EIR). The status returned in response to an IMEI query to the EIR is one of the following:

- White-listed: The terminal is allowed to connect to the network.
- Grey-listed: The terminal is under observation from the network for possible problems.
- Black-listed: The terminal has either been reported stolen, or is not type approved (the correct type of terminal for a GSM network). The terminal is not allowed to connect to the network.

The confidentiality of a call and anonymity of the GSM subscriber is only guaranteed on the radio channel and users need another major security step in achieving end-to-end security. The subscriber's anonymity is ensured through the use of temporary identification numbers. The confidentiality of the communication itself on the radio link is performed by the application of encryption algorithms and frequency hopping.

4.2 PGPfone

4.2.1 PGPfone overview

PGPfone (Pretty Good Privacy Phone) is a software package that turns a desktop or notebook computer into a secure telephone. It uses speech compression and strong cryptography protocols to give the ability to have a real-time secure telephone conversation. Secure voice calls are supported over the Internet, or through a direct modem-to-modem connection, or even over AppleTalk networks. PGPfone uses its own packet format (Type=1byte, Sequence=1byte, Data<1024, CRC=4bytes). In the next version, the packet format will be close to the Real Time Protocol (RTP) format [RFC 1889].

4.2.2 PGPfone encryption

When a call is first established, packets are sent across unencrypted while negotiation of the encryption parameters proceeds. However, as early as possible, the stream is encrypted to deny eavesdroppers information about the call. The data in the PGPfone packets is encrypted in counter mode (CTR-mode) using one of the three ciphers: TripleDES, CAST, or Blowfish. In this mode, a counter is encrypted and decrypted rather than the original data. The tranformed counter is then "XORed" with the data to transmit. Counter mode offers several advantages for this application. Because it does not actually encrypt or decrypt any of the original data, the counter encryption and decryption can be pre-computed for each packet, requiring only an XOR once the data becomes available. This allows us to further reduce latency in future implementations by pre-computing the encrypted keystream during inter-packet gaps before the voice data becomes available for encryption or decryption. Like CFB (Cipher Feedback) mode, counter mode can encrypt any byte length of data without padding.

4.2.3 User key exchange

PGPfone [PGPfone] uses the Diffie-Hellman key exchange (Section 3.2.1). The Diffie-Hellman key agreement protocol requires that both parties agree upon a generator and a prime. PGPfone uses a fixed generator of 2, which has speed advantages. The prime however is agreed upon somewhat interestingly. PGPfone assumes that each party generates all the primes that it will use by a common algorithm, beforehand. This way, each party can know that the primes aren't "weak" ones. The main reason for having multiple primes is to provide some room for security/speed tradeoffs.

The DH key exchange is sensitive to the man-in-the-middle attack (Section 3.3.4). Authenticated Diffie-Hellman key exchange solves this problem by using digital signature to authenticate the generator and the prime. PGPfone does not suppose that both parties have a digital certificate. It uses another mechanism to secure the public information exchange against the man-in-the-middle attack.

Alice and Bob could authenticate the key exchange by reading some binary key material to each other as hexadecimal bytes. This is awkward for non-technical humans and prone to errors. So PGPfone encodes each byte as one of 256 phonetically distinct English words, kind of like the military alphabet that pilots use to read letters over a noisy radio channel ("tango delta foxtrot"). PGPfone uses a bigger list of words, 256 instead of 26. We call this a "voice signature", because the human vocal tract is used as a way of authenticating or "signing" the information. It helps if Alice and Bob recognize each other's voices.

Figure 4D: PGPfone "voice signature" mechanism.

Actually, even if Alice has never spoken to Bob before, this voice signature scheme can still work, as long as the voice pronouncing the authentication words clearly matches the voice in the rest of the conversation.

One possible attack is to record Alice's voice saying the words. If Eve gets most of the 256 words, the "voice signature" mechanism can be bypassed by replaying the words that Bob expects to hear in the validation phase.

4.3 Session Initiation Protocol

4.3.1 SIP overview

The Session Initiation Protocol (SIP) [RFC2543] is an application-layer control protocol for creating and terminating sessions such as Internet telephone calls. SIP supports user location, user

capabilities, user availability, call setup and call handling. A SIP client or user agent client (UAC) sends SIP requests. A successful SIP invitation consists of an INVITE request and an ACK (acknowledgement) request. The client can either send the invitation to a local SIP proxy server, or send it to the IP address and port of the Request-URI (Uniform Resource Identifier) using TCP or UDP (default port is 5060). The client can find a SIP server by querying a DNS (Domain Name Server) [RFC2782]. Users may call a SIP server *sip.domainname*. The SIP server of the callee (Bob) can act as a proxy server (forwards call INVITE to Bob) or as a SIP redirect server (sends back the user location to the caller). If a user wants to change the parameters of an existing session, an INVITE request can be issued with the same callID but a higher CSeq. Users are identified by sip Uniform Resource Locations or URLs (sip:alice@registrar.com;transport=tcp). A SIP-message is either a request from a client to a server, or a response from a server to a client.

Response message codes are as follows:

- 1xx: Informational (e.g. call in progress)
- 2xx: Success (OK)
- 3xx: redirection
- 4xx: client failure
- 5xx: server failure
- 6xx: global failure

The callee can accept, redirect or reject the call. In any case, the callee copies the *To*, *From*, *Call-ID*, *Cseq* and *Via* fields from the request into their response. The callee may add a *Contact* header field in the response. When the caller receives the response to the initial request, it sends an ACK request to acknowledge or a BYE request to terminate the call.

All Requests and responses may contain message bodies. For responses, bodies contain information about the status or session description (for 2xx responses). The *Content-Type* header field must give the MIME (Multipurpose Internet Mail Extensions) type of the message body. A *Content-Encoding* field must indicate the encoding type if any. A typical content type is "application/sdp".

4.3.2 SIP architecture

A typical SIP call INVITE is described in Figure 4E below.

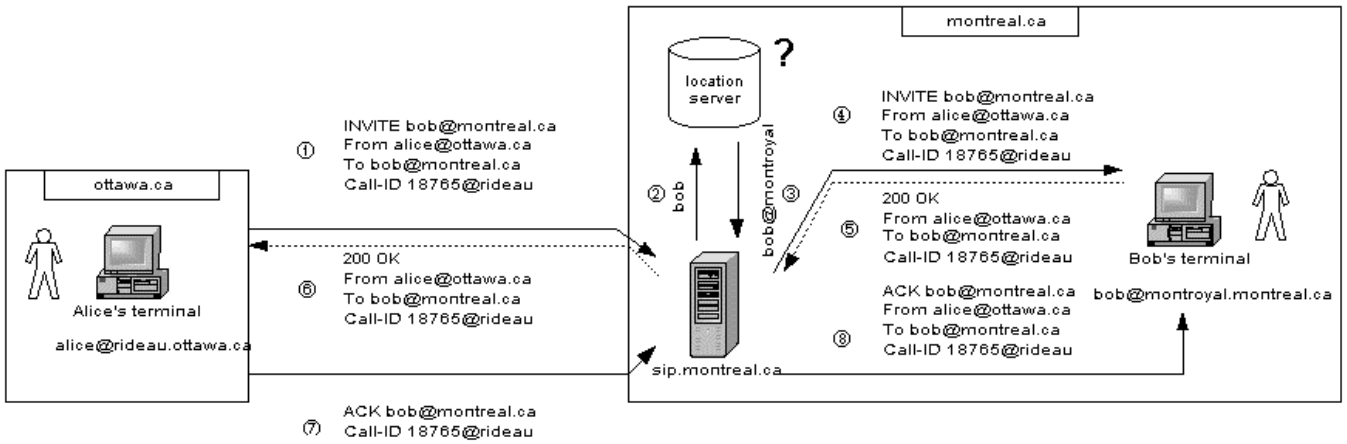

Figure 4E: SIP message exchanges.

Alice calls Bob from her machine *rideau.ottawa.ca*. Alice knows that Bob is registered in Montreal. She first sends an INVITE message to Montreal's SIP server (message 1). The latter server looks up in a database to find out Bob's current location (message 2). Bob is on *montroyal.montreal.ca* (message 3). Montreal's SIP server (*sip.montreal.ca*) forwards Alice's INVITE to Bob (message 4). Messages 5 and 6 are *OK* messages that send back a positive reply to Alice's INVITE. Messages 7 and 8 are Alice's acknowledgment to Bob's answer.

4.3.3 SIP message examples

In this section, we present a complete sequence of SIP messages for a two-party phone call request when a party is outside their home domain. Assume Alice registered in Ottawa as alice@ottawa.ca calls Bob registered in Montreal as bob@montreal.ca. Alice is currently in Paris and Bob is currently in Berlin. Four SIP servers are involved in this scenario: *sip.ottawa.ca*, *sip.montreal.ca*, *sip.paris.fr* and *sip.berlin.de*. Assume Alice is logged on *eiffel.paris.fr* and Bob is logged on *brandenburg.berlin.de* (see Figure 4F). We assume that Ottawa and Montreal SIP servers act as SIP proxy servers.

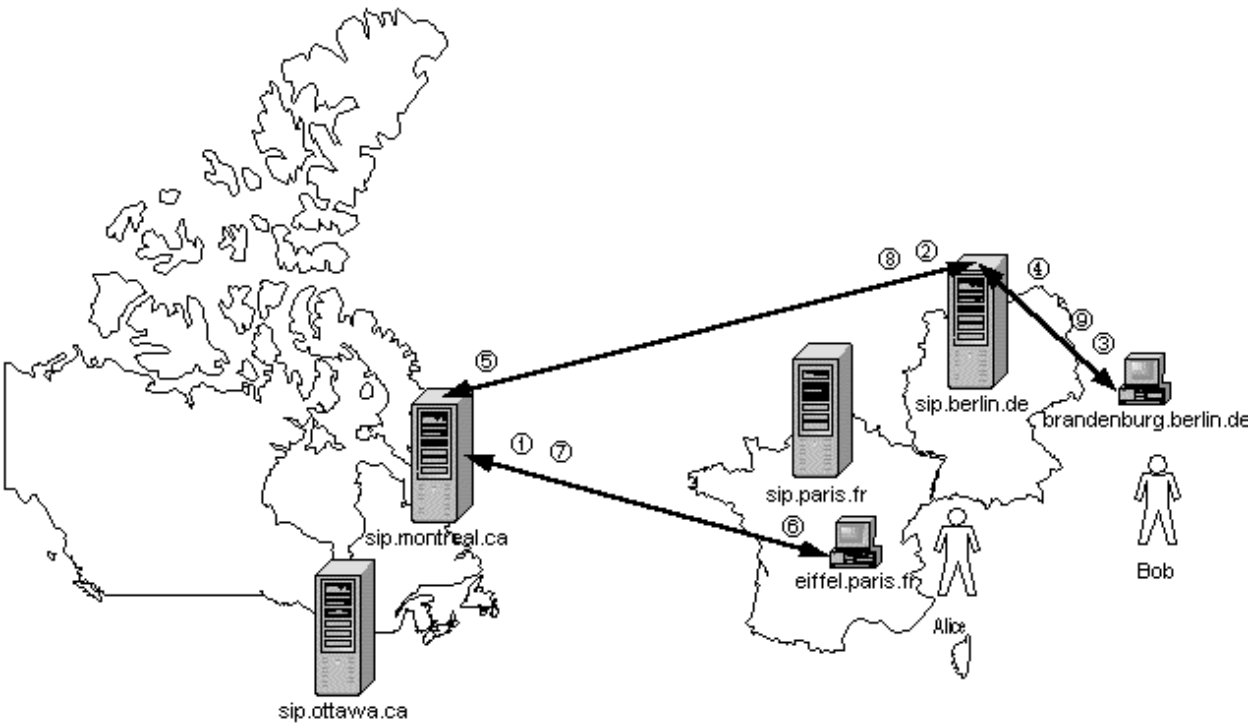

Figure 4F: SIP scenario.

All SIP messages are text-based and requests are one of the following types: INVITE, BYE, OPTIONS, STATUS, ACK, CANCEL and REGISTER. End of SIP headers is indicated by two "CRLF" (carriage return line feed noted '$').
We now give the complete sequence of messages. Only SIP headers are indicated here.

First Bob needs to register (e.g. on startup via multicast) to Berlin's SIP server (*sip.berlin.de*) as bob-berlin@berlin.de (his local name) and to his home SIP server as bob@montreal.ca. Thus the SIP client on *brandenburg.berlin.de* sends a REGISTER message to the local SIP server *sip.berlin.de* and to his home SIP server *sip.montreal.ca*.

REGISTER message to *sip.berlin.de*:

```
REGISTER sip:berlin.de SIP/2.0
Via: SIP/2.0/UDP brandenburg.berlin.de
From: bob-berlin@berlin.de //Bob does the registration
To: bob-berlin@berlin.de //Bob is known with this name
Call-ID: 7894635@brandenburg.berlin.de
Cseq: 1 REGISTER
```

```
Contact: sip:bob-berlin@brandenburg.berlin.de
Expires: 7200  // in 2 hours
```

REGISTER message to *sip.montreal.de*:

```
REGISTER sip:montreal.ca SIP/2.0
Via: SIP/2.0/UDP brandenburg.berlin.de
From: bob-berlin@berlin.de //Bob does the registration
To: bob@montreal.ca //Bob is known with this name
Call-ID: 7894635@brandenburg.berlin.de
Cseq: 1 REGISTER
Contact: sip:bob-berlin@berlin.de
Expires: 7200  // in 2 hours
```

Note that a Montreal administration agent (say Bob's colleague) could send a REGISTER message to Montreal's SIP server (*sip.berlin.fr*) on behalf of Bob:

```
REGISTER sip:sip.montreal.ca SIP/2.0
Via: SIP/2.0/UDP homeagent.montreal.ca
From: admin-server@montreal.ca
To: bob@montreal.ca //Bob is known with this name
Call-ID: 5834686@admin.ottawa.ca
Cseq: 1 REGISTER
Contact: sip:bob-berlin@berlin.de
Expires: 7200 //in 2 hours
```

Now Alice calls Bob knowing that Bob is registered in Montreal. Alice's SIP client on *eiffel.paris.fr* sends an INVITE message to *sip.montreal.ca*:

```
INVITE sip: bob@montreal.ca SIP/2.0$ //can be rewritten by proxies
Via: SIP/2.0/UDP sip.montreal.ca$
Via: SIP/2.0/UDP eiffel.paris.fr
From: Alice<sip:alice-paris@paris.fr>$ //caller
To: Bob<sip:bob@montreal.ca>$ //callee
Call-ID: 187602141351@eiffel.paris.fr
Cseq: 1 INVITE$
Contact: alice@eiffel.paris.fr //contact address
Subject: business
Content-Type: application/sdp
Content-Length: 102$
```

Montreal's SIP server (*sip.montreal.ca)* looks up for Bob's location and as a proxy server it forwards the invitation to Berlin's SIP server (*sip.berlin.de*):

```
INVITE sip: bobnameinberlin@berlin.de SIP/2.0$ //can be rewritten by proxies
Via: SIP/2.0/UDP sip.montreal.ca$
Via: SIP/2.0/UDP eiffel.paris.fr
From: Alice<sip:alice-paris@paris.fr>$ //caller
To: Bob<sip:bob@montreal.ca>$ //callee
Call-ID: 187602141351@eiffel.paris.fr
Cseq: 1 INVITE$
Contact: alice@eiffel.paris.fr //contact address
Subject: business
Content-Type: application/sdp
Content-Length: 102$
```

Berlin's SIP server (*sip.berlin.de*) looks up Bob's location and as a proxy server it forwards the INVITE message to *brandenburg.berlin.de*:

```
INVITE sip: bobnameinberlin@berlin.de SIP/2.0$ //can be rewritten by proxies
Via: SIP/2.0/UDP sip.montreal.ca$
Via: SIP/2.0/UDP eiffel.paris.fr
From: Alice<sip:alice-paris@paris.fr>$ //caller
To: Bob<sip:bob@montreal.ca>$ //callee
Call-ID: 187602141351@eiffel.paris.fr
Cseq: 1 INVITE$
Contact: alice@eiffel.paris.fr //contact address
Subject: business
Content-Type: application/sdp
Content-Length: 102$
```

If Bob accepts the call, Bob's user agent (at *brandenburg.berlin.de*) sends back an *OK* message to *sip.berlin.de:*

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP eiffel.paris.fr$
Via: SIP/2.0/UDP sip.montreal.ca$
From: Alice<sip:alice-paris@paris.fr>$ //caller
To: Bob<sip:bob@montreal.ca>$ //callee
Call-ID: 187602141351@eiffel.paris.fr
Cseq: 1 INVITE$
Contact: bob-berlin@brandenburg.berlin.de   //contact address
Subject: business
Content-Type: application/sdp
Content-Length: 102$
```

The latter server sends back an *OK* message to Montreal's SIP server: Finally sip.montreal.ca sends back an *OK* message to *eiffel.paris.fr*. Then Alice sends back an *ACK* message to Montreal's SIP server:

```
SIP/2.0 200 ACK
Via: SIP/2.0/UDP eiffel.paris.fr$
From: Alice<sip:alice-paris@paris.fr>$ //caller
To: Bob<sip:bob@montreal.ca>$ //callee
Call-ID: 187602141351@eiffel.paris.fr
Cseq: 1 ACK$
```

Montreal's SIP server forwards that message to *sip.berlin.de*. The latter server finally forwards that *ACK* message to *brandenburg.berlin.de*.

4.3.4 Security in SIP

4.3.4.1 Authentication in SIP

Authentication can be provided in three ways. First, transport-layer or network-layer authentication (IPsec-AH) may be used for hop-by-hop authentication. This may be defined out of band. SIP also extends the WWW-Authenticate and Authorization and their Proxy counterparts to include strong signature. Finally, SIP also supports the HTTP "basic" and "digest" schemes. HTTP basic scheme works the same way as PAP (Section 3.1.4): the password is transmitted in clear. HTTP digest scheme works the same way as CHAP: the password in transmitted in a challenge response including a digest.

A client, proxies and called user server agents may strongly sign their messages using the "Authorization" header field. When an "Authorization" header field is present, it indicates that all header fields following the Authorization header field, have been included in the signature that is the value of this field. Signature is always done after encryption if any. The "Authorization" field must remain in the clear if it contains a digital signature but may be encrypted if it contains "digest" authentication. A client may require that a server sign its response by including a "Require: signed-response" request header field and a "WWW-Authenticate" header that indicates the desired authentication method. Server message signatures are to prevent an eavesdropper from injecting unauthenticated responses that terminate, redirect or interfere with a call. However, typically, 4xx and 5xx responses will not be signed by called user agent. The best way to prevent this problem is to use hop-by-hop (or link) encryption of the SIP request.

Only UAC's can authenticate themselves to proxies in the "Proxy-Authorization" header field. The framework for SIP authentication parallels that for HTTP [RFC2617]. This provides authentication without message integrity. "Proxy-Authenticate" response-header value indicates a challenge and the authentication scheme and parameters applicable to the proxy for a Request-URI. "WWW-

Authenticate" response-header field must be included in 401 (unauthorized) response messages. The field value consists of at least one challenge that indicates the authentication scheme and parameters. Basic authentication asks for username and password that go in the clear on the network. Digest authentication response sends a nonce and asks for the digest including the nonce and the password.

4.3.4.2 Privacy in SIP

Route privacy can be provided in SIP. The "Hide" request header indicates that the client wants the path comprised of the "Via" header fields to be hidden form subsequent proxies and user agents. The client can ask to hide the route or only the next hop. This provides hop-by-hop encryption of "Via" fields to hide the route a request has taken. The algorithm chosen is entirely up to the proxy implementor.

Hop-by-hop encryption encrypts the entire SIP request or response on the wire so eavesdroppers cannot see who is calling whom. Proxies can still see who is calling whom. IPsec or TLS may protect SIP messages. "The use of a particular mechanism will generally to be specified out of band." [RFC2543].

The "Encryption" header field is intended for end-to-end encryption of requests and responses. This can also serve to authenticate the originator of the message. This mechanism relies on keys shared by the two user agents involved in the request. Requests are encrypted based on the public key belonging to the entity named in the "To" header field. Responses are encrypted based on the public key conveyed in the "Response-Key" header field. For any encrypted message, at least the message body and possibly other message header fields are encrypted. Encryption provides only privacy. The recipient has no guarantee that the request or response came from the party listed in the "From" message header. The "Response-Key" request-header field can be used by a client to request the key that the called user agent should use to encrypt the response with. Some headers that cannot be encrypted for end-to-end encryption such as Call-ID, Content-Length, Cseq, Encryption, From, Hide, Max-Forwards, MIME-Version, Proxy-Authenticate, Proxy-Authorization, Proxy-Require, To and Via.

Here is the INVITE message of Alice to Bob in the scenario studied last section ("{}" is the encrypted part):

