

# 國立交通大學

資訊科學與工程研究所

## 博士論文

行動通信全 IP 網路的安全機制  
Security for Mobile All-IP Network



研究生：許世芬

指導教授：林一平博士

中華民國九十八年七月

行動通信全 IP 網路的安全機制  
**Security for Mobile All-IP Network**

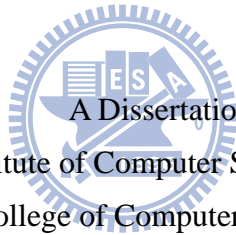
研究生：許世芬

Student : Shih-Feng Hsu

指導教授：林一平 博士

Advisor : Dr. Yi-Bing Lin

國立交通大學  
資訊科學與工程研究所  
博士論文



A Dissertation  
Submitted to Institute of Computer Science and Engineering  
College of Computer Science  
National Chiao Tung University  
in partial Fulfillment of the Requirements  
for the Degree of  
Doctor of Philosophy  
in  
Computer Science

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

# 行動通信全 IP 網路的安全機制

研究生：許世芬

指導教授：林一平 博士

國 立 交 通 大 學  
資 訊 工 程 學 系

## 摘 要

在全球行動通信系統 (UMTS) 全 IP (All-IP) 架構中，IP 多媒體子系統 (IMS) 提供 IP 多媒體服務供行動用戶使用。依照網路的功能，UMTS 全 IP 架構可以分成三個部份：IP 多媒體子系統 (IMS)、應用與服務網路、以及無線存取網路。透過無線存取網路，行動用戶可以存取 IMS 提供的 IP 多媒體服務；而應用與服務網路則提供彈性且有效率的開發平台，供 IP 多媒體服務的發展與建置。本論文探討 UMTS 全 IP 架構裡的安全機制議題，包括：認證、授權、以及保密機制。在提供 IP 多媒體服務之前，應用與服務網路需要跟 IMS 完成雙向認證；而為了能安全的存取 IP 多媒體服務，行動用戶亦需要跟 UMTS 全 IP 網路執行認證與保密機制。

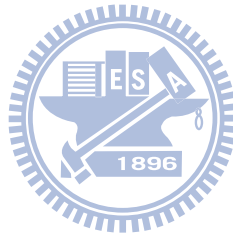
在這篇論文的第一部份，我們探討應用與服務網路跟 IMS 之間的認證授權機制。我們透過開放式服務存取 (OSA) 來描述應用與服務網路的設計概念；並介紹 OSA 應用伺服器提供 IP 多媒體服務之前，所執行的雙向認證流程。

在本論文的第二部份，我們著重在 UMTS 全 IP 網路的認證與保密機制。首先，我們探討在無線存取網路裡的認證機制：當透過無線存取網路來接取 IP 多媒體服務之前，行動用戶需與 UMTS 全 IP 網路達成雙向認證。然而，當行動用戶換手 (Handoff) 時，雙向認證的步驟會產生長時間的延遲，可能會中斷使用中的服務。為了解決這一問題，我們分別在無線區域網路 (WLAN) 與全球互通微波存取 (WiMAX) 系統中，研究如何省略不必要的認證步驟。執行完認證機制後，行動用戶還需執行保密機制來確保存取

的資料不會被竊取。本論文以點對點加密簡訊服務（SMS）來介紹 UMTS 全 IP 網路的保密機制。SMS 加密服務提供行動用戶跟應用與服務網路間保密的訊息交換機制。我們在標準的 UMTS 網路中，實作出二套 SMS 加密機制，並且評估加密的額外負擔。

以上的研究成果提供讀者在研究 UMTS 全 IP 網路裡認證與保密機制的議題上，可供參考之基礎。

**關鍵字：**認證、授權、與計費，IP 多媒體子系統，開放式服務存取，保密，簡訊服務，全球行動通信系統，無線區域網路，全球互通微波存取



# Security for Mobile All-IP Network

Student : Shih-Feng Hsu

Advisor : Dr. Yi-Bing Lin

Institute of Computer Science and Engineering  
National Chiao Tung University

## ABSTRACT

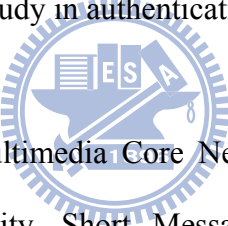
The *IP Multimedia Core Network Subsystem* (IMS) provides the IP multimedia services on the *Universal Mobile Telecommunications System* (UMTS) all-IP network. According to the network functionalities, the UMTS-all-IP architecture can be partitioned into three categories: IMS network, application and service network, and wireless access networks. Through the wireless access networks, the *Mobile Station* (MS) can access the IMS network for IP multimedia services. The application and service network supports flexible and efficient approaches for services development and deployment. This dissertation focuses on the authentication and security mechanisms in this UMTS all-IP architecture. Before providing IMS services, the application and service network should perform the authentication mechanism with the IMS network. Moreover, for secure IMS service access, the MS should perform the authentication and security mechanisms with the UMTS-all-IP network.

In the first part of this dissertation, we study on the authentication mechanism between the IMS network and the application and service network. We utilize the *Open Service Access* (OSA) to illustrate the concept of the application and service network, and study how the *OSA Application Server* (AS) mutually authenticates with *OSA Framework* before providing services.

In the second part of this dissertation, we demonstrate on the authentication and security mechanisms performed in the UMTS all-IP network. First, we study the authentication

mechanism in the wireless access network. Before accessing services through the wireless access networks, the MS should authenticate with the UMTS all-IP network. However, the execution of authentication on handoff may incur long delay and result in force-termination for real-time applications. To address this issue, we investigate how to eliminate the non-necessary authentication cost in *Wireless Local Area Network* (WLAN) and mobile *Worldwide Interoperability for Microwave Access* (WiMAX). After authentication, the MS should perform the security mechanism for secure service access. Thus we utilize the end-to-end secure *Short Message Service* (SMS) to illustrate the security mechanism between the MS and the application and service network. We implement two secure SMS mechanisms over the standard SMS network and estimate the encryption overhead.

These research results presented in this dissertation can be viewed as a useful foundation for further UMTS all-IP network study in authentication and security mechanisms.



**Keywords:** authentication, IP Multimedia Core Network Subsystem (IMS), Open Service Access (OSA), security, Short Message Service (SMS), Universal Mobile Telecommunications System (UMTS), Wireless Local Area Network (WLAN), Worldwide Interoperability for Microwave Access (WiMAX)

# Acknowledgements

I would like to express my sincere thanks to my advisor, Prof. Yi-Bing Lin. Without his supervision and perspicacious advice, I can not complete this dissertation. Special thanks are due to my committee members: Dr. Chung-Hwa Rao, Prof. Chu-Sing Yang, Prof. Han-Chieh Chao, Prof. Jean-Lien Chen, Prof. Ming-Feng Chang, and Prof. Rong-Jaye Chen for their valuable comments and helpful discussions.

I also express my appreciation to all the faculty, staff and colleagues in the Department of Computer Science. In particular, I would like to thank Prof. Shun-Ren Yang, Prof. Pei-Chun Lee, Dr. Lin-Yi Wu, Prof. Sok-Ian Sou, Mr. Yung-Chun Lin, Mr. Meng-Hsun Tsai, Ms. Ya-Chin Sung, Mr. Chien-Chun Huang-Fu, and Ms. Hsin-Yi Lee in the Laboratory 117 for their friendship and support in various ways.

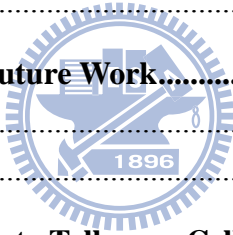
Finally, I am grateful to my dear parents, brother, sister, and girl friend Winnie Wang for their encouragement and firm support during these years.

# Contents

摘要 .....	i
ABSTRACT .....	iii
Acknowledgements .....	v
Contents .....	vi
List of Tables .....	viii
List of Figures .....	ix
Notations .....	x
<b>Chapter 1. Introduction .....</b>	<b>1</b>
1.1 UMTS All-IP Network Architecture .....	2
1.2 Open Service Access .....	4
1.3 AAA and Security Mechanisms in Wireless Access Network .....	6
1.4 Dissertation Organization .....	7
<b>Chapter 2. CCL OSA: a CORBA-based Open Service Access System.....</b>	<b>9</b>
2.1 Introduction to Open Service Access .....	10
2.2 CORBA-based OSA API .....	12
2.3 OSA Mutual Authentication for Initial Access .....	17
2.4 Detailed CORBA Interactions for CCL OSA .....	19
2.5 Application Server Architecture .....	23
2.6 Authentication between AS and Framework .....	24
2.7 Summary .....	27
<b>Chapter 3. Selecting Transition Process for WLAN Security .....</b>	<b>29</b>
3.1 Introduction to IEEE 802.11r .....	29
3.2 Initial MD Association Process (Inter-MD Scenario) .....	32
3.3 Fast BSS Transition (Intra-MD Scenarios) .....	34
3.4 Transition Process Selection Mechanism .....	35
3.5 Summary .....	37
<b>Chapter 4. A Key Caching Mechanism for Reducing WiMAX Authentication Cost in Handoff</b>	<b>39</b>
4.1 Introduction to WiMAX AAA Architecture .....	39
4.2 WiMAX Initial Network Entry Process .....	41

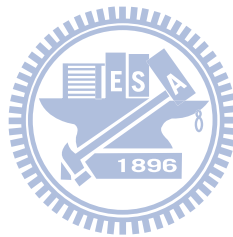


4.3 The Key Caching Mechanism .....	44
4.3.1 Derivation for Exponentially Distributed $t_M$ and Fixed $T$ .....	46
4.3.2 Derivation for Generally Distributed $t_M$ and Exponential $T$ .....	47
4.4 Simulation Validation .....	49
4.5 Numerical Examples .....	51
4.6 Summary.....	55
<b>Chapter 5. End-to-End Security Mechanisms for SMS .....</b>	<b>57</b>
5.1 Introduction to Wireless Cryptography .....	57
5.2 Public Key Cryptography .....	59
5.2.1 Certificate-based Public Key Cryptography.....	60
5.2.2 ID-based Public Key Cryptography .....	61
5.3 End-to-End Security for SMS .....	62
5.3.1 SMS Architecture .....	62
5.3.2 RSA Mechanism.....	63
5.3.3 ID-based Mechanism.....	65
5.4 Performance Comparison .....	68
5.5 Summary.....	71
<b>Chapter 6. Conclusions and Future Work.....</b>	<b>73</b>
6.1 Conclusions .....	73
6.2 Future Work.....	75
<b>Appendix A. OSA-based Push to Talk over Cellular Service.....</b>	<b>78</b>
A.1 Introduction to Push to Talk over Cellular.....	78
A.2 OSA AS Architecture for PoC Service .....	79
A.3 PoC Session Establishment .....	81
<b>Bibliography.....</b>	<b>87</b>
<b>Publication List.....</b>	<b>91</b>



# List of Tables

Table 2.1 POA policy modes for CCL OSA.....	16
Table 4.1 Comparison of analytic and simulation results.....	51
Table 5.1 Key length for equivalent security levels (in bits).....	69



# List of Figures

Figure 1.1 UMTS all-IP network architecture .....	3
Figure 1.2 Open Service Access architecture .....	5
Figure 2.1 Open Service Access architecture .....	10
Figure 2.2 CORBA architecture .....	13
Figure 2.3 Message flow for OSA mutual authentication .....	18
Figure 2.4 CORBA interaction flows between the AS and the FW .....	20
Figure 2.5 Application server architecture .....	23
Figure 2.6 Message flow for OSA mutual authentication .....	25
Figure 3.1 IEEE 802.11r security hierarchy .....	30
Figure 3.2 IEEE 802.11r initial MD association process .....	32
Figure 3.3 Derivation of PMK-R0, PMK-R1, and PTK.....	33
Figure 3.4 IEEE 802.11r fast BSS transition process.....	35
Figure 3.5 The proposed selection mechanism for transition process.....	36
Figure 4.1 WiMAX AAA architecture and protocol stack .....	40
Figure 4.2 WiMAX initial network entry process.....	42
Figure 4.3 WiMAX key derivation tree.....	44
Figure 4.4 Relationship of the MSK key lifetime and the MS movement.....	45
Figure 4.5 The simulation flowchart .....	49
Figure 4.6 Effect of $\mu$ .....	53
Figure 4.7 Effect of $V_M$ .....	54
Figure 5.1 The man-in-the-middle attack.....	60
Figure 5.2 The certificate-based public-key distribution.....	61
Figure 5.3 The ID-based public-key distribution .....	62
Figure 5.4 GSM SMS network architecture .....	63
Figure 5.5 The RSA procedure for sending an encrypted short message.....	64
Figure 5.6 ID-based end-to-end encryption mechanism .....	67
Figure 5.7 Encrypted short message experimental environment.....	68
Figure 5.8 Delivery delays of SMS .....	70
Figure A.1 Application server architecture for PoC Service .....	79
Figure A.2 Message flow for PoC session establishment.....	82
Figure A.3 Message flow for PoC session termination .....	85

# Notations

The notation used in this dissertation is listed below.

- $\alpha$ : the probability that the MS returns to the old ASN-GW before the MSK lifetime expires
- $E [t_K | t_M \geq t_K]$  : the expected unused key period under the condition that the MS does not return to the old ASN-GW before the MSK lifetime expires
- $E [t_K^* | t_M \leq t_K]$  : the expected reused key period under the condition that the MS returns to the old ASN-GW before the MSK lifetime expires
- $f(t_M)$  : the distribution of  $t_M$
- $f^*(s)$  : the Laplace transform of the  $t_M$  distribution
- $k_{ID}$  : the ID-based cipher key length
- $k_{RSA}$  : the RSA cipher key length
- $L_{ID}$  : the ID-based ciphered short message length
- $L_{RSA}$  : the RSA ciphered short message length
- $\frac{1}{\lambda} = E [t_M]$  : the mean MS residence time in new ASN-GWs
- $\frac{1}{\mu} = E [T]$  : the mean MSK lifetime
- $T$  : the MSK lifetime
- $t_K$  : the residual MSK lifetime after the MS moves from the old ASN-GW to the new ASN-GW
- $t_K^*$  : the reused key period after the MS returns to the old ASN-GW
- $t_M$  : the MS residence time in new ASN-GWs
- $V_M$  : the variance of MS residence time in new ASN-GWs

# Chapter 1.

## Introduction

*Universal Mobile Telecommunications System* (UMTS) evolved from *General Packet Radio Service* (GPRS) is a mobile telecommunication network providing high speed data services. To integrate *Internet Protocol* (IP) with UMTS, the UMTS all-IP architecture is proposed [21]. In UMTS release 5, the *IP Multimedia Core Network Subsystem* (IMS) is introduced on top of the UMTS all-IP architecture to support the IP multimedia services, such as voice over IP. Figure 1.1 illustrates the UMTS all-IP architecture. According to the network functionalities, the UMTS all-IP architecture can be partitioned into three categories:

**IMS Network** (Figure 1.1 (1)) provides the IP multimedia services to the *Mobile Station* (MS; Figure 1.1 (A)). The IMS is responsible for call routing and delivering voice data to the MSs in various access networks, including *Public Switch Telephony Network* (PSTN), wired access networks, and wireless access networks. In this dissertation, we focus on the wireless access networks.

**Wireless Access Networks** (Figure 1.1 (2)) provide wireless connectivity for the MS to access the IMS services. In this dissertation, we elaborate on three popular wireless access networks: the UMTS network (Figure 1.1 (4)), the *Wireless Local Area Network* (WLAN; Figure 1.1 (5)), and the mobile *Worldwide Interoperability for Microwave Access* (WiMAX; Figure 1.1 (6)).

**Application and Service Network** (Figure 1.1 (3)) supports flexible and efficient approaches for mobile service development and deployment through a service platform. With this platform, the service control is implemented in dedicated application servers with service-related databases, and is separated from the call and connection control. UMTS

defines several alternatives to construct the application and service network platform. In this dissertation, we use the *Open Service Access (OSA)* platform to illustrate the concept of the application and service network.

In the UMTS all-IP network, *Authentication, Authorization, and Accounting (AAA)* and encryption are two important mechanisms. Before the MS associates to the wireless access networks, the AAA mechanism is exercised to mutually authenticate the MS and the UMTS all-IP networks. Following the AAA mechanism, the encryption mechanism is performed to provide secure signaling and data delivery between MS and the wireless access networks and between the MS and the application and service networks.

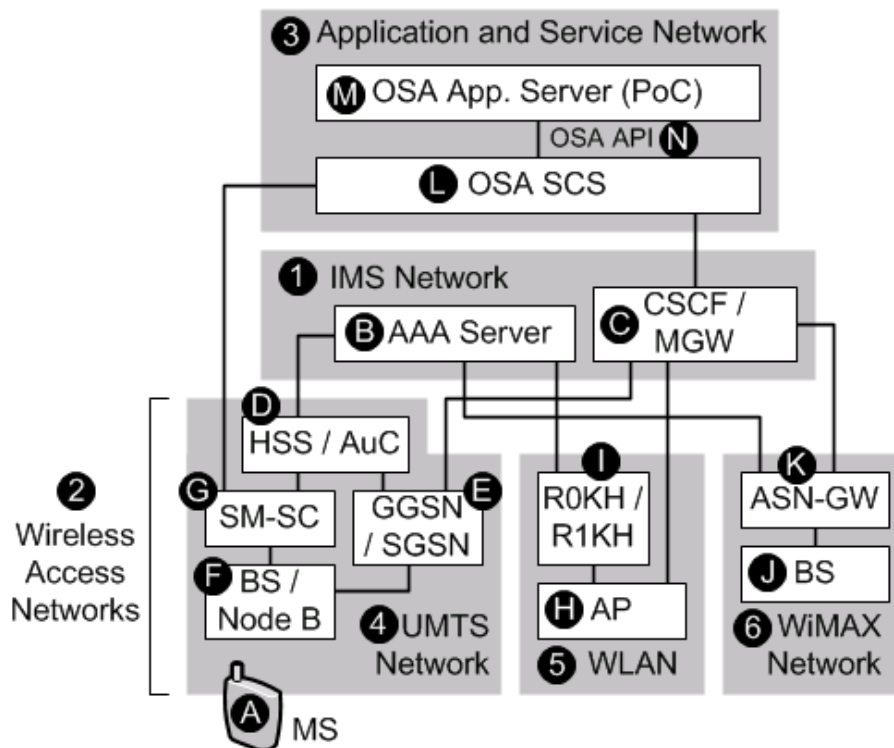
In the remainder of this chapter, we briefly introduce the UMTS all-IP network architecture. Then we present the OSA platform implemented in the application and service network. Finally, we describe the AAA and encryption mechanisms performed in the wireless access networks and describe the organization of this dissertation.

## 1.1 UMTS All-IP Network Architecture

Figure 1.1 illustrates the UMTS all-IP network architecture. In the IMS network, the AAA server (Figure 1.1 (B)) is responsible to mutually authenticate with the MS when the MS associates to the WLAN and the mobile WiMAX networks. The *Call Session Control Function (CSCF)* and *Media Gateway (MGW)* (Figure 1.1 (C)) are in charge of delivering the call control signaling and the voice data respectively, and interact with the application and service network to provide IP multimedia services.

The MS can access the IMS network through the UMTS network, the WLAN, or the mobile WiMAX network. In UMTS, the MS associates with the *Base Station (BS)* or *Node B* (Figure 1.1 (F)) through the air interface. The *Serving GPRS Support Node (SGSN)* and the *Gateway GPRS Support Node (GGSN)* (Figure 1.1 (E)) provide mobility and session managements to the MS. Through the GGSN, the UMTS network connects with the IMS

network to provide IP multimedia services. The *Home Subscriber Server* (HSS; Figure 1.1 (D)) is the master database containing all subscriber-related information. The *Authentication Center* (AuC; Figure 1.1 (D)) maintains the secret key of the MS. When performing the AAA mechanism, the SGSN, the GGSN, and the AAA server communicate with the HSS/AuC to retrieve authentication information and subscriber data of an MS. The *Short Message Service Center* (SM-SC; Figure 1.1 (G)) provides the *Short Message Service* (SMS) to the MS in UMTS. The SMS can be utilized to deliver control signaling between the MS and the application and service network.



- |   |  |
|---|--|
| AAA : Authentication, Authorization, and Accounting | API : Application Programming Interface  |
| AP : Access Point                                   | AuC : Authentication Center              |
| ASN-GW : Access Service Network Gateway             | CSCF : Call Session Control Function     |
| BS : Base Station                                   | HSS : Home Subscriber Server             |
| GGSN : Gateway GPRS Support Node                    | MS : Mobile Station                      |
| MGW : Media Gateway                                 | PoC : Push-to-Talk over Cellular         |
| OSA : Open Service Access                           | R0KH : Pairwise Master Key R0 Key Holder |
| R0KH : Pairwise Master Key R0 Key Holder            | R1KH : Pairwise Master Key R1 Key Holder |
| SCS : Service Capability Server                     | SGSN : Serving GPRS Support Node         |
| SM-SC : Short Message Service Center                |  |

Figure 1.1 UMTS all-IP network architecture

In WLAN, the MS connects to the IMS network through the *Access Point* (AP; Figure 1.1 (H)). In IEEE 802.11r WLAN, the authentication message is exchanged between the MS and the AAA server through the AP, the *Pairwise Master Key* (PMK) *R0 Key Holder* (R0KH), and *PMK R1 Key Holder* (R1KH; Figure 1.1 (I)). After authentication, R0KH, R1KH, and the MS generate an encryption key to provide secure signaling and data delivery between the MS and the AP.

The mobile WiMAX network consists of BS (Figure 1.1 (J)) and *Access Service Network* (ASN) *Gateway* (ASN-GW; Figure 1.1 (K)). The BS provides WiMAX radio access for the MS. Similar to the SGSN and the GGSN in UMTS, the ASN-GW provides mobility and session managements to the MS, and is responsible to forward the authentication messages between the MS and the AAA server. After authentication, the MS and the ASN-GW generate an encryption key to provide secure signaling and data transmission between the MS and the BS.

The IMS applications are implemented in the application and service network through the OSA platform. The OSA platform provides the third parties controlled access to the operator's network (i.e., the IMS and the UMTS networks). The third parties run their own applications in the OSA application server (Figure 1.1 (M)), and interworks with the OSA *Service Capability Server* (SCS; Figure 1.1 (L)) through the OSA *Application Programming Interface* (API; Figure 1.1 (N)). Details of OSA will be elaborated in Section 1.2.

## 1.2 Open Service Access

The OSA platform deployed in the application and service network is illustrated in Figure 1.2. The OSA consists of three parts.

**OSA Application Server** (AS; Figure 1.2 (A)) provides service by invoking the OSA SCS.

**OSA SCS** (Figure 1.2 (B)) provides the network functionalities to the OSA AS through a set of *Service Capability Features* (SCFs; Figure 1.2 (E)). These SCFs are offered by *Service*



*Capability* (SC; Figure 1.2 (F)), which interacts with the UMTS and IMS networks (Figure 1.2 (D)) to realize the network functionalities. To communicate with the UMTS and IMS networks, the SCs are implemented all kinds of interfaces. For example, the SC communicates with the CSCF through *Session Initiation Protocol* (SIP), and utilizes the *Message Transfer Part* (MAP) to retrieve authentication information from the HSS/AuC.

**Framework** (FW; Figure 1.2 (C)) is considered as one of the OSA SCSs. Before accessing the OSA SCS, the OSA AS should be authorized by the FW.

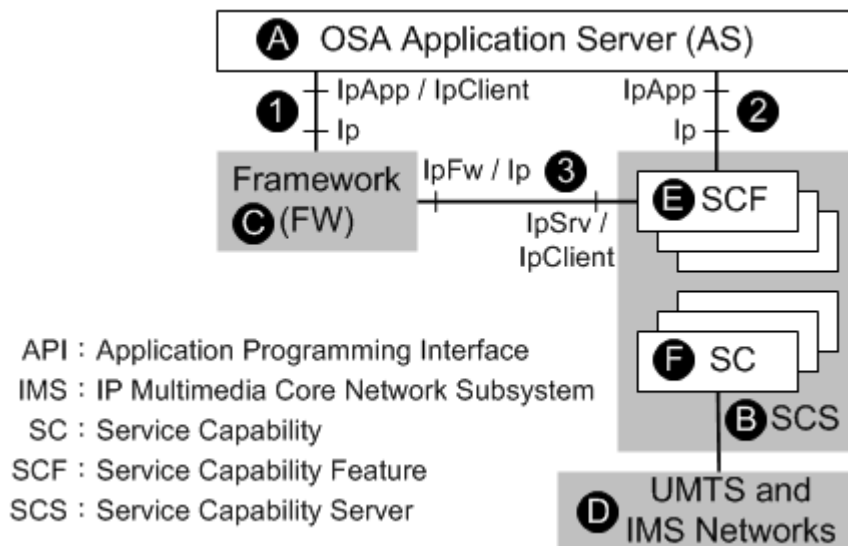


Figure 1.2 Open Service Access architecture

Through the standardized OSA APIs, an application needs not directly access the UMTS and IMS networks for services. Three classes of OSA APIs are defined among the AS, the FW, and the SCFs.