```
INVITE sip: bob@montreal.ca SIP/2.0$ //can be rewritten by proxies
Via: SIP/2.0/UDP sip.montreal.ca$
Via: SIP/2.0/UDP eiffel.paris.fr
From: Alice<sip:alice-paris@paris.fr>$ //caller
To: Bob<sip:bob@montreal.ca>$ //callee
Call-ID: 187602141351@eiffel.paris.fr
Cseq: 1 INVITE$
Contact: alice@eiffel.paris.fr //contact address
Subject: business
Content-Type: application/sdp
Content-Length: 102$
$
{Subject: talk about company.$
Content-Type: application/sdp$
Contact: Sip:bob@bob.site.uottawa.ca$
$
v=0$
o=bell 345637845 3456786578 IN IP4 128.3.4.5$
s= talk about company.$
t=0 0 $
c=IN IP$ 135.180.144.94$
m=audio 3456 RTP/AVP 0 3 4 5$}
```

4.3.4.3 Conclusion on SIP security

SIP message authentication can be provided by strong signature, challenge-digest scheme or by another protocol. End-to-end encryption of the whole SIP message can only be achieved with another protocol (such as IPsec or TLS). Built-in SIP encryption schemes provide message encryption (using PGP or another scheme) but certain header fields such as "From", "MIME-Version", "Proxy-Authenticate", "To", "Via" and encryption parameters have to remain in clear because they contain crucial information for message routing. *Via* fields can be hidden with a special mechanism to prevent eavesdroppers to see the route (and perform traffic flow attacks). Thus, it typically protects the subject, the content-type and the body of the message but not the identity of the caller and the callee. If the requirements of an architecture include privacy of the caller and the callee identities, a lower-layer security protocol must be used to encapsulate SIP (e.g. IPsec, TLS or another protocol).

4.4 Real Time Protocol

4.4.1 RTP/RTCP overview

The Real Time Protocol (RTP) [RFC1889] is a data transport protocol that provides sequencing, intra and inter-media synchronization (no data is sent for silence), payload identification and frame indication. RTP is multicast friendly, media independent and allows media mixers. These properties make RTP the most widely used protocol to carry media payloads. RTP is used on top of UDP. Real Time Control Protocol (RTCP) provides feedback and identification exchange, encryption (media streams encrypted with some non-RTP method e.g. SIP or SDP). RTCP is used over UDP. Media senders and receivers periodically send RTCP packets. RTP media payloads include H263, H261, JPEG, and MPEG.

4.4.2 RTP security

RTP specifies a standard way to encrypt RTP and RTCP packets using private key encryption schemes such as DES in the cipher block chaining (CBC) mode. "Key distribution and certificates are outside the scope of this document. "[RFC1889, section 9]. The first twelve octets of the RTP header are present in every RTP packet. Authentication and Message Integrity are not provided in RTP. It is expected that authentication and integrity services will be provided by lower layer protocols in the future." Security in RTP/RTCP was something temporary that became permanent. That is why RTP needs to be extended to carry secure media payloads.

Chapter 5 Mobility architectures and their security features

In terms of multimedia, the major challenge at present is to be able to preserve multimedia communications in spite of the mobility of end users. The implication of this mobility is that the underlying computer environment is prone to dramatic changes in terms of the quality of service attainable from the underlying network, the characteristics of the end system (screen size, processor capacity, battery life, etc), and indeed the physical location of the end system. All these have a profound impact on the (multimedia) service being offered. Given this, it is well recognized that mobility requires support for adaptation.

Adaptation is typically required at all levels of a system, from the application potentially right down to the operating system. However, this immediately introduces a number of problems. For example, adaptation at the operating system level can be quite dangerous in terms of affecting integrity and performance. The opposite extreme of leaving all adaptation to the application is also clearly unacceptable, as this would introduce an unacceptable burden for the application writer. A possible solution is to introduce a framework for managing adaptation in a middleware.

In this chapter, we first study the current main proposal to support mobility at the IP level i.e. Mobile IP. The Mobile IP architecture is typical of a scenario where a mobile user visits a foreign domain. The Internet Mobile Host Protocol (IMHP) is proposed as an improvement of Mobile IP. The mobility infrastructure proposed by Mobile IP and IMHP could be improved in several aspects. The main ones are scalability, the bottleneck at mobility agents, smoothness of handoffs, firewalls and security. Cellular IP deals with scalability. Many other proposals deal with user authentication. We present a few of them in this chapter. Finally we present current work on a next generation middleware platform capable of supporting mobility.

5.1 Protocols for mobility in the Internet

5.1.1 Mobile IP protocol

When IP routing was originally defined, mobility of hosts was not considered to be an issue. Routing methods were built for static networks, where the hosts were unlikely to move from one subnet to another. Routing takes advantage of a "network number" contained in every IP address. Thus, the IP address encodes the computer's physical location, and - by default - the location is fixed. Mobile IP defines protocols and procedures by which packets can be routed to a mobile node, regardless of its current point-of-attachment to the Internet, and without changing its IP address. Moreover mobile nodes should be able to communicate with hosts that don't implement Mobile IP. Other requirements are authentication of management packets, minimization of the number of management packets and no additional constraints on the IP address space.
The principle of mobile IP is illustrated in Figure 5A:

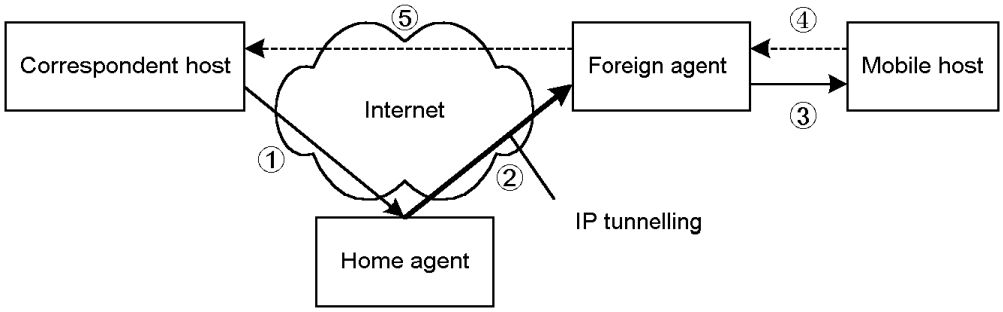

Figure 5A: Mobile IP architecture.

The three basic functional entities in mobile IP are the mobile host, the home agent and the foreign agent. Their role is explained in the following scenario. Assume a correspondent host who wants to communicate with a mobile host (MH). Each mobile host is registered in a home domain. Each domain has a *home agent* (HA) that takes care of mobile IP routing. A *home agent* in a foreign domain for a mobile host is called a *foreign agent*. (FA) Every mobile host has two addresses: the fixed *home address* and a *care-of address*. The home address is static and identifies the connection. The home agent has a most up-to-date database, which is able to tell where the host actually is, that is *the care-of address*

of the mobile host. This *care-of address* can be the address of a *foreign agent*, which has the same function as the base station in mobile telephony. The care-of address may also be a unique local address (*co-located care-of address*), which makes it possible to process the necessary functions inside the mobile host, without any foreign agent. This is especially useful in networks which have not deployed a foreign agent. For clarity we assume below that a foreign agent is available. Technically the care-of address is just the end point of a tunnel. The Mobile IP protocol describes mechanisms of *agent discovery*, *registration* and *IP tunneling*.

An extension of ICMP router discovery protocol is used by mobile hosts to discover foreign agents and to detect movements from one subnet to another. Mobility agents (HA and FA) periodically broadcast agent advertisements. A mobile node expects to receive periodic advertisements. If it doesn't receive them, it deduces that either it has moved or its agent has failed. Mobile nodes can also broadcast agent solicitation messages.

Registration is a mechanism by which the mobile host communicates reachability information to its HA. Registration messages create or modify a mobility binding at a HA, which is then valid for a certain lifetime period. It uses two control messages sent over UDP: *Registration Request* and *Registration Reply*. Registration provides FA smooth handoff. As part of registration, MH requests its new FA to notify its previous FA. The new FA sends a *binding update* to the previous FA. The previous FA updates its binding cache entry for the MH and sends a *binding Ack*. Authentication of binding update is based on a shared registration key between the new and the previous FA.

Routing is defined as follows and is illustrated in Figure 5A. Packets destined to a mobile node are routed first to its home network (message ①) - a network identified by the network prefix of the mobile node's (permanent) home address. At the home network, the mobile node's home agent intercepts such packets and tunnels them to the mobile node's most recently reported care-of address (message ②). At the endpoint of the tunnel, the inner packets are decapsulated and delivered to the mobile node (message ③). In the reverse direction, packets sourced by a mobile node are routed to their destination using standard IP routing mechanisms. In basic mobile IP operation, packets sent by the correspondent host to the mobile host are always sent to the mobile host's foreign agent (a Mobile-IP router) first (message ④), and then forwarded by the foreign agent to the

mobile host's correspondent (message ⑤). Packets originating from the mobile host are sent directly to the correspondent host, thus forming a triangular route. These packets use the mobile host's home address as their source address to preserve their home identity. The packets forwarded by the home agent to the care-of address are encapsulated in another IP packet. If a mobile host is on its home network, it may operate as though it had a fixed connection without any mobile IP specialties. When a FA receives a tunneled datagram for a MH for which it has no entry, it is tunneled back to the HA in a special tunnel. That gives the datagram one more chance of successful delivery.

5.1.2 Internet Mobile Host Protocol

The Internet Mobile Host Protocol (IMHP) is an extension of the basic Mobile IP protocol that features route optimization and authentication of management packets. IMHP defines four entities: Mobile Hosts, Foreign Agents, Cache Agents and Home Agents. Triangle routing in basic Mobile IP limits performance transparency and creates bottleneck at the Home Agent. Route optimization eliminates triangle routing. Any correspondent node can maintain a binding cache such that it tunnels datagrams directly to the care-off address of the mobile host.

Figure 5B illustrates the operation when a mobile host (MH1) within range of a foreign agent FA1, having a home agent HA1, wants to communicate with another mobile host (MH2) within range of a foreign agent FA2, having a home agent HA2. MH1 is supposed to understand *Binding Notify* packets which is not the case in basic mobile IP mode.

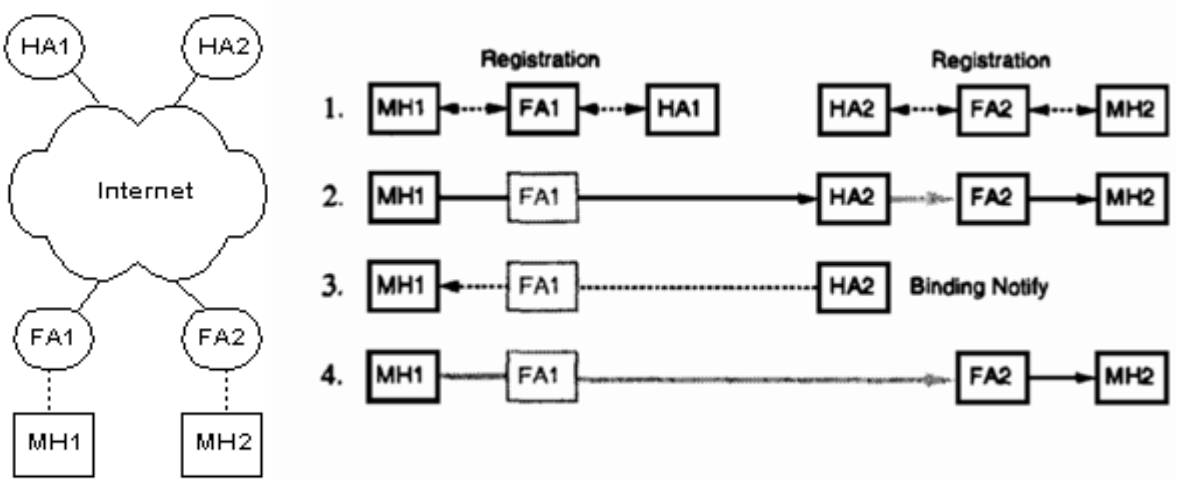

Figure 5B: Mobile host to mobile host communication.

1. MH1 and MH2 both register with their foreign agents and notify their home agents of their new bindings.
2. Suppose that MH1 wants to send a packet to MH2 and MH1 does not know the care-of address of MH2. MH1 transmits the packet relying on existing routing protocols (e.g. through FA1). The packet is received by MH2's home agent, HA2, which tunnels the packet to MH2's foreign agent FA2. When FA2 receives the tunneled packet, it decapsulates it and delivers it locally.
3. When HA2 receives and then tunnels the packet, it also sends to the source (here, MH1) a *Binding Notify* packet containing MH2's binding, as MH1 seems not to have a binding cached for MH2. MH1 requests and gets a binding update from HA2.
4. MH1 can transmit future packets for MH2 by tunneling them directly to MH2's current foreign agent, FA2. A close to optimum route is thus established.

When MH2 moves from one domain to another domain, handoff occurs as following.

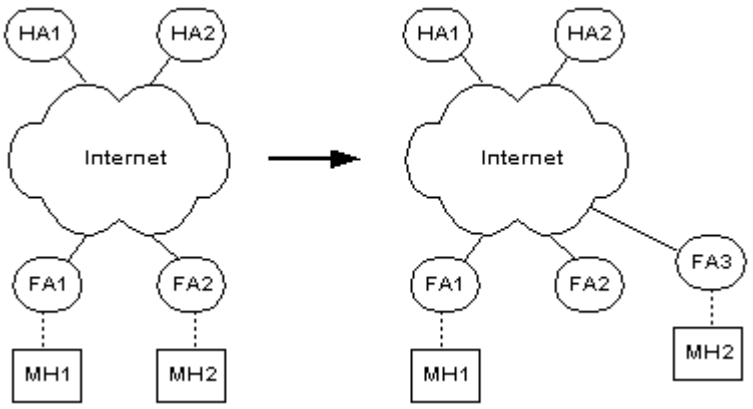

Figure 5C: IMHP smooth handoff.

1. MH2 registers with FA3 and HA2 and notifies FA2
2. MH1 sends a packet to MH2. MH1 forwards it to FA2, which tunnels it to FA3. When FA3 receives the tunneled packet, it decapsulates it and delivers it locally to MH2.

3. FA2 sends a *Bind Notification* to MH1. MH1 requests and gets a new binding for MH2 from HA2
4. MH1 can transmit future packets for MH2 by tunneling them directly to MH2's current foreign agent, FA3. A smooth handoff is thus carried out.

5.1.3 Security considerations

5.1.3.1 Security issues

A Mobile host is by nature more vulnerable as far as information security is concerned. There is for example a risk of address spoofing from a malicious host that could thus get through firewalls. Security issues appear in the three parts of Mobile IP and IMHP: *agent discovery*, *registration* and *IP tunneling*. Since the agent discovery process is not authenticated, a fake server can broadcast false agent advertisements. Forged registrations permit malicious hosts to remotely redirect packets destined for the mobile host. Default authentication between a mobile host and HA uses MD5 as a digest algorithm with a shared secret key between MH and HA but no authentication between MH and FA is performed. Replay protection needed to ensure that a malicious host does not replay registration messages is done using nonces or timestamps.

5.1.3.2 IMHP security models

IMHP is designed to support a range of security models, from no security, weak security to strong security. IHMP is defined to make use of strong authentication based on a shared secret for MH to HA communication and general authentication (a random number specified in the binding request is echoed in the reply by the HA). If IMHP operates with *no security*, a malicious host may intercept a packet stream to a mobile host very simply by sending a false *Binding Notify* to re-route packets to another address. This risk is totally unacceptable in an open environment like the Internet. Under the *strong security*, IMHP authenticates any *Binding Notify* messages or other information they receive about a mobile host. Public and private keys and trusted servers are used, but as a drawback it slows down the operation.

To gain a level of security available in the rest of the Internet, a *weak security* model can be utilized. In this model there are two main cases which have been protected against malicious attempts. First, the

home agent must have confidence that the care-of address of a mobile host is correct. Second, other IMHP compatible nodes in the network need to authenticate bindings, which are received in *Binding Notify* packets. When a host is migrating, it sends the home agent a *Registration Request* with a password in the weak security model. The password gives the same degree of security as available in the stationary Internet. In the weak security model, when someone asks the home agent about a mobile host's current binding, it sends also a nonce (random number). The home agent replies with the binding and authentication information that includes the same nonce. By doing this no one can repeat the reply with incorrect data. When a mobile host migrates, the new foreign agent receives a random number from the host. When the host migrates again, it sends it a *Binding Request* with the same random number.

5.1.3.3 Use of IPsec

The aim of using the IPsec ESP protocol in Mobile IP is to protect the redirected packets against attacks. It should also enable these packets to go through firewalls. The security models do not protect mobile IP traffic too well against deliberate attacks. The IPSec encryption would be used on mobile IP tunnels. Furthermore, these tunnels would be established by using an automatic key and a security association management protocol (e.g. IKE). However, the Mobile IPSec draft is over one year old and it has not received any RFC status. It is uncertain if this Internet-draft ever gets into the standardization process as it is.

5.2 Other issues on mobility support

5.2.1 Cellular IP

Both Mobile IP and 3rd generation cellular systems are being considered as possible candidate solutions for the delivery of IP data to mobile users but they have important shortcomings. Mobile IP has no fast and seamless handoff control. In contrast, third generation cellular systems provide smooth mobility support but are built on a complex infrastructure that lacks flexibility. Cellular IP [RFC2002] is presented as a "third way", which combines the strength of both. Mobile-IP is intended to provide macro-mobility whereas Cellular-IP is intended to provide micro-mobility.

A Cellular IP network is connected to the Internet through a gateway router. Mobility between gateways is managed by Mobile IP (tunneling from the home agent to the gateway). Mobility within access networks is handled by Cellular IP. Mobile hosts attached to the network use the IP address of the gateway as their Mobile IP care-of-address. If a correspondent node wishes to send a packet to a mobile host, the packet is first routed to the host's home agent and then tunneled to the gateway. The gateway detunnels packets and forwards them towards base stations. Inside the Cellular IP network, mobile hosts are identified by their home address.

Regular data packets transmitted by mobile hosts are used to establish host location information. The path taken by these packets is cached in intermediate stations. The reverse path is taken to send back packets to the host. When a mobile host initiates a handoff, it sends a *notify* packet to the new base station and immediately returns to listening to the old station. After a small delay, the host is sure that both the old and the new base station send packets. It can then perform a regular handoff. The old base station will continue to send packet until the soft-state timer expires. Idle mobile hosts are those that have not received packets for a specific time. These hosts transmit packets at regular intervals to re-activate their presence.

5.2.2 Authentication with Mobile-IP

In [ABOBA99-1], the author proposes to secure the mobile IP infrastructure and especially to authenticate mobile users. Within the classical roaming architecture, the roaming user typically does not maintain a pre-existing relationship with the local provider. As a result, unless authentication between the roaming user and the local provider is based on certificates, the local provider should contact the home authentication server in order to validate the user's identity. When certificate-based authentication is used between the roaming user and the local provider, as described in [ABOBA99-2], the local provider is able to determine whether the user possesses the private key corresponding to the offered certificate, and thus authentication by the home provider is not required.

"Where it is necessary for the local provider to contact the home authentication server, it is desirable for that communication to be secured using public key certificates, as supported in IKE. Since the local provider and home server typically do not maintain a pre-existing relationship, without public key certificate support it is typically necessary for one or more proxies to act as intermediaries in order to reduce the shared secret management problem. The introduction of proxies creates a number of security problems and therefore is undesirable." [ABOBA99-1]

In [ABOBA99-1], the author also suggests the use of public-key based authentication and security association between HA and FA (or AAAH and AAAF). The author advises to rely on IPsec and IKE protocols to establish the SAs between HA and FA. Although one can make use of the IPsec/IKE infrastructure, this mechanism leads to additional message exchanges that may affect latency and handoffs. Thus it seems that a suitable lightweight protocol especially designed according to the mobile IP architecture should be better to secure mobile IP than using a complex infrastructure. That is what Sufatrio and Lam have proposed in [SL99] (see next Section).

5.2.3 Proposed add-on on Mobile-IP for session-based mobility

In Mobile IP, a moving mobile node is permanently associated with one fixed home address and one home agent. In [SL99], mobile nodes may have multiple dynamically-allocated home addresses and home agents that might be different for each session. A *home agent service* could be offered as part of available resources on the foreign network. A home agent relocation mechanism can be used by HA to inform its client a better available HA that can support him. A dynamic allocation on the foreign network is also possible. A primary registration through an Authentication Authorization and Accounting (AAA) server is done to the "real" home agent. Secondary registrations depend on the connections and are done in the foreign networks.

The authors propose a mechanism for routing, roaming and a scalable authentication protocol to secure the operations. For authentication, it is assumed that the mobile node and AAAH (the home's AAA) share a security association i.e. a key or a password and no security association exists

between the mobile node and AAAF (the foreign's AAA). The proposed authentication protocol integrates the authentication information with Mobile-IP. We present here a simplified version of the flowcharts to present the principles of the protocol and not the actual details of Mobile-IP parameters. We supposed FA and AAAF, HA and AAAH are the same or related entities. FA and HA have digital certificates and AAAH can validates AAAF digital certificate based on an existing PKI.

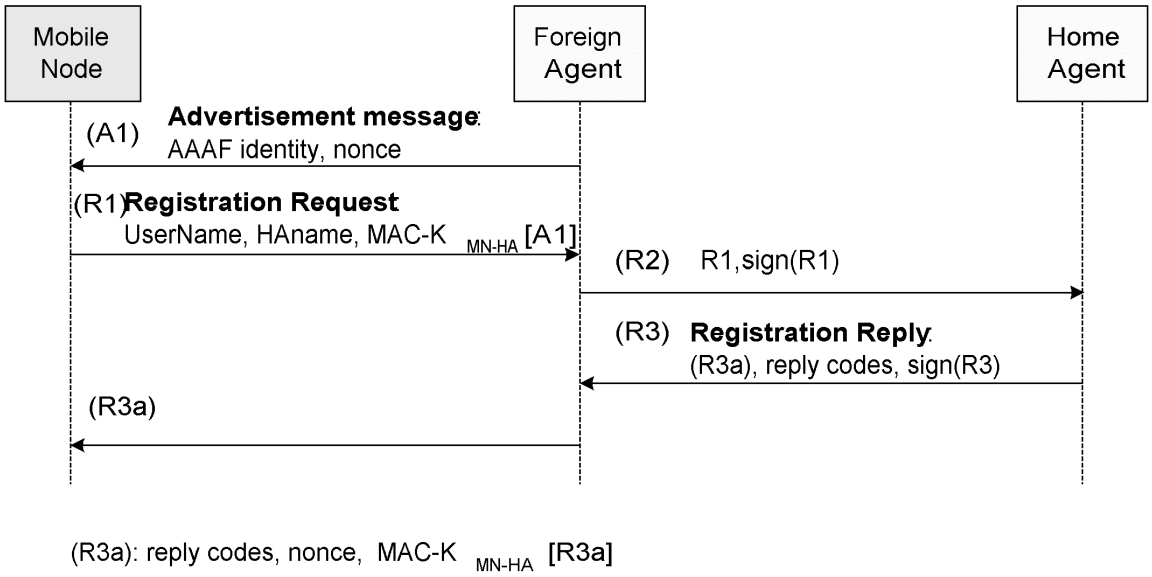

Figure 5D: session-based mobility protocol.

First the mobile node receives an advertisement message (A1) to determine the foreign agent location. (A1) also contains a nonce (challenge) that is used to compute the response i.e. the MAC-value (see Section 2.2.4). This MAC value is computed with K_{MN-HA} that is a key shared by the mobile node and the home agent. The foreign agent forwards the response to HA and signs it to authenticate the message and to ensure its integrity (R2). HA sends back the reply codes and the nonce that identifies the request (R3). (R3) is signed by HA and the reply codes are authenticated by the mobile node with the MAC-value contained in (R3a). That MAC-value is generated with the key shared by the mobile node and the home agent. Thus only HA could have generated such a response. The protocol may additionally distribute new derivative shared-key based security associations between MN and HA or MN and FA to avoid using the permanent shared key between

MN and HA. It should be noted that this protocol does not encrypt sensitive information such as the user name and the mobile node's care-of-address.

5.3 A proposed communication services infrastructure

A very interesting communication services infrastructure also known as the Mobile Internet Telecommunication (MobInTel) infrastructure has been proposed in [HEB00]. This infrastructure that uses the home directory agent concept (see Section 5.3.2) relies on an agent-based middleware that provides global mobility in the Internet. Mobility is not only offered at the IP level like in Mobile IP but rather offered for a set of applications. A middleware is defined as a set of services needed by many applications to function well in a networked environment. That particular middleware is designed to provide multimedia services for end users using the Information Communication Agent (ICA) model [ITU-ICA]. One of the main applications of MobInTel is to provide telephony services over the Internet. We first study the notion of global mobility. Then we present Mobintel architecture functionalities. Finally we compare Mobintel with some existing architectures.

5.3.1 Global mobility

The promising idea of having the Internet always available even as we move is about to become true with the active research and developments in mobile computing. Witness the huge development of pagers, Palm Pilots and soon the third generation mobile phones. In the parlance of telecommunications intelligent network services, full personal mobility (or global mobility) is defined as the ability of end users to originate and receive calls and access subscribed telecommunication services on any terminal in any location, and the ability of the network to identify end users as they

move. Personal mobility is based on the use of a unique personal identity (i.e., personal number). It includes three kinds of mobility that are user mobility, session mobility and terminal mobility.

User (or personal) mobility refers to the ability of a user to access telecommunication services at any terminal (e.g. desktop, laptop, cellular phone, Palm Pilot...) on the basis of an application-layer personal unique identifier, and the capability of the network to provide those services in accordance with the user's service profile. In the introductory example (Section 1.2), Alice can connect from the desktop at her office, her cellular phone, her desktop at home, and also a desktop in France.

Session (or service) mobility refers to the ability to continue a suspended session on another device. This way, users can have the capability to suspend a session at one desk and to pick it up elsewhere on the network, or hand the session over to the help-desk to have a problem fixed. In our main example (Section 1.2), Alice could see and speak with Bob without having to completely reconfigure her connection with Bob.

Terminal mobility is the ability to maintain communications when moving a single end system (e.g. cellular phone) from one subnet to another. Terminal mobility is typically associated with wireless access [Schul96]. In the example of section 1.2, Alice could, on her way home, speak with Bob.

5.3.2 MobInTel architecture overview

A mobility architecture, as considered in this section, includes all the three kinds of mobility described above. It typically involves three parties: the user (say Alice), the home agent (HA) and the foreign agent (FA). The MobInTel infrastructure provides personal mobility using the home directory concept and the agent-based infrastructure presented in [HEB00]. This architecture provides multimedia services with global mobility (terminal, user, and session). The Internet is divided in a large number of network domains (sub-networks).

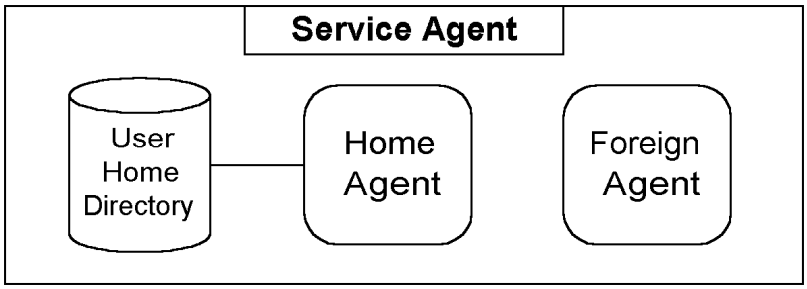

Figure 5E: Service Agent architecture.

Each network includes a Service Agent (Figure 5E) that acts as a Home Agent (HA) for users registered in that domain and as a Foreign Agent (FA) for other users. The Service Agent also includes a *user home directory*. This directory includes information about users registered in that domain concerning authentication, authorization, accounting, quality of service (QoS) preferences and location. The RDF (Resource Description Framework) format [RDF00] is used to store user profiles. The home directory includes information about ID, QoS preferences, contact and location. Only the HA has access to this directory. MobInTel service agents (Home agent and foreign agent) do not need to know each other before any communication.

Quality of service negotiation based on device capabilities and user QoS preferences stored in the user home directory is presented in [HEB00-2]. It is an example of use of information stored in the user home directory. Authentication information is used in the protocol we define in Chapter 7.

In our example, Alice's HA is in Ottawa and Bob's HA is in Montreal. Alice's FA is in Paris and Bob's FA is in Berlin. Thus, the MobInTel architecture is fully distributed and scalable. If either user is in their home domain, then FA does not exist and HA is the only agent contacted. This architecture is presented in figure 5F.

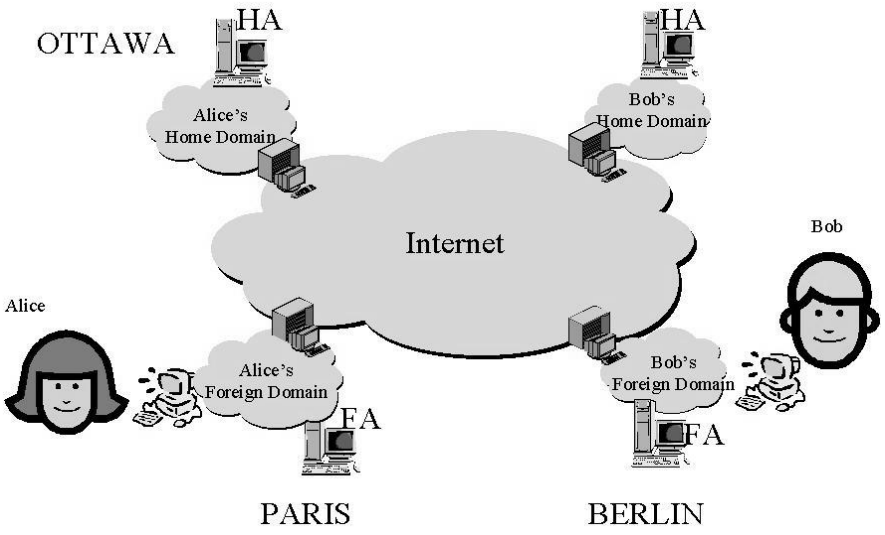

Figure 5F: MobInTel architecture.

Let us describe a typical logon procedure. Assume Alice who is registered in Ottawa arrives in Paris. She has first to identify herself to Paris' FA. Then Paris' FA communicates with Ottawa's HA to verify Alice's credentials and to inform Ottawa's HA that Alice is in Paris' FA domain (Figure 5G). Home and foreign agents manage the resources in their own domain. Alice needs to be known by Paris' FA if she wants for instance to use the local bandwidth broker if any. That is the main reason why we need foreign agents and we cannot only rely on home agents.

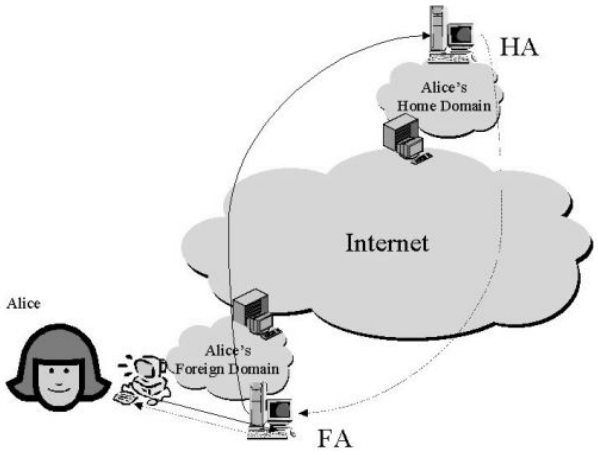

Figure 5G: Authentication process: message exchanges.

We assume Bob followed the same procedure described above to login. Now Alice wants to call Bob without knowing that he is in Berlin and not in Montreal. She calls Bob based on his personal

identifier (e.g. bob@montreal.ca). The call request is routed to Montreal and then either forwarded to Berlin or returned with the new location where to call Bob. Bob can receive the call request and decide to speak to Alice. All the session parameters (phone call parameters) are stored in the network. During the conversation, Alice can switch off her mobile phone and continue the phone call on her desktop computer.

Chapter 6 A proposed secure architecture for mobile Internet telephony

We have studied in Chapter 4 the limitations of current IP-Telephony standards in term of security and mobility. In the previous chapter we have shown the difficulty of providing a complete range of security services for mobile users. We have presented an infrastructure for mobile communication services (the MobInTel infrastructure). In this chapter we first study MobInTel security requirements to provide IP-Telephony (as an application example) with full mobility. We briefly review why existing security mechanisms in IP-telephony and mobility infrastructures do not meet MobInTel security requirements. We then propose a method to provide a complete range of security services for the MobInTel infrastructure.

6.1 MobInTel trust and security requirements

For now, many IP-telephony products focus on Intranets where security is of less concern. Security is crucial in every computer system and even more in the systems where the use of the Internet and wireless links is particularly vulnerable. With the current convergence of networks the future of IP-telephony is inevitably on the Internet where security is a main concern. It is required for any commercial IP-Telephony service to support security mechanisms and to build secure systems and applications. The traditional trade-off between security and computational power of devices is about to become obsolete. Microprocessor companies unveil new chips for cell phones and handheld computers that are especially suitable for multimedia communication and computations of security algorithms. Permanent Internet connections open doors towards new services but also new security concerns. The security requirements should include proper authentication of the user, anonymity of

the user for the environment, privacy and integrity of communications. We should also define minimal computing requirements and administration costs on the mobile node. We study below the security requirements in details.

We first study the authentication phase which involves a user (Alice), a Foreign agent (FA) (if the user is in a foreign domain) and a home agent (HA). Then we study the IP-telephony phase, which involves two users (Alice and Bob).

6.1.1 Authentication phase security requirements

The user logs on to the infrastructure to be identified. The basic requirements include:

- Security: A network eavesdropper should not be able to obtain the necessary information to impersonate a user.
- Reliability: Lack of availability of a service should only occur in case of normal service saturation.
- Transparency: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password (ideally one password query per user session). The system should be able to support a large number of clients and servers. This suggests a modular, distributed architecture.
- Choice for different security levels: Highest security option, Intermediate option, and Low security.
- In-call security option changes: it is possible to change of security level during a phone call when you are about to speak about a confidential issue.

6.1.1.1 Trust requirements

If Alice connects in her home domain, a direct trust relation can be established with the HA and the whole authentication process is much simpler. We assume in our scenario that Alice connects to the infrastructure from a foreign domain.

The trust relationship between Alice and the FA is based on their trust relationship with the home agent. After authenticating the HA, Alice relies on it to authenticate the FA. In the same way, the

FA relies on HA to authenticate Alice. Thus, a dual third-party authentication registration is performed. Although an explicit authentication between the FA and the user would be better, our scheme is not unreasonable since it spares bandwidth and computational power for the mobile user terminal.