- Interfaces between the AS and the FW (Figure 1.2 (1)) provide authentication, network functionality discovery, and service agreement establishment mechanisms that enable the AS to access the SCSs. The FW-side interfaces to be invoked by the AS are prefixed with “Ip”. The AS-side interfaces to be called back by the FW are prefixed with “IpApp” or “IpClient”.

- Interfaces between the AS and the SCFs (Figure 1.2 (2)) allow the AS to invoke network functionality for services. The SCF-side interfaces to be invoked by the AS are prefixed with “Ip”. The AS-side interfaces to be called back by the SCFs are prefixed with “IpApp”.
- Interfaces between the FW and the SCFs (Figure 1.2 (3)) provide the mechanisms for SCF registration. The FW-side interfaces to be invoked by the SCFs are prefixed with “IpFw”. The SCF-side interfaces to be called by the FW are prefixed with “IpSvc”.

With these OSA APIs, OSA provides a unified service creation and execution environments which is independent from the underlying network technologies. The OSA platform can be built via the object-oriented techniques, such as *Common Object Request Broker Architecture* (CORBA) and *Simple Object Access Protocol* (SOAP).

## 1.3 AAA and Security Mechanisms in Wireless Access



### Network

In UMTS, the MS secret key  $K_i$  is stored in the *Subscriber Identity Module* (SIM) card and the HSS/AuC. Based on this  $K_i$ , the MS mutually authenticate with the HSS/AuC and generate the cipher key  $K_c$  before accessing the UMTS network. The  $K_c$  is utilized to encrypt the signaling and data delivery between the MS and the BS/Node B.

In WLAN and mobile WiMAX, the MS should mutually authenticate with the AAA server and generate an encryption key before accessing the networks. The IEEE 802.1X *Extensible Authentication Protocol* (EAP) standard specifies authentication and authorization for WLAN and mobile WiMAX. When performing authentication, the ROKH (in IEEE 802.11r WLAN) and the ASN-GW (in mobile WiMAX) serve as the authenticator. The authenticator is responsible to forward authentication messages between the MS and the AAA server, and to maintain the MS related information (e.g., encryption key) after authentication. To reuse the

UMTS authentication mechanism, the SIM-based EAP authentication is introduced on top of the IEEE 802.1X EAP standard to authenticate the MS. Details of the SIM-based EAP authentication will be elaborated in Chapter 5. Following the authentication, the security mechanism is performed to provide secure signaling and data transmission between the MS and wireless access network. When accessing the IP multimedia services, the MS should perform the other security mechanism with the application and service network.

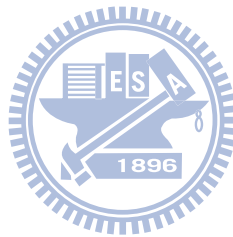
## 1.4 Dissertation Organization

Based on the above discussion, we study how UMTS all-IP applications can be implemented and exercised in OSA platform, and study the AAA and security mechanisms. In Chapter 2, we utilize the CORBA technique to develop the OSA platform in the application and service network. We use the authentication process between the OSA AS and the FW to illustrate how CORBA technique works in our OSA implementation. Then we propose an OSA AS architecture, and show how the OSA AS provides services by integrating the services supported by different SCFs. The mutual authentication procedure is used to demonstrate the interaction between the OSA AS and the FW. In Appendix A, we use the *Push to Talk over Cellular* (PoC) service to illustrate the interaction among the OSA AS and the SCFs for services. The PoC service is a walkie-talkie like service defined by the *Open Mobile Alliance* (OMA) PoC working group [25, 26].

Chapter 3 elaborates on the AAA and security mechanisms in WLAN. When switching from one AP to another, the MS executes the IEEE 802.1X EAP authentication, which may incur long delay and result in force-termination for real-time applications. The IEEE 802.11r is proposed to reduce the switching delay. However, we find an ambiguous *Mobility Domain Identifier* (MDID) issue in IEEE 802.11r WLAN. This chapter proposes a mechanism to resolve this issue, and therefore eliminates the cost for MDID management.

Chapter 4 describes the AAA and security mechanisms in mobile WiMAX. We propose a

key caching mechanism to eliminate the non-necessary IEEE 802.1X EAP authentication cost in mobile WiMAX handoff. This mechanism is investigated through analytic and simulation modeling. Our study indicates that the key caching scheme can effectively speed up the handoff process. In Chapter 5, we use two end-to-end security mechanisms for SMS to illustrate the security mechanism between the MS and the application and service network. Finally, we conclude this dissertation and give the future directions of this work.



## Chapter 2.

# CCL OSA: a CORBA-based Open Service Access

## System

Existing telecommunications services are considered as a part of network operation's domain and the development of services are achieved by, for example, *Intelligent Network* (IN) technology. By introducing internet and mobility into the telecommunications networks, more flexible and efficient approaches are required for mobile service deployment. Such approaches must allow network operators and enterprises to increase revenues via third party applications and service providers. To achieve the above goals, standardization bodies such as 3GPP CN5, ETSI SPAN12, ITU-T SG11 and the Parlay Group have been defining *Open Service Access* (OSA) specifications [1]. OSA provides unified service creation and execution environments to speed up service deployment that is independent from the underlying mobile network technology. In a research collaboration between National Chiao Tung University and Computer and Communications Laboratories (CCL)/Industrial Technology Research Institute (ITRI), we have developed the CCL OSA system. This chapter describes how the CCL OSA system can be implemented by the *Common Object Request Broker Architecture* (CORBA) technique. We use the authentication procedure for initial access to illustrate how the CORBA works in the CCL OSA system. Then we propose an OSA AS architecture. We show how the AS provides services by integrating the services supported by the *Service Capability Features* (SCFs). Then the OSA mutual authentication is utilized to illustrate the interaction among the AS modules and how the AS interacts with the *Framework* (FW). In Appendix A, we use the *Push to Talk over Cellular* (PoC) service to illustrate how the AS interacts with the SCFs for services.

## 2.1 Introduction to Open Service Access

This section describes the architecture and the features of the OSA system. As illustrated in Figure 2.1, the OSA consists of three parts: applications are implemented in one or more *Application Servers* (AS; Figure 2.1 (A)). *Framework* (FW; Figure 2.1 (C)) authorizes applications to access the *Service Capability* (SC; Figure 2.1 (F)) in the network. That is, an application can only access the OSA *Application Programming Interface* (API) via the FW for services. *Service Capability Servers* (SCS; Figure 2.1 (B)) provide the applications access to underlying network functionality through *Service Capability Feature* (SCF; Figure 2.1 (E)). These SCFs, specified in terms of interface classes and their methods, are offered by SCs within networks (and under network control; Figure 2.1 (D)). SCs are bearers needed to realize services.

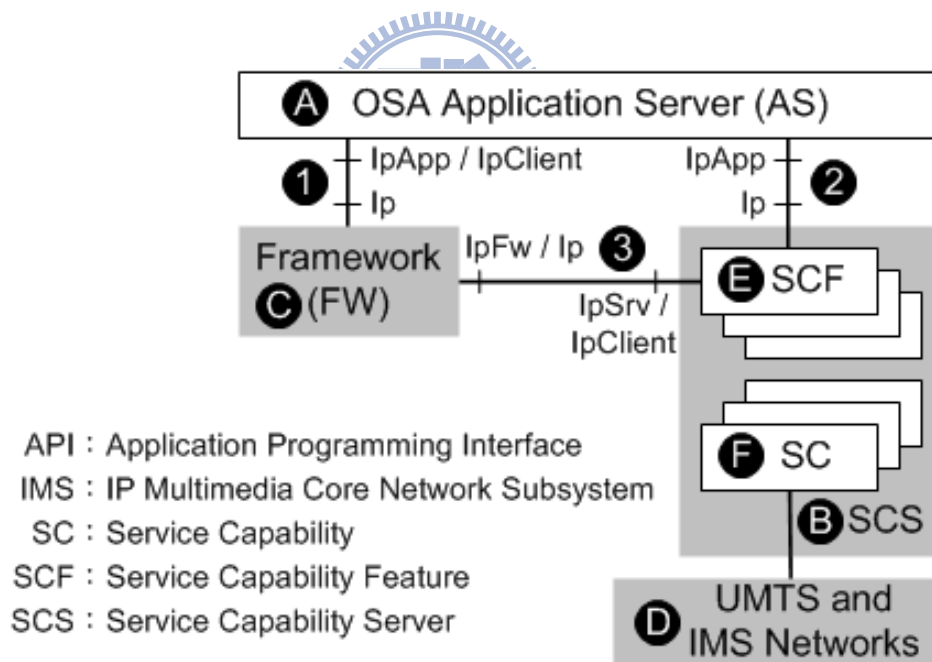


Figure 2.1 Open Service Access architecture

The FW is considered as one of the SCSs and is always present, one per network. The FW provides access control functions to authorize the access to SCFs or service data for any API method invoked by an application, with specified security level, context, domain, etc. Before any application can interact with a network SCF, an offline service agreement must be

established. Once the service agreement exists, mutual authentication can be performed between the application and the FW. Then the application can be authorized by the FW to access a specific SCF (in OSA, authentication must precede authorization). Finally the application can use the discovery function to obtain information on authorized network SCFs. The discovery function can be used at any time after successful authentication. SCFs offered by an SCS are typically registered at the FW. This information is retrieved when the application invokes the discovery function. Framework allows OSA to go beyond traditional IN technology through openness, discovery, and integration of new features. Based on TINA [28], the FW provides controlled access to the API by supporting flexibility in application location and business scenarios. Furthermore, the FW allows multi-vendorship and even the inclusion of non-standardized APIs, which is crucial for innovation and service differentiation. An SCS can be deployed as a standalone node in the network or directly on a node in the core network. In the distributed approach, the OSA gateway node contains the FW and zero or more SCS components. Other SCSs are implemented in different nodes. It is possible to add more SCSs and distribute the load from different applications over multiple SCSs. At the service selection phase, the FW may divert one application to one SCS and another to a different SCS. With middleware such as CORBA, it is possible to distribute load on a session basis without the application being aware that different sessions involve different SCSs. In some APIs, it is possible to add multiple application callbacks to the SCS so that the SCS can distribute the load of multiple sessions over different applications running on different servers. To allow applications from visited networks to use the SCSs in the home network, all communications between the application server and the SCSs must be secured through, for example, Secure Socket Layer or IPsec.

Examples of OSA SCFs are given as follows: call and session control SCFs provide capabilities for setting up basic calls or data sessions as well as manipulating multimedia conference calls. User and terminal related SCFs allow obtaining information from the end-user

(including user location and status) and the terminal capabilities, playing announcements, sending short text messages, accessing to mailboxes, etc. management related SCFs provision connectivity *Quality of Service* (QoS), access to end-user account and application/data usage charging. Interaction between an application and an SCS is always initiated by the application. In some scenarios, it is required to initiate the interaction from the SCS. An example is the call screening service. Suppose that the network routes a call to a user who has subscribed to this OSA service. Before the call reaches the user, the call screening application needs to be invoked. This issue is resolved by the OSA request of event notification mechanism. Initially, the application issues an OSA interface class method (API call) to the SCS. This OSA method allows the SCS to invoke the application (e.g. call screening) through a callback function when it receives events (e.g. incoming calls) from the network related to the application.

Since functionality inside a telecommunications network can be accessible via the OSA APIs, applications can access different network capabilities using a uniform programming paradigm. Three OSA API classes are defined among the applications, the FW and the SCFs as described in Section 1.2. To be accessible to a side developer community, the APIs should be deployed based on open *Information Technology* (IT). The details will be elaborated in the Section 2.2.

## **2.2 CORBA-based OSA API**

CORBA is an emerging open distributed object computing infrastructure, which provides the higher layers a uniform view of underlying heterogeneous network and OS layers. CORBA automates many common network programming tasks such as object registration, location, activation, request demultiplexing, framing and error-handling. Figure 2.2 illustrates the CORBA architecture.

In this architecture an *object* is a CORBA programming entity that consists of an identity, an interface and an implementation known as *servant* (Figure 2.2 (A)). An object reference



uniquely identifies the object across servers. This reference associates the object with one or more servant implementations. In CCL OSA API, every CORBA object is associated with one servant.

A *servant* is an implementation programming language entity that defines the operations to support an *Object Management Group (OMG) Interface Definition Language (IDL)* interface. Servants can be written in languages such as C, C++ or Java.

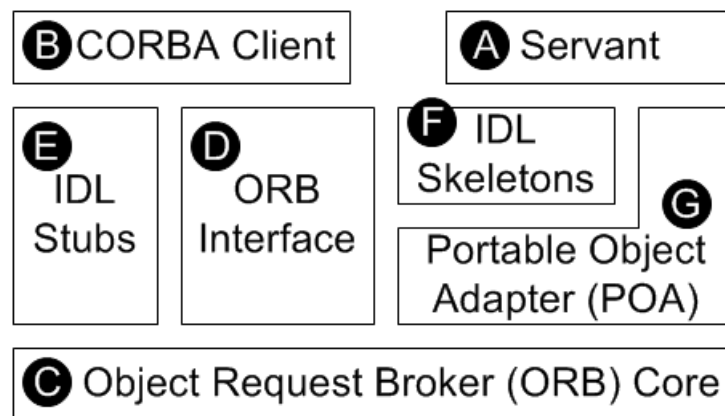


Figure 2.2 CORBA architecture

A *client* (Figure 2.2 (B)) is a program entity that invokes an operation on a servant. The client performs application tasks by invoking operations on object references. An object can be remote or local to the client. In CORBA, accessing a remote object should be as simple as calling an operation on a local object. A client always interacts with a servant through the corresponding object reference. Note that the CORBA concepts of client and servant are different from that of applications and servers in OSA (to be elaborated in Sections 2.3 and 2.4).

*Object Request Broker (ORB)* is a logical entity that can be implemented through several alternatives (such as one or more processes or a set of libraries). In CCL OSA, ORB is implemented as libraries. An ORB consists of an *ORB Core* (Figure 2.2 (C)) and an *ORB interface* (Figure 2.2 (D)). The ORB simplifies distributed programming by decoupling the client from the details of the method invocation. This makes client requests appear to be local

procedure calls. The ORB Core provides a mechanism for transparent delivery requests from clients to target servants. When a client invokes an operation, the ORB Core is responsible for finding the servant, transparently activating it if necessary, delivering the request to the object and returning any response to the client. An ORB Core is typically implemented as a run-time library linked into both client and server applications.

To decouple applications from implementation details, the CORBA specification defines an abstract ORB interface that provides various helper functions such as converting object references to strings. Specifically, *CORBA IDL stubs* (Figure 2.2 (E)) and *skeletons* (Figure 2.2 (F)) glue the clients, servants and the ORB. The CORBAIDL definitions are transformed into classes, structs, and functions in a particular language (e.g. C++, C, Java, etc.). Such transformation is automated by a CORBA IDL compiler. In CCL OSA, the target language is JAVA. We note that an object is an instance of an IDL interface. The corresponding servants implement the operations defined by the IDL. Several interfaces have been defined by IDL stubs to provide a strongly typed, *Static Invocation Interface* (SII) that marshals application (client) parameters into a common data-level representation. On the other hand, *skeletons* demarshal the data-level representation back into typed parameters that are meaningful to an application (server).

*Portable Object Adapter* (POA; Figure 2.2 (G)) is a CORBA portability enhancement. POA enables ORBs to support various types of servants that process similar requirements [37]. POA associates an IDL servant with objects, demultiplexes incoming requests to the servant and dispatches the appropriate operation on that servant. The POA design results in a smaller and simpler ORB that can still support a wide range of object granularities, lifetimes, policies, implementation styles and so on.

Several POA policies are specified in CCL OSA. *Threading policy* specifies the POA threading model. A POA can either be single-threaded or multithreaded concurrently controlled by the ORB. CCL OSA implementation uses ORB controlled multithreading model.

*Lifespan policy* specifies whether the CORBA objects created within a POA are *persistent* or *transient*. A transient object is destroyed when the process creating the object terminates. On the other hand, a persistent object can live beyond the life time of the process that created it. All POAs are transient. In any CORBA system, there is a root POA called `rootpoa`. In CCL OSA, `rootpoa` creates several POAs with the persistent life span policy. An example is POA `ipinitialpoa` created for the FW. In CCL OSA, `rootpoa` creates transient objects. For example, in Initial Access service, an authentication object (with the reference `ipAPIlevelauthentication_ref`) is transient because this object is session-oriented and is destroyed when the authentication procedure is complete. On the other hand, the `FWInitialContact` object (with the reference `ipinitial_ref`; see Step 1 in Figure 2.3) created by `ipinitialpoa` is persistent and should not be destroyed after the authentication action (between an AS application and the FW) is complete. The `FWInitialContact` object is the first contact point for any applications to start the authentication procedure. Therefore it must be persistent and active at any time. When the CCL OSA recovers from a failure, all persistent objects will be reactivated.

*Object Id uniqueness policy* specifies whether the servants activated in a POA must have unique Object Ids. In CCL OSA, since every object is implemented with one servant, the servant always has a unique Object Id. *Object Id assignment policy* specifies whether the Object Ids in a POA are generated by the application or the ORB. In CCL OSA, object Ids are generated by the ORB to avoid accidentally generating duplicated Object Ids by the programmer.

*Implicit activation policy* specifies whether implicit activation of servants is supported in a POA. With the implicit policy, the POA implicitly activates an object when the server application attempts to obtain a reference to a servant that is not already active. In CCL OSA, objects in `rootpoa` are implicitly activated. These implicitly created transient objects can be destroyed at the end of the session of a client application or are automatically cleaned up when the server process terminates. In CCL OSA POAs such as `ipinitialpoa`, objects are persistent

and should not be automatically clean up when the server process terminates. Therefore, these persistent objects are explicitly activated and should be explicitly destroyed by the programmer when he/she decides to stop providing the functionality.

*Servant retention policy* specifies whether the POA retains active servants in an active object map. A POA either retains the associations between servants and CORBA objects or establishes a new CORBA object/servant association for each incoming request. In CCL OSA, the POA retains the associations between servants and CORBA objects in an active object map.

*Request processing policy* specifies how requests are processed by the POA. The alternatives include:

1. to consult its active object map only
2. to use a default servant or
3. to invoke a servant manager

There are two types of servant managers: a manager of the *ServantActivator* type follows the RETAIN policy. A manager of the *ServantLocator* type follows the NON\_RETAIN policy. Since CCL OSA uses active object map, it uses first alternative for the request processing policy; that is, we consult POA's active object map for request processing.

Table 2.1 POA policy modes for CCL OSA

<i>Policy</i>	<i>Mode of Policy</i>
Thread Policy	ORB_CTRL_MODEL
Lifespan Policy	TRANSIENT (rootpoa) PERSISTENT (e.g. ipinitialpoa)
Object ID Uniqueness Policy	UNIQUE_ID
ID Assignment Policy	SYSTEM_ID
Servant Retention Policy	RETAIN
Request Processing Policy	USE_ACTIVE_OBJECT_MAP_ONLY
Implicit Activation Policy	IMPLICIT_ACTIVATION (for rootpoa) NO_IMPLICIT_ACTIVATION (e.g. ipinitialpoa)

The servant retention and request processing policies determine how the POA dispatches the request. The `RETAIN` and `USE_ACTIVE_OBJECT_MAP_ONLY` policies used in our approach allow simple and quick request processing at the cost of larger memory usage as compared with other servant retention and request processing policies. Table 2.1 lists the POA policy modes used in CCL OSA.

## 2.3 OSA Mutual Authentication for Initial Access

As mentioned in Section 2.1, an OSA application must authenticate with the FW before it can access any SCFs. Figure 2.3 illustrates the message flow of the mutual authentication procedure for initial access in CCL OSA. As shown in Figure 2.4, each of the AS and the FW implements both CORBA clients and CORBA servants. In the OSA mutual authentication procedure, the AS CORBA client uses the FW reference `ipinitial_ref` of initial contact interface `IpInitial`. In CCL OSA, this reference is obtained through a URL (e.g. `corbaname::pcs.csie.nctu.edu.tw:3500#IpInitial`) and the *naming service* of the FW. This *naming service* is a standard CORBA service that allows the CORBA client application to locate the object through a URL. `IpInitial` is implemented by the FW servant `ipinitialimpl` (Figure 2.3 (3)) to support the authentication function `initiateAuthenticationWithVersion`.

To authenticate the FW, the AS CORBA Client (Figure 2.3 (2)) invokes the `challenge` function in the FW interface `IpAPILevelAuthentication` (implemented by the servant `ipAPIlevelauthenticationimpl`; Figure 2.3 (4)). To authenticate the AS application, the FW CORBA client (Figure 2.3 (5)) invokes the `challenge (callback)` function in the AS interface `IpClientAPILevelAuthentication` (implemented by the servant `ipclientAPIlevelauthenticationimpl`; Figure 2.3 (1)). The detailed steps are described as follows.

**Step 1.** The AS CORBA client first generates the reference `ipclientAPIlevelauthentication_ref` of the AS interface `IpClientAPILevelAuthentication`. This reference will be called back by the FW to authenticate the AS application in Steps 5 and 6. The AS CORBA

client uses `ipinitial_ref` to invoke the FW function `initiateAuthenticationWithVersion` where the callback object reference `ipclientAPIlevelauthentication_ref` is included as a parameter. The FW servant `ipinitialimpl` returns a reference `ipAPIlevelauthentication_ref` of the FW interface `IpAPILevelAuthentication`. This FW interface is responsible for answering the authentication challenge from the OSA application.

**Step 2.** The AS CORBA client selects the authentication algorithm by invoking the FW function `selectAuthenticationMechanism` of the reference `ipAPIlevel authentication_ref`. In CCL OSA, authentication algorithm SHA-1 is utilized.

**Step 3.** The AS CORBA client authenticates the FW by invoking the function `challenge` of the FW reference `ipAPIlevelauthenticationi_ref`. The challenge function takes a random number as the parameter (in the byte-stream format). The FW servant `ipAPIlevelauthenticationimpl` executes the SHA-1 algorithm using the received random number and returns the result `fw_digest` to the AS CORBA Client.

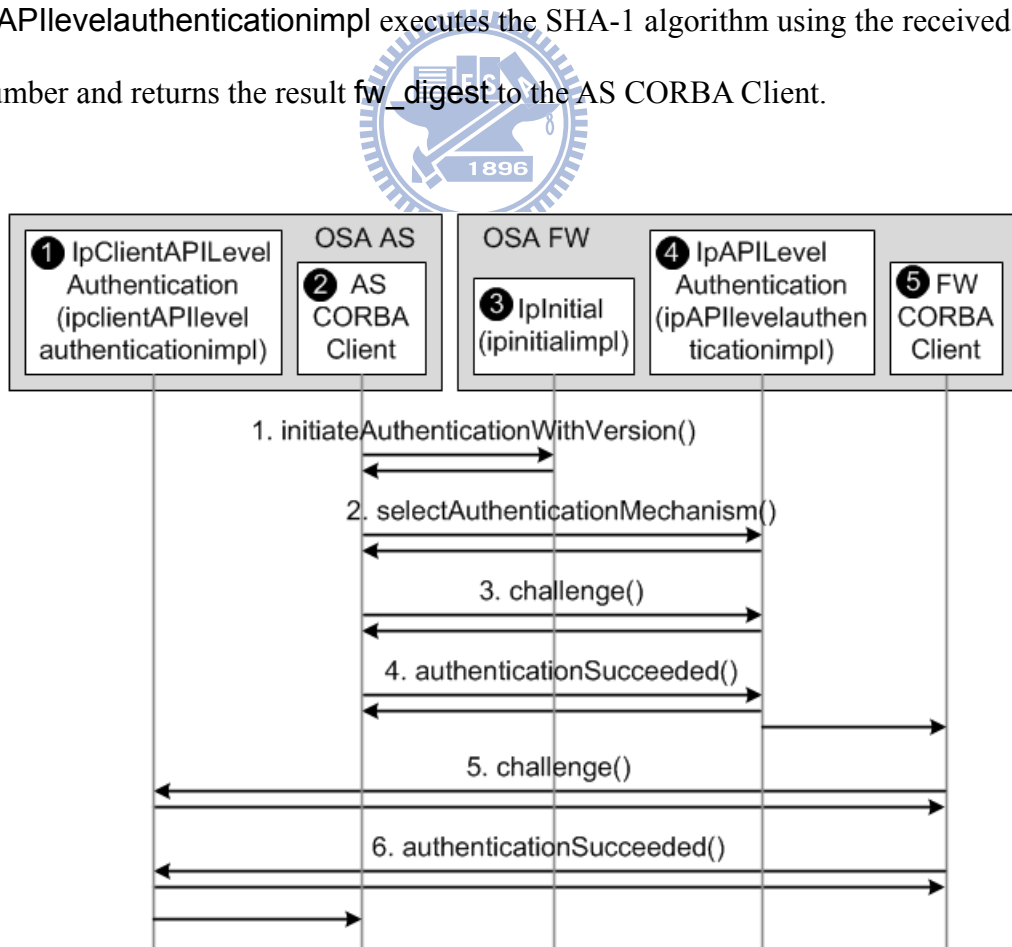


Figure 2.3 Message flow for OSA mutual authentication

**Step 4.** The AS CORBA client validates the result `fw_digest`. Suppose that the authentication is successful, the AS CORBA client invokes the function `authenticationSucceeded` of `ipAPIlevelauthentication_ref` to inform the FW servant `ipAPIlevelauthenticationimpl` of the result. Then the servant `ipAPIlevelauthenticationimpl` activates the FW CORBA client to authenticate the AS application.

**Step 5.** The FW CORBA client invokes the function `challenge` in the AS (callback) reference `ipclientAPIlevelauthentication_ref` (obtained in Step 1). Similar to Step 3, this function takes a random number as the parameter. The AS servant `ipclientAPIlevelauthenticationimpl` executes the SHA-1 algorithm and returns the result `app_digest` to the FW CORBA client.

**Step 6.** The FW CORBA client validates the result `app_digest`. Suppose that the authentication is successful, the `FWCORBAclient` invokes the function `authenticationSucceeded` of `ipclientAPIlevelauthentication_ref` to inform the AS servant `ipclientAPIlevelauthenticationimpl` of the result. This result is passed to the AS CORBA Client. At this point, the mutual authentication procedure is complete, and the OSA CORBA Client can access SCFs through the FW.

## 2.4 Detailed CORBA Interactions for CCL OSA

We use the `initiateAuthenticationWithVersion` function invoked from the AS to the FW (see Step 1 in Figure 2.3) and the callback `challenge` function invoked from the FW to the AS (see Step 5 in Figure 2.3) as examples to illustrate the CORBA interaction between the AS and the FW.

As we mentioned in the previous section, before the AS CORBA client requests mutual authentication, it first retrieves the FW reference `ipinitial_ref` of the `IpInitial` interface. The CORBA behavior for invoking `initiateAuthenticationWithVersion` is described as follows (which implement Step 1 in Figure 2.3).

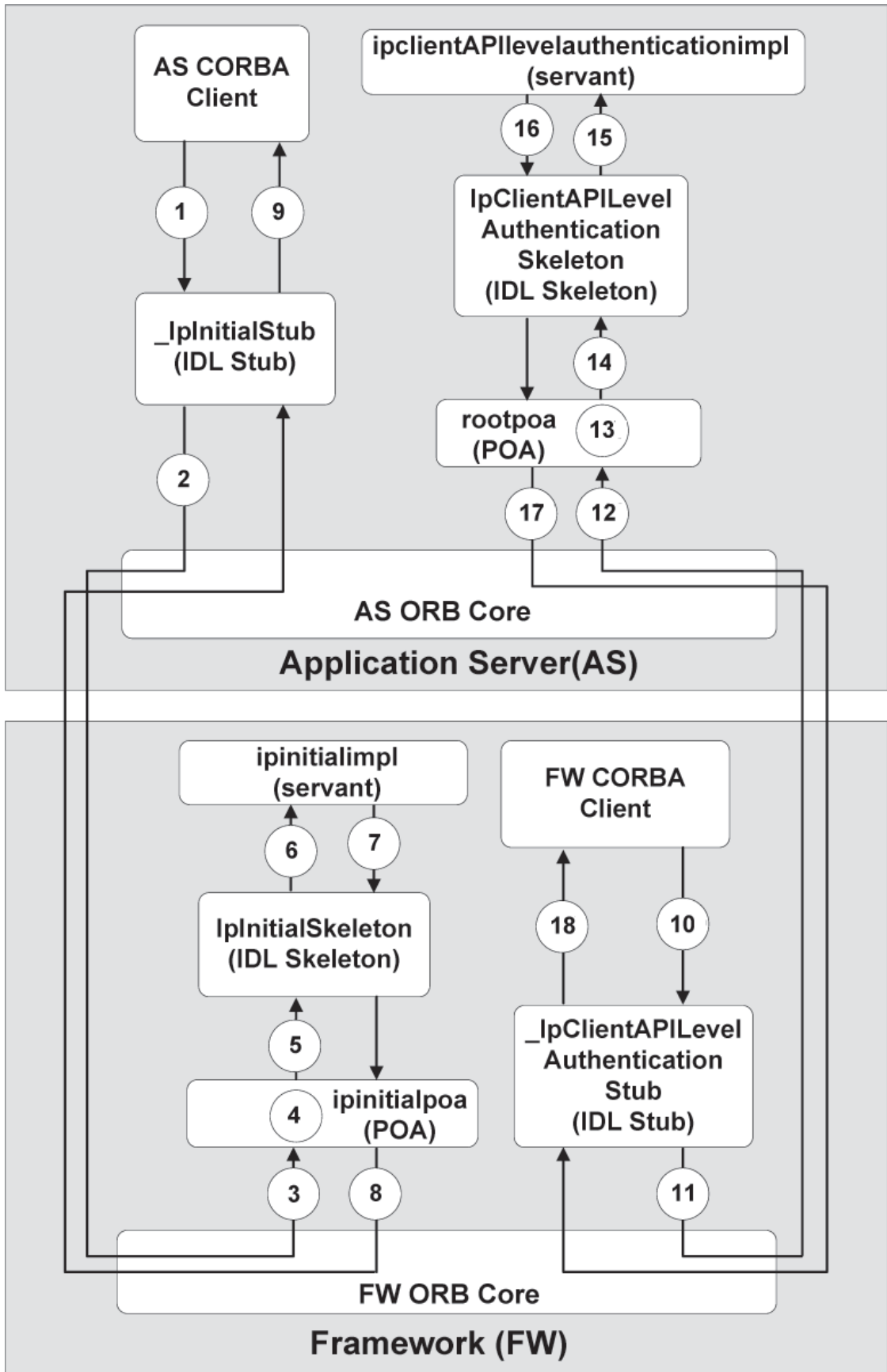


Figure 2.4 CORBA interaction flows between the AS and the FW



**Step 1.** The AS CORBA client uses `ipinitial_ref` to invoke the FW function `initiateAuthenticationWithVersion` where the callback object reference `ipclientAPIlevelauthentication_ref` is included as a parameter. This request is sent to the AS stub `_lpInitialStub`.

**Step 2.** The stub `_lpInitialStub` marshals the parameters `clientDomain`, `authType` and `frameworkVersion` into the *Common Data Representation* (CDR) format and forwards the CDR data and the operation name `initiateAuthenticationWithVersion` to the AS ORB Core. The AS ORB Core uses the CDR data, the operation name `initiateAuthenticationWithVersion`, an object key and other information to construct a request message. Then the AS ORB Core sends this request message to the FW ORB Core.

**Step 3.** The FW ORB Core uses the object key in the request message to locate the target POA `ipinitialpoa` and delivers the request message to `ipinitialpoa`.

**Steps 4 and 5.** The FW POA `ipinitialpoa` uses the received object key to locate the servant `ipinitialimpl` in the `ipinitialpoa`'s active object map, and then activates the skeleton `lpInitialSkeleton`.

**Step 6.** The skeleton demarshals the parameters in the request message into arguments. Three arguments `clientDomain` (containing the callback reference `ipclientAPIlevelauthentication_ref` to be used in Step 10), `authType` (`P_OSA_AUTHENTICATION`) and `frameworkVersion` (`FWv1.0`) are passed as parameters to the function `initiateAuthenticationWithVersion` of the servant `ipinitialimpl`.

**Step 7.** The servant `ipinitialimpl` performs the function `initiateAuthenticationWithVersion` and returns the result or exceptions to the skeleton `lpInitialSkeleton`.

**Step 8.** `lpInitialSkeleton` marshals the related results returned by the servant into the CDR format and forwards the CDR data to the FW ORB Core. The FW ORB Core uses the CDR data and other information to construct a reply message and sends the reply message back to the AS ORB Core.

**Step 9.** The stub `_IpInitialStub` demarshals the results in the reply message and returns the result (i.e, `ipAPIlevelauthentication_ref`; see Step 1 in Figure 2.3) to the AS CORBA client.

The CORBA behavior for invoking the callback `challenge` from the FW to the AS is described in Steps 10–18 (which implement Step 5 in Figure 2.3).

**Step 10.** The FW CORBA client uses `ipclientAPIlevelauthentication_ref` (obtained from the argument `clientDomain` in Step 6) to invoke the callback function `challenge`. This request is sent to the FW stub `_IpClientAPILevelAuthenticationStub`.

**Step 11.** The stub `_IpClientAPILevelAuthenticationStub` marshals the parameter `challenge` into the CDR format and forwards the CDR data and the operation name `challenge` to the FW ORB Core. The FW ORB Core uses the CDR data, the operation name `challenge`, an object key and other information to construct a request message. Then the FW ORB Core sends this request message to the AS ORB Core.

**Step 12.** The AS ORB Core uses the object key in the request message to locate and delivers the request message to the target POA `rootpoa`.

**Steps 13 and 14.** POA `rootpoa` uses the received object key to locate the servant `ipclientAPIlevelauthenticationimpl` in its active object map, and activates the skeleton `IpClientAPILevelAuthenticationSkeleton`.

**Step 15.** Upon receipt of the request message, the AS skeleton `IpClientAPILevelAuthenticationSkeleton` retrieves the argument `challenge` and instructs the servant `ipclientAPIlevelauthenticationimpl` to execute the function `challenge`.

**Step 16.** After the function `challenge` is executed, the AS servant `ipclientAPIlevelauthenticationimpl` returns the result to the skeleton `IpClientAPILevelAuthenticationSkeleton`.

**Step 17.** `IpClientAPILevelAuthenticationSkeleton` marshals the related results returned by the servant into the CDR format and forwards the CDR data to the AS ORB Core. The AS

ORB Core uses the CDR data to construct a reply message and sends the reply message back to the FW ORB Core.

**Step 18.** The FW stub `_lpClientAPILevelAuthenticationStub` demarshals the result (`app_digest`) in the reply message and returns the result to the FW CORBA Client.

The above two examples illustrate how CCL OSA interaction can be implemented using CORBA technology.

## 2.5 Application Server Architecture

This section describes the proposed AS architecture with six modules (see Figure 2.5).

- An `appService` module (Figure 2.5 (a)) implements services by accessing SCF through the “lp” interfaces. Furthermore, it generates the interface objects in the corresponding `appService` callback module (to be elaborated at Step 3 in Figure 2.6, Section 2.6) and passes the references of these objects as the callback references to the SCFs.

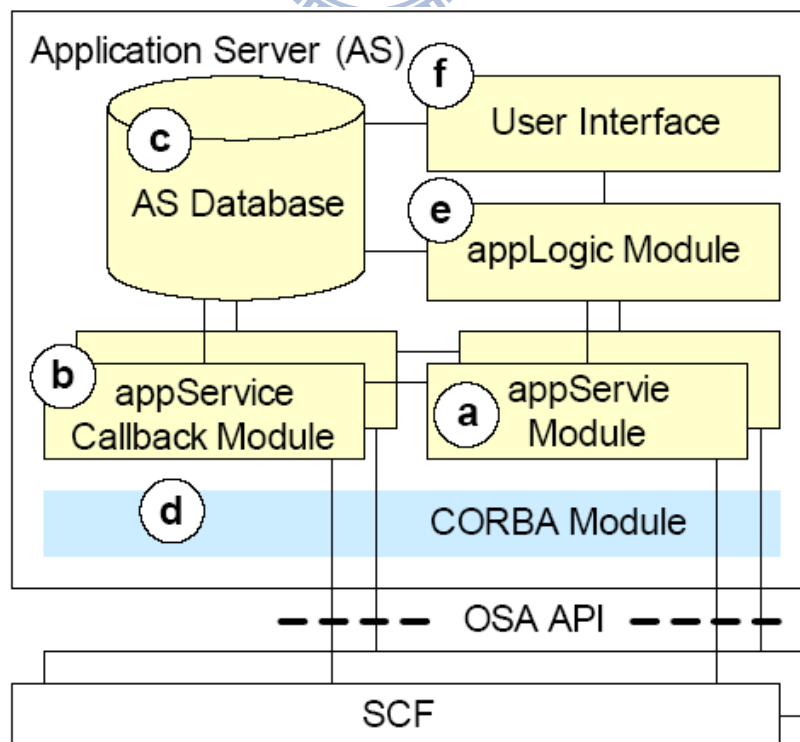


Figure 2.5 Application server architecture

- The appService callback module (Figure 2.5 (b)) provides the “IpApp” and “IpClient” callback interfaces to the SCFs/FW. Through these interfaces, the AS can receive specific events from the SCFs/FW. Upon receipt of the events, the appService callback module stores them in the AS database and may inform the appService module to handle the events. For each SCF, there exists one appService module and one appService callback module in our AS design.
- The AS database (Figure 2.5 (c)) stores the event information to be accessed by other modules. The information includes the list of the authorized SCFs, the events received from the SCFs, the log file of the AS, etc. The AS database module can be implemented by SQL database.
- The CORBA module (Figure 2.5 (d)) is an emerging open distributed object computing infrastructure. It provides the higher layers (i.e., other AS modules) a uniform view of underlying heterogeneous network, which is described in Section 2.2.
- The appLogic module (Figure 2.5 (e)) implements the logic of an AS application. It integrates services supported by different SCFs and the FW through appService modules without directly accessing the CORBA module and OSA interfaces.
- The user interface module (Figure 2.5 (f)) provides interfaces to monitor the AS operations, obtain the records stored in the AS database, and invoke the services offered by the SCFs.

## 2.6 Authentication between AS and Framework

An OSA AS must authenticate with the FW before it can access any SCFs. Before executing the OSA mutual authentication, the AS must obtain the FW reference `ipinitial_ref` of the initial contact interface object `IpInitial` from the CORBA module. This reference is obtained through a URL (e.g., `corbaname::pcs.csie.nctu.edu.tw:3500#IpInitial` in our implementation) and the *naming service* of the FW. This naming service provided by the CORBA module allows the CORBA client to locate the object through a URL.

Figure 2.6 illustrates the message flow of mutual authentication for initial access in our implementation. The FW mutually authenticates with the AS through the fwLogic module and the Access Session API module.

On the FW side, the fwLogic module (Figure 2.6 (a)) is the control logic of the FW that authenticates the AS at Steps 10 and 11 in Figure 2.6. The Access Session API module (Figure 2.6 (b)) provides the initial contact interface IpInitial and the FW authentication interface IpAPILevelAuthentication to communicate with the AS (see Steps 4, and 6~8 in Figure 2.6). Details of the OSA mutual authentication are described as follows:

**Steps 1 and 2.** The AS control logic appLogic first generates the appFw object to access the FW. appLogic invokes the appFw function authenticationReq to start mutual authentication procedure with the FW.

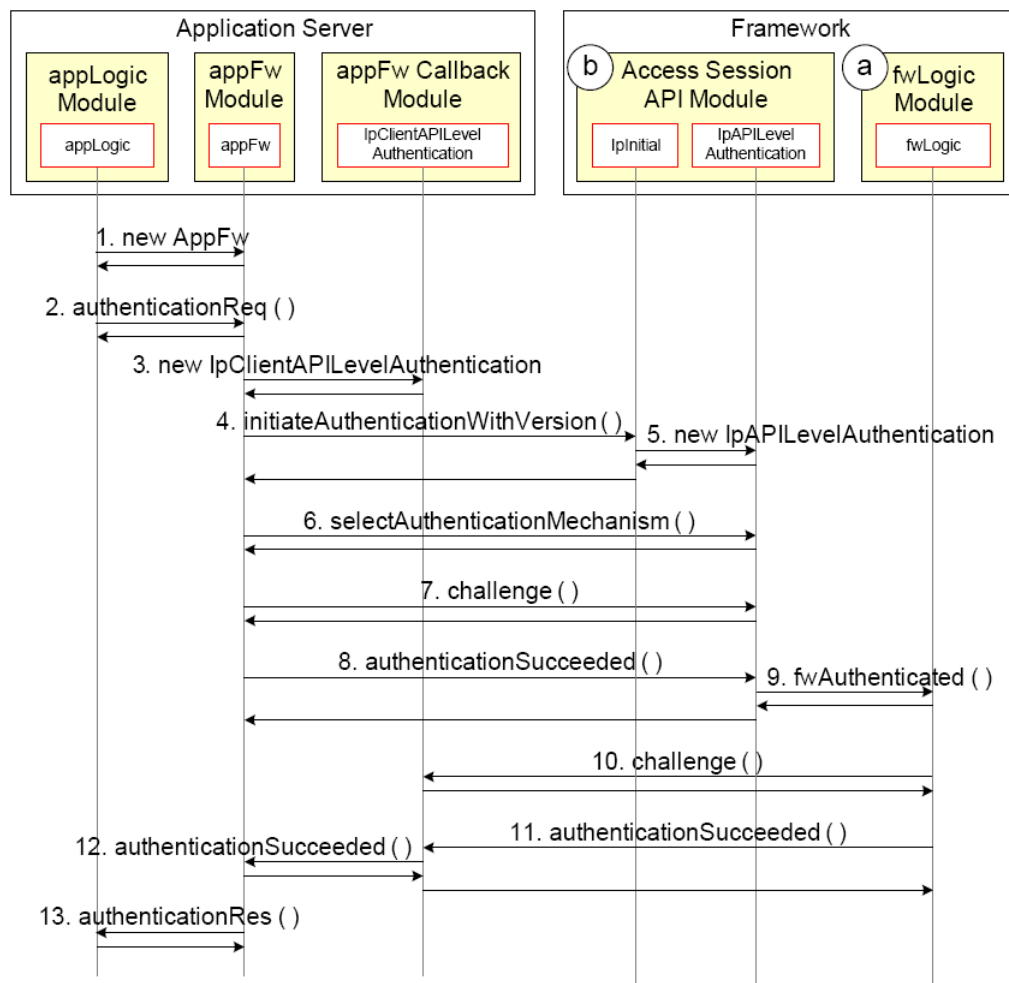


Figure 2.6 Message flow for OSA mutual authentication

**Step 3.** appFw creates the IpClientAPILevelAuthentication object that provides authentication-related functions to be invoked by the FW (e.g., function challenge invoked at Step 10 and function authenticationSucceeded invoked at Step 11).

**Step 4.** appFw generates the reference ipclientAPIlevelauthentication\_ref of the IpClientAPILevelAuthentication object. This reference will be used by the FW to invoke the callback functions to authenticate the AS at Steps 10 and 11. appFw uses ipinitial\_ref to invoke the FW function initiateAuthentication- WithVersion that includes the callback object reference ipclientAPIlevel- authentication\_ref as a parameter.

**Step 5.** The FW creates the IpAPILevelAuthentication object that provides the authentication-related functions to be invoked by the AS (e.g., function selectAuthenticationMechanism invoked at Step 6, function challenge invoked at Step 7, and function authenticationSucceeded invoked at Step 8). Then IpInitial generates the reference ipAPIlevelauthentication\_ref of IpAPILevelAuthentication. This reference is returned to the AS as the return value of function initiateAuthenticationWithVersion invoked at Step 4.

**Step 6.** Through the ipAPIlevelauthentication\_ref reference, appFw selects the authentication algorithm by invoking the FW function selectAuthentication- Mechanism. In our implementation, the default algorithm is SHA-1 [10, 23].

**Step 7.** appFw authenticates the FW by invoking function challenge through the FW reference ipAPIlevelauthentication\_ref. This function takes a random number as the parameter (in the byte-stream format). Then the FW IpAPILevelAuthentication object executes the authentication algorithm SHA-1 using the received random number and returns the result to the AS.

**Step 8.** appFw checks the returned result from the FW. If the FW is successfully authenticated, appFw invokes function authenticationSucceeded through ipAPIlevelauthentication\_ref to inform the FW of the authentication result.

**Step 9.** The FW `IpAPILevelAuthentication` object invokes the function `fwAuthenticated` to inform `fwLogic` of the result.

**Step 10.** `fwLogic` authenticates the AS by invoking the callback function `challenge` through the AS reference `ipclientAPIlevelauthentication_ref`. Similar to Step 7, this function takes a random number as the parameter. The AS `IpClientAPILevelAuthentication` object executes the authentication algorithm SHA-1 using the received random number and returns the result to the FW.

**Steps 11 and 12.** `fwLogic` checks the returned result from the AS. If the AS is successfully authenticated, `fwLogic` invokes the callback function `authenticationSucceeded` through `IpClientAPILevelAuthentication` to inform `appFw` of the result.

**Step 13.** At this point, the mutual authentication procedure is complete, and `appFw` invokes function `authenticationRes` to inform `appLogic` of the successful mutual authentication.

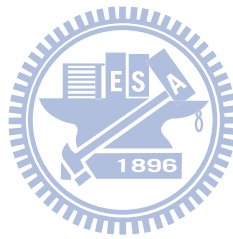
After mutual authentication with the FW, the AS can identify the authorized SCFs through the FW discovery mechanism. To select the SCFs for service (e.g., the PoC SCF and the GLMS SCF), the AS first establishes the service agreement with the FW. Then the AS can obtain the object references of the PoC SCF and the GLMS SCF for PoC service. The details are omitted.

## 2.7 Summary

This chapter described a CORBA-based OSA system we designed and developed in CCL/ITRI, and proposed an OSA AS architecture. In the CORBA-based OSA system, we showed how OSA API interfaces and functions can be implemented by CORBA clients, stubs, servants and skeletons. We described how CORBA POA and ORB are set up for CCL OSA. Then we used the authentication procedure for initial access to illustrate how CORBA mechanism works for CCL OSA. One of the challenges for future OSA extension is to deploy open API-based services that support both legacy circuit-switched networks and all-IP based networks.

In our proposed OSA AS architecture, a new application is created by implementing the

appLogic module that invokes the SCFs through the appService modules and appService callback modules. To interact with an SCF, the AS implements one appService module and one appService callback module for that SCF. When the existing SCFs are reused for new services, the corresponding appService and appService callback modules for accessing these SCFs can also be reused. Through this modularized AS design, the service deployment can be sped up. We used the OSA mutual authentication to illustrate the interaction within the AS modules and how the AS interacts with the FW. Details of the interaction among the OSA AS and the SCFs for PoC services will be elaborated in Appendix A.





## Chapter 3.

### Selecting Transition Process for WLAN Security

*Wireless local area network* (WLAN) was originally designed for cable replacement. In the UMTS All-IP network, WLAN functionality has been extended for users with mobility and has even been integrated with cellular system to serve as an access technology to the cellular system, and therefore scales up the coverage of mobile services. In WLAN, an *Mobile Station* (MS) accesses the IMS network through an *access point* (AP). When switching from one AP to another, the MS executes the transition process, which may incur long delay and result in force-termination for real-time applications. The IEEE 802.11r proposes the fast *basic service set* (BSS) transition to speed up the transition process for a MS moving within the same *mobility domain* (MD). This scheme requires unique MD assignment so that the MS knows whether it should conduct fast BSS transition process (for intra-MD scenario) or the expensive initial MD association process (for inter-MD scenario). However, how to guarantee unique *MD identifier* (MDID) assignment is not mentioned in the specification. This chapter proposes a mechanism for IEEE 802.11r fast transition without using MDID, and therefore eliminates the cost for MDID management.