6.1.1.2 Authentication and integrity requirements

Three mutual authentications should be fulfilled: mutual authentication between A and FA (non explicit), mutual authentication between A and HA (explicit) and mutual authentication between FA and HA (non explicit).

Authentication is a critical issue for the Foreign Agent since it will provide resources to the user on behalf of the HA. FA wants to be paid for the resources it will grant and thus it wants to authenticate Alice to know the location of her HA which may be the place to send the bill. A needs to authenticates FA to avoid giving information about herself to a fake server. In some cases where Alice has no Internet access before the authentication process (e.g. wireless access), Alice cannot contact HA directly without going through FA.

FA needs to authenticate HA before sending an authentication request to protect the user's privacy. For his part, HA needs to authenticate FA to be sure of the location of FA and possibly certify to A that FA can be trusted if A has limited trust verification capabilities (e.g. if A cannot check digital certificates). Moreover, FA's authentication is important for HA for service availability purposes. Only authenticated foreign servers can send requests to HA outside HA's domain. This reduces the risk of denial of service attacks (see Section 3.3).

HA needs to authenticate A through FA (i.e. to receive an authenticated message from A through FA) to allow Alice to use FA resources and to charge her thereafter. A needs to authenticate HA through FA to be sure that FA communicates with HA. However, it is possible that FA never communicate with HA. A does not mind if FA is not willing to be paid for the resources it grants. However A needs to ensure that HA knows A's location to forward to her communication calls from other people.

Integrity is necessary in any communication to protect data from any possible change.

6.1.1.3 Privacy requirements

Our goal is to minimize, as much as possible, information available to a passive or an active attacker about the privacy of the user. Communication between A and FA should be encrypted for privacy. If the user asks for it, even the user name should be protected so that an attacker cannot determine which user is logged on. In the same way, privacy is intended to prevent traffic analysis (see Section 3.3). Data between FA and HA on the one hand and, between A and HA on the other hand, need to be protected for privacy.

6.1.1.4 Non-repudiation requirements

Non-repudiation service is needed for communication between FA and HA and between A and HA, for the purpose of billing. FA bills HA which, in turn, bills A for the used resources. We need a non-repudiation service in case Alice claims she never used these resources. We do not need non-repudiation between A and FA since no billing transactions are done between these two entities.

6.1.1.5 Service access control and key management

FA controls access to local services (e.g. bandwidth brokerage) since it is the only one who has authenticated A in the local domain. The right to use a specific service can be granted by HA to A through FA. At the end of the authentication process, A must have established a shared session-key with FA to encrypt future communications (for privacy). This key will be used to authenticate A in future communications.

6.1.2 IP telephony security requirements

MobInTel's basic services are voice signaling and voice transfer via an IP-network. This is called IP-telephony or "Voice over IP" (VoIP). We consider one-to-one telephony (e.g. Alice calling Bob). One-to-many applications addresses special issues raised by the use of multicast. We do not deal with them in this work.

6.1.2.1 Trust requirements

Alice does not trust the HA to establish secure communication with parties other than FA. Thus Alice can establish a session key with Bob without HA knowing it. Alice should also be able to phone Bob with privacy and anonymity. To provide the latter, signaling messages should be encrypted all along the way.

6.1.2.2 Basic tasks and attacks

We distinguish three basic tasks that IP-Telephony systems and protocols have to carry out, the same as in the old PSTN. That reduces the complexity of defining security requirements and mechanisms possibly used to fulfil the requirements of these three basic tasks. These basic tasks [RRAS1] are the same in every scenario and have to be done for providing every service:

- *Signaling* provides for set up and tear down of calls, including the setup and maintaining of databases and entities used for call routing and number translation.
- *Transmission* is the carrying of the audio (and / or video).
- *Operation* implies the provision, configuration and maintenance of all services, which are used by providers and different users, such as location services or charging and billing services.

Different attacks can be undertaken against these basic tasks:

- Change the caller identity or the callee identity is an attack against the *signaling* service. Traffic analysis can be realized by eavesdropping the signaling data.
- Eavesdropping is the simplest attack on the *transmission* service.
- Operation Systems can be directly attacked or forged configuration or management commands can be used. These are attacks against the *operation* service.

6.1.2.3 Security requirements

The five different security services (Section 2.1), which are authentication, access control, confidentiality, integrity and non-repudiation, are necessary for the IP-Telephony tasks of transmission, signaling and operation. We illustrate how these services are used for IP-Telephony, to figure out which services are essential. It is not possible to describe all security requirements in detail in this thesis (see [RRAS1] for more details). In addition to the above categories, Table 6A shows an executive summary of the requirements, whereby the essential requirements are marked in grey.

	Signaling	Transmission	Operation
Authentication, Integrity, Confidentiality	End to end and Hop by hop	End to end	End to end
Non-repudiation	Only services which are liable for costs	Only for special services e.g. voice information on call	End to end

Table 6A: IP-telephony security requirements.

Signaling should be encrypted to protect the user privacy and authenticated to prevent spoofing. End-to-end (end-user to end-user) encryption protects signaling information that are not necessary for routing (e.g. call invite content). Hop by hop (link or server to server) encryption protects signaling information that are necessary for routing (e.g. caller and callee identities, server address). Non-repudiation is only necessary for services which are liable for costs (e.g. voice mailbox consulting).

Media data transmission should be end-to-end encrypted since only the caller and the callee should know about the content of the conversation. Non-repudiation is only necessary for services which are liable for costs (e.g. voice information on call). Call operations should be fully protected since that concerns very sensitive operations such as billing.

6.1.2.4 Security concerns related to IP-Telephony

Key management is not a fundamental security requirement, but it is necessary for many cryptographic security mechanisms to ensure their scalability. End-systems and network servers like gatekeepers and gateways systems must achieve the same general security requirements as existing computer systems and applications. Firewalls network address translations sets an issue for IP-telephony protocols. This problem is described in detail in [RAS00] and also discussed in [RARR99], [Biggs00], [RDS00].

6.2 Review of existing approaches

We briefly review why existing security mechanisms in IP-telephony and mobility infrastructures do not meet MobInTel security requirements.

6.2.1 Mobile Communication Infrastructure

Most existing mobile systems, such as the Global System for Mobile Communications (GSM) (section 4.1), do not transmit all communications on the Internet, and thus lead to different security requirements. GSM provides terminal mobility only and it is based on a fixed signaling network that is supposed to be secure (Section 4.1.3). In such homogeneous mobile user environments, no operations between the foreign domain and the home domain are needed, or these operations are static (e.g. roaming agreements).

The Internet is formed by a set of heterogeneous networks which are however administrated locally. No trust relationship exists between a home domain and a foreign domain before they authenticate each other. As a result, the approach taken by GSM cannot be simply transposed to the Internet environment. Moreover, it is not scalable to consider defining security associations between all pairs of foreign and home agents.

As well, a user can only switch from one GSM device to another GSM device and not to another type of terminal (e.g. a desktop computer). Thus user mobility is not fully provided. It is also impossible to reopen a suspended session.

A centralized key distribution center (KDC) is used in Kerberos (Section 3.1.3) to assist authentication and key management. In the Internet, it is very common that a long distance exists between KDC and the foreign/home domain, and thus a long delay will be introduced in communication with the KDC. A reasonable authentication and key distribution scheme should be managed on a distributed, rather than centralized basis, since the application environment is completely distributed. These observations strongly suggest that we take the public key approach for designing an authentication and key distribution scheme. In addition, a user cannot connect from an

external realm to his home realm. He can only access resources of a foreign realm from his home realm.

An architecture only based on a home directory cannot control the use of local resources. For example, Alice can connect via *ssh* (Section 3.1.4) to a Unix account (comparable to our home directory) but all services remain granted by Alice's home server. No local resources (bandwidth reservation for example) that are especially useful for multimedia telecommunications can be made available with a direct connection to a home agent unless they are freely available. In the Mobintel architecture, local services can be provided by means of the foreign agent that has identified the user. Thus, the FA is able to limit or to bill local resource utilization.

In SET (see Section 3.3.4), the merchant can be compared to our FA and the bank to our HA. This system provides user mobility since the user can connect from any device that supports a SET Credit Card reader. It is a transactional system however so it cannot support session mobility.

Radius (Section 3.1.4), a widespread network access system can provide terminal mobility but does not provide full user mobility. MobInTel provides user, session and terminal mobility on a password-based access that allows a user to consume local resources on the local agent control.

A few schemes related to Internet mobility have already been proposed. Several of them deal with the registration or authentication process, without considering communication with other users at the application level. Some of them are related to mobility at the IP level (i.e. Mobile IP, Section 5.1). The latter involves a user, a foreign agent and a home agent, similar to the MobInTel architecture. The Mobile IP specifications define only authentication services. Data encryption and absolute location privacy are not addressed there. The secure registration protocol for Mobile IP presented in Section 5.2.2 assumes, as in the Mobile IP specification, that the user owns a shared key with the home agent. User identity is not protected in the local domain. Moreover, a user without a certified public-key cannot be authenticated directly by a party that didn't know him or her before. Such a user cannot digitally sign a message as well. The same remarks apply for the authentication framework for mobile IP as proposed in [SufLam99].

6.2.2 Secure telephony on the Internet

Telephony on the Internet means that both signaling and communication data are transmitted through the Internet. General public Internet telephony products are currently not secured [RUAS00]. Some telephony software introduced various kinds of security features but no architecture takes into account both QoS and security requirements. In Microsoft Netmeeting™ version 3 [Netmeeting], only user authentication, and data encryption (excluding audio and video) is provided. PGPfone (Section 4.2) makes use of a "voice signature" scheme based on voice recognition to authenticate users but this scheme is not completely reliable and is not convenient for the user. in the latter case however data encryption can be provided.

There are two main telephony-signaling protocols on the Internet: one defined by ITU (International Telecommunication Union) within H.323 and SIP (Session Initiation Protocol) (Section 4.3) defined by the IETF (Internet Engineering Task Force). H.323 is a set of recommendations, which defines how voice, data, and video traffic will be transported over IP–based local area networks and the Internet. SIP is an application-layer control protocol for creating and terminating sessions such as Internet telephone calls. SIP in itself supports user mobility by redirecting requests to the user's current location. Users can register their current location. SIP supports user location, device capabilities, user availability, call setup and call handling.

Achieving security in IP-networks requires a higher degree of sophistication than in PSTNs (Public Switched Telephone Network). The risk of being attacked using the IP-telephony infrastructure is higher than using the PSTN for a telephone call due to the differences of the networks and system architectures. IP-Networks are neither centrally managed nor centrally controlled. Thus many systems (local telephony servers) have to be administrated and secured. In IP-Telephony, voice transmission and signaling is done over the same network, so a user may forge signaling information. Some companies offer to use their VPN (Virtual Private Network) based on IPsec to transmit signaling information. We consider here the worst case where no security link initially exists in the infrastructure. Active elements of an IP-network, like routers or network servers are, by design, accessible from the network they control, so they are more vulnerable, especially to tapping. Endpoints of IP-networks, personal computers and servers can also be attacked [RRAS00]. We

study the specific security risks of IP-Telephony in the MobInTel architecture. We do not study the risks of general network services which are not especially IP-Telephony related (name server attacks or denial of service). An overview of the latter risks was given in Section 3.3.

We have studied MobInTel security requirements. We showed that existing infrastructures do not meet MobInTel security requirements and cannot be simply adapted to this infrastructure. Therefore in the next section, we propose a new scheme that could fulfill the expected requirements. We justify our choices on whether to partly reuse existing mechanisms or not.

6.3 Security scheme proposal

6.3.1 User Logon

6.3.1.1 User authentication scheme

In our scenario, we provide user global mobility that is mobility of location, terminal and session. User and session mobility imply the user's secret (user's secret shared with its home agent) moves from one device to another.

Several user logon applications have been reviewed in Section 3.1.5. We cannot directly rely on a "user action" authentication method (such as writing a signature with the mouse) for the Mobintel architecture since we cannot afford even a small percentage of mis-verification. This is an interesting idea however to be followed by further work. In the same way, for the time being, we cannot rely on biometrics systems to authenticate every user in the Mobintel architecture. However, it is possible to use certain biometrics to access some high-security level resources. Voice recognition can be used on devices allowing voice transmission. Smartcards could be used in the Mobintel architecture by Alice and Bob to store a digital certificate signed by their respective Home Agent (HA). The HA can himself be certified by another Certification Authority (CA). These smartcards could be used to authenticate users from any smart-card reader device. However, this solution is still costly and requires a unique standardized reader for all devices. Relying on the presence of such reader device reduces user mobility. Passwords (or passphrases) are the easiest way to carry the user's secret to

different locations since they are immaterial. This is the choice adopted in our work. This choice raises security concerns that are discussed later.

6.3.1.2 Server identification schemes

HA and FA do not know each other before A connects to FA. So we cannot rely on a previous shared secret between these two entities. We need mutual authentication and privacy. Privacy can be established with public-key cryptography that allows to establish a secure channel. Parties send to their correspondant their public-key so that they can exchange a session key. This can be done via SSL, IKE or any other protocol that establish a secure channel. A digital certificate helps to provide authentication. A digital certificate binds an entity's public-key to its identity. If servers can check certificate's validity (relying on a PKI), FA could authenticate HA by sending to it an unpredictable challenge. Then it could decipher the response with FA's public-key. If the deciphered message matches the challenge, that means only the entity that owns FA's private key can have sent this message. Therefore we can assume that every server (HAs and FAs) owns a digital certificate.

6.3.2 IP Telephony

6.3.2.1 General requirements for a security solution

Quality of Service is very important for any real-time multimedia system. The voice transmission delay should be no more than 30 ms. The call setup delay is also a major concern for IP telephony. The use of cryptographic mechanisms and especially public-key mechanisms can result in a longer delay. The overhead produced by security features should be considered. Any proposed solution must be scalable and should work in large environments. The infrastructure used by cryptographic mechanisms (e.g. certificate authorities) must be available and operational. Security mechanisms themselves must contain no single point of failure. We should evaluate the grade of security of such mechanisms in term of performance and necessary time to perform a brute-force attack. However it is difficult to precisely quantify the degree of security.

6.3.2.2 Security in IP-Telephony Standards

Security mechanisms should be part of IP-telephony standards to achieve a better compatibility between systems. Two different protocol families for IP-telephony currently exist: H.323 [ITU98]

and SIP [Chapter3, HSSR99]. The H.323 umbrella standard is proposed by ITU whereas the SIP signaling protocol is proposed by the IETF. The two protocols are compared in [DF99, Dous00].

Confidentiality is the main requirement for the transmission service. RTP [SCFJ96] and RTCP are the underlying protocols used for transmission in both H.323 [ITU98] and SIP. Datastreams (RTP payload) can be encrypted but RTP-Headers cannot be encrypted because they include essential routing information. Symmetric encryption is used for a better performance. The ITU suggests the latter encryption solution for data streams [ITU98-2]. The encryption capabilities can be negotiated during signaling. RTCP security is not defined sofar. Symmetric encryption requires that both users have exchanged a key. Key management is part of H.245 signaling but it is not part of SIP. We need to add key management in the SIP message body if we want to use this mechanism.

Authentication and Confidentiality are major concerns for the signaling service. The ITU H.235 recommendation deals with the security aspects of H.323. In H.235, authentication of users is done during call establishment. The signaling channel (H.245, [ITU97]) can be secured hop-by-hop by a challenge-response mechanism or with a certificate exchange. End-to-end authentication is not provided. TLS (Section 3.1.3) or IPsec (Section 3.1.2) (or other similar mechanisms) are the only way to provide authentication and call authorization before the call is accepted. Confidentiality of the signaling information is not provided, except using TLS or IPsec. Key management is partly supported with the exchange of certificates and Diffie-Hellmann (Section 3.2.1) key exchange.

A SIP request can be authenticated using the *Authorization* header field to include a digital signature of certain header fields (not all because some of them have to be changed by proxies) and the payload. PGP or HTTP-authentication is used. So, end-to-end authentication is not completely provided. TLS or IPsec can be used to provide complete authentication. SIP offers three forms of encryption: hop-by-hop encryption of certain header fields (not all because some of them are used for call routing) and the SIP body, hop-by-hop encryption to prevent eavesdropping tracking who is calling whom, hop-by-hop encryption of *Via* fields to hide the route packets have taken. Additional security features can be provided through a network or application level protocol such as IPsec or TLS.

Major requirements are covered in the current standards but many additional features required for new commercial services like non-repudiation are not performed. A great disadvantage is the use of IPsec that is a low-level security protocol and it is not a widespread mechanism yet. Performance can be greatly lowered (for example during call establishment) if certificate verification has to take place and additional messages are processed. Moreover, some mechanisms may not be scalable. Further work is needed on integration of authentication mechanisms in signaling protocols.

SIP combines both simplicity and user mobility. Many security features are already defined. We choose to use SIP for call signaling and RTP that is the most widespread use protocol for media transmission. We use additional security features as described in the next chapter.

Chapter 7 Protocol proposal

In this chapter we propose a protocol that meets the MobInTel security requirements described in Section 6.1. We first specify the authentication protocol in an general manner and then in detail with the data interchange coding. We rely on the authentication protocol to present a protocol based on SIP that offers an increased degree of security. The message exchanges are detailed. In the second part, we examine the degree of security of this infrastructure. Finally we give a brief review of possible security attacks and how they are avoided in our scheme.

7.1 Protocol specification

7.1.1 Authentication abstract specification

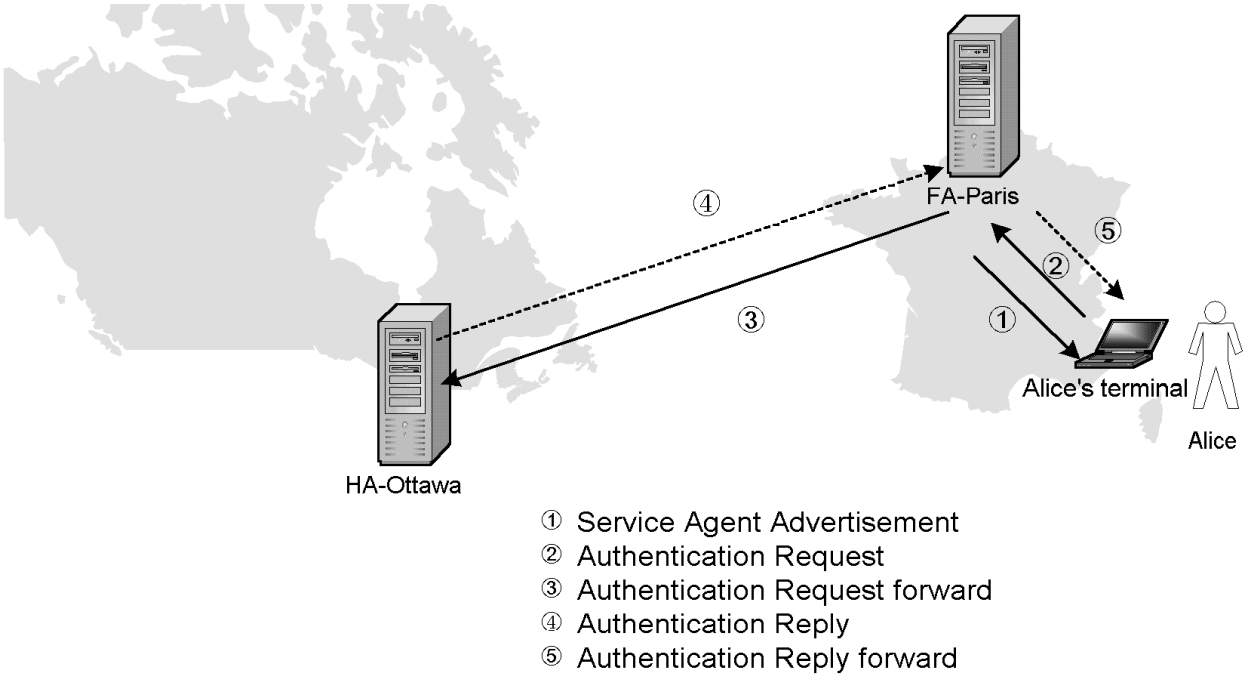

Figure 7A: Authentication protocol overview.

7.1.1.1 Abstract specification

We first recall a few cryptographic notations that we use to describe our protocol message exchanges. The comma means concatenation.

ID	UserID (ex: alice@ottawa.ca)
KU_X/KR_X	Public-key of X/private-key of X
Ks	Session key (secret key)
N	Nonce value
TS	Timestamp
HV	Hash-value
DP	Device profile
SecCx	Secure connection
K[data]	Data are encrypted/decrypted using key K
pwd	Password
Ack	Acknowledgement
Nak	Negative acknowledgement
Cert	Digital certificate

Table 7A: Cryptographic notations.

The device profile (DP) contains parameters specific to a device such as the media (phone=Y/N, text=Y/N…) or the coding (PMC=Y/N, MPEG=Y/N…) it accepts.

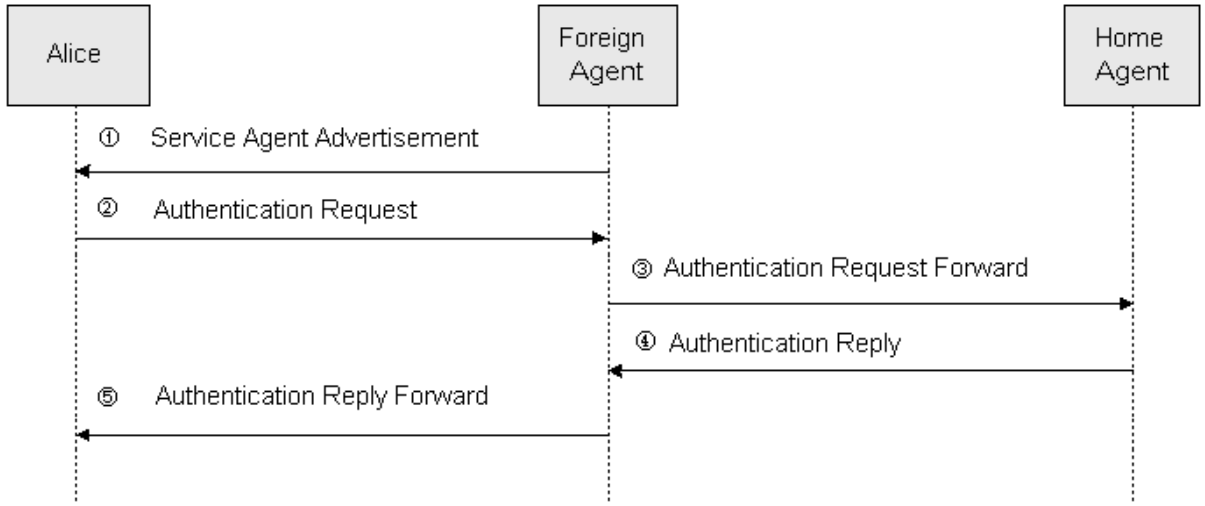

Figure 7B: Authentication message exchanges overview - Ack.

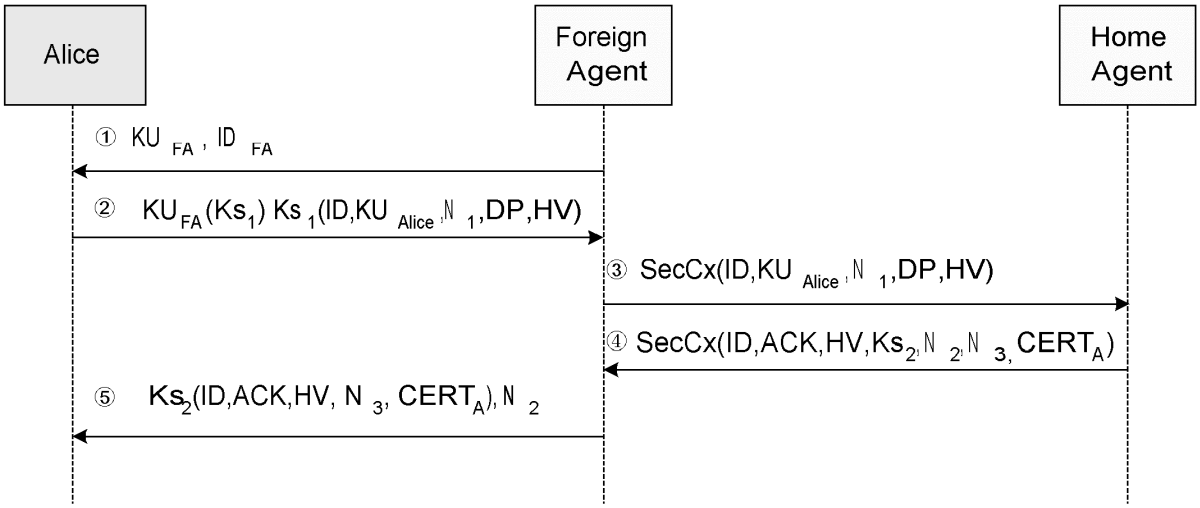

Figure 7C: Authentication message exchanges - Ack.

Where: Ks_2=f(N1,N2,pwd) (f could be a one-way function), Ks_3=f(N1,N3,pwd), HV= H(ID,pwd,N1,DN,DP,KU_{Alice}) (H is a one-way hash function).

We explain the five authentication message exchanges in the case of successful authentication (Figure 7C) above.

① Service Agent Advertisement: KU_{FA} , ID_{FA}

When Alice arrives in a visited network, she first needs to know where the foreign agent is located. We can either assume this agent has a standardized name (e.g. fa.domain_name) or assume that Alice learn this information by broadcast messages (e.g. through DHCP messages). This information includes the IP address and port number of FA and FA's public key KU_{FA}. Only the presence of FA's public key is mentioned in Figure 7C. FA could send its certificate containing its public key but we assume that Alice cannot check certificates and thus she could only read the public key from FA's certificate. Note that this public-key is not authenticated and could be the one of an intruder. We explain below that this has limited consequences on the global security of the system.

② Authentication Request: $KU_{FA}(Ks_1)$ $Ks_1(ID,KU_{Alice},N1,DP,HV)$

Alice picks a random session key Ks_1 that will be used only for message ② to avoid encrypting the whole message with FA's public key. The message includes Alice's identity (e.g. alice@ottawa.ca) with her home domain address so FA knows in which domain to forward the authentication request. Alice generates a pair of keys KU_{Alice} and KR_{Alice} on her terminal. KR_{Alice} stays on the terminal and is never sent to a MobInTel agent. Alice sends KU_{Alice} with the authentication request so that HA can bind Alice's name and KU_{Alice}. In other words, Alice sends a Certificate Signing Request (CSR) to HA, which computes a digital certificate acting as a certificate authority. Note that HA can be just one branch of the certification tree so that HA's authority can be signed by a higher level authority. N_1 is a random number that will be used to calculate the session key between Alice and FA. It also prevents replay attacks. The device profile is sent to let HA know which type of media this device supports. The hash-value contains information that allow HA to authenticate Alice such as Alice's ID and her password:

$$HV= H(ID,pwd,N1,DN,DP,KU_{Alice})$$

Additional parameters included in the input of the hash function provide message integrity.

③ Authentication Request Forward: $SecCx(ID,[KU_{Alice}],N1,DP,HV)$

Before forwarding the request, FA keeps track of certain parameters: Alice's ID, Ks_1 (current session key with Alice), N_1 that will be use to create the new session key with Alice and DP that gives FA information about the type of media Alice can receive on her device. FA forwards the message to Alice's HA. FA should be able to retrieve HA's location knowing only the name of the domain. That operation could be done through a DNS lookup in HA's domain. FA opens a secure connection with HA. This secure link can be established by several ways since both FA and HA owns a digital certificate. There are several methods to exchange a session key including authenticated Diffie-Hellman exchange (Section 3.3) that is supported by IPsec, TLS and IKE.

④ Authentication Reply: $SecCx(ID,ACK,HV,Ks_2,N2,N3,CERT_A)$

Through the same secure connection, HA sends back the answer including Alice's ID, the answer of the authentication process (ACK or NACK), the hash-value sent by Alice that uniquely identifies the request and Ks_2 the session-key that will be used between Alice and FA. N2 and N3 are used to calculate Ks_2 and Ks_3 knowing N1 and Alice's password. FA is given Ks_2 in clear.

⑤ Authentication Reply Forward: Ks_2(ID,ACK,HV, N3,$CERT_A$),N2

1) Alice computes Ks_2=f(N1,N2,pwd)
2) Alice try to decipher Ks_2(ID,ACK,HV,N3).
3) If it succeeds, Alice knows FA was given the key from HA. That means FA communicated with HA and was authenticated as a valid agent (HA checked FA's certificate). Moreover Alice can now use the certificate $CERT_A$ signed by HA.

4) Alice computes Ks_3=f(N1,N3,pwd) the session key between Alice and HA.

We explain the two specific authentication message exchanges in case of an unsuccessful authentication (Figure 7C).

In case A is not authenticated, the message exchanges are:

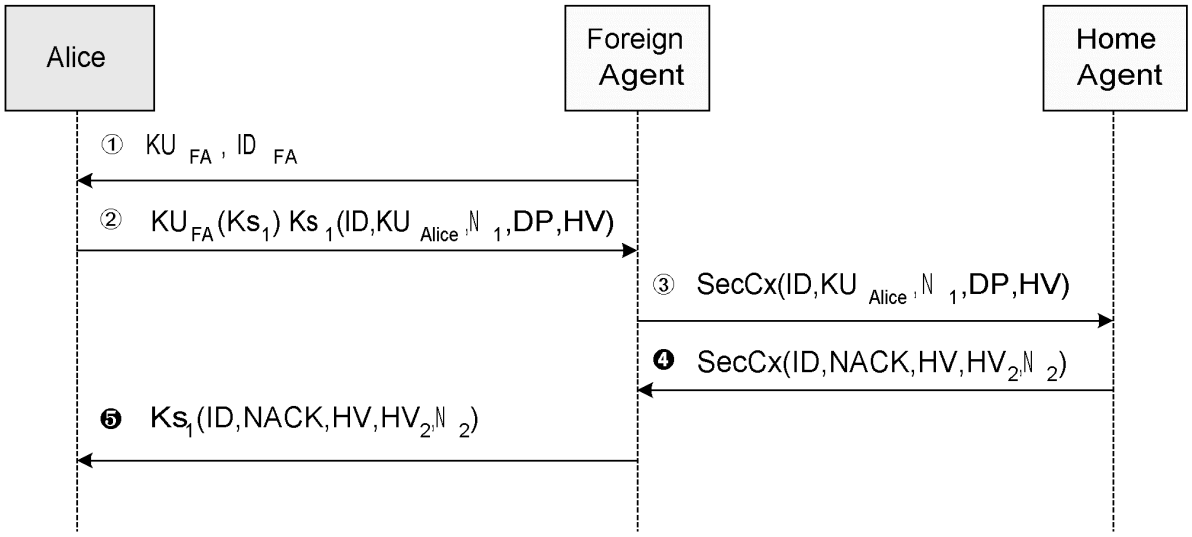

Figure 7D: Authentication message exchanges – Nack.

Where: HV2 = H(HV,pwd,N2) (H is a one-way hash function).

❹ Authentication Reply: SecCx(ID,NACK,HV,HV2)

In case Alice is not authenticated, that is $HV_{HA} \neq HV$ (HV_{HA} being the hash-value calculated by HA), the authentication reply message includes a NACK (negative acknowledgement), previous values to identify the request (ID and HV) and an additional value HV2.

$$HV2 = H(HV,pwd,N2)$$

HV2 is the digest of HV, a nonce and Alice's password. HV2 will be sent to Alice as a proof that NACK is the answer from HA and that FA has communicated with HA to get the answer. The nonce N2 is there to make sure that a cryptanalyst cannot get any information about the password from the pair (HV, HV2).

❺ Authentication Reply Forward: Ks_1(ID,NACK,HV,HV2)

FA answers to Alice using Ks_1. This message indicates the authentication failure and the authentication identifier (ID and HV). As previously said, HV2 is sent to Alice as a proof that NACK is the answer from HA and that FA has communicated with HA to get the answer. Indeed only HA could have generated HV2. Moreover, N2 is part of HV2 to avoid a possible chosen-plaintext attack (knowing H(X) and H(x,pwd)) from an opponent.

If A is in her home domain, the message exchanges are:

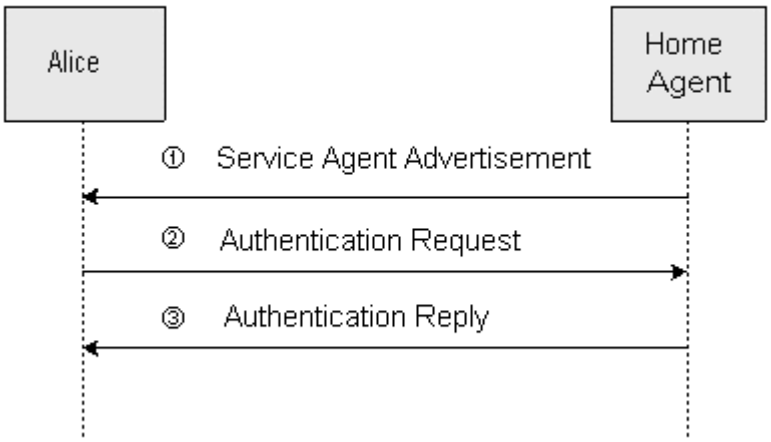

Figure 7E: Home domain message exchanges overview - Ack.

In case of positive acknowledgment:

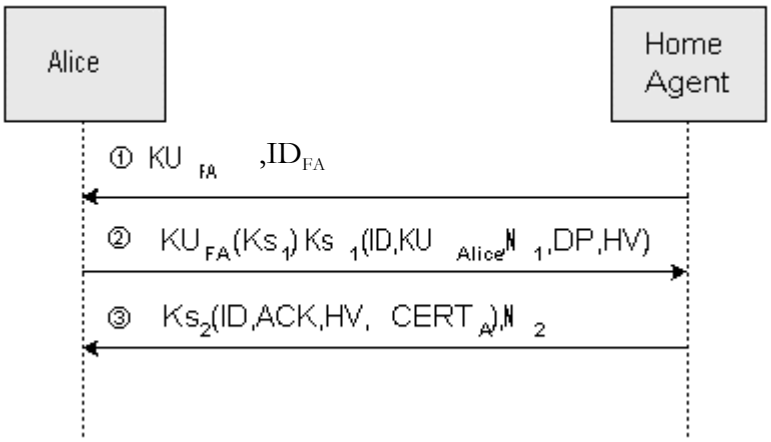

Figure 7F: Home domain message exchanges - Ack.

In case of negative acknowledgement:

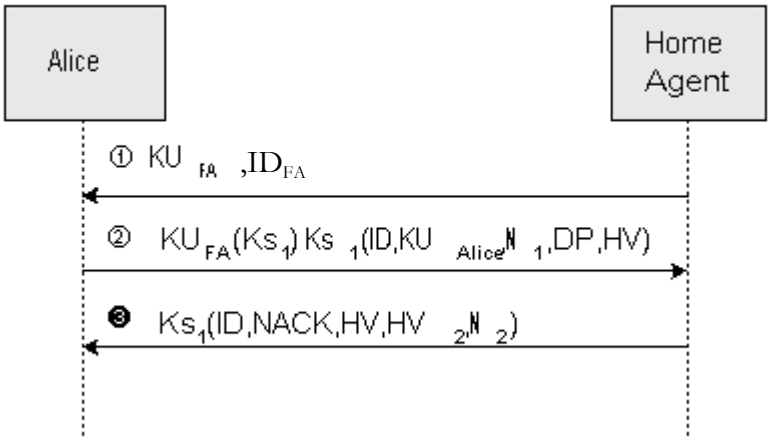

Figure 7G: Home domain message exchanges - Nack.

7.1.2 Authentication message coding

7.1.2.1 Possible choices for message coding and transmission

The authentication protocol could be implemented in several ways as shown in Table 7B.

	1	2	3	4
Data Processing	Java	Web server with script language	Corba client/server model	Any language
Data Encoding	XML	HTTP header values or XML	IDL parameters or XML	Specific encoding
Used Protocol Stack	UDP/TCP IP	HTTP TCP IP	Corba TCP IP	UDP IP

Table 7B: Authentication phase implementation choices.

In the first scenario, data are encoded in the XML format and sent through a TCP (or UDP) connection. Each field is defined by a specific tag. Data are parsed by the receiver to extract and process information. This can be done, using the Java language for example. The latter language has a lot of built-in security features that are very useful to develop the protocol implementation quickly. XML is platform independent and can be sent between heterogeneous applications.

In the second scenario home agents are implemented using Web servers (e.g. Apache) with security support (SSL-aware) using a user preferences and location database (e.g. MySQL) through a script language (e.g. PHP). A user can send parameters to a web page through the script language (POST method) a receive the result through a HTTP GET method. This is a good way to get an efficient implementation.

Using the Corba paradigm, server interfaces are described in IDL. XML [XML00] (the Extensible Markup Language) is a growing standard format for structured documents and data on the Web. There are several ways to integrate XML into Corba. The initial approach is to "stringify" the XML document but it is inefficient in term of space, time and type safeness. Indeed, the later thing means

XML inner data types cannot be checked by the Corba infrastructure. A second approach would be to map XML to IDL using value types but this feature is only implemented in the latest Corba version (2.3). Moreover, the semantics of the document is not checked. The third approach is to map XML into Corba object types. The mapping between XML and IDL types can be done given the DTD. This can be done on existing CORBA implementations. This would be the best solution to implement the protocol on the Corba middleware.

Protocols such as TLS only work over TCP and TCP is prone to certain kind of attacks (TCP SYN attacks). Finally, the RPC scheme (over UDP) may be used as it is more lightweight.

We have chosen the first scenario using Java to implement the client/server and XML to format data. This choice guarantees the portability of our implementation on different plateforms.

7.1.2.2 Data coding

We now explicit the data coding of the different messages of the protocol. The data interchange format is XML (Extensible Markup Language) [XML00]. XML documents are defined by Document Type Definition (DTD). We review here the nine different messages in the protocol.

SrvAdvertisement: The service advertisement message is a broadcast message that indicates the location of the local service agent. It includes the public key of the service agent. The representation of this message is defined by the following DTD:

```
<?xml version=”1.0”?>
<!ELEMENT mobintel (header, body)>
<!ATTLIST mobintel version "1.0" #REQUIRED>
<!ELEMENT header (messagetype, time))>
<!ELEMENT messagetype
(SrvAdvertisement|AuthRequest|AuthRequestFwd|AuthReplyAck|AuthReplyAckHome|AuthReplyNa
ck|AuthReplyNackHome|AuthReplyAckFwd|AuthReplyNackFwd)>
<!ELEMENT time (#PCDATA)>
<!ATTLIST time reference CDATA “”>
<!ELEMENT body (serverinfo, authenticationinfo)>
<!ELEMENT serverinfo (ipaddress, port, serversessionpreferences)>
<!ELEMENT ipaddress (#PCDATA)>
<!ELEMENT port (#PCDATA)>
<!ELEMENT serversessionpreferences (qosecparameters, qosparameters)>
<!ELEMENT qosecparameters (#PCDATA)>
<!ELEMENT qosparameters (#PCDATA)>
<!ELEMENT authenticationinfo (publickey)>
<!ELEMENT publickey (#PCDATA)>
```

```
<!ATTLIST publickey
          algorithm CDATA #REQUIRED
          encoding "base64" #REQUIRED>
```

Figure 7H: MobInTel DTD – 1.

An example of message implementing MobInTel DTD-1 is presented below:

```
<?xml version="1.0"?>
<MOBINTEL version="1.0">
  <HEADER>
    <MessageType>SrvAdvertisement</MessageType>
    <Time reference="server">12:34:17</Time>
  </HEADER>
  <BODY>
    <ServerInfo>
      <IPaddress>mobintel.paris.fr</IPaddress>
      <port>10000</port>
      <ServerSessionPreferences>
        <QoSecParameters>DES,DESede,Blowfish,RSA<QoSecPreferences>
        <QoSParameters>networkResourceUtilization=20%<QoSParameters>
      </ServerSessionPreferences>
    </ServerInfo>
    <AuthenticationInfo>
      <PublicKey algorithm="RSA/1024"
encoding="base64">MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwb
UBfsRExY QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB</SessionKey>
    </AuthenticationInfo>
  </BODY>
</MOBINTEL>
```

AuthRequest: The authentication request message is sent by a MobInTel client to register on the infrastructure.