#### 3.1 Introduction to IEEE 802.11r

In IEEE 802.11 WLAN, a *Mobile Station* (MS; Figure 3.1 (1)) accesses the IMS network through an *Access Point* (AP; Figure 3.1 (2)) over the air [9, 15]. Because WLAN is much easier to be intercepted than a wired network, it is important to exercise authentication and encryption between the MS and the AP with a security key. If the MS and the AP are not assigned the same security key through an offline process, the key must be generated in the

transition process. When an MS connects to an AP for the first time or switches from one AP to another, the transition process consisting of the following four procedures is exercised.

- *IEEE 802.11 open system authentication* is performed between the MS and the AP. The AP will grant any authentication request from the MS unless the open system authentication is disabled in the AP.
- *Association* enables the MS and the AP to negotiate the security policy and the encryption algorithm.
- *IEEE 802.1X authentication* is executed between the MS and an authentication, authorization, and accounting server (AAA server; Figure 3.1 (3)) [21]. A *Master Session Key* (MSK; or AAA-Key) is generated independently in both the MS and the AAA server.
- *IEEE 802.11i 4-way handshake* generates the *Pairwise Transient Key* (PTK) from the MSK. This PTK provides data integrity and confidentiality by encrypting data transmitted between the MS and the AP.

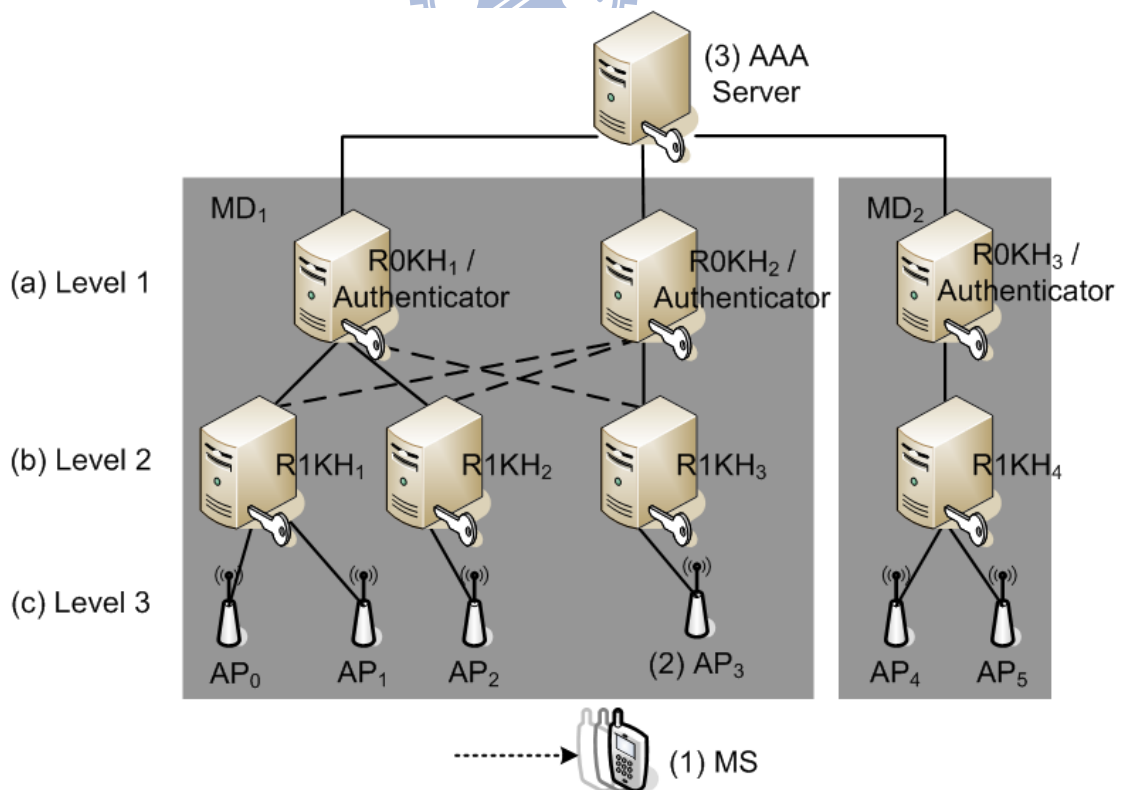


Figure 3.1 IEEE 802.11r security hierarchy

This transition process may incur long delay and result in force-termination for real-time applications such as voice over IP. To reduce the latency for security key generation in the transition process, IEEE 802.11r proposed fast *Basic Service Set* (BSS) transition implemented in a three-level key hierarchy [15]. In this hierarchy, the first-level key is *Pairwise Master Key first level* (PMK-R0) derived from the MSK and is shared between the MS and the *PMK-R0 Key Holder* (R0KH; Figure 3.1 (a)). The R0KH is the authenticator that maintains the MSK received from the AAA server. The second-level key is *PMK second level* (PMK-R1) shared between the MS and the *PMK-R1 Key Holder* (R1KH; Figure 3.1 (b)). PMK-R1 is derived from PMK-R0 and is used to derive PTK at the third-level (i.e., the AP; Figure 3.1 (c)).

*Mobility domain* (MD) is defined in this three-level key hierarchy. An MD consists of several R0KHs. Figure 3.1 illustrates two mobility domains MD<sub>1</sub> and MD<sub>2</sub>. MD<sub>1</sub> consists of two R0KHs (i.e., R0KH<sub>1</sub> and R0KH<sub>2</sub>) and MD<sub>2</sub> consists of one R0KH (i.e., R0KH<sub>3</sub>). Each R0KH directly associates with several nearby R1KHs that can acquire PMK-R1 from this R0KH. In Figure 3.1, R0KH<sub>1</sub> directly associates with two R1KHs (i.e., R1KH<sub>1</sub> and R1KH<sub>2</sub>; the connectivity is represented by solid links) and R0KH<sub>2</sub> directly associates with one R1KH (i.e., R1KH<sub>3</sub>). An R1KH can obtain PMK-R1 from an R0KH which does not directly associate with this R1KH, if both key holders are in the same MD. In Figure 3.1, R1KH<sub>1</sub> and R1KH<sub>2</sub> can acquire PMK-R1 from R0KH<sub>2</sub>, and this indirect association is represented through the dashed links. Based on the MD structure, there are three MS transition scenarios: intra-R1KH transition (e.g., from AP<sub>0</sub> to AP<sub>1</sub> in Figure 3.1), inter-R1KH transition in an MD (e.g., from AP<sub>1</sub> to AP<sub>2</sub> or AP<sub>3</sub>), and initial MD association (or inter-MD transition; e.g., from AP<sub>3</sub> to AP<sub>4</sub>) scenarios. In this chapter, the third scenario is described in Section 3.2. The first two scenarios are intra-MD scenarios described in Section 3.3. Section 3.4 proposes a selection mechanism that automatically selects the appropriate transition process for intra- and inter-MD scenarios.

### 3.2 Initial MD Association Process (Inter-MD Scenario)

When an MS first associates to the WLAN or moves from one MD to another (inter-MD transition), PMK-R0 and PMK-R1 for this MS are generated and stored in the R0KH and the R1KH, respectively [15]. Assume that an MS first connects to AP<sub>1</sub> in Figure 3.1. The steps in Figure 3.2 are executed as follows.

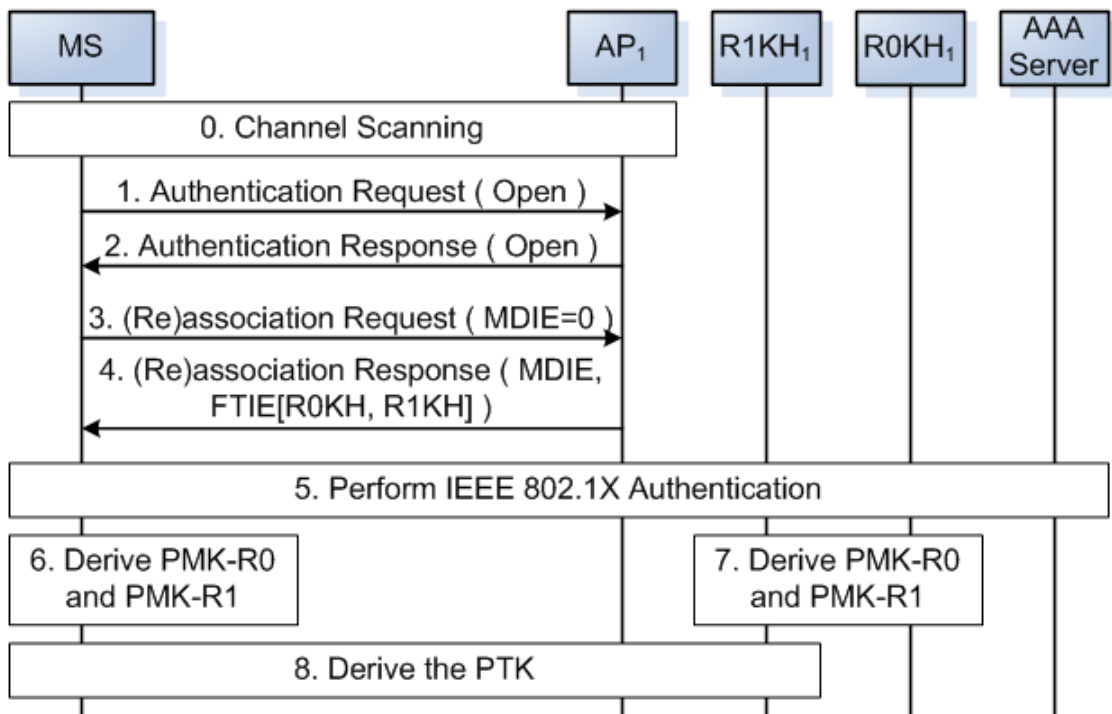


Figure 3.2 IEEE 802.11r initial MD association process

**Step 0.** The MS scans multiple channels for an AP with good signal, and checks the *information elements* (IEs) advertised in the signal (i.e., Beacon and Probe Response frames). If the *mobility domain IE* (MDIE) is included in the frames, it means that the AP supports the IEEE 802.11r fast BSS transition. The MDIE contains the *mobility domain identifier* (MDID) of MD<sub>1</sub> and the transition policy.

**Steps 1 and 2.** Suppose that AP<sub>1</sub> is selected. The MS sends an authentication request message to AP<sub>1</sub> and performs the open system authentication.

**Steps 3 and 4.** The MS sends an association request message to AP<sub>1</sub>. In this message, the MDIE field is set to 0 to indicate that the initial MD association process is exercised. AP<sub>1</sub> replies an association response message to indicate the MDID of MD<sub>1</sub>, and *fast BSS transition IE* (FTIE) which includes the identifiers of R0KH<sub>1</sub> and R1KH<sub>1</sub>.

**Step 5.** The IEEE 802.1X authentication is executed between the MS and the AAA server. A new MSK is independently generated in the MS and the AAA server. This key is also passed from the AAA server to R0KH<sub>1</sub>. Details of the IEEE 802.1X authentication will be illustrated at Steps 2~6 in Figure 4.2 and will be described in Section 4.2.

**Steps 6 and 7.** The MS and R0KH<sub>1</sub> independently generate the PMK-R0 key by executing the *key derivation function* (KDF; Figure 3.3 (1)) with the R0KH<sub>1</sub> identifier, the MSK (generated at Step 5), the MD<sub>1</sub> identifier, and the *supplicant address* (SPA; the MS's *Medium Access Control* (MAC) address). Then PMK-R1 is derived from PMK-R0, SPA, and the R1KH<sub>1</sub> identifier (Figure 3.3 (2)).

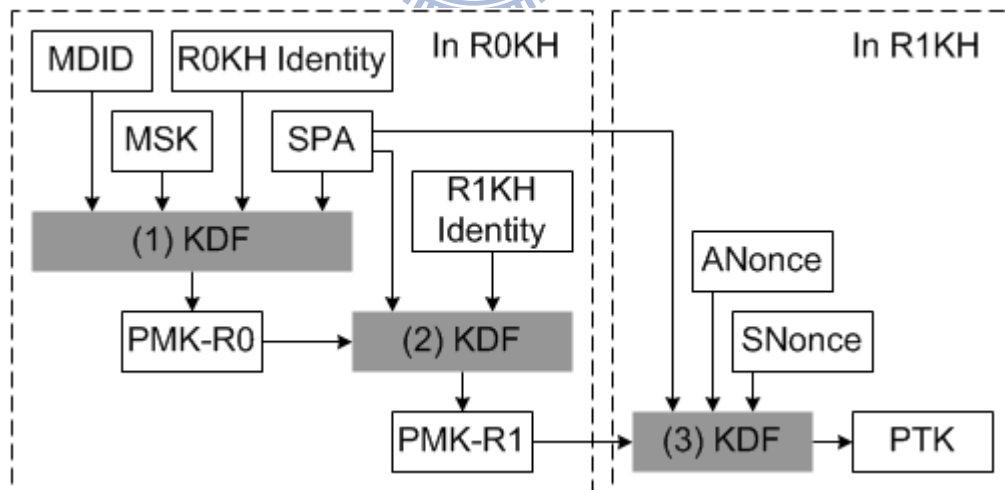


Figure 3.3 Derivation of PMK-R0, PMK-R1, and PTK

**Step 8.** The MS and AP<sub>1</sub> perform the IEEE 802.11i 4-way handshake procedure by exchanging two random numbers (i.e., ANonce generated by AP<sub>1</sub> and SNonce generated by the MS). To derive PTK, AP<sub>1</sub> sends the related parameters to R1KH<sub>1</sub>, including the R0KH<sub>1</sub>

identifier, SNonce, ANonce, and SPA. Then the MS and R1KH<sub>1</sub> independently derive the PTK key from inputs including the PMK-R1 generated at Steps 6 and 7 (Figure 3.3 (3)). After PTK is generated, R1KH<sub>1</sub> passes it to AP<sub>1</sub> for encrypting and decrypting data transmitted between the MS and AP<sub>1</sub>.

After this initial MD association process, PMK-R0 and PMK-R1 are kept in R0KH<sub>1</sub> and R1KH<sub>1</sub>, respectively. When the MS moves to another AP in MD<sub>1</sub>, these two keys are reused to generate new PTK without executing the IEEE 802.1X authentication.

### 3.3 Fast BSS Transition (Intra-MD Scenarios)

The IEEE 802.11r fast BSS transition is exercised in intra-MD transition scenarios. When the MS performs inter-R1KH transition from AP<sub>1</sub> to AP<sub>3</sub> in MD<sub>1</sub>, the following steps are executed (see Figure 3.4).

**Step 1.** Since AP<sub>1</sub> and AP<sub>3</sub> are in the same MD, the MS sends an authentication request message with *fast BSS transition* (FT) authentication to AP<sub>3</sub>. This message contains the MDID of MD<sub>1</sub> and FTIE (containing the R0KH<sub>1</sub> identifier and a random number SNonce for PTK derivation).

**Step 2.** From the MDIE and the FTIE provided by the MS, AP<sub>3</sub> knows that the inter-R1KH transition occurs. AP<sub>3</sub> sends an authentication response message to the MS. This response message contains the identifiers of R0KH<sub>2</sub>, R1KH<sub>3</sub>, MD<sub>1</sub>, and a random number ANonce for deriving PTK.

**Step 3.** Upon receipt of ANonce and the identifiers from AP<sub>3</sub>, the MS generates a new PMK-R1 key from the R1KH<sub>3</sub> identifier, SPA, and PMK-R0 (Figure 3.3 (2)). This PMK-R1 key is shared between the MS and R1KH<sub>3</sub>, and is used together with SPA, ANonce, and SNonce to derive the PTK key (Figure 3.3 (3)). If the MS moves from AP<sub>1</sub> to AP<sub>0</sub> connecting to the same R1KH in Figure 3.1, the old PMK-R1 is reused to generate the new PTK key.

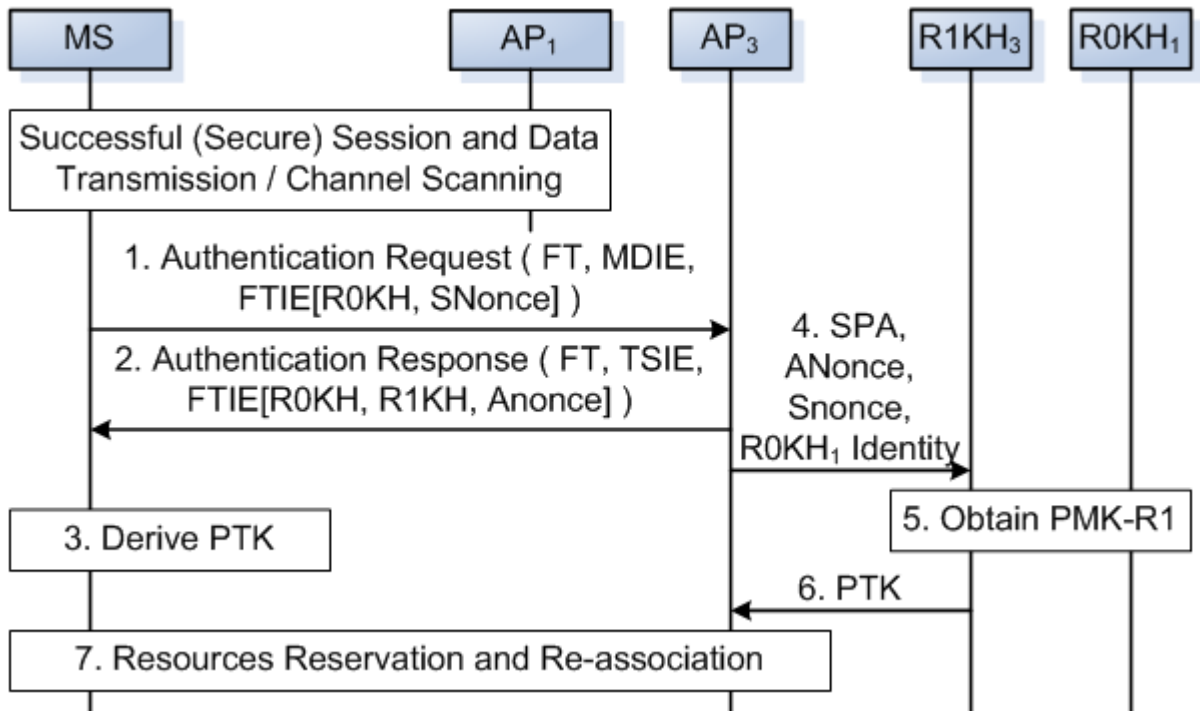


Figure 3.4 IEEE 802.11r fast BSS transition process

**Step 4.** AP<sub>3</sub> sends SPA, ANonce, SNonce, and the R0KH<sub>1</sub> identifier to R1KH<sub>3</sub> for deriving new PTK.

**Step 5.** According to the R0KH<sub>1</sub> identifier, R1KH<sub>3</sub> acquires the new PMK-R1 from R0KH<sub>1</sub>. If the MS moves from AP<sub>1</sub> to AP<sub>0</sub> in Figure 3.1, this step is omitted.

**Step 6.** R1KH<sub>3</sub> derives PTK by executing KDF (Figure 3.3 (3)) and sends the generated PTK to AP<sub>3</sub>. After this step, PTK is kept in both the MS and AP<sub>3</sub>.

**Step 7.** The MS switches from AP<sub>1</sub> to AP<sub>3</sub> after resource reservation and re-association between the MS and AP<sub>3</sub>.

In the fast BSS transition, the IEEE 802.1X authentication procedure is omitted. Instead, PMK-R0 is reused to derive the new PTK key to speed up the transition process.

### 3.4 Transition Process Selection Mechanism

Since every AP advertises the MDID in the Beacon and Probe Response frames, the MS can select the appropriate transition process for intra-MD or inter-MD scenarios. Specifically, the

MAC addresses are used as the identifiers for R0KH and R1KH to ensure global uniqueness. The MDID is assumed to be managed by vendors [15]. However, it is not clear how to guarantee unique MDID among vendors. If ambiguity of MDID does occur, this error will be detected at Step 5 in Figure 3.4 because the new PMK-R1 cannot be acquired. Therefore the MS is forced to stop the fast BSS transition process and is switched to perform the IEEE 802.11r initial MD association process.

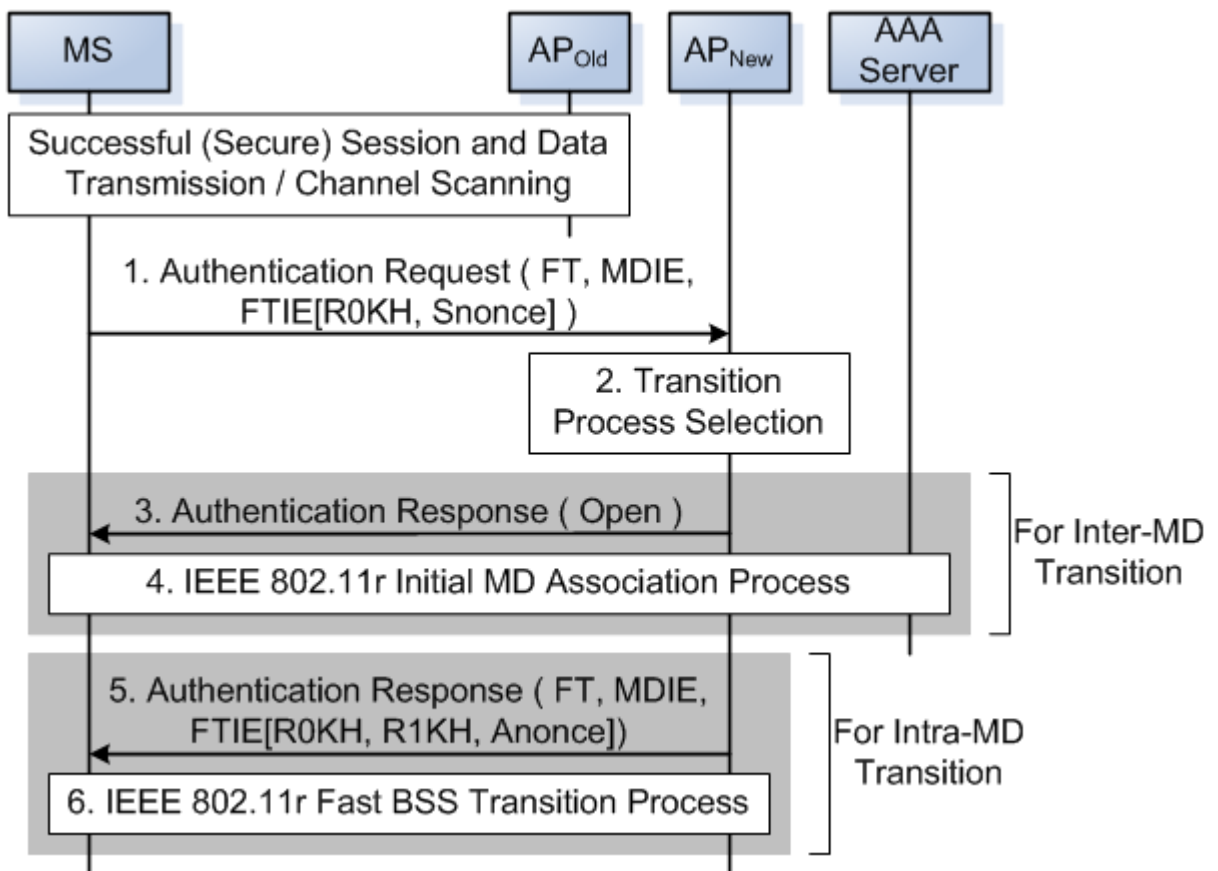


Figure 3.5 The proposed selection mechanism for transition process

To resolve the ambiguous MDID issue, we propose a new method that does not require the MDID for transition. In our approach, every AP maintains an R0KH table recording all R0KHs that can be accessed by the AP. In Figure 3.1, the identifiers of R0KH<sub>1</sub> and R0KH<sub>2</sub> are recorded in AP<sub>0</sub>, AP<sub>1</sub>, AP<sub>2</sub>, and AP<sub>3</sub>, and the R0KH<sub>3</sub> identifier is recorded in AP<sub>4</sub> and AP<sub>5</sub>. Upon receipt



of the authentication request message (i.e., Step 1 in Figure 3.4), an AP queries its R0KH table to determine whether the MS comes from another MD, and selects the appropriate transition process for execution. Suppose that the MS moves from AP<sub>Old</sub> to AP<sub>New</sub>. The following steps are executed (see Figure 3.5).