```
<?xml version="1.0"?>
<!ELEMENT mobintel (digitalenvelope, encryptedelement)>
<!ELEMENT digitalenvelope (#PCDATA)>
<!ATTLIST digitalenvelope
          algorithm CDATA #REQUIRED
          contentType CDATA #FIXED "text/xml"
          encoding CDATA #REQUIRED
          iv CDATA>
<!ELEMENT encryptedelement (#PCDATA)>
<!ATTLIST encryptedelement
          algorithm CDATA #REQUIRED
          contentType CDATA #FIXED "text/xml"
          encoding CDATA #REQUIRED
          iv CDATA>
```

Figure 7I: MobInTel DTD-2.

The protocol parameters follow the notations of for information about standard cipher algorithm names (see Appendix A of [JCE00]) e.g. *DES/ECB*, *RSA/1024*.

Encryption with KU_{FA} is done in *DigitalEnvelope*. Encryption with Ks_1 is done in *EncryptedElement*. Data are sent on the wire as:

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <DigitalEnvelope algorithm="RSA/1024" contentType="text/xml" encoding="base64"
iv="">GZI70PHfNDg=</DigitalEnvelope>
  <EncryptedElement algorithm="DES/ECB" contentType="text/xml" encoding="base64"
iv="">C6/dSOgZt7xLQm6PyGxfrY8dxQDjUBJwo37w3+Ll3O36/r6XIFTEniWMsIj5oUSiZLAWEmYfOf2x
b44Vfzkg5rhZHiIpAG8GH4/9E00nGx7UPZdXYb4Rzy525K/aXdXqQ/EJDIU6UnmxkNbihuvP6o6O
XeVbR4+Ofv7QHDHJBjjHwcugMFi958nicIAN3KIl1M0SXXOCEkb5+l/t8OkILq7MCjmIgIe4KHyM
ueiVA87sY+8AlFaUYtFglLlGMr8KNRqABZsP3nXx2/WCKk36bFLL9dnmXmIqPkE4FA/gjNmy7vx3
Y56C1kEmQfepV7Dvnzkwmpqhb0EsiZfLH5vmq7u6Ps2+iUnlKZCUVkImq6OE03CfrgWiIaXa53SI
G+1lV7Tl7JDmTxmeujGdKgO2JGVfQgto7QnDIs8Q+qR4ryn04kP3awL4piarYbXn7rBAhQvt3TqF
rVZ3jURsNMv79m+2Qd2+t5QLInDwEvDdP7xliAd0YqRfnJVHcnmreGerFbQAVirtdZUsWSCbAzES
Cm/wJLypB2390dhTSJzSjtw89RdZ1d6W8vmZTYop79toyD3Z0pEaeLro1FoVUF9ylMzlX15KACsM
LwmtLuQ122Ia6pXa+zGIo04ZDcUQrvkzD6/kWLOXPdRVoa/OS7y528zVhfGsBxqDZHcPThDX47aq
SyOSzbD8vwHt56801LxcG7VRSCO1o+T5mU2KKe/baC6DfiP9wcUu6NRaFVBfcpTM5V9eSgArDC8J
rS7kNdtiGuqV2vsxiKNXpFKU8gNsVal1leWK2fqBsSI5e0oZJSIWNhhB240q5RN34jaOYIRuwBog
xn9jaj90KEfE4lJRevd+eq0BlXJTHkLPgC2Wie7zTgUkI+Al8yjMVMAow7YSCr45NiZfMke2sRAr
VsRdoRu1UUgjtaPk8m5DQNeUrN0GkzlxI6ZOzp7u2/ZQ5SB7cThA/s8BkCnQzlgwMZ375fRnDLmy
CgXDL3K2pAP0Imi/qaalIIPlN4zCe05s0K6gRMhL3FLeaP0OldkSef7LUwAM+6Ef4W/vUVEbhglv
AOFJ2DEc71OctxrqldR7MYijV6RSlPIDbFUwC8RZL4mK8T86MM8m/xiyiZuNgPrC8MjkKFE4/LB0
8XEVoF/ji79AzKYROA+QP9oT0EUWuvKX+bJCm9y4mgaf1V9cwjyejmgDnMKEIEesVZtoACqdZaUy
KuZrrq+L8YZY9W/SbsI/LnUuPaMyYljusj60H5q0xVCNxK4JcKcyUm6rUJT4YSy3KQnjwGIwXQsq
rmuhnJZhUoIYOnjZe/QhsNalLJbE+DrGRUuzzyO3OiUieJOsOqCD+Bq0s3FRu/aCgjdwWvAUj21S
47+OxH3uTu1X2eMDRzFscZp0HY+5dh5k5DMe2Eaypp45T05KcsU3rAvEazF3xqluNCodHKfRPBd7
Dd7Go48wOeRQCxZvXiguOBVJIRBvstlZAmcPL2BVp8Q30CwBedgnLQSIApl568Ay5r4tzGWy2VkC
Zw8vYACWSD/c7akzoNBV2S8734f3/dCLkIzWFLtjuztN9KpRhyxIjAfW97ZZYJ2anbBYWkn+c8Sn
gdbqUsKVZoCY2pI=  </EncryptedElement>
</MOBINTEL>
```

The EncryptedElement element contains (when decrypted) parsed character data (PCDATA) following the DTD below:

```
<!ELEMENT encryptedelement (header, body)>
<!ATTLIST encryptedelement
          algorithm CDATA #REQUIRED
          contentType CDATA #FIXED "text/xml"
          encoding CDATA #REQUIRED
          iv CDATA>
<!ELEMENT header (messagetype, time))>
<!ELEMENT messagetype
(SrvAdvertisement|AuthRequest|AuthRequestFwd|AuthReplyAck|AuthReplyAckHome|AuthReplyNa
ck|AuthReplyNackHome|AuthReplyAckFwd|AuthReplyNackFwd)>
<!ELEMENT time (#PCDATA)>
<!ATTLIST time reference CDATA #IMPLIED "">
<!ELEMENT body (usersessioninfo, authenticationinfo)>
<!ELEMENT usersessioninfo (uid, usersessionpreferences, deviceprofile)>
```

```
<!ELEMENT uid (#PCDATA)>
<!ELEMENT usersessionpreferences (qosecpreferences, qospreferences)>
<!ELEMENT qosecpreferences (#PCDATA)>
<!ELEMENT qospreferences (#PCDATA)>
<!ELEMENT deviceprofile (#PCDATA)>
<!ELEMENT authenticationinfo (hashvalue+, nonce+, sessionkey+, certrequest*,
authstatus, authmessage)>
<!ELEMENT hashvalue (#PCDATA)>
<!ATTLIST hashvalue
          no CDATA #REQUIRED>
          algorithm CDATA #REQUIRED
          encoding CDATA #REQUIRED>
<!ELEMENT nonce (#PCDATA)>
<!ATTLIST nonce no CDATA>
<!ELEMENT sessionkey (#PCDATA)>
<!ATTLIST sessionkey
          no CDATA #REQUIRED>
          algorithm CDATA #REQUIRED
          encoding CDATA #REQUIRED>
<!ELEMENT certrequest (#PCDATA)>
<!ATTLIST certrequest attributes CDATA>
<!ELEMENT authstatus (#PCDATA)>
<!ELEMENT authmessage (#PCDATA)>
```

Figure 7J: MobInTel DTD-3.

An example of a message implementing *MobInTel DTD-3* is presented below:

```
  <EncryptedElement algorithm="DES/ECB" contentType="text/xml" encoding="base64"
iv="">   </EncryptedElement>
  <HEADER>
    <MessageType>AuthRequest</MessageType>
    <Time reference="user">12:34:17</Time>
  </HEADER>
  <BODY>
    <UserSessionInfo>
      <UID>alice@ottawa.ca</UID>
      <UserSessionPreferences>
        <QoSecPreferences />
        <QoSPreferences />
      </UserSessionPreferences>
      <DeviceProfile>deviceprofile</DeviceProfile>
    </UserSessionInfo>
    <AuthenticationInfo>
      <HashValue no="1" algorithm="SHA-1"
encoding="base64">DC47ECC1976E0B11056BCDB5FDEF7D268C521FA4</HashValue>
      <HashValue no="2" algorithm="SHA-1" encoding="base64">0</HashValue>
      <Nonce no="1">41956325747236519496606774855 5</Nonce>
      <Nonce no="3">0</Nonce>
      <SessionKey no="2" algorithm="DES" encoding="base64">0</SessionKey>
      <SessionKey no="3" algorithm="DES" encoding="base64">0</SessionKey>
      <CertRequest
attributes="">MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwbUBfs
RExY QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB</CertRequest>
      <AuthStatus>0</AuthStatus>
      <AuthMessage>0</AuthMessage>
    </AuthenticationInfo>
  </BODY>
```

Note that on the example, the first hash-value is processed on the following data:

```
<HEADER>
  <MessageType></MessageType>
  <Time reference="user">12:34:17</Time>
</HEADER>
<BODY>
  <UserSessionInfo>
    <UID>alice@ottawa.ca</UID>
    <HomeDomainServer>0</HomeDomainServer>
    <UserSessionPreferences>
      <QoSecPreferences />
      <QoSPreferences />
    </UserSessionPreferences>
    <DeviceProfile>deviceprofile</DeviceProfile>
  </UserSessionInfo>
  <AuthenticationInfo>
    <HashValue no="1" algorithm="SHA-1" encoding="base64"></HashValue>
    <HashValue no="2" algorithm="SHA-1" encoding="base64">0</HashValue>
    <Nonce no="1">356607969072864616863357647322</Nonce>
    <Nonce no="3">0</Nonce>
    <SessionKey no="2" algorithm="DES" encoding="base64">0</SessionKey>
    <SessionKey no="3" algorithm="DES" encoding="base64">0</SessionKey>
    <CertRequest
attributes="">MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwbUBfs
RExY QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB</CertRequest>
    <AuthStatus>0</AuthStatus>
    <AuthMessage>0</AuthMessage>
  </AuthenticationInfo>
</BODY>
```

Note that the *MessageType* field is put to null because it is a mutable field (i.e. it is changed at each end of link). Thus the hash-value cannot check integrity on that field.

Assume first that Alice connects from her home domain:

AuthReplyAckHome: the authentication request forward message is sent by the home agent in case of authentication. This message is encoded following *MobInTel DTD-2*.

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <DigitalEnvelope algorithm="mobintel/nonce" contentType="text/xml"
encoding="plaintext" iv="">400634324830450825164119745798</DigitalEnvelope>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64"
iv="S5Rirg//pNQ=">ZntkTPAFNiE6QvSuVbznC3GQkMCWhVQcF101r3pBrnZgdu6oT+h9aq24M62h+vfzPeVL
4lrdH4LS hU6OGo7guAeUXGqurnO3TDTc0+3AvYWxU6yN10aa6iIwh9IUju9gCDkFKa9ug5FgRs3/QvrJmK3f
                                                ...
CUkg06oeAG0y63N/+6Vp+KD53P5FM+NcBaZ62UwLDXDvbjQPn+nT6+JgWxxhgPvozDwz7/nKrDuf
SHhkdNhHZ8vkCRuTVd6TYdbdJ7gGCzH/2LMG2NqxbG0MT6u5bam5YxGMtSN11AJwFBbxBnkakSAW iFU=
</EncryptedElement>
</MOBINTEL>
```

The decoded version of the previous message is:

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <AuthNonce>98672162059467664612566492635 7</AuthNonce>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64" iv="S5Rirg//pNQ=">   </EncryptedElement>
  <HEADER>
    <MessageType>AuthReplyAckHome</MessageType>
    <Time reference="user">12:34:17</Time>
  </HEADER>
  <BODY>
    <UserSessionInfo>
      <UID>alice@ottawa.ca</UID>
      <HomeDomainServer>0</HomeDomainServer>
      <UserSessionPreferences>
        <QoSecPreferences />
        <QoSPreferences />
      </UserSessionPreferences>
      <DeviceProfile>deviceprofile</DeviceProfile>
    </UserSessionInfo>
    <AuthenticationInfo>
      <HashValue no="1" algorithm="SHA-1"
encoding="base64">B08522A2872A12206F490C8A6854929961F3320A</HashValue>
      <HashValue no="2" algorithm="SHA-1" encoding="base64">0</HashValue>
      <Nonce no="1">14952713190818318737566850793 1</Nonce>
      <Nonce no="3">0</Nonce>
      <SessionKey no="2" algorithm="DES" encoding="base64">0</SessionKey>
      <SessionKey no="3" algorithm="DES" encoding="base64">0</SessionKey>
      <CertRequest
attributes="">MIICCTCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UECBMC
T04x DzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
aXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
MDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
BAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
EEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
cAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy4AMCsCFDmkexNu/f6Uige5C93apnG+8D1lAhMJigvzt9Kl
nRgq9eTsqSGOvBaw</CertRequest>
      <AuthStatus>Ack</AuthStatus>
      <AuthMessage>0</AuthMessage>
    </AuthenticationInfo>
  </BODY>
```

```
</MOBINTEL>
```

If Alice is not authenticated, the *AuthReplyNackHome* message is sent. The authentication reply negative acknowledgement home message includes the following data:

```
<MOBINTEL version="1.0">
  <AuthNonce>44644151389559804314204181515 9</AuthNonce>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64"
iv="S5Rirg//pNQ=">C6/dSOgZt7xLQm6PyGxfrY8dxQDjUBJw5JWaHWCOoWr9jOFJ+PkLU9ktMAz0Y+ue5z3x
hj3zUqMf
VhflS3UHo//WNupJ3Oo1yAwC5C+tIkiqDysiSsBwG+rEf8WCp7aHd2CJlFdgab4O7HANg/866hn+
XtjETY2Ayc5zMPd+m0+yoE1KILop1DYsSNHD4PgRrEmp2qhrp9Em/K4ndKDsjP3aRH0YNBEgNixI
                                                ...
w+D4EQ5ffRb6vCNF/nas/Jp+xyW8vu1HR1FJ6uQGLfEui1JneJDPM3AbWhVQaqSAN9uBaTsAp43r
gtJfHqWW8YjZ/SDf3zcUzU/8w3isqbSIBF9eKdHiVarPnvRMZotErzESD0tTNHvuoI1V
</EncryptedElement>
</MOBINTEL>
```

The previous message contains the following data:

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <AuthNonce>10158255134386278349675647523 23</AuthNonce>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64" iv="S5Rirg//pNQ=">   </EncryptedElement>
  <HEADER>
    <MessageType>AuthReplyNackHome</MessageType>
    <Time reference="user">12:34:17</Time>
  </HEADER>
  <BODY>
    <UserSessionInfo>
      <UID>alice@ottawa.ca</UID>
      <HomeDomainServer>0</HomeDomainServer>
      <UserSessionPreferences>
        <QoSecPreferences />
        <QoSPreferences />
      </UserSessionPreferences>
      <DeviceProfile>deviceprofile</DeviceProfile>
    </UserSessionInfo>
    <AuthenticationInfo>
      <HashValue no="1" algorithm="SHA-1"
encoding="base64">BDA2AB32AA3A81B1F476EAE5D4F068AB059AE0FE</HashValue>
      <HashValue no="2" algorithm="SHA-1" encoding="base64">0</HashValue>
      <Nonce no="1">10158255134386278349675647523 23</Nonce>
      <Nonce no="3">11770349262387349574262648489 19</Nonce>
      <SessionKey no="2" algorithm="DES" encoding="base64">0</SessionKey>
      <SessionKey no="3" algorithm="DES" encoding="base64">0</SessionKey>
      <CertRequest
attributes="">MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwbUBfs
```

```
RExY QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB</CertRequest>
      <AuthStatus>Nack</AuthStatus>
      <AuthMessage>0</AuthMessage>
    </AuthenticationInfo>
    </BODY>
</MOBINTEL>
```

If Alice is not in her home domain, FA sends an authentication request forward message (*AuthRequestFwd*):

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <AuthNonce>0</AuthNonce>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64" iv="S5Rirg//pNQ=">
    <HEADER>
      <MessageType>AuthRequestForward</MessageType>
      <Time reference="user">12:34:17</Time>
    </HEADER>
    <BODY>
      <UserSessionInfo>
        <UID>alice@ottawa.ca</UID>
        <HomeDomainServer>0</HomeDomainServer>
        <UserSessionPreferences>
          <QoSecPreferences />
          <QoSPreferences />
        </UserSessionPreferences>
        <DeviceProfile>deviceprofile</DeviceProfile>
      </UserSessionInfo>
      <AuthenticationInfo>
        <HashValue no="1" algorithm="SHA-1"
encoding="base64">E6652304F467BB7CEB9D11462973627B4F6FB99B</HashValue>
        <HashValue no="2" algorithm="SHA-1" encoding="base64">0</HashValue>
        <Nonce no="1">69937888823055532092850693376 7</Nonce>
        <Nonce no="3">0</Nonce>
        <SessionKey no="2" algorithm="DES" encoding="base64">0</SessionKey>
        <SessionKey no="3" algorithm="DES" encoding="base64">0</SessionKey>
        <CertRequest
attributes="">MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwbUBfs
RExY QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB</CertRequest>
        <AuthStatus>0</AuthStatus>
        <AuthMessage>0</AuthMessage>
      </AuthenticationInfo>
    </BODY>
  </EncryptedElement>
</MOBINTEL>
```

This message is sent over a secure connection to HA.

At this point, HA can authenticate Alice. An *AuthReplayAck* message is sent:

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <AuthNonce>98219309140527546797696146473l</AuthNonce>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64" iv="S5Rirg//pNQ=">   </EncryptedElement>
  <HEADER>
    <MessageType>AuthReplyAck</MessageType>
    <Time reference="user">12:34:17</Time>
  </HEADER>
  <BODY>
    <UserSessionInfo>
      <UID>alice@ottawa.ca</UID>
      <HomeDomainServer>0</HomeDomainServer>
      <UserSessionPreferences>
        <QoSecPreferences />
        <QoSPreferences />
      </UserSessionPreferences>
      <DeviceProfile>deviceprofile</DeviceProfile>
    </UserSessionInfo>
    <AuthenticationInfo>
      <HashValue no="1" algorithm="SHA-1"
encoding="base64">479DD4B34BC8F0E611D5CB11F47846EA7ED3BB7E</HashValue>
      <HashValue no="2" algorithm="SHA-1" encoding="base64">0</HashValue>
      <Nonce no="1">624488238982740733895378112353</Nonce>
      <Nonce no="3">20992402843902685566234333584l</Nonce>
      <SessionKey no="2" algorithm="DES"
encoding="base64">MTJDOTEyQUI5NENBNkQyRjVEQzQ4N0U0MEQxQTY0NzU3QjQ5Q0MyRA==</SessionKey
>
      <SessionKey no="3" algorithm="DES" encoding="base64">0</SessionKey>
      <CertRequest
attributes="">MIICCjCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UECBMC
T04x
DzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
aXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
MDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
BAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
EEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
cAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy8AMCwCFFs36kSUneNncotlbgk8JToOWzVdAhRV9BZEaG1f
fAzu7aOuVOlmzTd6fA==</CertRequest>
      <AuthStatus>Ack</AuthStatus>
      <AuthMessage>0</AuthMessage>
    </AuthenticationInfo>
  </BODY>
</MOBINTEL>
```

FA forwards the message to Alice *AuthReplyAckFwd*:

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <AuthNonce>553856117064195320592280829591</AuthNonce>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64" iv="S5Rirg//pNQ="> </EncryptedElement>
  <HEADER>
    <MessageType>AuthReplyAck</MessageType>
    <Time reference="user">12:34:17</Time>
  </HEADER>
  <BODY>
    <UserSessionInfo>
      <UID>alice@ottawa.ca</UID>
      <HomeDomainServer>0</HomeDomainServer>
      <UserSessionPreferences>
        <QoSecPreferences />
        <QoSPreferences />
      </UserSessionPreferences>
      <DeviceProfile>deviceprofile</DeviceProfile>
    </UserSessionInfo>
    <AuthenticationInfo>
      <HashValue no="1" algorithm="SHA-1"
encoding="base64">CB47B1AC37A93BD8DC94DBD9B90C1E5AD2D50C0E</HashValue>
      <HashValue no="2" algorithm="SHA-1" encoding="base64">0</HashValue>
      <Nonce no="1">224218033972645084977438012567</Nonce>
      <Nonce no="3">525967407930546212722068658105</Nonce>
      <SessionKey no="2" algorithm="DES"
encoding="base64">NEEyOTlCMUZFRDc3NDNENTEwQTUwNTRFODU5RDExMjA2MTlEMjNEQw==</SessionKey
>
      <SessionKey no="3" algorithm="DES" encoding="base64">0</SessionKey>
      <CertRequest
attributes="">MIICCjCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UECBMC
T04x DzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
aXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
MDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
BAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
EEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
cAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy8AMCwCFHzmv9y9frjuQWBGYuN0MgMgfUxfAhQL+YfFELdu
grK7MLQnoasNiJicKg==</CertRequest>
      <AuthStatus>Ack</AuthStatus>
      <AuthMessage>0</AuthMessage>
    </AuthenticationInfo>
  </BODY>
</MOBINTEL>
```

This message is transmitted as:

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <AuthNonce>23927944397818741083360075726 3</AuthNonce>
```

```
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64"
iv="S5Rirg//pNQ=">gueq9EaMLUK9OYTbYC9UGGkg6aT5i7MDHN8u1aXZtMPD1Ww1PysNiaaUE4rylPLYtsqe
g3p7UJz3
4bHxAkseuK/FBA+SWPRyCnOTaDlOAcA1gdQ74xwd7rjrbaX01pPRh6Zx9/KCaaHoeCD45S4sARJl
tEK0V4mPwJ4wauQtKVNtdn5hBqZUEPejaiSpQeqaxJWVQ+RLgWLKi8TERI6sLJEacxzmI+u+5PQI
S7yZioCt8MlmP5WslRzaHmlgIN9KN4F7wzh5ex0wAb/dmA1VIXLTXf1heZlkzSjI0PF6hLtP6AiT
                                                ...
wMnoQGiAgNaW9kfi+au7wnPuLxHWujdzHmqdq9ZUO0JvE7dHz9ntJJD21D5y7Smjnv9unQQwS8I9
CdWo/f5XTyTioNFVS87VSGmG6KyfsXHcdO+Q/05BM9TLB/7kI7lOVd2R9KcYQlmPyDMhCafenKTO
d7If+fP119fZAzcFvohw/i6lIuFb+7R6kg==</EncryptedElement>
</MOBINTEL>
```

If Alice is not authenticated, HA sends a *AuthReplyNack* message.

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <AuthNonce>0</AuthNonce>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64" iv="S5Rirg//pNQ=">   </EncryptedElement>
  <HEADER>
    <MessageType>AuthReplyNack</MessageType>
    <Time reference="user">12:34:17</Time>
  </HEADER>
  <BODY>
    <UserSessionInfo>
      <UID>alice@ottawa.ca</UID>
      <HomeDomainServer>0</HomeDomainServer>
      <UserSessionPreferences>
        <QoSecPreferences />
        <QoSPreferences />
      </UserSessionPreferences>
      <DeviceProfile>deviceprofile</DeviceProfile>
    </UserSessionInfo>
    <AuthenticationInfo>
      <HashValue no="1" algorithm="SHA-1"
encoding="base64">E6E86A3623F259987BF1D2AFFD7A7BDD0249AA6D</HashValue>
      <HashValue no="2" algorithm="SHA-1"
encoding="base64">33CFDC47C3C9174C73DF934FCCA01528B052CF94</HashValue>
      <Nonce no="1">695594411478204104310780083711</Nonce>
      <Nonce no="3">266440091424567463308964935448</Nonce>
      <SessionKey no="2" algorithm="DES" encoding="base64">0</SessionKey>
      <SessionKey no="3" algorithm="DES" encoding="base64">0</SessionKey>
      <CertRequest
attributes="">MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwbUBfs
RExY QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB</CertRequest>
      <AuthStatus>Nack</AuthStatus>
      <AuthMessage>0</AuthMessage>
    </AuthenticationInfo>
  </BODY>
</MOBINTEL>
```

FA forwards the message to Alice a *AuthReplyNackFwd* message.

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <AuthNonce>6217409443363254615420948602O5</AuthNonce>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64" iv="S5Rirg//pNQ="> </EncryptedElement>
  <HEADER>
    <MessageType>AuthReplyNack</MessageType>
    <Time reference="user">12:34:17</Time>
  </HEADER>
  <BODY>
    <UserSessionInfo>
      <UID>alice@ottawa.ca</UID>
      <HomeDomainServer>0</HomeDomainServer>
      <UserSessionPreferences>
        <QoSecPreferences />
        <QoSPreferences />
      </UserSessionPreferences>
      <DeviceProfile>deviceprofile</DeviceProfile>
    </UserSessionInfo>
    <AuthenticationInfo>
      <HashValue no="1" algorithm="SHA-1"
encoding="base64">A805C235C7067F24E7293E16BE9D0D7143A5F3DA</HashValue>
      <HashValue no="2" algorithm="SHA-1"
encoding="base64">BC1097BD7550D5AD9E8E1F372A6E892DEF2F1CDD</HashValue>
      <Nonce no="1">6217409443363254615420948602O5</Nonce>
      <Nonce no="3">99480750273731278127229358O725</Nonce>
      <SessionKey no="2" algorithm="DES" encoding="base64">0</SessionKey>
      <SessionKey no="3" algorithm="DES" encoding="base64">0</SessionKey>
      <CertRequest
attributes="">MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwbUBfs
RExY QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB</CertRequest>
      <AuthStatus>Nack</AuthStatus>
      <AuthMessage>0</AuthMessage>
    </AuthenticationInfo>
  </BODY>
</MOBINTEL>
```

This message is transmitted as:

```
<?xml version="1.0"?>

<MOBINTEL version="1.0">
  <AuthNonce>13072046506525916214085307933O</AuthNonce>
  <EncryptedElement algorithm="RSA/CBC/PKCS5Padding" contentType="text/xml"
encoding="base64"
iv="S5Rirg//pNQ=">C6/dSOgZt7xLQm6PyGxfrY8dxQDjUBJw5JWaHWCOoWqS+vFRUcjCLeaSQb9FaiVyLdBj
9H2NNtrc I9u63WD3qD5TYsmejYH4NRFoZ/HFwmJ11yCpKGBP/DCz8PRyRoMzTKlzmh7+d0Nx0XvjvXLf9rcJ
IBycVDzHsV5o7wakSknEUyzjhdx4+YPIYKK1mDxSe6pyPB1XYuK8OGiWcguTBhJzp8Q6WIm1Azcy
                                        ...
c+wgbLQTo74eXK6gr9rJaWnGgv0FkepZ1aA2jPHBfZjHr8AAlkg/3O2pM3KocT1+4DPSx7WPpYiH
```

```
p76lRJ4pbxx84phFD+mCK07m8srsdZIj7ik=</EncryptedElement>
</MOBINTEL>
```

7.1.1.4 Underlying Transport Protocol

A service agent listens on port 10000 for requests from clients and listens on port 10001 for secure connections from other servers. TCP is used because it is reliable to transmit messages. We need the different messages to arrive safely.

7.1.1.5 Multiple device authentication

The authentication process has to be done for each device Alice uses in the foreign domain. For instance if Alice travels with a PDA and a cellular phone and arrives in her office, she might prefer to receive her phone calls on her cellular phone and short messages on her PDA. To avoid entering her password three times to be connected to the MobInTel infrastructure, a *master device* can take care of delivering security credentials locally to the other devices. Say Alice chooses the desktop computer as a *master device*. Alice specifies a short hash of the physical address of the other devices, for instance, a short hash-value of the equivalent of a IMSI in GSM (Section 4.1). These devices can connect to the *master device* through a one-way secure connection (e.g. SSL). The other devices are authenticated by the *master device* using the code written by Alice in the desktop's screen. The *master device* can thus deliver the security credentials to the other devices. Alice is authenticated with her desktop computer and she can now make secure phone calls from her cellular phone.

Note that Alice could type her password on each device. The different devices could authenticate to the master device instead of HA by that means. We prefer Alice to avoid entering her password in several terminals however for both practical and security reasons.

7.1.3 Phone call protocol

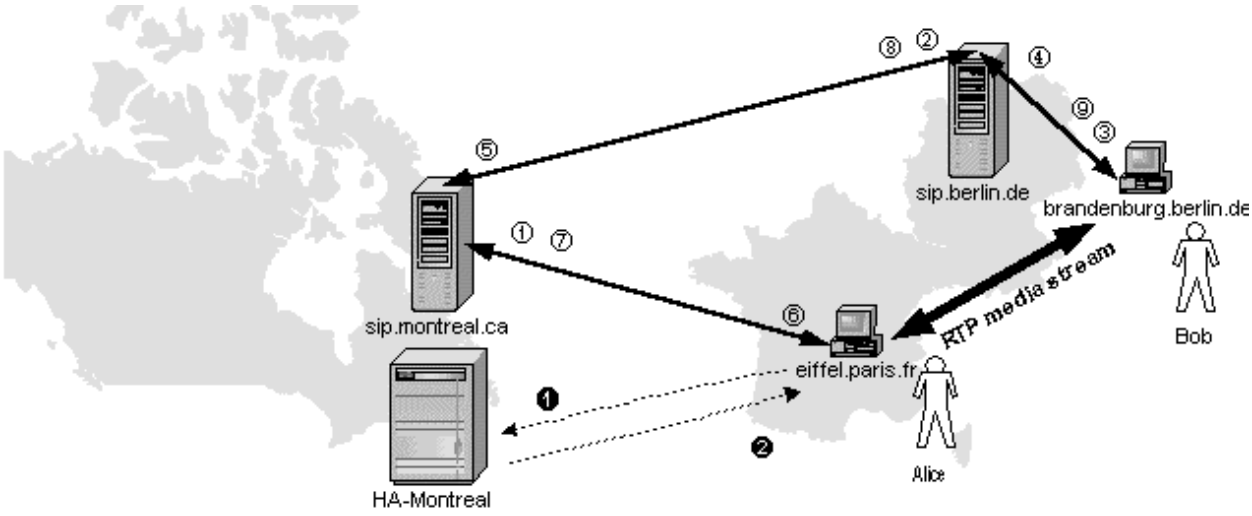

Figure 7K: Phone call protocol overview.

We assume both Alice and Bob have registered to Paris' and Berlin's SIP server respectively. Registration in a foreign domain requires two register messages. That can be done securely in two ways depending on assumptions. Assume each SIP server has a certificate, both Alice and Alice's home agent can communicate securely with a SIP server (e.g. through a SSL connection). If not, Alice and Alice's Home agent can communicate with the foreign agent who has a way to communicate securely with the SIP server located in its domain (e.g. with a previous shared key or a locally-valid certificate).

7.1.2.1 Protocol specification

Assume Alice in Paris wants to call Bob in Berlin.

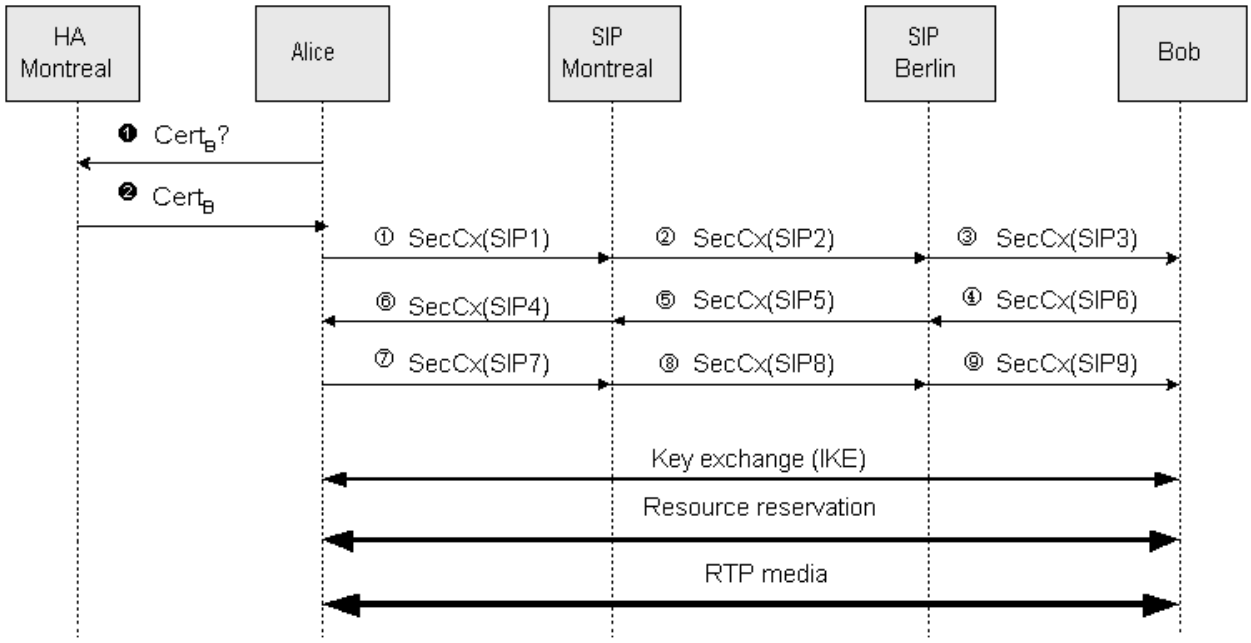

Figure 7L: Phone call message exchanges.

The phone call setup is divided is three phases: Bob's certificate request (2 exchanges), SIP INVITE (9 exchanges) and the key exchange and resource reservation phase. We study in detail these three phases in the following.

7.1.2.1.1 Bob's certificate request

First Alice needs to get Bob's certificate to encrypt end-to-end sensitive information in the SIP INVITE message. Messages ❶ is a certificate request and ❷ the reply. Bob's home agent sends to Alice Bob's certificate that corresponds to the device suitable to receive a call.

7.1.2.1.2 SIP INVITE message

① SIP1: The start-line indicates an INVITE message to bob@berlin.de. The first part of the header (the whole header but sensitive information such as *Subject*, *Content* and *Contact*) is signed by Alice. The second part of the SIP message header (non-mutable SIP fields) and the SIP message body are encrypted with Bob's public-key and signed by Alice with her private-key. The signature field of the

Authorization header line contains Alice's certificate (in a Base64 form). A Response-Key line is not needed since the response key is included in the certificate.

```
INVITE sip: bob@montreal.ca SIP/2.0$
Via: SIP/2.0/UDP eiffel.paris.fr$
To: Bob<sip:bob@montreal.ca>$
From: Alice<sip:alice-paris@paris.fr>$
Encryption: pkc version=1.0$
Authorization: pkc version="1.0", realm="alice-paris",
nonce="913082051",
signature="MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwbUBfsREx
Y QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB"$
Certificate: type=PKCS7 cert="
MIICCjCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04x
DzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
aXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
MDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
BAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
EEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
cAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy8AMCwCFHzmv9y9frjuQWBGYuN0MgMgfUxfAhQL+YfFELdu
grK7MLQnoasNiJicKg=="$
Require: org.ietf.sip.encrypt-response
Content-Length: 102$  // length of the encrypted body
Call-ID: 187602141351@eiffel.paris.fr$
Content-Type: message/sip$
Cseq:488$
$
{Subject: talk about company.$
Content-Type: application/sdp$
Contact: Sip:alice-paris@eiffel.paris.fr$
$
v=0$
o=alice 345637845 3456786578 IN IP4 128.3.4.5$
s= talk about company.$
t=0 0 $
c=IN IP$ 135.180.144.94$
m=audio 3456 RTP/AVP 0 3 4 5$}
```

② SIP2: We assume Montreal's SIP server is a proxy server. SIP2 is the message forwarded by Montreal's SIP server to Berlin's SIP server. SIP message's start-line is changed changed to *bob-berlin@berlin.de* and a "Via" line is added.

```
INVITE sip: bob-berlin@berlin.de SIP/2.0$
Via: SIP/2.0/UDP sip.montreal.ca$
Via: SIP/2.0/UDP eiffel.paris.fr$
To: Bob<sip:bob@montreal.ca>$
From: Alice<sip:alice-paris@paris.fr>$
Encryption:PKC version=1.0$
```

```
Authorization: pkc version="1.0", realm="alice-paris",
nonce="913082051",
signature="MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwbUBfsREx
Y QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB"$
Certificate: type=PKCS7 cert="
MIICCjCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04x
DzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
aXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
MDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
BAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
EEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
cAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy8AMCwCFHzmv9y9frjuQWBGYuN0MgMgfUxfAhQL+YfFELdu
grK7MLQnoasNiJicKg=="$
Content-Length: 102$  // length of the encrypted body
Call-ID: 187602141351@eiffel.paris.fr$
Content-Type: message/sip$
Cseq:488$
$
{Subject: talk about company.$
Content-Type: application/sdp$
Contact: Sip:alice-paris@eiffel.paris.fr$
$
v=0$
o=alice 345637845 3456786578 IN IP4 128.3.4.5$
s= talk about company.$
t=0 0 $
c=IN IP$ 135.180.144.94$
m=audio 3456 RTP/AVP 0 3 4 5$}
```

③ SIP3: Berlin's SIP server with SIP3 does the same process as Montreal's SIP server for SIP2. SIP3 is the message forwarded by Berlin's SIP server to Montreal's SIP server. SIP message's start-line is changed changed to *bob-berlin@brandenburg.berlin.de* and a "Via" line is added.

```
INVITE sip: bob-berlin@brandenburg.berlin.de SIP/2.0$
Via: SIP/2.0/UDP sip.berlin.ca
Via: SIP/2.0/UDP sip.montreal.ca
Via: SIP/2.0/UDP eiffel.paris.fr$
To: Bob<sip:bob@montreal.ca>$
From: Alice<sip:alice-paris@paris.fr>$
Encryption:PKC version=1.0$
Authorization: pkc version="1.0", realm="alice-paris",
nonce="913082051",
signature="MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCySptbugHAzWUJY3ALWhuSCPhVXnwbUBfsREx
Y QitBCVny4V1DcU2SAx22bH9dSM0X7NdMObF74r+Wd77QoPAtaySqFLqCeRCbFmhHgVSi+pGeCipT
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB"$
Certificate: type=PKCS7 cert="
MIICCjCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04x
DzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
aXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
MDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
BAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
EEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
cAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
```

```
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy8AMCwCFHzmv9y9frjuQWBGYuN0MgMgfUxfAhQL+YfFELdu
grK7MLQnoasNiJicKg==”$
Content-Length: 102$  // length of the encrypted body
Call-ID: 187602141351@eiffel.paris.fr$
Content-Type: message/sip
Cseq:488$
$
{Subject: talk about company.$
Content-Type: application/sdp$
Contact: Sip:alice-paris@eiffel.paris.fr$
$
v=0$
o=alice 345637845 3456786578 IN IP4 128.3.4.5$
s= talk about company.$
t=0 0 $
c=IN IP$ 135.180.144.94$
m=audio 3456 RTP/AVP 0 3 4 5$}
```

When Bob receives SIP3, Alice's message signature is first checked with Alice's public-key provided in the certificate. He decrypts the second part of the header and the body with his private key.

④ SIP4: For the reply, Bob encrypts the second part of the SIP message header and the SIP message body with Alice's public-key and signs these two parts using his private-key. The start-line indicates an "200 OK" response.

```
SIP/2.0 200 OK$
Via: SIP/2.0/UDP sip.berlin.ca
Via: SIP/2.0/UDP sip.montreal.ca
Via: SIP/2.0/UDP eiffel.paris.fr$
To: Bob<sip:bob@montreal.ca>$
From: Alice<sip:alice-paris@paris.fr>$
Encryption:PKC version=1.0$
Authorization: pkc version=”1.0”, realm=”alice-paris”,
nonce=”567823071”,
signature=”eXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE
1 FDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB”$
Certificate: type=PKCS7 cert=”
jAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
OzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
FDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
lIICCjCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04x
qAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
JEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
eXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy8AMCwCFHzmv9y9frjuQWBGYuN0MgMgfUxfAhQL+YfFELdu
grK7MjdyIasNiJiJUr==”$
Content-Length: 102$  // length of the encrypted body
Call-ID: 187602141351@eiffel.paris.fr$
Content-Type: message/sip
Cseq:488$
$
{Subject: talk about company.$
```

```
Content-Type: application/sdp$
Contact: Sip:alice-paris@eiffel.paris.fr$
$
v=0$
o=bob 4858949 4858949 IN IP4 192.1.2.3$
s= ok.it is confidential$
t=0 0 $
c=IN IP$ 135.180.144.94$
m=audio 3456 RTP/AVP 3
a=rtpmap:3 GSM/8000$}
```

⑤ SIP5: Berlin's SIP server removes a *Via* line.

```
SIP/2.0 200 OK$
Via: SIP/2.0/UDP sip.montreal.ca
Via: SIP/2.0/UDP eiffel.paris.fr$
To: Bob<sip:bob@montreal.ca>$
From: Alice<sip:alice-paris@paris.fr>$
Encryption:PKC version=1.0$
Authorization: pkc version="1.0", realm="alice-paris",
nonce="567823071",
signature="eXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE
1 FDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB"$
Certificate: type=PKCS7 cert="
jAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
OzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
FDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
lIICCjCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04x
qAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
JEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
eXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy8AMCwCFHzmv9y9frjuQWBGYuN0MgMgfUxfAhQL+YfFELdu
grK7MjdyIasNiJiJUr==”$
Content-Length: 102$  // length of the encrypted body
Call-ID: 187602141351@eiffel.paris.fr$
Content-Type: message/sip
Cseq:488$
$
{Subject: talk about company.$
Content-Type: application/sdp$
Contact: Sip:alice-paris@eiffel.paris.fr$
$
v=0$
o=bob 4858949 4858949 IN IP4 192.1.2.3$
s= ok.it is confidential$
t=0 0 $
c=IN IP$ 135.180.144.94$
m=audio 3456 RTP/AVP 3
a=rtpmap:3 GSM/8000$}
```

⑥ SIP6: Montreal's SIP server removes another *Via* line.