**Step 1.** The MS sends an authentication request message with FT authentication algorithm to AP<sub>New</sub>. This message is similar to that in Step 1 of Figure 3.4, but does not include the MDID in MDIE.

**Step 2.** Upon receipt of the authentication request, AP<sub>New</sub> checks the R0KH identifier in FTIE. There are two possibilities. If the R0KH identifier is not found in the R0KH table of AP<sub>New</sub>, Steps 3 and 4 are executed (for inter-MD scenario). Otherwise, Steps 5 and 6 are executed (for intra-MD scenarios).

**Steps 3 and 4.** AP<sub>New</sub> exercises open system authentication and replies the authentication response message with parameter “open system authentication”. Then the MS proceeds to execute the IEEE 802.11r initial MD association process (Steps 2~8 in Figure 3.2).

**Steps 5 and 6.** AP<sub>New</sub> exercises FT authentication and replies the IEEE 802.11 authentication response message with parameter “FT authentication”. The IEEE 802.11r fast BSS transition process is executed (Steps 2~7 in Figure 3.4).

Through the R0KH table in an AP, the above mechanism correctly distinguishes the inter-MD scenario from the intra-MD scenarios without using the MDID.

A few other studies have been reported in the literature which has also carried out research similar to that reported in this chapter [5, 17, 27].

### 3.5 Summary

This chapter describes the IEEE 802.11r transition process for WLAN, where a three-level key hierarchy was proposed to speed up the transition process without executing the expensive IEEE 802.1X authentication for some scenarios. This hierarchy requires assignment of unique

MDIDs worldwide. However, how to guarantee the uniqueness of MDID is not clear. This chapter proposed a mechanism that does not need MDID, and therefore MDID management is eliminated. This mechanism also saves four message exchanges incurred in the original fast BSS transition when MDID ambiguity occurs.



## Chapter 4.

# A Key Caching Mechanism for Reducing WiMAX

## Authentication Cost in Handoff

IEEE 802.16e mobile *Worldwide Interoperability for Microwave Access* (WiMAX) provides broadband wireless services with wide service coverage, high data throughput, and high mobility. To access the IMS network with mobile WiMAX, several mobile telecommunications network issues, e.g., mobility management, voice quality, and power saving, must be addressed in the mobile WiMAX environment. Among them, security is probably one of the most important and essential issue that must be carefully addressed, which includes authentication and encryption aspects. This chapter will focus on the authentication aspect for mobile WiMAX.

The IEEE 802.1X is utilized in mobile WiMAX authentication. This procedure incurs long delay in WiMAX handoff. To resolve this issue, this chapter proposes a key caching mechanism to eliminate the non-necessary IEEE 802.1X authentication cost in WiMAX handoff. This mechanism is investigated through analytic and simulation modeling. Our study indicates that the key caching scheme can effectively speed up the handoff process.

### 4.1 Introduction to WiMAX AAA Architecture

To support security network access, the *authentication, authorization, and accounting* (AAA) mechanism is exercised in WiMAX [38]. Figure 4.1 shows the AAA architecture and protocol stack for WiMAX. In this architecture, the *Access Service Network* (ASN; Figure 4.1 (2)) consists of *Base Stations* (BSs; Figure 4.1 (4)) and *ASN Gateways* (ASN-GWs; Figure 4.1 (5)).

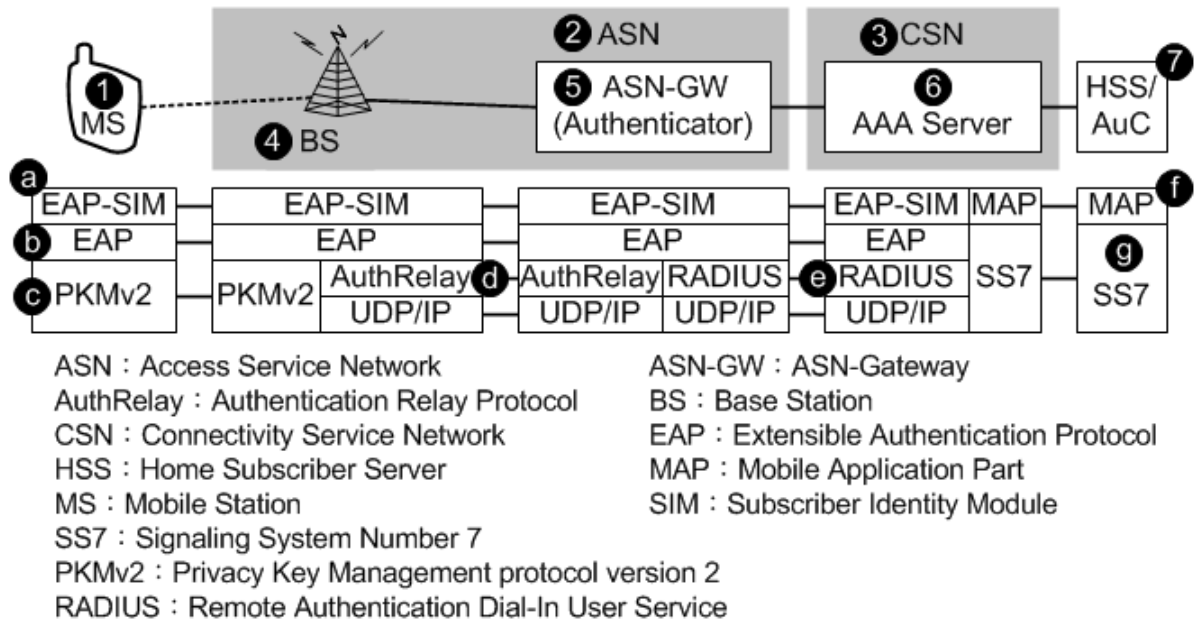


Figure 4.1 WiMAX AAA architecture and protocol stack

An ASN-GW controls several BSs. A BS provides WiMAX radio access for *Mobile Stations* (MSs; Figure 4.1 (1)) after the MSs are authenticated by the AAA server (Figure 4.1 (6)) in the *Connectivity Service Network* (CSN; Figure 4.1 (3)). In the WiMAX AAA architecture, the ASN-GW serves as the authenticator for the MS. The authenticator is responsible for forwarding authentication messages between the MS and the AAA server, and for maintaining the MS related information (e.g., encryption keys) after authentication. We assume that the *Subscriber Identity Module* (SIM)-based *Extensible Authentication Protocol* (EAP) is utilized for AAA [12]. Note that this approach reuses the authentication mechanism in mobile telecommunications [8]. In the authentication procedure, an EAP-SIM message (Figure 4.1 (a)) is encapsulated in an EAP message (Figure 4.1 (b)). The MS then encapsulates the EAP message in *Privacy Key Management protocol version 2* (PKMv2; Figure 4.1 (c)) before it is transmitted to the BS. The BS exercises the *Authentication Relay protocol* (AuthRelay; Figure 4.1 (d)) to forward the received EAP message to the authenticator (i.e., the ASN-GW). Upon receipt of an EAP message, the authenticator translates it into a *Remote Authentication Dial-In User Service* (RADIUS; Figure 4.1 (e)) message. Then the RADIUS message is sent to the

AAA server. Upon receipt of the RADIUS message, the AAA server utilizes the *Mobile Application Part* (MAP; Figure 4.1 (f)) of the *Signaling System Number 7* protocol (SS7; Figure 4.1 (g)) to communicate with the *Home Subscriber Server* (HSS)/Authentication Center (AuC; Figure 4.1 (7)). The HSS is the mobility database of the GSM/UMTS mobile telecommunication networks [8, 21]. The AuC maintains the secret keys of the MSs, and provides the authentication information to the AAA server.

## 4.2 WiMAX Initial Network Entry Process

By using the protocols described in Figure 4.1, the WiMAX authentication works as follows. Suppose that an MS first connects to the WiMAX network, the following steps are executed for the initial network entry process (see Figure 4.2):

**Step 1.** The MS, the BS, and the ASN-GW (authenticator) negotiate the security policy (i.e., to select the encryption and decryption algorithms) and the authorization policy; specifically, to select the *message authentication code* (MAC) type.

**Step 2.** The authenticator sends an EAP Request message to the MS. This message initiates the IEEE 802.1X authentication procedure by requesting the user identity.

**Steps 3 and 4.** The MS replies an EAP Response message with the user identity to the authenticator. The user identity consists of two elements: the AAA server address AAA-addr and the user account User-acct. In the SIM-based EAP authentication, the user account is set to the *International Mobile Subscriber Identity* (IMSI) of the MS [21]. According to AAA-addr, the authenticator forwards the EAP Response message to the AAA server.

**Step 5.** Upon receipt of the user identity, the AAA server performs the SIM-based EAP authentication with the MS as follows:

**Step 5.1.** The AAA server issues an EAP Request message with type “Start” to the MS.

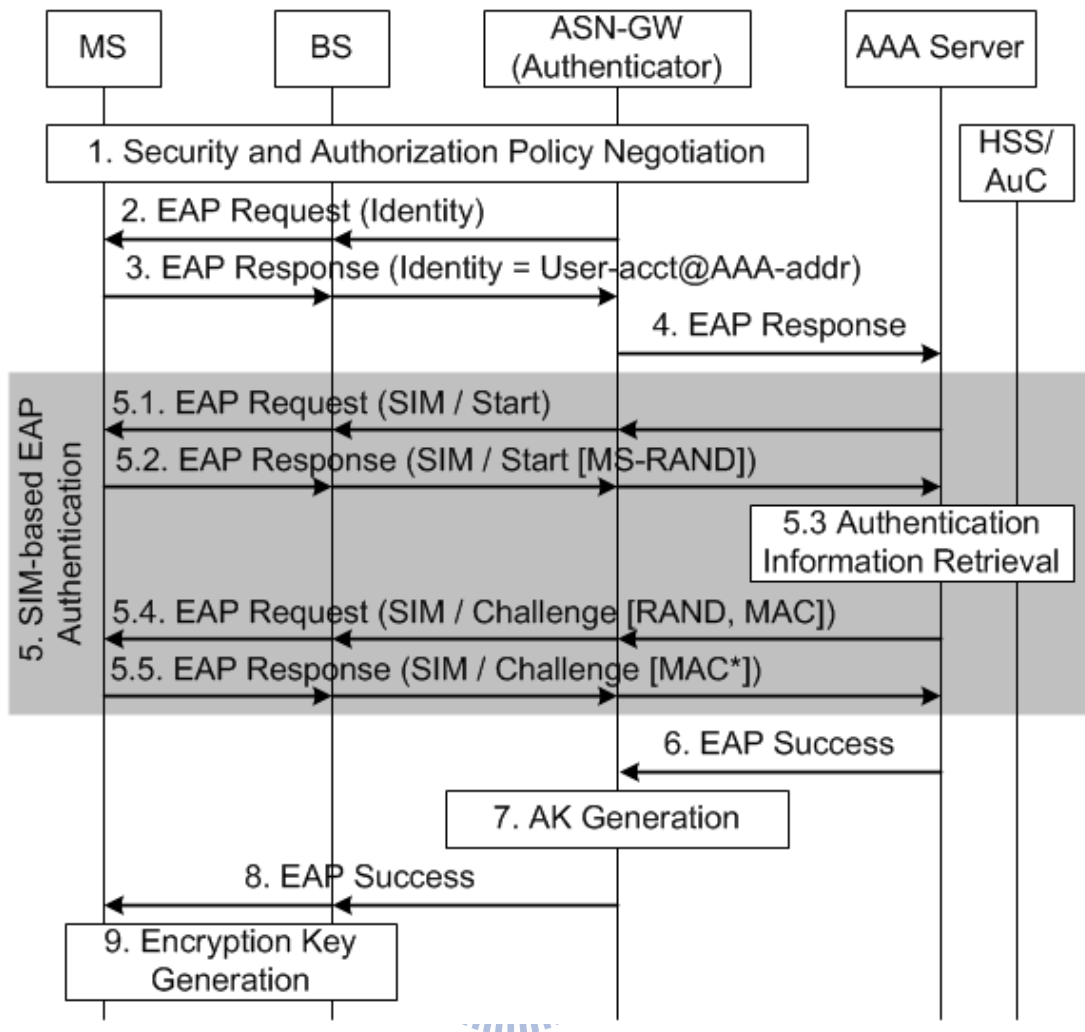


Figure 4.2 WiMAX initial network entry process

**Step 5.2.** The MS replies the EAP Response message containing a random number MS-RAND. This random number is used to derive the encryption keys in Steps 5.3 and 5.5.

**Step 5.3.** Based on the IMSI received in Step 4, the AAA server communicates with the HSS/AuC to obtain the authentication information, including a random number RAND, a signed result SRES, and a cipher key Kc. Both the MS and the HSS/AuC utilize the RAND and the secret key Ki (stored in the SIM card and the HSS/AuC) to execute the A3 and the A8 algorithms for deriving the signed result SRES and the cipher key Kc [20]. Then the AAA server utilizes Kc and MS-RAND (received in Step 5.2) to derive the *Master Session Key* (MSK) and

the EAP integrity key  $K_{EAP}$ .

**Step 5.4.** The AAA server sends a challenge EAP Request message with the RAND and the MAC. This MAC is derived from  $K_{EAP}$  and is used to ensure integrity of this message.

**Step 5.5.** Upon receipt of the EAP Request message, the MS utilizes RAND, MS-RAND (generated in Step 5.2), and  $K_i$  (stored in the SIM card) to generate  $SRES^*$ ,  $K_c$ , MSK, and  $K_{EAP}$ . With  $K_{EAP}$  and the received RAND, the MS verifies the received MAC. If the MAC is correct, the AAA server is successfully authenticated by the MS. Then the MS replies a challenge EAP Response message with a code  $MAC^*$  derived from  $K_{EAP}$  and  $SRES^*$ .

**Step 6.** The AAA server verifies  $MAC^*$  by using  $K_{EAP}$  (generated in Step 5.3) and SRES (received in Step 5.3). If  $MAC^*$  is correct, the MS is successfully authenticated by the AAA server. The AAA server sends the EAP Success message to the authenticator containing MSK (generated in Step 5.3), the MSK lifetime, and the MS authorization profile (e.g., service restrictions and supplementary services). The MSK lifetime is the period that the MS is authorized to access the ASN-GW. When the MSK lifetime is expired, the MS should execute the IEEE 802.1X authentication with the AAA server again.

**Step 7.** The ASN-GW stores MSK, the MSK lifetime, and the authorization profile. Then it derives the *Authentication Key* (AK) by using the MSK and the BS address. This AK is shared between the MS and the BS.

**Step 8.** The ASN-GW forwards the EAP Success message to the BS with AK. The BS passes the EAP Success message to inform the MS that the authentication is successful. Upon receipt of this message, the MS generates its version of AK.

**Step 9.** The BS generates the final encryption key *Traffic Encryption Key* (TEK). This encryption key is used to provide data integrity and confidentiality for a communication

session between the MS and the BS. The BS passes the generated TEK (encrypted by AK) to the MS.

The relationship of WiMAX encryption keys and the locations maintaining these keys are shown in Figure 4.3.

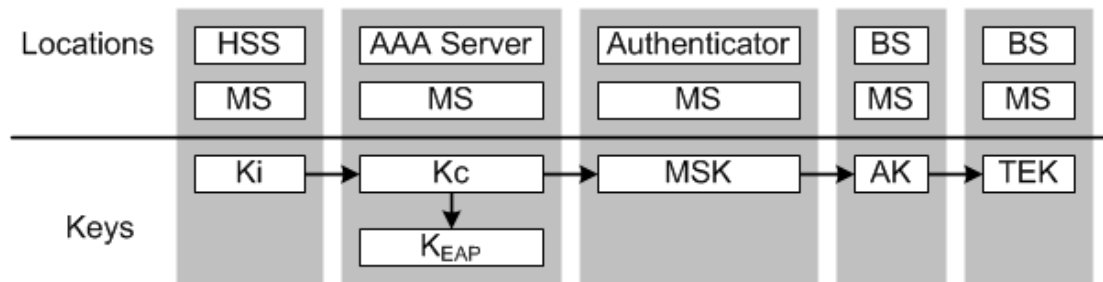


Figure 4.3 WiMAX key derivation tree

If the MS moves from the old BS to the new BS connecting to a different authenticator (ASN-GW), a new MSK must be generated in this inter-ASN-GW handoff process, which is the same as the initial network entry process described in Figure 4.2. In this case, the authenticator (ASN-GW) of the old BS will remove the MS key record (i.e., MSK, the MSK lifetime, and the MS authorization profile). When the MS moves back to the old ASN-GW again, another inter-ASN-GW handoff process should be performed, which may incur long delay.

### 4.3 The Key Caching Mechanism

To speed up the inter-ASN-GW handoff process, we propose a *key caching* mechanism. The idea is simple: When the MS moves from the old ASN-GW to the new ASN-GW, the old ASN-GW still keeps the MS key record. If the MS returns to the old ASN-GW before the MSK lifetime expires, it can reuse the MSK without executing the IEEE 802.1X authentication. That is, only Steps 1 and 9 in Figure 4.2 are executed to speed up the inter-ASN-GW handoff process. In Figure 4.2, Step 1 contains 2 message exchanges and Step 9 contains 5 message exchanges



[38]. Therefore, the caching mechanism speeds up the process by saving 50% ( $= 7/14$ ) message exchanges between the MS and the BS.

Although the key caching mechanism may effectively avoid the execution of IEEE 802.1X authentication, it consumes extra storage to keep the MS key records at the old ASN-GW, where a stored key record includes 512 or 1024 bits for MSK, 32 bits for the MSK lifetime, and 512 or 1024 bits for the MS authorization profiles. Therefore, it is desirable to select an appropriate MSK lifetime to eliminate the IEEE 802.1X authentication without consuming too much extra storage in the ASN-GW. We investigate the effect of the MSK lifetime on the caching performance by an analytic model described below.

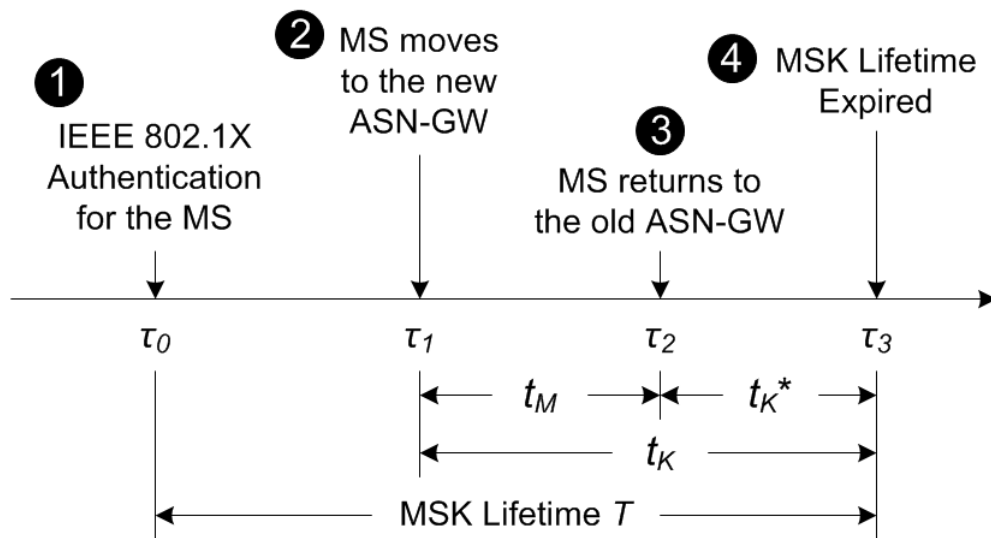


Figure 4.4 Relationship of the MSK key lifetime and the MS movement

Figure 4.4 illustrates the relationship between the movement of an MS and its MSK lifetime. In this figure, the IEEE 802.1X authentication is executed at time  $\tau_0$  (Figure 4.4 (1)), and the MSK lifetime expires at time  $\tau_3$  (Figure 4.4 (4)). At time  $\tau_1$  (Figure 4.4 (2)), the MS moves from the old ASN-GW to the new ASN-GW. The residual MSK lifetime is  $t_K = \tau_3 - \tau_1$ . If the MS will not return to the old ASN-GW before the MSK lifetime expires, we call this  $t_K$  period the unused key period. At time  $\tau_2$  (Figure 4.4 (3)), the MS returns to the old ASN-GW.

Let  $t_M = \tau_2 - \tau_1$  be the period between when the MS leaves the old ASN-GW and when it returns. If the MS returns before the MSK lifetime expires, the MS can reuse the MSK for period  $t_K^* = t_K - t_M$  without executing the IEEE 802.1X authentication. Period  $t_K^*$  is referred to as the reused key period. We make the following assumptions:

- We consider two distributions for the MSK lifetime  $T$ . That is,  $T$  is either an Exponential period with rate  $\mu$  or a fixed period.
- The MS residence time  $t_M$  in new ASN-GWs has the density function  $f(t_M)$  with mean  $1/\lambda$  and variance  $V_M$ .

Three output measures are evaluated in our study.

- $\alpha$ : the probability that the MS returns to the old ASN-GW before the MSK lifetime expires.
- $E[t_K | t_M \geq t_K]$ : the expected unused key period under the condition that the MS does not return to the old ASN-GW before the MSK lifetime expires (therefore, the cached MSK will not be reused).
- $E[t_K^* | t_M \leq t_K]$ : the expected reused key period under the condition that the MS returns to the old ASN-GW before the MSK lifetime expires (the cached MSK is reused).

We derive the above output measures for Exponentially distributed  $t_M$  with fixed  $T$ , and then generalize the derivation for Generally distributed  $t_M$  with Exponentially distributed  $T$ .

### 4.3.1 Derivation for Exponentially Distributed $t_M$ and Fixed $T$

Suppose that the departure of the MS from the old ASN-GW is a random observer to the MSK lifetime. For fixed MSK lifetime  $T$ , from the residual life theorem [32],  $t_K$  has the Uniform distribution over  $0 \leq t_K \leq T$ . Then  $\alpha$  is derived as

$$\alpha = Pr[t_M \leq t_K] = \int_{t_K=0}^T \left(\frac{1}{T}\right) \times \left(\int_{t_M=0}^{t_K} \lambda e^{-\lambda t_M} dt_M\right) dt_K = \frac{e^{-\lambda T} + \lambda T - 1}{\lambda T} \quad (4.1)$$

$E[t_K | t_M \geq t_K]$  is expressed as:

$$\begin{aligned}
E[t_K | t_M \geq t_K] &= \int_{t_K=0}^T t_K \times \Pr[t_K | t_M \geq t_K] dt_K \\
&= \int_{t_K=0}^T t_K \times \frac{\Pr[t_K \cap (t_M \geq t_K)]}{\Pr[t_M \geq t_K]} dt_K = \int_{t_K=0}^T \frac{t_K \times \int_{t_M=t_K}^{\infty} \frac{1}{T} \times \lambda e^{-\lambda t_M} dt_M}{1 - \alpha} dt_K \\
&= \left( \frac{1}{1 - \alpha} \right) \times \left( \frac{1 - e^{-\lambda T}}{\lambda^2 T} - \frac{e^{-\lambda T}}{\lambda} \right) = \frac{1}{\lambda} - \frac{T e^{-\lambda T}}{1 - e^{-\lambda T}}
\end{aligned} \tag{4.2}$$

Similarly,  $E[t_K^* | t_M \leq t_K]$  is expressed as:

$$\begin{aligned}
E[t_K | t_M \geq t_K] &= \int_{t_K^*=0}^T t_K^* \times \Pr[t_K^* | t_M \leq t_K] dt_K^* \\
&= \int_{t_K^*=0}^T t_K^* \times \frac{\Pr[t_K^* \cap (t_M \leq t_K)]}{\Pr[t_M \leq t_K]} dt_K^* \\
&= \int_{t_K^*=0}^T \frac{t_K^* \times \int_{t_M=0}^{T-t_K^*} \frac{1}{T} \times \lambda e^{-\lambda t_M} dt_M}{\alpha} dt_K^* \\
&= \left( \frac{1}{\alpha} \right) \times \left( \frac{T}{2} - \frac{1}{\lambda} + \frac{1 - e^{-\lambda T}}{\lambda^2 T} \right) \frac{\lambda T^2}{2(\lambda T + e^{-\lambda T} - 1)} - \frac{1}{\lambda}
\end{aligned} \tag{4.3}$$

### 4.3.2 Derivation for Generally Distributed $t_M$ and Exponential $T$

Since the departure of the MS from the old ASN-GW is a random observer to the MSK lifetime, from the residual life theorem,  $t_K$  is Exponentially distributed with mean  $E[T] = 1/\mu$ . Let  $t_M$  have a General distribution with density function  $f(t_M)$  and Laplace transform  $f^*(s)$ . Then  $\alpha$  is derived as

$$\alpha = \int_{t_K=0}^{\infty} \mu e^{-\mu t_K} \times \left[ \int_{t_M=0}^{t_K} f(t_M) dt_M \right] dt_K = f^*(\mu) \tag{4.4}$$

$E[t_K | t_M \geq t_K]$  is expressed as

$$\begin{aligned}
E[t_K | t_M \geq t_K] &= \int_{t_K=0}^{\infty} t_K \times \Pr[t_K | t_M \geq t_K] dt_K \\
&= \int_{t_K=0}^{\infty} \frac{t_K \times \int_{t_M=t_K}^{\infty} \mu e^{-\mu t_K} \times f(t_M) dt_M}{1 - \alpha} dt_K \\
&= \left[ \frac{1}{1 - f^*(\mu)} \right] \times \left\{ \frac{1}{\mu} + \left[ \frac{df^*(s)}{ds} \right] \Big|_{s=\mu} - \frac{f^*(\mu)}{\mu} \right\}
\end{aligned} \tag{4.5}$$

$E [ t_K^* | t_M \leq t_K ]$  is derived as

$$\begin{aligned} E[t_K | t_M \geq t_K] &= \int_{t_K^*=0}^{\infty} t_K^* \times \Pr[t_K^* | t_M \leq t_K] dt_K^* \\ &= \left[ \frac{1}{f^*(\mu)} \right] \times \left[ \frac{f^*(\mu)}{\mu} \right] = \frac{1}{\mu} \end{aligned} \quad (4.6)$$

Equation (4.6) says that  $E [ t_K^* | t_M \leq t_K ]$  is not affected by the  $t_M$  distribution.

To further investigate (4.4) and (4.5), we assume that  $t_M$  has the Gamma distribution, which has been used in telecommunication modeling [22, 39]. The Gamma distributed  $t_M$  has the mean  $1/\lambda$ , variance  $V_M$  and Laplace transform

$$f^*(s) = \left( \frac{1}{\lambda V_M s + 1} \right)^{\frac{1}{\lambda^2 V_M}}$$

Then from (4.4),

$$\alpha = f^*(\mu) = \left( \frac{1}{\lambda \mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M}} \quad (4.7)$$

and  $E [ t_K | t_M \geq t_K ]$  is expressed as

$$\begin{aligned} E[t_K | t_M \geq t_K] &= \left\{ \frac{1}{\mu} + \left[ \frac{df^*(s)}{ds} \right] \Big|_{s=\mu} - \frac{f^*(\mu)}{\mu} \right\} \times \left[ \frac{1}{1 - f^*(\mu)} \right] \\ &= \left[ \frac{1}{\mu} - \frac{1}{\lambda} \times \left( \frac{1}{\lambda \mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M} + 1} - \frac{1}{\mu} \times \left( \frac{1}{\lambda \mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M}} \right] \times \left[ \frac{1}{1 - \left( \frac{1}{\lambda \mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M}}} \right] \end{aligned} \quad (4.8)$$

When  $t_M$  is Exponentially distributed (i.e.,  $V_M = 1/\lambda^2$ ), (4.7) is rewritten as

$$\alpha = \left( \frac{1}{\lambda \mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M}} = \frac{\lambda}{\lambda + \mu} \quad (4.9)$$

and (4.8) is expressed as

$$\begin{aligned} E[t_K | t_M \geq t_K] &= \left[ \frac{1}{\mu} - \frac{1}{\lambda} \times \left( \frac{1}{\lambda \mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M} + 1} - \frac{1}{\mu} \times \left( \frac{1}{\lambda \mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M}} \right] \times \left[ \frac{1}{1 - \left( \frac{1}{\lambda \mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M}}} \right] \end{aligned}$$

$$= \frac{1}{\lambda + \mu} \quad (4.10)$$

Equations (4.1), (4.2), (4.3), (4.6), (4.9) and (4.10) provide the mean value analysis to show the “trends” of the output measures. These equations are also used to validate the simulation experiments in the next section.

## 4.4 Simulation Validation

We utilize simulation experiments to validate the analytic model described in Section 4.3. The MSK lifetimes  $T$  are either produced from a random number generator with mean  $1/\mu$  or set as a fixed period  $1/\mu$ . According to the residual life theorem, the residual MSK lifetime  $t_K$  can be generated directly. The  $t_M$  periods are first drawn from an Exponential random number generator with mean  $1/\lambda$  to validate against the analytic model. Then we use Gamma random number to investigate the impact of General  $t_M$ .

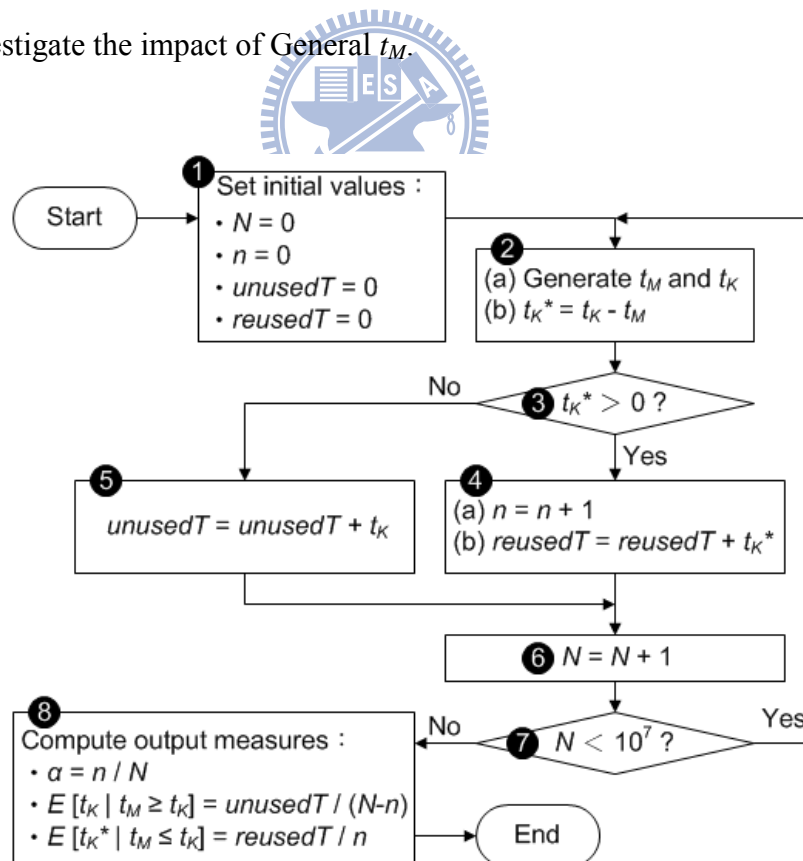


Figure 4.5 The simulation flowchart

To ensure that the results are stable, we simulate  $10^7$  departures from the old ASN-GW. In the simulation process, the counter  $N$  records the number of simulation runs,  $n$  counts the number of MSK reused times,  $unusedT$  calculates the sum of periods which the unused MSK stored in the old ASN-GW, and  $reusedT$  calculates the sum of periods which the MSK reused in the old ASN-GW. The simulation flowchart is shown in Figure 4.5, and the details are described as follows.

**Step 1.** The variables  $N$ ,  $n$ ,  $unusedT$ , and  $reusedT$  are initialized to 0.

**Step 2.** The  $N$ th departure from the old ASN-GW is performed. Periods  $t_M$  and  $t_K$  are generated, and  $t_K^*$  is set to  $t_K - t_M$ .

**Step 3.** If  $t_K^* > 0$ , it means that the MS returns to the old ASN-GW before the MSK lifetime expires, and the flow goes to Step 4. Otherwise, go to Step 5.

**Step 4.** The MSK is reused in the old ASN-GW, and the MSK reused counter  $n$  is incremented by 1. The sum of the MSK reused periods  $reusedT$  is added by  $t_K^*$ .

**Step 5.** Since the MSK cannot be reused, the sum of the MSK unused periods  $unusedT$  is added by  $t_K$ .

**Step 6.**  $N$  is incremented by 1.

**Step 7.** Check if  $10^7$  MS departures from the old ASN-GW have been simulated. If so, go to Step 8. Otherwise, go to Step 2.

**Step 8.** The simulation is complete, and the output measures are computed as follows.

$$\alpha = \frac{n}{N}$$

$$E[t_K | t_M \geq t_K] = \frac{unusedT}{N - n}$$

$$E[t_K^* | t_M \leq t_K] = \frac{reusedT}{n}$$

Based on equations (4.1), (4.2), (4.3), (4.6), (4.9) and (4.10), Table 4.1 shows that the simulation is consistent with the analytic analysis and all errors are within 1%. Therefore, the analytic and the simulation results are consistent.

Table 4.1 Comparison of analytic and simulation results

$E[T]$ ( $1/\lambda$ )	Fixed $T$			Exponential $T$		
	Analytic	Simulation	Errors (%)	Analytic	Simulation	Errors (%)
$10^{-2}$	0.005	0.005	0.2187	0.0099	0.0098	0.7171
$10^{-1}$	0.0484	0.0484	0.0068	0.0909	0.0909	0.0444
$10^0$	0.3679	0.3676	0.0645	0.5	0.4998	0.0492
$10^1$	0.9	0.9	0.005	0.9091	0.9092	0.0122
$10^2$	0.99	0.99	0.0008	0.9901	0.9901	0.0007

(a)  $\alpha$  (Exponential  $t_M$ )

$V_M$ ( $1/\lambda^2$ )	$E[T]=0.1/\lambda$			$E[T]=10/\lambda$		
	Analytic	Simulation	Errors (%)	Analytic	Simulation	Errors (%)
$10^{-1}$	0.0995	0.0996	0.0575	0.5364	0.5363	0.0261
$10^0$	0.0909	0.0909	0.0372	0.9091	0.91	0.0947
$10^1$	0.0831	0.0831	0.0257	3.0336	3.0425	0.2914
$10^2$	0.086	0.0861	0.0505	6.2541	6.229	0.4019
$10^3$	0.0892	0.0893	0.1398	7.8596	7.8467	0.1651

(b)  $E[t_K | t_M \geq t_K]$  (Gamma  $t_M$  and Exponential  $T$ )

$V_M$ ( $1/\lambda^2$ )	$E[T]=0.1/\lambda$			$E[T]=10/\lambda$		
	Analytic	Simulation	Errors (%)	Analytic	Simulation	Errors (%)
$10^{-1}$	0.1	0.1006	0.5986	10	10.005	0.0501
$10^0$	0.1	0.1001	0.1492	10	10.001	0.0137
$10^1$	0.1	0.1001	0.1074	10	10.012	0.1206
$10^2$	0.1	0.1	0.0222	10	10.003	0.032
$10^3$	0.1	0.1	0.0104	10	9.9999	0.001

(c)  $E[t_K^* | t_M \leq t_K]$  (Gamma  $t_M$  and Exponential  $T$ )

## 4.5 Numerical Examples

According to the analytic and the simulation models, we use numerical examples to investigate how the MSK lifetime  $T$  affects the performance of the key caching mechanism. Figure 4.6 plots the results for Exponential  $t_M$ . Figure 4.6 (a) plots  $\alpha$  against  $E[T]$ . The figure indicates that  $\alpha$  is an increasing function of  $E[T]$ . It is intuitive that if  $E[T]$  is large, then it is more likely that

the MS will return before the MSK lifetime expires. From (4.1)

$$\lim_{E[T] \rightarrow \infty} \alpha = \lim_{T \rightarrow \infty} \left( \frac{e^{-\lambda T} + \lambda T - 1}{\lambda T} \right) = 1$$

Since  $E[T]=1/\mu$ , from (4.9), we have

$$\lim_{E[T] \rightarrow \infty} \alpha = \lim_{E[T] \rightarrow \infty} \left[ \frac{\lambda}{\lambda + (1/E[T])} \right] = 1$$

This figure also shows that the Exponential  $T$  outperforms the fixed  $T$  in terms of  $\alpha$ .

Figure 4.6 (b) plots the unused key period  $E[t_K | t_M \geq t_K]$  as the function of  $E[T]$ . The figure shows that the unused key period increases as  $E[T]$  increases. From (4.2), we have

$$\lim_{E[T] \rightarrow \infty} E[t_K | t_M \geq t_K] = \lim_{T \rightarrow \infty} \left( \frac{1}{\lambda} - \frac{T e^{-\lambda T}}{1 - e^{-\lambda T}} \right) = \frac{1}{\lambda}$$

and from (4.10)

$$\lim_{E[T] \rightarrow \infty} E[t_K | t_M \geq t_K] = \lim_{E[T] \rightarrow \infty} \left[ \frac{1}{\lambda + (1/E[T])} \right] = \frac{1}{\lambda}$$

Therefore, the maximum unused key period is  $E[t_M] = 1/\lambda$ . When  $E[T]$  is small (e.g., less than  $1/\lambda$ ), the fixed  $T$  outperforms the Exponential  $T$ . When  $E[T]$  is large, the Exponential  $T$  yields better performance in terms of the unused key period.

Figure 4.6 (c) plots the reused key period  $E[t_K^* | t_M \leq t_K]$  as the function of  $E[T]$ . The figure indicates the intuitive result that the key reused period increases as  $E[T]$  increases. From (4.3), we have

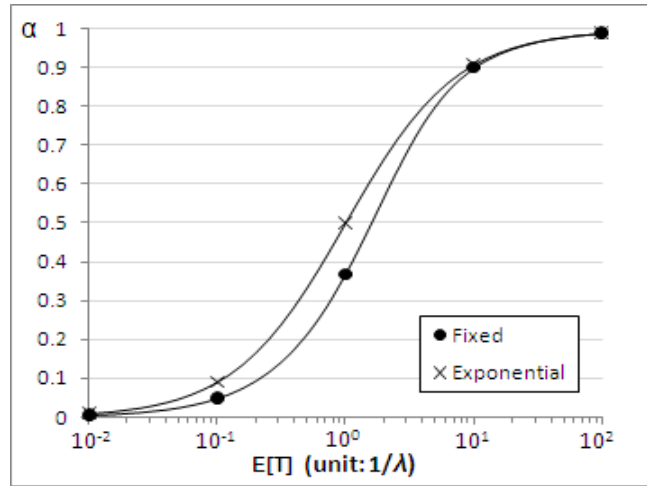
$$\lim_{E[T] \rightarrow \infty} E[t_K^* | t_M \leq t_K] = \lim_{T \rightarrow \infty} \left[ \frac{\lambda T^2}{2(\lambda T + e^{-\lambda T} - 1)} - \frac{1}{\lambda} \right] = \infty$$

and from (4.6)

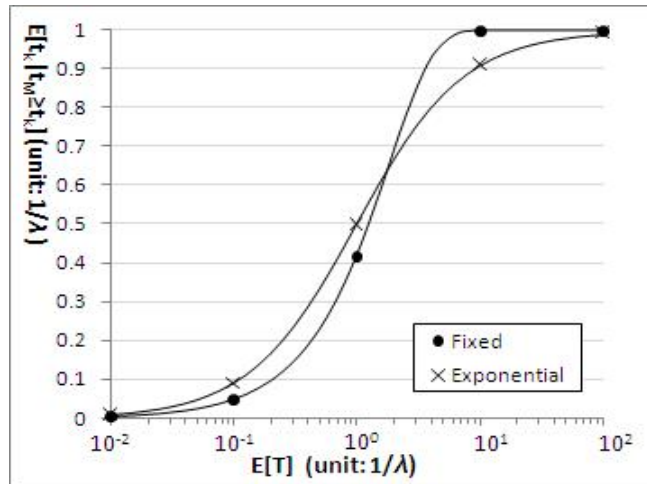
$$\lim_{E[T] \rightarrow \infty} E[t_K^* | t_M \leq t_K] = \lim_{E[T] \rightarrow \infty} \left( \frac{1}{1/E[T]} \right) = \infty$$

The figure also indicates that the Exponential  $T$  outperforms the fixed  $T$  in terms of the reused key period.

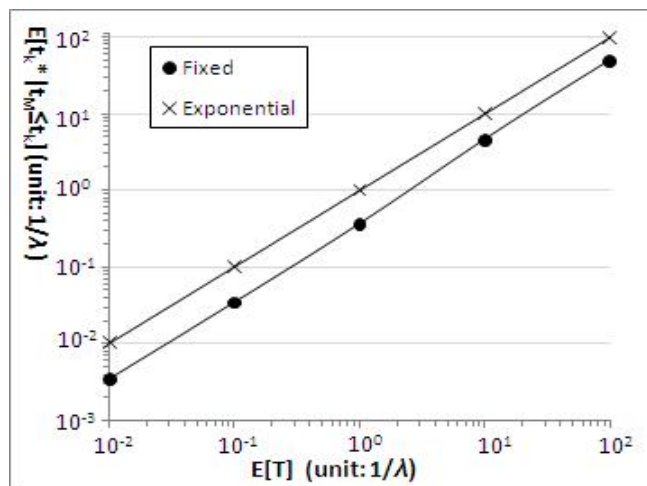




(a) Effect of  $E[T]$  on  $\alpha$

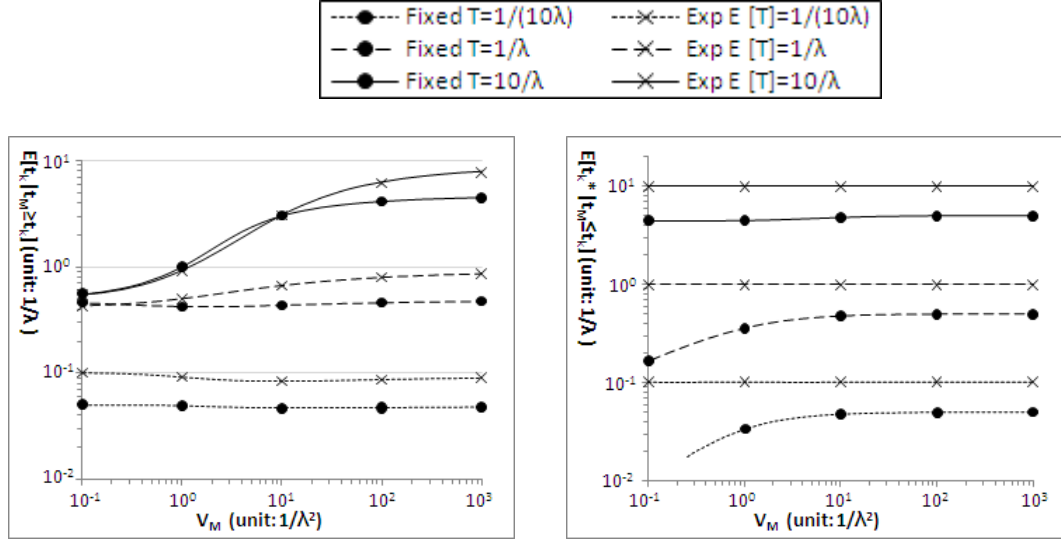


(b) Effect of  $E[T]$  on  $E[t_K | t_M \geq t_K]$



(c) Effect of  $E[T]$  on  $E[t_K^* | t_M \leq t_K]$

Figure 4.6 Effect of  $\mu$



(a) Effect of  $V_M$  on  $E[t_K | t_M \geq t_K]$

(b) Effect of  $V_M$  on  $E[t_K^* | t_M \leq t_K]$

Figure 4.7 Effect of  $V_M$

Figure 4.7 (a) plots the unused key period  $E[t_K | t_M \geq t_K]$  against  $E[T]$  and  $V_M$ . When  $E[T] \geq 1/\lambda$ , the unused key period increases as  $V_M$  increases. This phenomenon is explained as follows. As  $V_M$  increases, more long and short  $t_M$  are observed. Since a random observer (an MS movement) tends to observe long  $t_M$ , short  $t_M$  will not contribute to  $E[t_K | t_M \geq t_K]$ . Therefore, more long  $t_K$  are observed as  $V_M$  increases. From (4.8),

$$\lim_{V_M \rightarrow \infty} E[t_K | t_M \geq t_K] = \lim_{V_M \rightarrow \infty} \left\{ \frac{\left[ \frac{1}{\mu} - \frac{1}{\lambda} \times \left( \frac{1}{\lambda\mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M} + 1} - \frac{1}{\mu} \times \left( \frac{1}{\lambda\mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M}} \right]}{1 - \left( \frac{1}{\lambda\mu V_M + 1} \right)^{\frac{1}{\lambda^2 V_M}}} \right\} = \frac{1}{\mu}$$

When  $E[t_K] < 1/\lambda$ ,  $E[t_K | t_M \geq t_K] = E[t_K]$ , which is not sensitive to  $V_M$ .

Figure 4.7 (b) plots the reused key period  $E[t_K^* | t_M \leq t_K]$  against  $E[T]$  and  $V_M$ . For Exponential  $T$ , according to (4.6),  $E[t_K^* | t_M \leq t_K] = E[T]$ . This phenomenon is explained as follows. Since the residual MSK lifetime  $t_K$  is Exponentially distributed, the arrival of the MS to the old ASN-GW is a random observer to  $t_K$ . Thus, from the residual life theorem,  $t_K^*$  is also

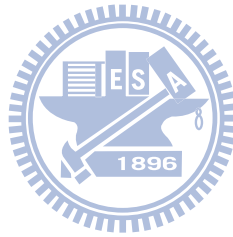
Exponentially distributed with the mean  $E[T]$ . For fixed  $T$ ,  $E [ t_K^* | t_M \leq t_K ]$  increases as  $V_M$  increases. Since we only consider the case when  $t_M \leq t_K$ , as  $V_M$  increases, short  $t_K$  periods are observed and long  $t_K$  will not contribute to  $E [ t_K^* | t_M \leq t_K ]$ . Thus, for fixed  $T$ , the reused key period increases as  $V_M$  increases and eventually approaches to  $E [t_K] = T/2$ .

Figure 4.7 shows that the Exponential  $T$  outperforms the fixed  $T$  in terms of the reused key period. On the other hand, for the unused key period, the fixed  $T$  outperforms the Exponential  $T$  in most cases. Another advantage of Exponential  $T$  over the fixed  $T$  is that the reused key period  $E [ t_K^* | t_M \leq t_K ]$  performance is not affected by the variance  $V_M$ . This stability property is important for telecom-grade system.

## 4.6 Summary

This chapter proposed a key caching mechanism to speed up the inter-ASN-GW handoff for mobile WiMAX. With this mechanism, when an MS leaves the old ASN-GW, the MS key record (e.g., the MSK) is cached in the old ASN-GW. If the MS returns to the old ASN-GW before the MSK lifetime expires, it can reuse the MSK without executing the IEEE 802.1X authentication. On the other hand, the old ASN-GW consumes extra storage to maintain the MS key records when the MS leaves the old ASN-GW. This chapter investigated how the period  $T$  of the MSK lifetime affects the key caching performance by an analytic model and simulation experiments. Three output measures are evaluated: the key reused probability, the unused key period, and the reused key period. We showed that the caching mechanism can effectively speed up the inter-ASN-GW handoff. We also observed that the Exponential  $T$  outperforms the fixed  $T$  in most cases. Moreover, for the reused key period, the Exponential  $T$  is not affected by the variance of MS residence period in new ASN-GWs, and is more suitable for telecommunications system. As a final remark, the operator uses our study and the number of serving MSs to calculate the storage budget at an ASN-GW. Our study indicates that  $E[T] > 10/\lambda$  will not improve the performance. Therefore, if  $E[T] < 10/\lambda$  is selected, the extra storage

can be computed from Little's Law  $N = xE[T] < 10x/\lambda$ , where  $N$  is the extra storage (the number of MS key records) and  $x$  is the rate of the MSs leaving the ASN-GW [18].