```
SIP/2.0 200 OK$
Via: SIP/2.0/UDP eiffel.paris.fr$
To: Bob<sip:bob@montreal.ca>$
From: Alice<sip:alice-paris@paris.fr>$
Encryption:PKC version=1.0$
Authorization: pkc version="1.0", realm="alice-paris",
nonce="567823071",
signature="eXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE
1 FDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
pueefSkz2AX8Aj+9x27tqjBsX1LtNWVLDsinEhBWN68R+iEOmf/6jGWObQIDAQAB"$
Certificate: type=PKCS7 cert="
jAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
OzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
FDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
lIICCjCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04x
qAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
JEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
eXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy8AMCwCFHzmv9y9frjuQWBGYuN0MgMgfUxfAhQL+YfFELdu
grK7MjdyIasNiJiJUr==”$
Content-Length: 102$  // length of the encrypted body
Call-ID: 187602141351@eiffel.paris.fr$
Content-Type: message/sip
Cseq:488$
$
{Subject: talk about company.$
Content-Type: application/sdp$
Contact: Sip:alice-paris@eiffel.paris.fr$
$
v=0$
o=bob 4858949 4858949 IN IP4 192.1.2.3$
s= ok.it is confidential$
t=0 0 $
c=IN IP$ 135.180.144.94$
m=audio 3456 RTP/AVP 3
a=rtpmap:3 GSM/8000$}
```

The "ACK" message is represented by messages ⑦⑧⑨. They are encrypted and signed the same way as ①②③ (without sending Alice's certificate).

7.1.2.1.3 Key exchange and resource reservation

After the call signaling phase, that is, after the SIP exchanges, both Alice and Bob have the certificate of the other party. Key exchange could be done independently of the call signaling messages or within them. Doing both call signaling and key exchange at the same time saves several message exchanges. Parameters of a Diffie-Hellman exchange could be part of the SIP multipart MIME (Multipurpose Internet Mail Extensions) message body. Since the body is encrypted end-to-

end, an authenticated Diffie-Hellman could occur avoiding the man-in-the-middle attack. This way of creating a private key seems to provide higher security.

For clarity, we presented in Figure 7H a key exchange in a separate phase. This could typically be done with IKE. That method would have the advantage of using a standard method of establishing a security association. Moreover, IKE can be done directly between Alice and Bob without having to go to the SIP servers.

Resource reservation is important for multimedia applications. This should occur after the security association is established since we should avoid reserving resources if no security association can be established.

7.2 Security analysis

7.2.1 General remarks

Some particular users may require anonymity or location privacy. In order to acquire location privacy, the user name is encrypted with the public key of the foreign agent. This does provide user location privacy. A unique alias to replace the real user identity could be used. But once the static mapping between real identity and alias is disclosed the user location will be exposed as well.

Billing is the keystone of commercial use. When the foreign agent (or a local service provider) sends billing information to the home agent, it may use a similar scheme as the one of Secure Electronic Transaction (SET). The user's dual signature related to the purchase could be sent to both FA and HA (acting as a payment gateway) so that FA doesn't know about the payment information and HA doesn't know about the service for which Alice is asking to pay. This way, purchase privacy is guaranteed.

It should also be noted that a good user password choice is essential. General weakness of knowledge-based authentication (such as one based on a password) is presented in many papers

including [DP2000]. The home agent should prevent the user from keeping a weak password that is a password that can be guessed or found easily. These passwords should be identified before they are broken by constantly running password cracking programs. In our scheme, even the hash-value (HV) is encrypted. This technique increases the computational overhead of cracking passwords as advocated in [DP2000].

7.2.2 Attacks on the protocol

7.2.2.1 Spoofing

A wrong user, say user E (Eve) may try to usurp Alice's identity. Authentication information is included in HV sent in ②. Since E does not know Alice's password, HA while calculating $(HV)_{HA}$ on its own side will find a different value from HV and E will not be authenticated as Alice. In ④, HA sends the authentication result to FA so that FA knows Alice (actually E) is not authenticated.

A fake server can try to masquerade as FA for Alice. This fake server called FZ may broadcast (in parallel to FA) its own public key KU_{FZ} and a message indicating its location. Thus Alice would send her authentication information to FZ encrypted with FZ's public key! Since Alice may have no Internet access at this point of authentication, it is virtually impossible for Alice to check a certificate at this point (and that's why we use public-keys instead of digital certificate). Note that a certificate can be sent but only the public-key information will be used and the authenticity of the certificate will not be checked. Alice can only try to authenticate with the information she is given. Is this a major security failure? The answer is no because the only information sent by Alice to FZ is { ID,KU_{Alice},N1,DP,HV}. The only things FZ could know about Alice is her ID, the public-key she is requesting to be signed and her device profile. These are not sensitive information since none of them can be exploited in order to masquerade as Alice or to do anything on behalf on Alice. Only Alice's name is revealed. We could hope that the real MobInTel server in that domain (if any) would notice that another server broadcasts information on behalf of it.

A fake server FZ will not be authenticated by HA since it has no valid "agent sever" certificate so FZ cannot answer anything valid to Alice. In ⑤, FZ cannot generate a matching pair (Ks_2 ,N2) and

send {Ks_2(ID,ACK,HV,N3),N2} properly. In ❺, FZ cannot generate HV2 and send Ks_1(ID,NACK,HV,HV2,N2).

A fake HA server will not be authenticated by FA since it does not have a valid "agent server" certificate. FA will not be able to generate HV2. Thus Alice will know that something wrong went on.

7.2.2.2 Replay attack

Replay attacks are impossible thanks to the nonces. If an attacker tries to replay ③, (ie. {KU_{FA}(Ks_1) Ks_1(ID,KU_{Alice},N1,DP,HV)}), this will be detected by HA that keeps all successful login nonce of the few last days. Since the nonce N1 includes the day date, this prevents any replays. Another way to do it would be to ask Alice to send a confirmation message to HA saying she has decrypted message ⑤ to ensure this is not a replay attack.

7.2.2.3 Denial of Service attack

Denial of service (a.g.a. "DoS") attacks are possible since each authentication request consumes both bandwidth and processing time for FA and HA. This is a general issue for any service on the Internet. This could be avoided by using adaptive firewalls or intrusion/attack detector systems.

7.2.3 Attacks on the phone call protocol

7.2.3.1 Spoofing

Spoofing is denied by the systematic use of digital certificates. Before sending a message to SIP(HA_B), Alice is authenticated with her certificate so nobody can send an *INVITE* message to Bob on behalf of Alice. The SIP servers are also authenticated by both Alice and Bob. These authentications are required for hop-by-hop encryption. Because hop-by-hop encryption is a chain of trust, if a SIP server (say SIP(FA_B)) is a rogue one, privacy of the start-line and the first part of the header is lost but end-to-end encryption still applies. Bob authenticates Alice with her certificate.

7.2.3.2 Replay attack

Messages ① and ③ cannot be replayed if the secure connection between Alice and SIP(HA_B) or BOB and SIP(FA_B) includes replay attack prevention such as SSL.

7.2.3.3 Denial of Service attack

Denial of Service ("Dos") are possibly made easier since each INVITE message requires a lot of computation. This is an inevitable tradeoff between efficiency and security. In [Section 13.4 RFC2543], the authors underline that unauthenticated 6xx messages should be ignored because they could be sent by a rogue proxy if hop-by-hop encryption and authentication is not systematically chosen.

7.3 Conclusion

The protocol defined in this chapter should be further studied to guarantee a high level of security for the end user. A public review could be the best approach to ensure that this protocol contains no flaws. We should particularly study sophisticated attacks (e.g. timing attacks) against the proposed authentication protocol.

A suitable implementation should be able to validate the protocol defined in this chapter. The criteria for the verification are both the level of security and the setup delay from the user's point of view.

Chapter 8 Implementation and analysis

An implementation of the protocol described in chapter 7 is presented here. We discuss our different implementation choices and give an overview of our implementation. We explain in detail the programming environment and the difficulties we met during implementation process. We finally discuss some test results, provide some performance analysis, and suggest alternative ways of implementing this protocol.

8.1 Implementation design

8.1.1 Implementation choices

8.1.1.1 Authentication protocol

We chose to implement the authentication client in Java because of Java portability (see Section 7.1.2.1). Moreover, Java is suitable to program GUIs.

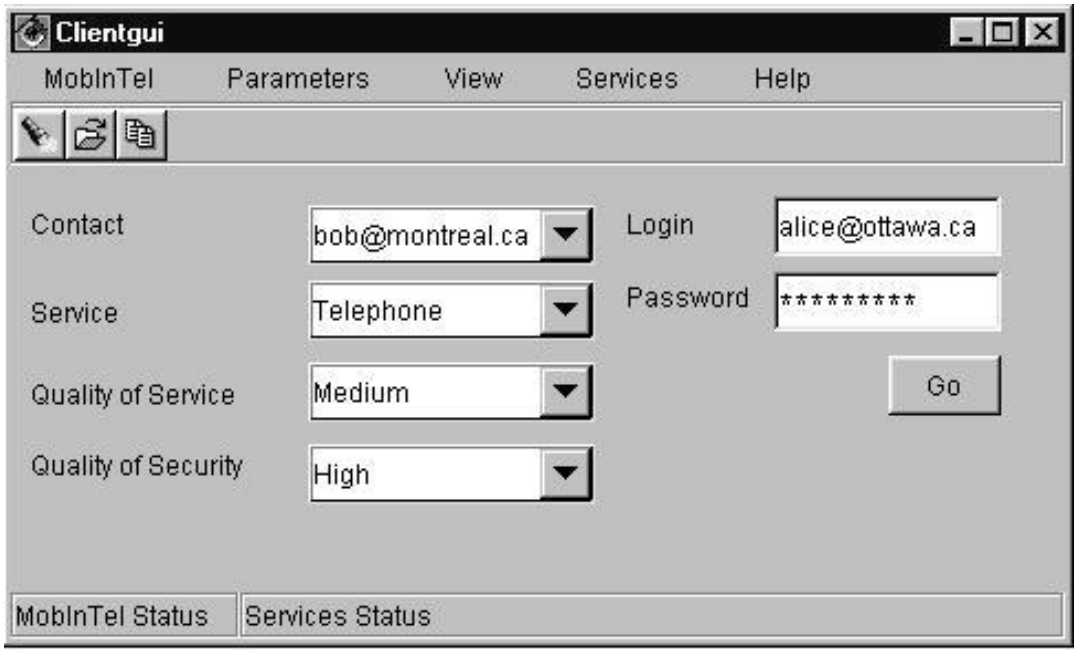

Figure 8A: MobInTel authentication module GUI.

8.1.1.2 Telephony protocol

Our main concern for implementing the phone call protocol is to reuse an existing implementation of the SIP protocol, ideally in Java. The only freely available source code of a SIP implementation we have found is in C++. Columbia University (NY, USA) provides a SIP daemon (*sipd*), a SIP client (*sipc*) and a SIP user agent (*sipua*) [SIPsoft00]. *Sipc* is a complete telephony software using SIP. Sipua allows you to make and receive SIP calls. We use *Sipc* instead of *Sipua* since we try to reuse as much code as possible. Note that a user authentication module that implements the security features added to SIP should be written in C++ so that it is easier to integrate the security module in the existing SIP implementation. SSL can encapsulate SIP to provide a secure connection.

8.1.2 Main blocks

The protocol implementation involves several constituent blocks (Figure 8B). We mainly deal here with the Java authentication module and the Service Agent to implement the proposed authentication protocol.

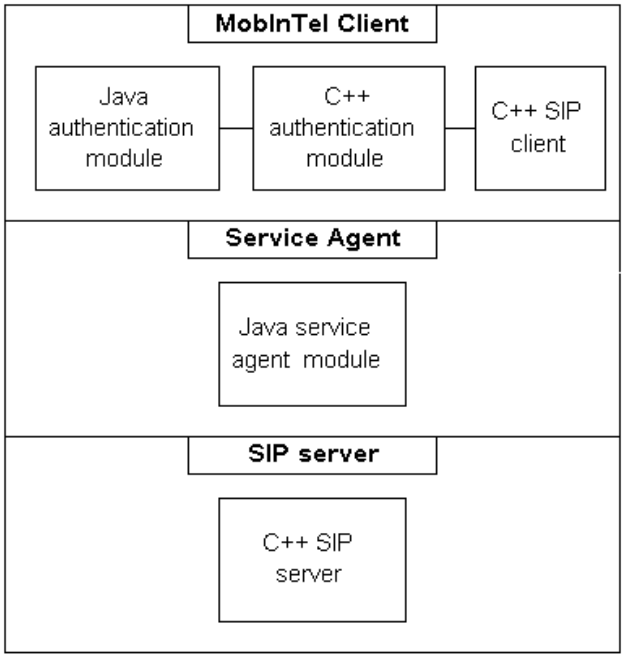

Figure 8B: MobInTel implementation main blocks.

In an implementation including both the authentication protocol and SIP, the Java client module would communicate with the C++ authentication module of the modified SIP telephony software.

Note that a proper implementation should be use the same language to avoid disclosing keys during the transfer between the authentication module and the other applications (e.g. between Java and C++). Indeed, the private key corresponding to the public key of the certificate needs to be transferred from a memory space managed by Java and a memory space managed by C++. If the key goes through the memory space that is not secure, the transfer of key becomes a security flaw. In that way our implementation is improper.

8.1.3 Protocols and languages

8.1.3.1 Implementation of the authentication phase

8.1.3.1.1 MobInTel APIs and packages

SUN's Java development kit (JDK version 1.3) includes several packages that provide security APIs.

- Java.security part of the JS2E (Java Standard Entreprise Edition) provides interfaces for certificate management and signature (RSA and DSA key generation).
- package *java.security.cert* provides X.509 certificate manipulation and certificate (parsing and generation of certificates but cannot sign a certificate.

We use two extensions of the standard development kit:

- JCE (Java Cryptographic Extension) provides javax.crypto cryptographic tools such for encryption and key agreement.
- JSSE (Java Secure Socket Extension) provides a full implementation of SSL.
- package *sun.security.x509* provides certificate signing (Class X509CertImpl method sign())

XML parsers are not part of the standard JDK. We use several other packages:

- JAXP global project provides two awaited future Java extensions (javax.xml.parsers, javax.xml.transform), three packages org.xml.sax, org.w3c.dom and org.apache.crimson for XML tree manipulation and package org.apache.xalan for XSL (Extensible Stylesheet Language) transforming. The latter package is not used in our implementation.
- X project provides org.xml.sax, org.w3c.dom and com.sun.xml packages. The com.sun.xml package provides an XML parser and utilities to go through an XML tree.

Our MobInTel authentication phase implementation consists of four packages:

- *ca.uottawa.site.dsrg.mobintel.client*. This package with the *Client* class is used by the client to logon and to format requests.
- *ca.uottawa.site.dsrg.mobintel.common*. This package contains the *User* class that contains all the parameters related to a particular user (such as ID and keys). This package is needed by both the client and the service agent.
- *ca.uottawa.site.dsrg.mobintel.utils*. This package contains several utility tools such as XML parsing (*XmlTools* class), key and certificate generation (*MobintelCertificateSigner* class) and debugging information (*Debug* class).
- *ca.uottawa.site.dsrg.mobintel.server*. This package contains the classes for the socket server (*ServerAgent* class), the service agent (*ServiceAgent* class), the authentication agent (*AuthAgent* class) and the particular classes for the home agent (*HA* class) and foreign agent (*FA* class).

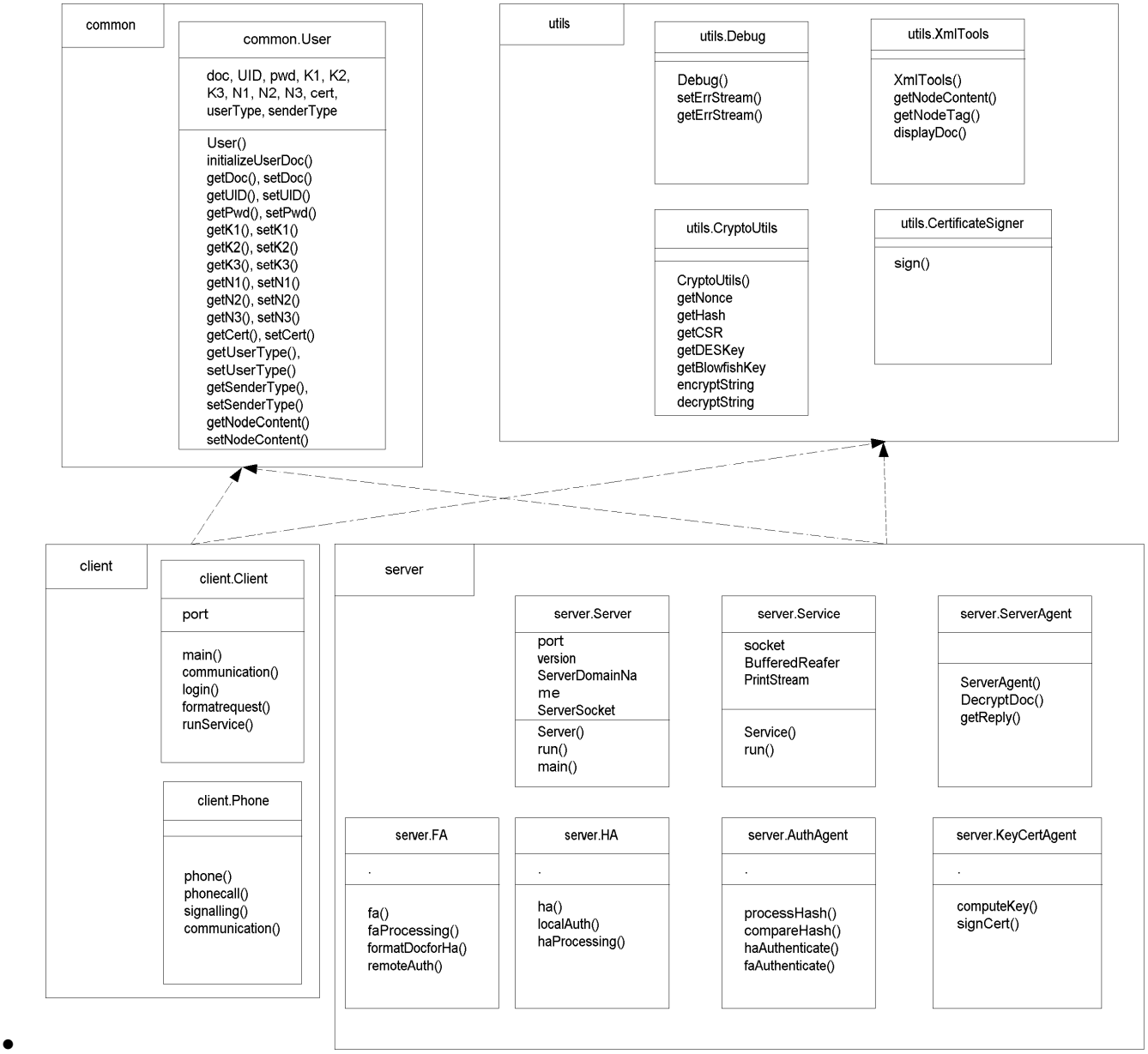

Figure 8C: Class diagram with package dependence.

MobInTel client uses *client*, *common* and *utils* packages. MobInTel server uses *server*, *common* and *utils* packages.

8.1.3.2 APIs and materials

8.1.3.1.3 Key material storage

Java has the built-in keystore facility. Key store is a collection of keys and certificates that contain a list of aliases. There is a key pair and a series of information per alias. It manages two types of entries:

- Key Entry: this type of keystore entry holds very sensitive cryptographic key information, which is stored in a protected format to prevent unauthorized access. Typically, a key stored in this type of entry is a secret key, or a private key accompanied by the certificate chain for the corresponding public key.
- Trusted Certificate Entry: This type of entry contains a single public key certificate belonging to another party. It is called a trusted certificate because the keystore owner trusts that the public key in the certificate indeed belongs to the identity identified by the subject (owner) of the certificate.

A keystore uses X.500 distinguished names, with components Common Name (CN), Organizational Unit (OU), Organization (O), Location (L), State (ST), and Country (C) to identify key owners and certificate issuers.

Package *sun.security.x509* permits certificate manipulation. No API documentation is provided. The documentation (generated with *javadoc* from the sources) shows that Class X509CertInfo permits the change of information in a certificate.

Outside keystores, it is better to store keys as byte arrays than as String because byte arrays are more often collected by the Java garabge collector when it is not used anymore.

8.1.3.1.4 Cryptographic tools

The X.509 standard should be used for every certificates. PGP certificates may be accepted by some implementations due to their large public acceptance. Symmetric encryption algorithms must support at least DES (CBC mode) and should support also 3-DES, Blowfish, IDEA and AES. Asymmetric encryption algorithms must support RSA (1024 bits) and should support ECC that requires only 160 bits for an equivalent security. MD5 and SHA-1 one-way hash functions should be

used to compute digests. Algorithm negotiation is out of scope of this document. It is provided by ISAKMP/IKE.

RTP encryption supports DES and may support other encryption algorithms. New implementations should support AES ("Rijndael") that is an algorithm that supports variable key lengths. The key length could be adapted to the processing power of the device in the IKE process.

8.1.4 Difficulties on the implementation stage

8.1.4.1 Secure Random

We present below three methods to generate keys using *java.security.KeyPairGenerator*. This object has several methods to be initialised. One of them uses a secure random generator: initialize(AlgorithmParameterSpec params, SecureRandom random). This method takes about 30 seconds to give the result whereas the default initialisation takes only 5 seconds. We discuss the drawbacks of this approach below.

8.4.1.2 Base64 conversion

Keys cannot be transmitted as is within XML. XML supports (by default) only ASCII characters. A key represented as an array of bytes may correspond to any type of character especially those, which are not in the ASCII set. Thus a conversion with Base64 [Stallings99] is needed. Keys are transmitted as Base64. XML considers a '*\n*' character as a space. We have to convert by hand the Base64 output to a transmittable form that is by replacing '*\n*' with a space character. For instance, consider Element *CertRequest* below:

```
<CertRequest attributes=""
algorithm="">MIICCzCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkG
A1UECBMCT04x
DzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
aXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
MDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
BAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
EEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
cAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAzAAMC0CFQCRvwMHFb31JL7NodTCY7YcEsEyaQIUTk9s9Nal
nTKumZwQ0CfB4eWbuAo=</CertRequest>
```

This element will be sent as:

```
<CertRequest attributes=""
algorithm="">MIICCzCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkG
A1UECBMCT04x zANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVB
AsTE1NlY3Vy aXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwH
hcNMDEwMTE1 MDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT
04xDzANBgNV BAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxG
TAXBgNVBAMT EEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm
1u6AcDNZQlj cAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3v
tCg8C1rJKoU uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3r
xH6IQ6Z//qM ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAzAAMC0CFQCRvwMHFb31JL7NodTCY7YcEsEya
QIUTk9s9Nal nTKumZwQ0CfB4eWbuAo=</CertRequest>
```

The invert operation of replacing '\n' character by the space character is done when receiving the message.

8.2 Tests scenarios

In this section, we describe the network architecture used to test our protocol and then evaluate it using scenarios.

8.2.1 Test architecture

Ideally, eight computers in different domains would be used to test the implementation: two for the home agents, two for the foreign agents, two for the SIP servers and two for Alice and Bob. We propose here a testing architecture using three computers.

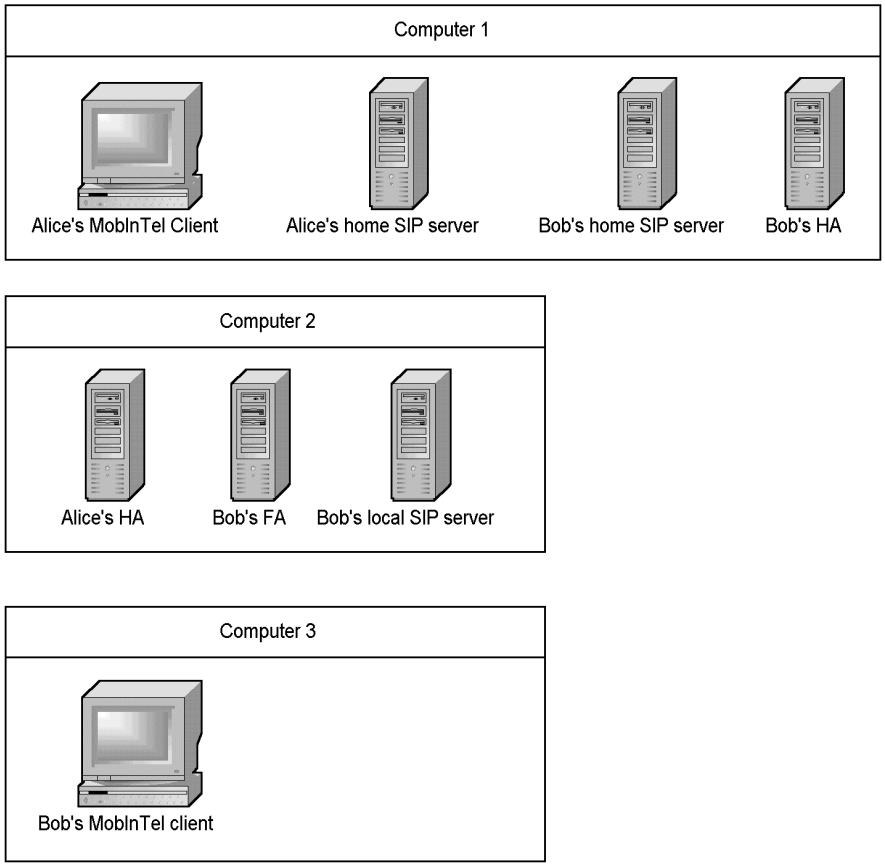

Figure 8D: Testing architecture.

The tests have been conducted on Intel-450 MHz computers with 128 Mega bytes of RAM (Read Access Memory).

8.2.2 Test scenario

8.2.2.1 Authentication test parameters

Several parameters can be changed in the tests:

- User connecting from its home or foreign domain;
- Type of authentication: Ack or Nack;
- Security Level (with or without certificate);
- Cipher algorithm, cipher mode, key length.

We could define several levels of encryption:

- Low security level: DES/ECB for symmetric encryption, RSA/512 for asymmetric encryption, MD5 as a cryptographic one-way hash function and MD5 with RSA for certificate signing. A mathematical (non secure) random generator is used for number generation.
- Medium security level: DES/CBC for symmetric encryption, RSA/1024 for asymmetric encryption, SHA1 and SHA1 with RSA for certificate signing. A secure pseudo random number generator is used to pick up nonces.
- High security level: AES/256 for symmetric encryption, RSA/2048 for asymmetric encryption, SHA1 as a cryptographic one-way hash function and SHA1 with RSA for certificate signing. A secure pseudo random number generator is used to pick up nonces.

8.2.2.3 Test results

The cipher used in these tests are the ones defined above in the medium security level. The network delay is negligible (Intranet with machines side by side)

8.2.2.3.1 Functional test

In this scenario, Alice authenticates to the MobInTel infrastructure from a foreign domain. She is asked a login and a password to use a certain Service (telephone, email…).

At the end of the authentication process, Alice gets three security credentials: Ks2, Ks3 and a certificate:

- Session key number 2:


```
<SessionKey no="2" algorithm="DES"
encoding="base64">NEEyOTlCMUZFRDc3NDNENTEwQTUwNTRFODU5RDExMjA2MTlEMjNEQw==</SessionKey
>
```


- Session key number 3 calculated from nonce 3:


```
<Nonce no="3">525967407930546212722068658105</Nonce>
```


- Certificate:


```
      <CertRequest algorithm="SHA1withRSA"
encoding="base64">MIICCjCCAcgCBDpiV2YwCwYHKoZIzjgEAwUAMIGDMQswCQYDVQQGEwJDQTELMAkGA1UE
CBMCT04x DzANBgNVBAcTBk90dGF3YTEXMBUGA1UEChMOY2l0eSBvZiBPdHRhd2ExHDAaBgNVBAsTE1NlY3Vy
aXR5IGRlcGFydG1lbnQxHzAdBgNVBAMTFlNlY3VyaXR5IEFkbWluaXN0cmF0b3IwHhcNMDEwMTE1
MDE1MDMwWhcNMDEwNDE1MDE1MDMwWjBrMQswCQYDVQQGEwJDQTELMAkGA1UECBMCT04xDzANBgNV
BAcTBk90dGF3YTEUMBIGA1UEChMLVSBvZiBPdHRhd2ExDTALBgNVBAsTBFNJVEUxGTAXBgNVBAMT
EEFsaWNlIFdvbmRlcmxhbmQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALJKm1u6AcDNZQlj
cAtaG5II+FVefBtQF+xETFhCK0EJWfLhXUNxTZIDHbZsf11IzRfs10w5sXviv5Z3vtCg8C1rJKoU
uoJ5EJsWaEeBVKL6kZ4KKlOm5559KTPYBfwCP73Hbu2qMGxfUu01ZUsOyKcSEFY3rxH6IQ6Z//qM
ZY5tAgMBAAEwCwYHKoZIzjgEAwUAAy8AMCwCFHzmv9y9frjuQWBGYuN0MgMgfUxfAhQL+YfFELdu
grK7MLQnoasNiJicKg==</CertRequest>
```

This certificate can be printed:

```
[
[
  Version: V1
  Subject: CN=Alice Wonderland, OU=SITE, O=U of Ottawa, L=Ottawa, ST=ON, C=CA
  Signature Algorithm: SHA1withDSA, OID = 1.2.840.10040.4.3

  Key:  com.sun.rsajca.JSA_RSAPublicKey@1c8a71
  Validity: [From: Sat Jan 14 20:50:30 EST 2001,
              To: Sat Jan 14 22:50:30 EST 2001]
  Issuer:  CN=Security  Administrator,  OU=Security  department,  O=city  of  Ottawa,
L=Ottawa, ST=ON, C=CA
  SerialNumber: [    3a625766 ]

]
  Algorithm: [SHA1withDSA]
  Signature:
0000: 30 2E 02 15 00 88 4D 4D   BB 5B DA 99 FE DE DE A2  0.....MM.[......
0010: D3 37 4E 18 2B 39 D3 FA   32 02 15 00 8D 6D 29 8B  .7N.+9..2....m).
0020: 39 36 AA 29 FF 63 FC 62   62 18 9F A5 13 41 77 A9  96.).c.bb....Aw.