## Chapter 5.

### End-to-End Security Mechanisms for SMS

In the UMTS all-IP network, the security mechanism (i.e., encryption) offered by the network operator applies only on the wireless link in the wireless access networks. Data delivered through the mobile core network may not be protected. Existing end-to-end security mechanisms are provided at application level and typically based on public key cryptosystem. This chapter introduces two encryption mechanisms for *Short Message Service* (SMS) to illustrate the security mechanism between the MS and the application and service network. These two mechanisms are based on the *Rivest-Shamir-Adleman* (RSA) scheme and the *Identity-based* (ID-based) scheme. We implement these two mechanisms over the standard SMS network and estimate the encryption overhead. Our study indicates that the ID-based mechanism has advantages over the RSA mechanism in key distribution and scalability of increasing security level for mobile service.

#### 5.1 Introduction to Wireless Cryptography

In today's mobile communication systems, the symmetric key technology is used to authenticate the *Mobile Station* (MS). The authentication method is based on the pre-shared secret key  $K_i$  (created at the service subscription) between the user and the mobile operator [20]. The symmetric session key  $K_c$  derived during the authentication phase (by using  $K_i$ ) only applies on the wireless link, that is, from the MS to the base station. End-to-end security or confidentiality and integrity over the whole path between two parties (e.g. an MS to another MS) is not provided by mobile systems (such as *Global System for Mobile Communication* (GSM) and *Universal Mobile Telecommunications System* (UMTS)) and therefore has to be

provided at application level. Asymmetric cryptography is typically used in mobile systems to provide end-to-end security.

Asymmetric cryptography or *Public Key Cryptography* (PKC) involves two distinct keys, public key  $K_U$  and private key  $K_R$ .  $K_U$  can be widely distributed without compromising its corresponding private key  $K_R$ . In some systems,  $K_R$  remains known only to the user that generated it, while in other systems,  $K_R$  is given to a user by another trusted entity. To generate the key pair, one first chooses a private key  $K_R$  and applies some trapdoor one-way function to  $K_R$  to obtain a random and uncontrollable public key  $K_U$ . The main concern in a public key setting is the authenticity of the public key. If an attacker convinces a sender that the receiver's public key is some key of the attacker's choice instead of the correct public key, he/she can eavesdrop and decrypt messages intended for the receiver. This is the well known man-in-the-middle attack [36]. This authentication problem is typically resolved by the use of verifiable information called certificate, which is issued by a trusted third party and consists of the user name and his/her public key.

In 1984, Shamir (1984) introduced the concept of *Identity-based* (ID-based) cryptography where the public key of a user can be derived from public information that uniquely identifies the user. For example, the public key of a user can be simply his/her e-mail address or telephone number, and hence implicitly known to all other users. A major advantage of ID-based cryptosystem is that no certificate is needed to bind user names with their public keys.

ID-based cryptosystem transparently provides security enhancement to the mobile applications without requiring the users to memorize extra public keys. For example, sending an ID-based encrypted short message is exactly the same as sending a normal short message [14] if the mobile phone number of the short message recipient is used as the public key. Therefore, the mobile user (the sender) does not need to memorize an extra public key for the receiver. This feature is especially desirable for transaction-type mobile applications such as

bank or stock transactions [21].

In this chapter, we implement two encryption systems for *Short Message Service* (SMS) and estimate the encryption overheads compared with the original non-ciphered message transmissions. We first introduce the certificate-based and the ID-based public key cryptosystems, which provide authentic solutions for public key distribution. Then we propose two applicable end-to-end encryption mechanisms for SMS based on the certificate-based public key cryptosystem and the ID-based public key cryptosystem, respectively. Finally, we also evaluate and compare the delivery overheads between these two mechanisms.

## 5.2 Public Key Cryptography

Public key techniques utilize the asymmetric key pairs. In an asymmetric key pair, one key is made publicly available, while the other is kept private. Because one of the keys is available publicly, there is no need for a secure key exchange. However, it is required to distribute the public key authentically. Because there is no need for pre-shared secrets prior to a communication, public key techniques are ideal for supporting security between previously unknown parties. Authentication is achieved by proving possession of the private key. One mechanism for doing this is digital signature. A digital signature is generated with the private key and verified using the corresponding public key. Since the public key of a key pair is usually published in a directory, the overhead associated with distributing keys is reduced significantly in comparison with secret key techniques.

A main concern in public key distribution is the authenticity of the public key. Figure 5.1 illustrates how an adversary between a sender B and a receiver A can impersonate the receiver A in the public key encryption scheme. The adversary achieves this by replacing A's public key  $KU_A$  with a false public key  $KU_A'$  which is then received by B (see Figure 5.1 (1) and (2)). User B uses the false public key  $KU_A'$  to encrypt the message  $\mathcal{M}$  (see Figure 5.1 (3)). The

adversary obtains the secret message  $\mathcal{M}$  (see Figure 5.1 (4)) and delivers the re-encrypted cipher to user A (see Figure 5.1 (5)). In this way, the secret message  $\mathcal{M}$  is acquired by both user A (see Figure 5.1 (6)) and the adversary. Similar impersonation settings exist between the signer and verifier in the signature schemes. This is the well known man-in-the-middle attack. The following issue arises from the need to prevent these kinds of attacks: how does B know (or authenticate) which particular public key is A's? To answer this question, authentication of public key distribution is required. Authenticating public keys provides assurance to the entity that the received public key corresponds to the sender's identity.

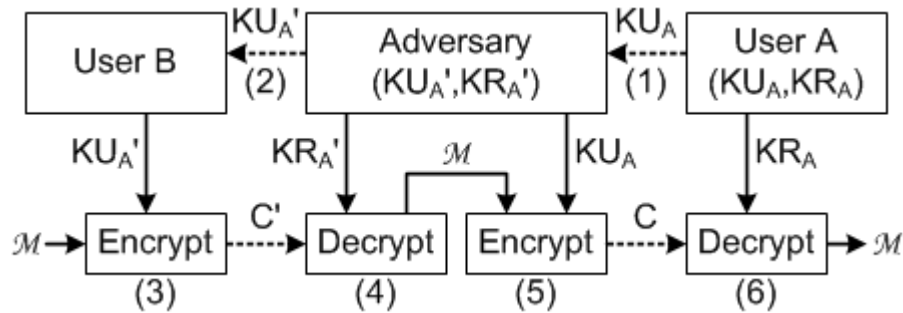


Figure 5.1 The man-in-the-middle attack

### 5.2.1 Certificate-based Public Key Cryptography

A typical approach to guarantee the authentication of the public key holder relies on a trusted agent named *Certificate Authority* (CA). The CA's digital signature binds entity A's identity  $ID_A$  to the corresponding public key  $KU_A$ . The CA's signature, when sent along with the identity (e.g. name or telephone number) and public key, forms a digital certificate, which can be verified by any entity in possession of the CA's public key.

This certificate provides a binding between the identity and the public key. Digital certificates can contain extra information, such as cryptographic algorithms to be used in conjunction with the public key in the certificate. The most widely adopted certificate format is based on the X.509 standard [16]. A basic certificate issued by a CA for entity A is of the



form:

$$\text{Cert}_A = (\text{ID}_A, \text{KU}_A, \text{Sign}_{\text{KR}_{\text{CA}}}(\text{ID}_A, \text{KU}_A))$$

where  $\text{Sign}_{\text{KR}_{\text{CA}}}(\cdot)$  denotes the signing algorithm with the CA's private key as the signing key.

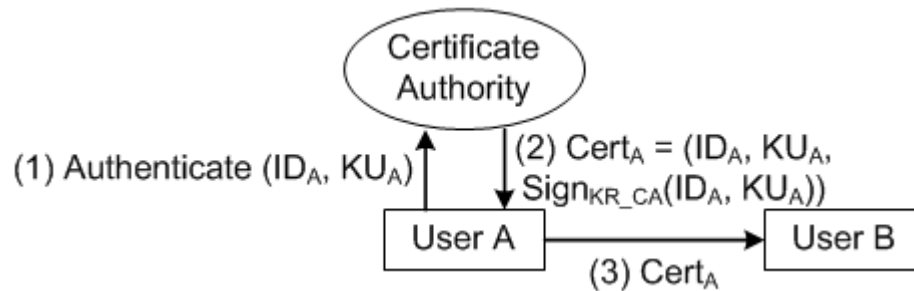


Figure 5.2 The certificate-based public-key distribution

The certificate-based public-key distribution works as follows. User A first chooses a public key cryptosystem, and generates his/her own key pair  $(\text{KU}_A, \text{KR}_A)$ , where  $\text{KU}_A$  denotes the public key and  $\text{KR}_A$  is the private key. To attain the authenticity of public-key distribution, user A has to subscribe to the trusted CA (see Figure 5.2 (1)), and requests a certificate  $\text{Cert}_A$  for his/her public-key from CA (see Figure 5.2 (2)). The CA signs the certificate with its private key. Then user A can send his/her certificate directly to another user B (see Figure 5.2 (3)) or put it on the public key directory. Once user B is in possession of A's certificate, B verifies the certificate with the CA's public key and has confidence that the messages he/she encrypts with A's public key will be secure from eavesdropping and that messages signed with A's private key are unforgeable.

### 5.2.2 ID-based Public Key Cryptography

Shamir (1984) proposed the ID-based public key approach to support PKC without the use of certification. In ID-based PKC user A's public key  $\text{KU}_A$  is not delivered to user B, and therefore eliminates the attack shown in Figure 5.1. User B encrypts a message for user A or verifies a signature from user A using a public key which is derived from user A's identifier

$ID_A$  (e.g. e-mail address or telephone number; see Figure 5.3 (3)). The trusted agent has a new role in ID-based public key cryptosystem, and is renamed as the *Private Key Generator* (PKG). The PKG issues the private key corresponding to the public key (derived from the identifier  $ID_A$ ) to user A over a secure channel (see Figure 5.3 (2)). This issuing action takes place after user A is authenticated by the PKG (see Figure 5.3 (1)). To generate private keys, the PKG makes use of a master key which must be kept in secret. The requirement to have an authentic CA's public key for verifying certificates in certificate-based cryptosystem is replaced by the requirement to have authentic PKG's system parameters in ID-based cryptosystem. Notice that both the PKG and the user A know the private key  $KR_A$ .

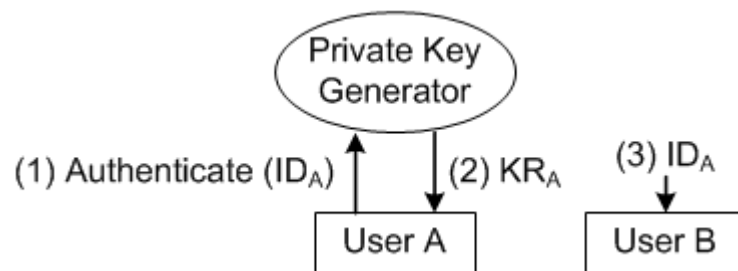


Figure 5.3 The ID-based public-key distribution

## 5.3 End-to-End Security for SMS

This section first introduces the SMS for GSM [20, 21]. Then we describe the *Rivest-Shamir-Adleman* (RSA) and the ID-based encryption mechanisms for SMS.

### 5.3.1 SMS Architecture

The network architecture of SMS in GSM is illustrated in Figure 5.4. In this architecture, the short message is first delivered from the MS A (Figure 5.4 (A)) to a *Short Message Service Centre* (SM-SC; Figure 5.4 (E)) through the *Base Station System* (BSS; Figure 5.4 (B)), the *Mobile Switching Centre* (MSC; Figure 5.4 (C)) and then the *Interworking MSC* (IWMSC; Figure 5.4 (D)). The SM-SC then forwards the message to the GSM network through a

specific GSM MSC called the *SMS gateway MSC* (SMS GMSC; Figure 5.4 (F)). The SM-SC may connect to several GSM networks and to several SMS GMSCs in a GSM network. Following the GSM roaming protocol, the SMS GMSC locates the current MSC of the message receiver and forwards the message to that MSC. The MSC then broadcasts the message through the BSS to the destination MS B (Figure 5.4 (G)). In Section 5.3.2, we will describe two encryption mechanisms for end-to-end secure SMS based on certificate-based and ID-based cryptosystems.

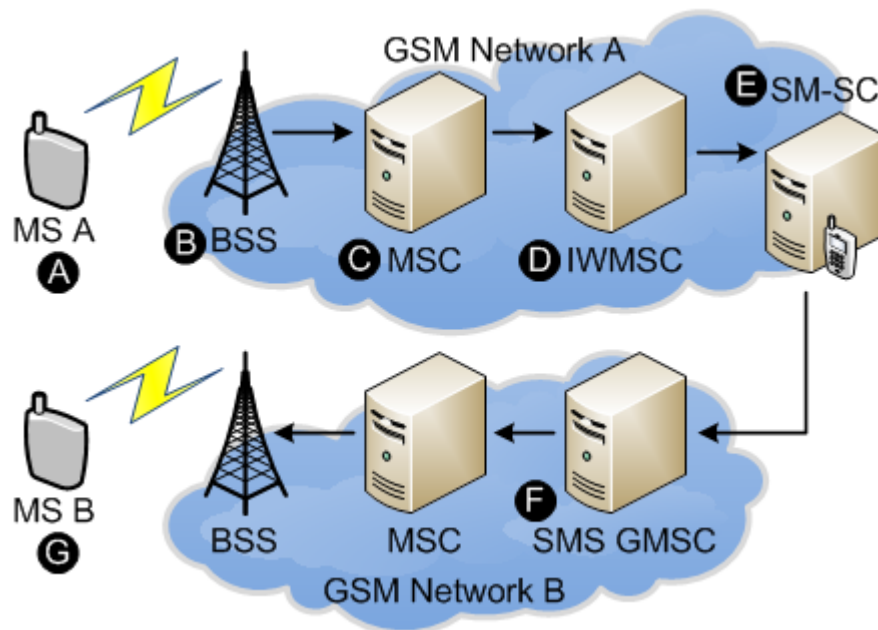


Figure 5.4 GSM SMS network architecture

### 5.3.2 RSA Mechanism

The most widely implemented approach to public key encryption is the RSA scheme [30]. The RSA scheme is a block cipher in which the original non-ciphered text and cipher text are integers between 0 and  $n-1$  for some  $n$ . That is, the block size  $k_{RSA}$  is determined by the bit length of the integer  $n$  and regarded as the key size of the RSA scheme. This scheme consists of the following three functions:

- *Key generation*: a user first selects two prime numbers  $p$  and  $q$ , randomly chooses  $e$  with  $\text{gcd}(e, (p-1)(q-1)) = 1$  and calculates  $d \equiv e^{-1} \pmod{(p-1)(q-1)}$ . Then the public key is

$KU = (e, n)$  and the private key is  $KR = (d, n)$ , where  $n = pq$ .

- *Encryption*: for a given message represented as an integer  $M < n$ , the cipher text is computed by  $C = M^e \bmod n$ .
- *Decryption*: for a given cipher text  $C$ , the original non-ciphered text is computed by  $M = C^d \bmod n$ .

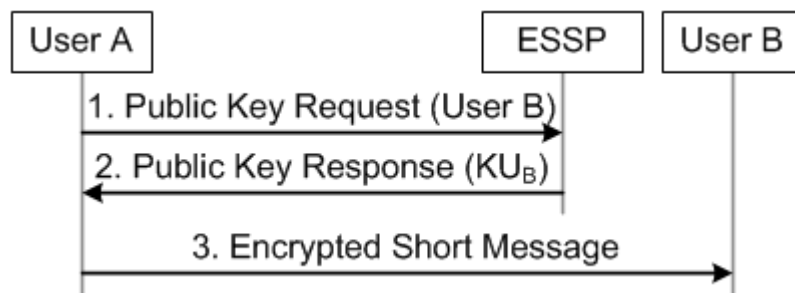


Figure 5.5 The RSA procedure for sending an encrypted short message

A RSA mechanism for end-to-end secure SMS is introduced as follows. The *End-to-end Security Service Provider* (ESSP) plays a role as the CA in the certificate-based public key cryptosystem. To access the end-to-end security service, a user first chooses his/her own key pair  $(KU, KR)$  and subscribes to the ESSP for requesting a certificate of his/her public key  $KU$ . The ESSP signs the certificate with its private key and publishes the certificate in the public key directory for public access. When a mobile user A (the sender) wants to encrypt a short message to user B, he/she first sends a public key request (see Figure 5.5 (1)) to the public key directory in short message format. The public key directory retrieves user B's certificate. If it succeeds, user B's certificate is sent to user A as the public key response (see Figure 5.5 (2)). Once user A is in possession of B's certificate, he/she verifies the certificate with the ESSP's public key and uses the user B's public key to encrypt short message for B (see Figure 5.5 (3)). If the request fails (due to unavailability of user B's certificate), the ESSP will inform user B to subscribe to end-to-end security service if he/she wants to securely communicate with user A.

### 5.3.3 ID-based Mechanism

In the above RSA approach, the sender needs to communicate with the public key directory for requesting the public key. If the request fails (e.g. the directory server is down or there is no certificate exist for the receiver), the sender cannot encrypt short message for the receiver. On the other hand, in an ID-based encryption scheme, the sender simply uses the receiver's ID (i.e. the telephone number) as his public key without any request and verification. Thus, the sender does not need to access any public key directory before sending an encrypted short message.

The first complete and efficient ID-based encryption scheme was proposed by Boneh and Franklin (2001) which uses a *bilinear map* called Weil pairing over elliptic curves. The bilinear map transforms a pair of elements in group  $G_1$  and sends it to an element in group  $G_2$  in a way that satisfies some properties. The most important property is the bilinearity that it should be linear in each entry of the pair. Assume that  $P$  and  $Q$  are two elements (e.g. points on elliptic curves) of an additive group  $G_1$ . Let  $e(P, Q)$  be the element of a multiplicative group  $G_2$ , which is the pairing applied to  $P$  and  $Q$ . Then the pairing must have the following property:

$$e(rP, Q) = e(P, Q)^r = e(P, rQ)$$

where  $r$  is an integer and  $rP$  denotes the element generated by  $r$  times of additions on  $P$ , for example,  $2P = P + P$ ,  $3P = P + P + P$  and so on. Weil pairing on elliptic curves is selected as the bilinear map. That is, the elliptic curve group (the set of point collection on elliptic curves) is used as  $G_1$  and the multiplicative group of a finite field is used as  $G_2$ .

The ID-based scheme consists of four algorithms: *Set-up*, *Extraction*, *Encryption* and *Decryption*. Set-up is run by the PKG to generate the master key and the system parameters. This is done on input of a security parameter  $k_{ID}$ , which specifies the bit length of the group order and is regarded as the key size of the ID-based scheme. The extraction algorithm is

carried out by the PKG to generate a private key corresponding to the identity of a user. As with regular PKC, the encryption algorithm takes a message and a public key as inputs to produce a cipher text. Similarly, the decryption algorithm is executed by the owner of the corresponding private key to decrypt the cipher text. These four functions are described as follows.

- *Set-up*: with the parameter  $k_{ID}$ , the algorithm works as follows:
  - Generate a random  $k_{ID}$ -bit prime  $p$ , two groups  $(\mathbf{G}_1; +)$ ;  $(\mathbf{G}_2; *)$  of order  $p$ , and the Weil pairing  $\mathbf{e}: \mathbf{G}_1 \times \mathbf{G}_2 \rightarrow \mathbf{G}_2$ . Choose an arbitrary generator  $P \in \mathbf{G}_1$
  - Pick a random number  $s \in \mathbb{Z}_p^*$  and set  $P_{\text{pub}} = sP$
  - Choose cryptographic hash functions  $h_1: \{0, 1\}^* \rightarrow \mathbf{G}_1^*$  and  $h_2: \mathbf{G}_2 \rightarrow \{0, 1\}^n$  for some  $n$ .

The public system parameters are  $\{p, \mathbf{G}_1, \mathbf{G}_2, \mathbf{e}, n, P, P_{\text{pub}}, h_1, h_2\}$  and the master key  $s$  is kept in secret by the PKG.

- *Extraction*: for a given string  $ID \in \{0, 1\}^*$  as the public key, the algorithm works as follows:
  - compute  $Q_{ID} = h_1 (ID) \in \mathbf{G}_1$
  - set the private key  $KR = sQ_{ID}$ , where  $s$  is the master key held by PKG.
- *Encryption*: to encrypt a message  $\mathcal{M}$  under the public key  $KU = ID$ , the algorithm works as follows:
  - compute  $Q_{ID} = h_1 (ID) \in \mathbf{G}_1$
  - choose a random  $r \in \mathbb{Z}_p^*$
  - set the cipher text to be  $C = (U, \mathcal{V}) = (rP, \mathcal{M} \oplus h_2 (\mathbf{e} (Q_{ID}, sP)^r))$ .
- *Decryption*: to decrypt a cipher  $C = (U, \mathcal{V})$  encrypted using the public key  $KU = ID$ , the algorithm uses the private key  $KR = sQ_{ID}$  to compute  $\mathcal{M} = \mathcal{V} \oplus h_2 (\mathbf{e} (sQ_{ID}, U))$ . This decryption procedure yields the correct message due to the bilinearity of the Weil pairing (i.e.  $\mathbf{e} (sQ_{ID}, U) = \mathbf{e} (sQ_{ID}, rP) = \mathbf{e} (Q_{ID}, sP)^r$ ).

Details of Weil pairing for ID-based cryptosystem can be found in [14], and will not be elaborated further in this chapter.

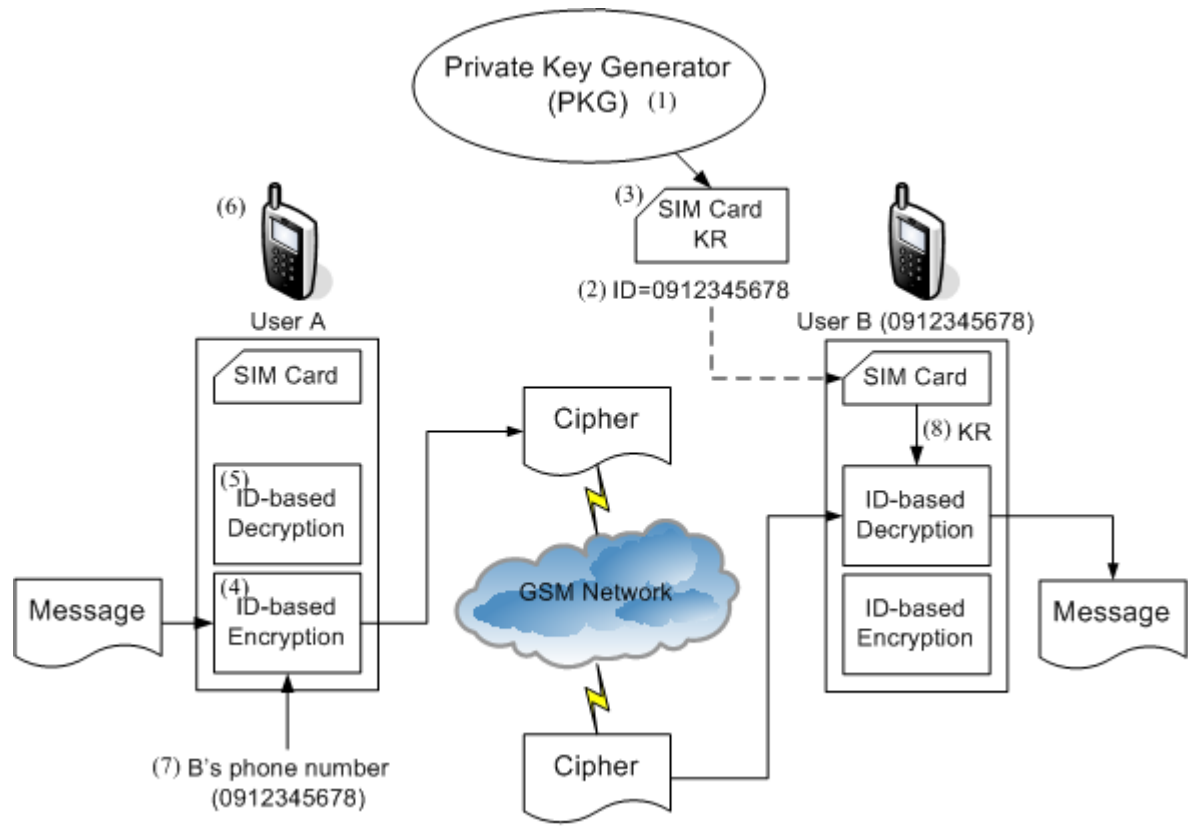


Figure 5.6 ID-based end-to-end encryption mechanism

Based on an improved algorithm we proposed in [14], an efficient ID-based end-to-end encryption mechanism for mobile services as illustrated in Figure 5.6. The PKG (Figure 5.6 (1)) constructs the ID-based cryptosystem and uses, for example, the phone number as the ID (see Figure 5.6 (2)). Every mobile user involved in the ID-based cryptosystem is given a *Subscriber Identity Module* (SIM) card (Figure 5.6 (3)) at the subscription time. The ID (phone number; e.g. 0912345678 in Figure 5.6) and its corresponding private key KR are loaded in the SIM card by the ESSP. Note that for standard GSM/UMTS service, SIM card is always given to a mobile user at the subscription time and the proposed ID-based encryption scheme can be preloaded into the SIM card without incurring any extra overhead. The MS contains two security modules: ID-based encryption module (Figure 5.6 (4)) and ID-based

decryption module (Figure 5.6 (5)). When a mobile user A (the sender; Figure 5.6 (6)) wants to encrypt a short message to user B (the receiver), A uses B's phone number 0912345678 (see Figure 5.6 (7)) as the public key and encrypts the message through the ID-based encryption module. Once user B receives the cipher (the encrypted message), he/she uses the private key  $K_R$  (see Figure 5.6 (8)) stored in the SIM card to decrypt the cipher through the ID-based decryption module and obtain the original non-ciphered message.