]
```

Thus, Alice gets all the security tokens needed to communicate securely with FA, HA and any other peer.

8.2.2.3.2 Authentication delay

In this scenario, Alice authenticates to the MobInTel infrastructure from an outside domain. We measure the delay for an authentication session of high security level (secure random source for RSA generation of keys and for DES generation of key). The delay is from the moment that Alice knows the location of the MobInTel server (receives a broadcast message) and thus is able to send a request. Delays have been measured by the implementation using timestamps (*Class java.util.Date*).
In the scenario of Alice connecting from Paris and being authenticated, the measured delay is 22s:

- 21 s at the client side to prepare the request (including the nonces and keys generation).
- 565 ms for FA processing;
- 256 ms for HA processing.

Generation of keys and encryption are the main source delay of the protocol. The client generates several pairs of keys. Key generation is also done on the server side as well but using a hash function.

The delay is about 10 s when the default key generation method is used (not a high secure pseudo-random generator). That means the keys generated may be not secure against sophisticated attacks.

- 9s for encryption at the client side
- 702 ms for HA and FA processing.

This delay is more reasonable but it is still high for the client.

We measured the delay of several operations with a system with the following properties: a Pentium II 400MHz computer with 128 MBytes of RAM using Red Hat Linux 6.2, Sun JDK v1.3 and JCE 1.2.1 software packages. The encryption/ decryption operations are processed on the data of the protocol. The values correspond to the median of a series of twenty experiments. The confidence (for 95% values included) intervals are indicated inside the parentheses.

- CSR generation: 3449 ms (± 16 ms)
- PRNG (pseudo random number generator) seed (10B): 5712 ms (± 30 ms)
- Random number generation: <1 ms (± 1 ms)
- MD5 hashing: 3 ms (± 1 ms)
- SHA-1 hashing: <1 ms (± 1 ms)
- Sym key generation: 11478 ms (± 85 ms)
- DES encryption: 3 ms (± 1 ms)
- DES decryption 3 ms (± 1 ms)
- Blowfish56 encryption: 1 ms (± 1 ms)
- Blowfish56 decryption: 1 ms (± 1 ms)
- 3-DES-112 encryption: 3 ms (± 1 ms)
- 3-DES-112 decryption: 3 ms (± 1 ms)
- RSA128 encryption (of 56 bits): 62 ms (± 2 ms)
- RSA128 decryption (of 56 bits): 33 ms (± 2 ms)
- RSA128 encryption (of 112 bits): 176 ms (± 4 ms)
- RSA128 decryption (of 112 bits): 180 ms (± 4 ms)

- RSA1024 encryption (of 112 bits): 62374 ms (± 2050 ms)
- RSA1024 decryption (of 112 bits): 28137 ms (± 1276 ms)
- Certificate signing: 152 ms (± 5 ms)

Discussion:

Key generation is an important problem on the client side. We could use less secure mechanisms to generate a key. Client keys could be generated with a nonce and a digest (as the server does). This would take an estimated time of 1 ms. This is far less than a few seconds. As observed in the above method, we cannot see any flaw in this method for key generation, we need to determine why this method is hundred times faster than the built-in secure key generation implemented in Java. The slowness of Java would be the only explanation. If there is a security level difference, how could it be quantified? If the client key generation method is not secure, that means we should use another function *f* to generate Ks_2 and Ks_3, not simply based on a hash function. Function *f* could be chosen as a function of the security level decided by the user. One should determine whether or not this increased security level is needed for most applications.

8.2.2.3.2 Bandwidth utilization study

Table 8C presents the average amount of data sent during a series of exchange. They have been calculated with the *ifconfig* command that reports the number of packets sent by the Ethernet card. Knowing the MTU (Maximum Transfer Unit) on the considered network interface we can find an upper bound of the transferred amount of data. The number of packets indicates the volume of processing time for packet switching.

Type of situation	Average message length sent over the network
Alice in Paris – Ack	72 kBytes
Alice in Paris – Nack	64 kBytes
Alice in Ottawa – Ack	40 kBytes
Alice in Ottawa –Nack	32 kBytes

Table 8A: Bandwidth consumption comparison

As expected, local authentication consumes less bandwidth than remote authentication. When Alice is not authenticated, cryptographic materials such as digital certificates and session keys are not sent. That explains the 8 kBytes difference between a positive authentication message exchange (*Ack*) and a negative one (*Nack*).

Note that the number of packets measured does not take into account the packets for DNS lookup and ARP requests (since the tests were on a local Ethernet network).

Chapter 9 Conclusions

9.1 Contributions

This thesis contributes to the final goal of convergence of telecommunications services: the integration of all existing and growing new services into a unified framework (such as a web-based personalized communications portal) including data information exchanges and most person-to-person communications. The ability to deliver disparate data and streaming multimedia demands the convergence of many technologies. In our work, we combine full mobility, telephony and security on the Internet in the context of a new proposed infrastructure (MobInTel).

The contributions of this thesis are the following:

1. Discussion of the MobInTel trust and security requirements: In chapter 6 we underlined the advantages and shortcomings of existing and proposed telephony and mobility architectures. We showed that no current infrastructure could be adapted to meet MobInTel security requirements.
2. Our major contribution is the proposal of a new protocol that meets the specific security requirements of the MobInTel infrastructure. We presented different options to meet MobInTel requirements and we justified our choices. We specified in detail the proposed authentication protocol in chapter 7. We explained the different coding choices. The XML message coding was fully detailed.
3. Application of the proposed protocol in the context of IP-telephony: We provided secure IP-telephony with SIP using the MobInTel infrastructure. We analyzed the security threats against this infrastructure and explained how they were addressed.
4. Implementation: We implemented the proposed authentication protocol and discussed our implementation choices. We presented our design and the implementation environment. We gave the results of our protocol functional and performance testing. We discussed issues that arose during testing and we suggested some improvements.

9.2 Further work

Wireless technologies and other pervasive computing will change the way we communicate in the future. Further work could be done to validate our protocol specifications on mobile terminals especially on cellular phones and PDAs. It would be very interesting to study performance such as authentication delay and bandwidth consumption with limited computation and limited memory devices.

Further work could be undertaken on the protocol itself. We could first study the resistance of our protocol to sophisticated attacks. We could make use of formal logic such as BAN-logic to validate our protocol even if these formal methods may contain some flaws. Indeed these methods could validate protocols that contain flaws. They could also invalidate protocols that contain no flaws. These methods could however reduce possible flaws. Moreover, an automated key management protocol should be proposed to ensure the periodic renewal of session keys. The integration of IKE to negotiate security associations would be particularly interesting. AbdelAziz proposed in [BAA00] such an integration of IKE.

The security infrastructure is not a final goal in itself. We should now develop applications that rely on that infrastructure to provide multimedia applications that the next generation network will be able to carry. Thus, further work in bandwidth brokerage (QoS mechanisms) for resource reservation, media streaming (authentication, quality of service and multicasting) is expected. Research is currently very active in these domains.

While bandwidth over-provisioning continues, most industry experts agree that QoS mechanisms such as bandwidth brokerage are needed to address the needs of converging networks. Because not all QoS mechanisms are created alike, selecting the right QoS mechanism for the network could affect results significantly. QoS mechanisms have been slow to deploy because of the lack of adequate QoS management, accounting, and control. They manipulate router/switch queues so that when congestion occurs, priority "VIP" traffic is serviced quickly, while less important traffic experiences delays and drops

There exists a variety of approaches. Some router/switches are capable of setting filters to classify traffic and map it to specific queues. The Differentiated Services (DiffServ) approach separates the classification and queuing functions. The ReSerVation Protocol (RSVP) is a set-up protocol providing a receiver-based, guaranteed, end-to-end QoS pipe. In the integrated services (IntServ) approach, RSVP-enabled applications dynamically request and reserve network resources necessary to meet their specific QoS requirements. Another approach to QoS can be to preset tunnels across the network and provision QoS to each such tunnel. This concept has existed for quite some time in Layer 2 protocols such as ATM and frame relay. Recently, Multi-Protocol Label Switching (MPLS) has been picking up steam and could become the IP equivalent of ATM's permanent virtual circuits. Today much research is currently undertaken in these fields.

As the popularity of streaming media grows, that becomes one of the fields in which research is highly growing. The ability to stream content from the edge of the network can help to ensure that users will receive unbroken high-quality audio and video. Because the network requirements for streaming media and HTTP differ greatly, network managers need to understand how caching technology can be deployed to keep streaming service at acceptable levels.

When available, the future unified interface for all communication services will move beyond current delivery platforms. The network and application levels technologies including security details will be transparent to the user. It will be the basis of the new economy and, for that reason, it must provide freedom, integration and security.

Bibliography

[RFC .]: Reference to a Request For Comments (RFC) document issued by the IETF (Internet Engineering Task Force).

Chapter 1: Introduction

[PPYSSMS99] C. A. Polyzois, K. H. Purdy, P. Yang, D. Shrader, H. Sinnreich, F. Ménard and H. Schulzrinne: *From POTS to PANS -- A Commentary on the Evolution to Internet Telephony*, IEEE Network, vol. 13, no. 3, pp. 58--64, May/June 1999.

[HEB-00-1] X. HE, K. El-Khatib, G. v. Bochmann: *Quality of service negociation based on device capabilities and user preferences*. Technical report. U. of Ottawa, Canada. May 2000.

[HEB00-2] He, El-Khatib, Bochmann. *A communication services infrastructure including home directory agents* May 2000. U. of Ottawa.

[SR99] H. Schulzrinne and J. Rosenberg. *Internet Telephony: architecture and protocols – an IETF perspective. Computer Networks*, Vol. 31, No. 3. February 11, 1999.

Chapter 2: Security Fundamentals

[HA94] N. Haller, R. Atkinson. *On Internet Authentication*. RFC 1704, 1994.

[ITU91] *X.800, Security Architecture for Open Systems Interconnection for CCITT Applications*. ITU. 1991.

[Neum93] B. C. Neuman. *Proxy-Based Authorization and Accounting for Distributed Systems*. Proceedings of the 13th International Conference on Distributed Computing System, 1993.

[Amor94] E. Amoroso. *Fundamentals of Computer Security Technology*. Prentice-Hall, 1994.

[Schn96] B. Schneier. *Applied Cryptography*. Second edition, Wiley, 1996.

[AES00] NIST . Advanced Encryption Standard. http://csrc.nist.gov/encryption/aes/ 2000.

[Clipper94] NIST. Clipper chip technology. http://csrc.nist.gov/keyrecovery/clip.txt 1994

[Stallings99] W. Stallings. *Cryptography and network security*. Prentice Hall. 1998.

[KBC97] H. Krawczyk M. Bellare R. Canetti. HMAC: *Keyed-Hashing for Message Authentication.* February 1997 RFC2104.

[BKR94] M. Bellare, J. Kilian, P. Rogaway. *The Security of the Cipher Block Chaining Message Authentication Code*. Crypto 94 procedeeings. LNCS 839. 1994.

[BGR95] M. Bellare, J. Kilian, P. Rogaway. *XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions*. Crypto 95. LNCS 963. 1995.

Chapter 3: Security Applications

[X509-93] CCITT recommendation. *X509 – The Directory Authentication Framework – version 3*. 1995.

[Stallings99] W. Stallings. *Cryptography and network security*. Prentice Hall. 1998.

[RFC1510] J. Kohl C. Neuman. *The Kerberos Network Authentication Service (V5)*. September 1993.

[RFC2138, RFC2139] C. Rigney, A. Rubens, W. Simpson, S. Willens. *Remote Authentication Dial In User Service.* April 1997

[RFC2748] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, A. Sastry. *The COPS (Common Open Policy Service) Protocol.* January 2000.

[RFC1492] C. Finseth. *An Access Control Protocol, Sometimes Called TACACS*. July 1993.

[RFC1334] B. Lloyd W. Simpson. *PPP Authentication Protocols*. October 1992.

[RFC1994] W. Simpson. *PPP Challenge Handshake Authentication Protocol (CHAP)*. August 1996.

[RFC1938] N. Haller, C. Metz. *A One-Time Password System*. May 1996.

[RFC2284] L. Blunk, J. Vollbrecht. *PPP Extensible Authentication Protocol (EAP)*. March 1998.

[SSH00] T. Ylonen T. Kivinen M. Saarinen T. Rinne S. Lehtinen. *SSH Connection Protocol.* IETF draft. November 2000

[SOM98] A.F. Syukri, E. Okamoto and M. Mambo: *User Identification System Using Signature Written with mouse* - LNCS Vol. 1438, p403. July 98.

[RFC2408] D. Maughan M. Schertler. M. Schneider, J. Turner. *Internet Security Association and Key Management Protocol.* November 1998

[RFC2409] D. Harkins, D. Carrel. *The Internet Key Exchange (IKE)*. November 1998.

[RFC2412] H. Orman. *The OAKLEY Key Determination Protocol*. November 1998.

[RFC2401] S. Kent, R. Atkinson. *Security Architecture for the Internet Protocol.* November 1998.

[RFC1826-1829] R. Atkinson, P. Karn P. Metzger W. Simpson. *IP Authentication Header. IP Encapsulating Security Payload (ESP). IP Authentication using Keyed MD5. The ESP DES-CBC Transform.* August 1995

[RFC2402-2406] R. Atkinson, S. Kent, C. Madson, R. Glenn, N. Doraswamy. *The Use of HMAC-MD5-96 within ESP and AH. IP Authentication Header. The ESP DES-CBC Cipher Algorithm With Explicit IV. The Use of HMAC-SHA-1-96 within ESP and AH. IP Encapsulating Security Payload (ESP).* November 1998

[SSL98] Netscape. Secure Socket Layer. http://home.netscape.com/eng/ssl3/ssl-toc.html

[RFC2246] T. Dierks, C. Allen. The TLS Protocol Version 1.0. January 1999.

[SET00] SET specifications. http://www.setco.org/set_specifications.html.

Chapter 4: Telephony

[GSMSEC] *GSM Security and Encryption*. http://www.security.ece.orst.edu/seminars/sunar1/

[MoPa92] Michel Mouly and Marie-Bernadette Pautet. *The GSM System for Mobile Communications*. Published by the authors, 1992

[PGPfone] *PGPfone - Pretty Good Privacy Phone* http://web.mit.edu/network/pgpfone/

[RFC2543] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg. *SIP: Session Initiation Protocol.* March 1999

[RFC2782] A. Gulbrandsen, P. Vixie. *A DNS RR for specifying the location of services (DNS SRV).* February 2000

[RFC1889] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications* . January 1996.

[RFC2617] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart. *HTTP Authentication: Basic and Digest Access Authentication*. June 1999

Chapter 5: Mobility

[Katz, R., "Adaptation and Mobility in Wireless Information Systems," IEEE Personal Communications Magazine, Vol. 1, No. 1, pp. 6-17, 1994.]

[RFC2002] C. Perkins. IP Mobility Support. October 1996.

[SL99] Inernet Mobility Support optimized dor client access and its scalable authentication framework. Lecture notes in computer science 1748, pp220-229, Dec 1999.

[ABOBA99-2] Aboba, B., "Certificate-based Roaming", Internet Draft (work in progress), draft-ietf-roamops-cert-01.txt, April 1999.

[ABOBA99-1] Roaming Support in Mobile IP. draft-ietf-roamops-mobileip-02.txt. Bernard Aboba. Microsoft. 25 June 1999

[Henning Schulzrinne *European Workshop on Interactive Distributed Multimedia Systems and ervices*, (Berlin, Germany), Mar. 1996.]

[HEB00] He, El-Khatib, Bochmann. *A communication services infrastructure including home directory agents* May 2000. U. of Ottawa.

[RDF00] W3C. Resource Description Framework (RDF). http://www.w3.org/RDF/

[ITU-ICA] ITU, study group 13. *Information Communication Architecture*. September 1999.

[HEB00-2] He, El-Khatib, Bochmann. *Quality of service negotiation based on device capabilities and user preferences*. May 2000, U. of Ottawa

Chapter 6: Proposed architecture

[RRAS1] *A Survey of Requirements and Standardization Efforts for IP-Telephony-Security*.
Christoph Rensing, Utz Roedig, Ralf Ackermann, Ralf Steinmetz. Darmstadt University of Technology,Germany. Proceedings of the Workshop "Sicherheit in Mediendaten", September 2000

[RAS00] Utz Roedig, Ralf Ackermann, Ralf Steinmetz: *Evaluating and Improving Firewalls for IP-TelephonyEnvironments* In Proceedings of the 1st IP-Telephony Workshop (IPTel2000), ISSN 1435-2702, GMD-Forschungszentrum Informationstechnik GmbH, April 2000

[RARR99] U. Roedig, R. Ackermann, C. Rensing and R. Steinmetz: "DDFA Concept" Technical Report. KOM-TR-1999-04, KOM, December 1999

[Biggs00] B. Biggs: "A SIP Application Level Gateway for Network Address Translation" Internet Draft, draft-biggs-sip-nat-00.txt.

[RDS00] J. Rosenberg, D. Drew, H. Schulzrinne: "Getting SIP through Firewalls and NATs", Internet Draft, draft-rosenberg-sip-firewalls-00.txt.

[Sufatrio, K.-Y. Lam: *Internet Mobility Support optimized for Client access and its scalable authentication framework*. MDA'99. LNCS 1748. Dec. 1999.].

[C. Rensing, U. Roedig, R. Ackermann, R. Steinmetz: *A Survey of Requirements and Standardization Efforts for IP-Telephony-Security*. Darmstadt University of Technology, Germany. Proceedings of the Workshop "Sicherheit in Mediendaten", September 2000.].

[Netmeeting] Web site of Netmeeting: http://www.microsoft.com/windows/netmeeting/

[HSSR99] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg: "SIP: Session Initiation Protocol" RFC 2543, March 1999.

[ITU98] ITU-T Recommendation H.323 V.2 "Packet-Based Multimedia Communication Systems", Februar 1998.

[DF99] I. Dalgic, H. Fang: "Comparsion of H.323 and SIP for IP Telephony Signaling" In Proceedings of Photonics East, Boston, Massachusetts, September 20-22, 1999.

[Dous00] B. Douskalis: "IP Telephony - The Integration of Robust VoIP Services" Prentice Hall, 2000.

[RFCSCFJ96] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: "RTP: A Transport Protocol for Real-Time Applications" RFC 1889, IETF, Jan. 1996

[ITU98-2] ITU-T Recommendation H.235 "Security and Encryption for H. Series (H.323 and other H.245 based) Multimedia Terminals", Februar 1998.

[ITU97] ITU-T Recommendation H.245, Version 3 "Control Protocol for Multimedia Communication" September 1997

[RFC2408] D. Maughan M. Schertler M. Schneider J. Turner. *Internet Security Association and Key Management Protocol (ISAKMP)*. November 1998

Chapter 7: Protocol proposal

[DP2000] R. Dhamija, A. Perrig: *Déjà Vu: A User Study using Images for Authentication*. 9th Usenix Security Symposium. August 2000.

[XML00] W3C. The Extensible Markup Language. http://www.w3.org/XML/

[JCE00] Appendix A in the Java Cryptography Architecture API Specification & Reference.

Chapter 8: Implementation

[SIPsoft00] Session Initiation Protocol. http://www.cs.columbia.edu/sip/implementations.html

Chapter 9: Conclusions

[BAA00] B. AbdelAziz. Using IKE with MobInTel. Internal project. SITE, U.of Ottawa. 2000.

Appendix:

A - Table of cryptographic algorithms

X_{number} means that algorithms X uses a *number*-length key or has a result of legth *number* for Hash/MAC. Encryption and decryption speed (e.g. RSA1024 4kB c/2kB d) are calculated for a Pentium 200 MHz. What really counts is the speed order of magnitude between different types of algorithms.

Cryptographic algorithms					
Symmetric-Key		Public-Key		Hash/MAC	
Block Cipher	Stream-Cipher	Encryption	Signature	Hash	MAC
NIST: DES-40_{40}-ECB DES_{56}-ECB 3-DES_{168} AES $Skipjack_{80}$ NSA: $Cast_{128}$ Enigma Comsec ClipperChip "Independents": $IDEA_{128}$ 3-IDEA $Blowfish_{32\text{-}448}$ $RC5_{0\text{-}2040}$ $CDMF_{40}$	$RC4\text{-}40_{40}$ $RC4\text{-}128_{128}$ DES-CBC DES-CFB DES-OFB DES-PCBC	$RSA_{0\text{-}\infty}$ $ECC_{0\text{-}\infty}$ SHEN (cern)	$RSA_{0\text{-}\infty}$ DSA $ECC_{0\text{-}\infty}$ El Gamal $DSS_{0\text{-}\infty}$	$SHA\text{-}1_{160}$ $MD5_{128}$ $RIPEMD_{160}$	HMAC-SHA HMAC-MD5 DES/CBC-MAC XOR-MAC
Short keys		Long Keys		Short outputs	
Fast (DES-CBC: 3MB/3MB)		Slow (RSA_{1024} 4kB enc/2kB dec)		Very Fast (MD5 36 MB/s)	

Table App-A: cryptographic algorithms.

B - Relationships between some protocols in the OSI layer model

<table>
<tr><td>Application</td><td colspan="3">HTTP, FTP, SMTP, Telnet, LDAP DNS</td></tr>
<tr><td>Presentation</td><td colspan="3">MPEG, H261</td></tr>
<tr><td>Session</td><td colspan="3">SSL, RTSP RTP/RTCP,SIP, H.323</td></tr>
<tr><td>Transport</td><td>TCP</td><td>TCP or UDP</td><td>UDP</td></tr>
<tr><td>Network</td><td colspan="3">IPv4-6/ICMP, IPsec</td></tr>
<tr><td>Data link</td><td colspan="3">Local Network Protocols:
MAC(Eth, TR, FDDI), AAL3/4, AAL5, PPP, WAP, GSM, CDMA…</td></tr>
<tr><td>Physical link</td><td colspan="3">Physical Network Access Protocols:
RJ45, 10-Base-T, PSTN, ATM, Sonet, V.34</td></tr>
</table>

Table App-B: protocol stack.

C – Acronyms

AAA: Authentication, Authorization and Accounting protocol

AES: Advanced Encryption Standard

AH: Authentication Header

API: Application Programming Iterface

AS: Authentication Server

AuC: Authentication Center

BSC: Base Station Controller

BTS: Base Transceiver Station

CA: Certificate Authority

CBC: Cipher Block Chaining

CDP: Certificate Distribution Points

CFB: Cipher FeedBack

CHAP: CHallenge Authentication Protocol

CORBA: Common Object Request Broker Architecture

COPS: Common Open Policy Service

CP: Certification Policy

CPS: Certificate Practice Statement

CRL: Certificate Revocation List

CRLF: Carriage Return Line Feed

CSPDN: Circuit-Switched Public Data Network

CSR: Certificate Signing Request

CTR: CounTeR (mode)

CV: Challenge Value

DAC : Discretionary Access Control

DES: Data Encryption Standard

DCE : Distributed Computing Environment

DH: Diffie-Hellman

DoS: Denial of Service

DS: Dual Signature

DSS: Digital Signature Standard

EAP: Extensible Authentication Protocol

ECB: Electronic CodeBook

ECC: Elliptic Curve Cryptography

EIR: Equipment Identity Register

ESP: Encapsulating Secure Payload

GSM: Global System for Mobile communications

GUI: Graphical User Interface

HLR: Home Location Registrar

HMAC: Hash Message Authentication Code

HTTP: Hypertext Transfer Protocol

HV: hash-value

ICA: Information Communication Architecture

IDEA: International Data Encryption Algorithm

IDL: Interface Definition Language

IETF: Internet Engineering Task Force

IKE: Internet Key Exchange

IMEI: International Mobile Equipment Identity

IMHP: Internet Mobile Host Protocol

IMSI: International Mobile Subscriber Identity

IN: intelligent network

IntServ: Integrated Services

IP Internet protocol

IPsec: Internet Protocol security (protocol)

ISAKMP: Internet Security Association and Key Management Protocol

J2SE: Java 2 platform Standard Edition

JAXP: Java API for XML parsing

JCE: Java Cryptographic Extension

JSSE: Java Secure Socket Extension

KDC: Key Distribution Standard

LDAP: Lightweight Directory Access Protocol

LNCS: Lecture Notes in Computer Science

MAC : Message Authentication Code

MAC : Mandatory Access Control

MD5: message digest (version 5)

MIT : Massachussets Institute of Technology

MobInTel : Mobile Internet Telephony

MSC : Mobile services Switching Center

NAS: Network Access Server

NFS: Network File System

NGN: Next Generation Network

OCSP: Online Certificate Status Protocol

OFB: Output FeedBack

OTP: One-Time Pad

PAP: Password Authentication Protocol

PFS: Perfect Forward Secrecy

PGP: Pretty Good Privacy

PIN: Personal Identification Number

PKI: Public Key Infrastructure

PMI: Privilege Management Infrastructure

PPP: Point-to-Point Protocol

PSPDN: Packet Switched Public Data Network

PSTN: Public Switched telephone Network

POTS: Plain Old Telephone Service

PSE: Personal Security Environment

QoS: Quality of Service

RADIUS: Remote Authentication Dial-In User Service

RAS: registration, admissions, and status

RDF: Resource Description Framework

RIPEMD: RACE Integrity Primitives Evaluation Message Digest

ROM: Read Access Memory

RPC: Remote Procedure Call

RSA: Rivest Shamir-Adelman

RSVP: Resource reSerVation Protocol

RTCP: Real-Time Control Protocol

RTP: Real-Time Protocol

SDP: Session Description Protocol

SecCx: Secure Connection

SET: Secure Electronic Transaction

SHA1: Secure Hash Algorithm (version 1)

SIM: Subscriber Identity Module

SIP: Session Initiation Protocol

SMS: Short Message Service

SMTP: Simple Mail Transfer Protocol

SPKI: Simple Public Key Infrastructure

SRES: Signed RESponse

SS7: signaling system 7

SSH: Secure SHell

SSL: Secure Socket Layer

TACACS: Terminal Access Controller Access Control System

TCP: Transport Control Protocol

TDMA: Time Division Multiple Access

TEMPEST: Transient Electromagnetic Pulse Emanation Standard

TGS: Ticket Granting Server

TLS: Transport Layer Security

TS: Time stamp

UPT: Universal Personal Telecommunication

URI: Uniform Resource Identifier

VLR: Visitor Location Registrar

VoIP: Voice over the Internet Protocol

VPN: Virtual Private Network

XML: Extensible Markup Language