To estimate the encryption overheads between the RSA and the ID-based mechanisms, we implement these two encryption schemes and give the evaluation in the next section.

## 5.4 Performance Comparison

This section compares the delivery (encryption, decryption and message delivery delay of ciphered short messages based on the RSA and the ID-based approaches. The experimental environment is illustrated in Figure 5.7. Both the sender and the receiver are notebooks (Figure 5.7 (1) and (3)) configured with a Pentium-III 500 MHz CPU, 256MB main memory and 20GB disk space and are running on the Windows XP Professional operating system. To deliver short messages, every notebook is plugged in a NOKIA Card Phone version 2.0 and the short message is sent via the ChungHwa GSM network (Figure 5.7 (2)) from the sender to the receiver.



Figure 5.7 Encrypted short message experimental environment

We first note that to support the same security level, the key length for the ID-based and the RSA approaches are different. The ID-based cryptosystem using Weil pairing is over



elliptic curves, thus its security level depends on the key length of *Elliptic Curve Cryptosystem* (ECC). As listed in Table 5.1 [11], a 108-bit ECC key provides the same security level as a 512-bit RSA key, a 160-bit ECC key provides the security level equivalent to a 1024-bit RSA key and a 224-bit ECC key is equivalent to a 2048-bit RSA key. For a fair comparison, we measure the delivery delays of ID-based system and RSA system over the same security level, and the results are shown in Figure 5.8.

Table 5.1 Key length for equivalent security levels (in bits)

<i>ECC ( ID-based )</i>	<i>RSA</i>
108	512
160	1024
224	2048

Figure 5.8 plots delivery delays of the RSA and ID-based approaches for the same non-ciphered length (in bytes), where the ● curves represent the RSA delivery delays, the dashed curves represent the ID-based delivery delays and the solid curves represent the non-ciphered message delays.

Based on the RSA encryption algorithm described in Section 5.3.2, for a non-ciphered message of length  $i$ , the RSA ciphered message length is

$$L_{\text{RSA}}(i) = k_{\text{RSA}} \times \left\lceil \frac{i}{k_{\text{RSA}}} \right\rceil$$

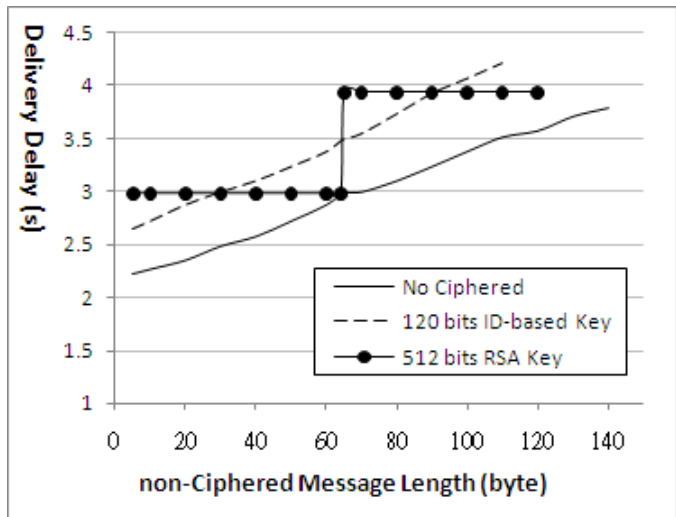
where  $k_{\text{RSA}}$  is the key length of RSA approach. For  $k_{\text{RSA}} = 512$ ,

$$L_{\text{RSA}}(i) = \begin{cases} 512 & i \leq 512 \\ 1024 & 512 < i \leq 1024 \end{cases}$$

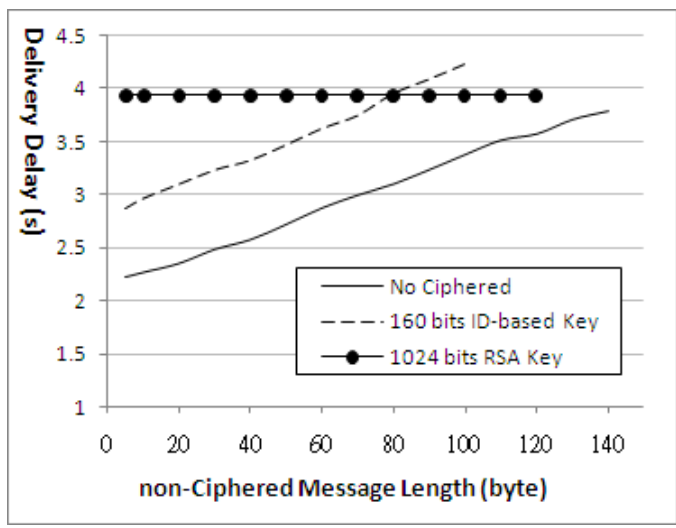
Therefore, in Figure 5.8 (a), we observe a step curve for the RSA ciphered message delivery.

For  $k_{\text{RSA}} = 1024$  and  $k_{\text{RSA}} = 2048$ , if  $i \leq 1024$ ,  $L_{\text{RSA}}(i)$  is 1024 and 2048, respectively.

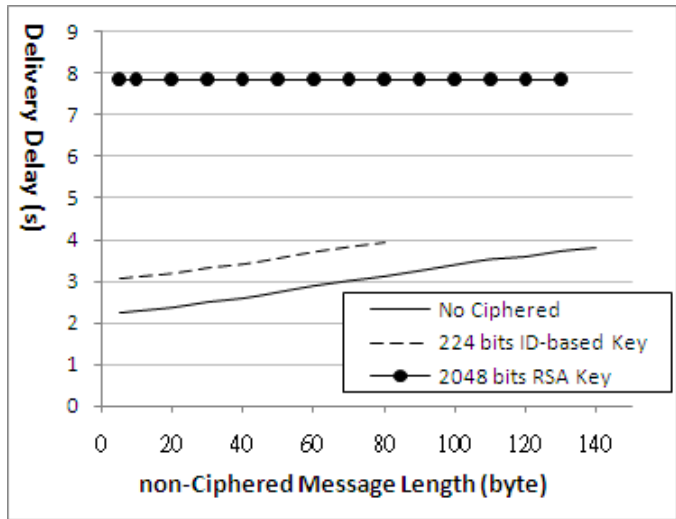
Therefore, in Figure 5.8 (b) and (c), we observe horizontal lines for the RSA ciphered message delivery.



(a) 512-bit RSA key and 120-bit ID-based key cipher



(b) 1024-bit RSA key and 160-bit ID-based key cipher



(c) 2048-bit RSA key and 224-bit ID-based key cipher

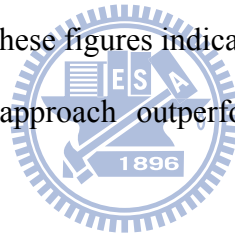
Figure 5.8 Delivery delays of SMS

Based on step 3 of ID-based encryption algorithm described in Section 5.3.3, the ID-based ciphered message length is

$$L_{ID}(i) = i + \frac{k_{ID}}{4}$$

where  $k_{ID}$  is the key length of ID-based approach. For a fixed  $k_{ID}$ ,  $L_{ID}(i)$  increases as  $i$  increases. Therefore, in Figure 5.8 (a)–(c), we observe linear lines for ID-based ciphered message delivery.

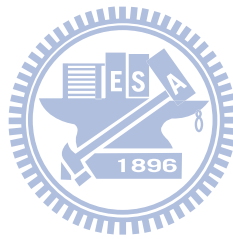
Based on the above delivery delay analysis, Figure 5.8 (a) shows that the ID-based approach outperforms the RSA approach when the non-ciphered message length is less than 30 bytes or is between 65 and 90 bytes. Figure 5.8 (b) shows that the ID-based approach outperforms the RSA approach when the non-ciphered message length is less than 79 bytes. Figure 5.8 (c) shows that the ID-based approach outperforms the RSA approach up to 140-byte message limit of SMS. These figures indicate that as the security level increases, it is more likely that the ID-based approach outperforms the RSA approach for the SMS applications.



## 5.5 Summary

In this chapter, two applicable end-to-end security mechanisms for SMS based on the RSA scheme and the ID-based scheme are introduced and implemented. The ID-based scheme provides a greater simplification of key distribution. That is, all public keys can be derived from the identities of the users. Therefore obtaining someone's public key, for encryption or verification, becomes a simple and transparent procedure. This is in contrast to the RSA scheme, where one has to look up the corresponding certificate and verify the CA's signature. Another advantage of the ID-based scheme is the linear scalability of increasing security level. When the security level increases, the key size of the RSA scheme increases faster than that of the ID-based scheme and may not be practical for the SMS applications. Our study concludes

that the ID-based scheme offers a convenient end-to-end security mechanism for mobile service such as SMS.



## Chapter 6.

### Conclusions and Future Work

The *IP Multimedia Core Network Subsystem* (IMS) provides the IP multimedia services on the *Universal Mobile Telecommunications System* (UMTS) all-IP network. To increase revenues via third party applications and service providers with IMS, the *Open Service Access* (OSA) platform is proposed. OSA provides unified service creation and execution environments to speed up service deployment that is independent from the underlying UMTS and IMS network technologies. In the UMTS all-IP network, *Authentication, Authorization, and Accounting* (AAA) and security mechanisms are two important issues. Before providing IMS services, the OSA applications should mutually authenticate with the OSA Framework. For secure IMS service access, the *Mobile Station* (MS) should perform the AAA and security mechanisms with the UMTS all-IP network. In this dissertation, we first described the OSA platform development and the mutual authentication with Framework. Then we investigated the AAA and security mechanisms performed between the MS and UMTS all-IP network. This chapter concludes our work in Section 6.1, and briefly discusses directions of the future work in Section 6.2.

#### 6.1 Conclusions

In Chapter 2, we described a CORBA-based OSA system we designed and developed in CCL/ITRI. We showed how OSA *Application Programming Interface* (API) interfaces and functions can be implemented by CORBA mechanism. Then we used the authentication procedure for initial access to illustrate how CORBA mechanism works for CCL OSA. Based on this CORBA mechanism, an OSA *Application Server* (AS) architecture is proposed. In this

architecture, a new application is created by implementing the appLogic module that invokes the SCFs through the appService modules and appService callback modules. When the existing SCFs are reused for new services, the corresponding appService and appService callback modules for accessing these SCFs can also be reused. Through this modularized AS design, the service deployment can be sped up.

For secure IMS service access, Chapters 3 and 4 described the AAA and security mechanisms between the *Mobile Station* (MS) and the wireless access networks. However, the execution of AAA mechanism on handoff may incur long delay and result in force-termination for real-time services. In Chapter 3, we illustrated the IEEE 802.11r transition process for WLAN, where a three-level key hierarchy was proposed to speed up the transition process without executing the expensive IEEE 802.1X authentication for some scenarios. The IEEE 802.11r transition process requires the assignment of unique *Mobility Domain Identities* (MDIDs) worldwide. However, how to guarantee the uniqueness of MDID is not clear. We proposed a mechanism that does not need MDID, and therefore MDID management is eliminated. This mechanism also saves four message exchanges incurred in the original fast BSS transition when MDID ambiguity occurs.

Chapter 4 proposed a key caching mechanism to speed up the inter-*Access Service Network Gateway* (ASN-GW) handoff for mobile WiMAX. With this mechanism, when an MS leaves the old ASN-GW, the MS key record (e.g., the *Master Session Key* (MSK)) is cached in the old ASN-GW. If the MS returns to the old ASN-GW before the MSK lifetime expires, it can reuse the MSK without executing the IEEE 802.1X authentication. On the other hand, the old ASN-GW consumes extra storage to maintain the MS key records when the MS leaves the old ASN-GW. Thus, we investigated how the period  $T$  of the MSK lifetime affects the key caching performance by an analytic model and simulation experiments. We showed that the caching mechanism can effectively speed up the inter-ASN-GW handoff. This chapter also observed that the Exponential  $T$  outperforms the fixed  $T$  in most cases. Moreover, for the reused key

period, the Exponential  $T$  is not affected by the variance of MS residence period in new ASN-GWs, and is more suitable for telecommunications system.

The end-to-end security mechanism between the MS and the OSA AS is also an important security issue in the IMS network. To address this issue, Chapter 5 illustrated the end-to-end secure *Short Message Service* (SMS) service. In this chapter, two applicable end-to-end security mechanisms for SMS based on the RSA scheme and the ID-based scheme are introduced and implemented. The ID-based scheme provides a greater simplification of key distribution by reducing the procedure of obtaining someone's public key. Another advantage of the ID-based scheme is the linear scalability of increasing security level. Our study concludes that the ID-based scheme offers a convenient end-to-end security mechanism for mobile service such as SMS.

## 6.2 Future Work



Based on the research results of this dissertation, the following issues can be investigated further:

**OSA AS Level Authentication:** In this dissertation, we demonstrated that the MS should perform the mutual authentication procedure with the AAA server before associating to the wireless access networks. However, before the MS accesses the OSA AS for service, other authentication procedures may be performed between the MS and the IMS network and between the MS and the OSA AS. In these procedures, the authentication information is most likely retrieved from the HSS/AuC to authenticate the MS. To reduce the redundant authentication, we should integrate these authentication procedures. Moreover, the integrity solution should correctly authenticate the MS with the AAA server and the OSA AS.

**AAA Mechanism:** In the real world, the MS movement may exhibit locality in user's living space, and many MSs are likely to move back to the old ASN-GW in a short period of

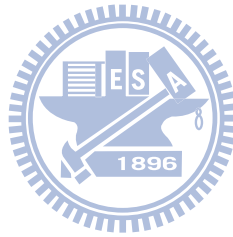
time. For example, a person works in an office for 8 hours, and during the work time, he may leave his office 1 hour for lunch, afternoon tea, or meeting. For this kind of occasionally irregular user's behavior, our key caching mechanism will result in good performance. However, if the ASN-GWs cover the BSs in the highways, the IEEE 802.1X authentication still should be performed when the MS handoff to a new ASN-GW. For smooth real-time service access, a fast inter-authenticator handoff mechanism is needed to reduce the handoff latency without cached MSK. Then we will estimate the new mechanism and the caching mechanism against the MS moving behavior, and show how to select appropriate mechanism.

**Security Mechanism:** After authentication with the AAA server, the MS should perform the security mechanism with the wireless access networks. In both IEEE 802.11r WLAN and mobile WiMAX, the MSK is stored in the authenticator and reused to generate the encryption key (i.e., PTK and TEK) for intra-MD handoff and intra-ASN-GW handoff, respectively. The security mechanism is implemented in a two-level key hierarchy in mobile WiMAX network (i.e., ASN-GW and BS), and is implemented in a three-level key hierarchy in IEEE 802.11r WLAN (i.e., R0HK, R1HK, and AP). In the three-level key hierarchy, the key retrieval procedure from authenticator is omitted in some scenario (see Step 5 in Figure 4.4). Thus the handoff procedure is further speed up. However, extra equipments are required to maintain the security keys (i.e., R1KHs). To prevent the force-termination of real-time service, how to efficiently reduce the handoff latency without maintaining extra equipments is an important issue. We will propose analytic and simulation models to investigate this issue, and show how to select appropriate key hierarchy for various wireless access network environments.

**Key Caching Mechanism:** In this dissertation, we propose a key caching mechanism to speed up the inter-ASN-GW handoff for mobile WiMAX. When an MS leaves the old ASN-GW, the MS key record is cached in the old ASN-GW before the key lifetime expire. However,



when the MSK lifetime  $T \rightarrow \infty$ , the old ASN-GW will keep the unused MS key record for a long time if the MS will not return. To solve this problem, after the departure of the MS, the old ASN-GW should keep the MS key record for a time interval called the *key caching* period instead of the residual MSK lifetime. This key caching period is assigned by the ASN-GW to prevent from caching unused MS key record for a long time. We should propose an analytic model to investigate the impact of the key caching period on the performance.



## Appendix A.

### OSA-based Push to Talk over Cellular Service

This appendix uses the *Push to Talk over Cellular* (PoC) service to illustrate how the OSA *Application Server* (AS) interacts with the *Service Capability Features* (SCFs). Details of the OSA architecture and functionalities are described in Sections 1.2, 2.1, and 2.5. This appendix is organized as follows. Section A.1 introduces the PoC service. Section A.2 shows the OSA AS architecture for PoC service. Section A.3 describes PoC session establishment and termination procedures.

#### A.1 Introduction to Push to Talk over Cellular

The PoC service is a walkie-talkie like service defined by the OMA PoC working group [25]. A PoC session consists of connections among the participants defined in a PoC group. The session is half duplex; that is, at any time, only one participant can speak and all other participants can only listen. When two or more participants attempt to speak, they are arbitrated by a centralized floor control mechanism to select a speaker. In current OMA PoC v2.0 release specification [26], *Session Initiation Protocol* (SIP) is used for session management [33] and both *Real-Time Transport Protocol* (RTP) and *Real-Time Transport Control Protocol* (RTCP) are used for media streams transport and control [34].

A PoC group is a predefined set of participative members. The addresses of these members are maintained in the group member list. The attributes of a PoC group include the display name (the nickname for the group), PoC address (the identifier of the PoC group), etc. There are two alternatives to specify a PoC address: a *Telephone Uniform Resource Identifier* (TEL URI; e.g., tel:+886-3-5131350) or a SIP URI (e.g., sip:poc\_Lab117@pcs1.csie.nctu.edu.tw). The

group member list and the group attributes are maintained in the *Group and List Management Server* (GLMS). Note that, in OMA PoC v2.0 release specification, the GLMS server has been replaced by three components: *shared list XML Document Management Server* (XDMS), *Group XDMS*, and *aggregation proxy*. The shared list XDMS manages user information; e.g., phone book. The group XDMS manages the group member list and the group attributes. The aggregation proxy is the contact point for the PoC participants to access the XDMSs, and the PoC participants should be authenticated by the aggregation proxy.

## A.2 OSA AS Architecture for PoC Service

To offer the PoC service, the AS first accesses the FW, and then communicates with two SCFs: PoC SCF and GLMS SCF. The PoC SCF provides methods to create and control PoC sessions. The GLMS SCF supports methods to retrieve and manage user and group information stored in the GLMS server. The relationship among the FW, SCFs and the AS modules is illustrated in Figure A.1.

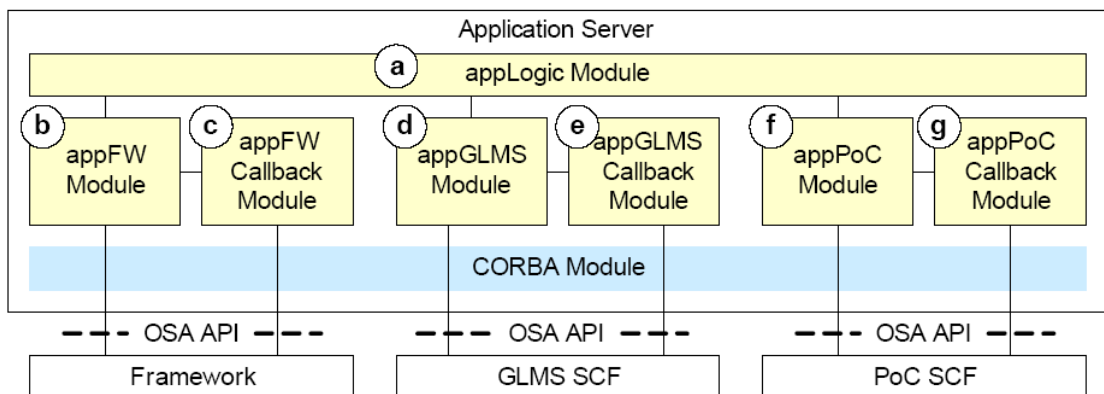


Figure A.1 Application server architecture for PoC Service

- The appLogic module (Figure A.1 (a)) implements the PoC service by integrating the services supported by the FW, the PoC SCF, and the GLMS SCF.
- The appFW module (Figure A.1 (b)) accesses the FW to execute the authentication,

service discovery, and service agreement establishment procedures.

- The appFW callback module (Figure A.1 (c)) provides interfaces to be called back by the FW. Details of these callback interfaces can be found in [3]. In the authentication example described in Section 2.6, this module provides the `IpClientAPILevel- Authentication` interface for the FW to authenticate the AS.
- The appGLMS module (Figure A.1 (d)) accesses the GLMS server through the GLMS SCF.
- The appGLMS callback module (Figure A.1 (e)) provides callback interfaces to the GLMS SCF. Through these interfaces, the GLMS SCF can pass the results to the AS after accessing the GLMS server.
- The appPoC module (Figure A.1 (f)) manages and establishes the PoC sessions by accessing the PoC SCF.
- The appPoC callback module (Figure A.1 (g)) provides callback interfaces. The PoC SCF uses these interfaces to report specific events to the AS. In our implementation, three callback interfaces are provided. The `IpAppPoCCallControlManager` interface provides the management functions for PoC service (see Steps 6, 19, and 30 in Figure A.2). The `IpAppPoCCall` interface provides the PoC SCF with the control management functions for PoC sessions (see Step 4 in Figure A.3). The `IpAppPoCCallLeg` interface is used to receive specific events of group members from the PoC SCF (see Step 26 in Figure A.2 and Step 1 in Figure A.3).

PoC session establishment consists of four stages:

**Stage 1.** The appLogic module accesses the FW to perform mutual authentication, service discovery, and service agreement establishment through appFw and appFw callback modules. After this stage, the AS obtains the object references for the PoC SCF and the GLMS SCF. The details are described in Section 2.6.

**Stage 2.** The appLogic module requests the appPoC callback module to provide the callback

interface `IpAppPoCCallControlManager` through the `appPoC` module. With this callback interface, the PoC SCF notifies the `appLogic` module when a PoC session request occurs. The details are described in Section A.3.

**Stage 3.** When receiving a request, the `appLogic` module retrieves the group member list by accessing the GLMS SCF through the `appGLMS` and the `appGLMS` callback modules or by accessing the AS database.

**Stage 4.** After the group member list is obtained, the `appLogic` module establishes the PoC session for members in the list through the `appPoC` and the `appPoC` callback modules. The details are described in Section A.3.

### A.3 PoC Session Establishment

In our OSA-based PoC design, the AS first sets up the callback interface `IpAppPoCCallControlManager` (see Steps 1 ~ 3 in Figure A.2) and subscribes notifications for specific PoC-related events from the PoC SCF (see Steps 4 and 5 in Figure A.2). With this subscription, when a PoC event (e.g., PoC session establishment request) occurs at the SCF, a notification will be sent to the AS. When receiving an event notification from the PoC SCF (see Step 6 in Figure A.2), the AS starts to initiate a PoC session. In the following example, a PoC session for PoC group with the display name “Lab117” is requested to be established. This group is identified by the SIP URI “`sip:poc_Lab117@pcs1.csie.nctu.edu.tw`”, which contains two members: party A and party B. The PoC session establishment procedure is described in the following steps (see Figure A.2).

**Step 1.** The AS control logic `appLogic` invokes `appPoC` function `setCallbackForIpAppPoCCallControlManager`. This function instructs `appPoC` to generate the `IpAppPoCCallControlManager` interface object and to pass the object reference to the PoC SCF. The `IpAppPoCCallControlManager` interface provides management functions for PoC service (e.g., function `reportMediaNotificationWithGroupID` invoked

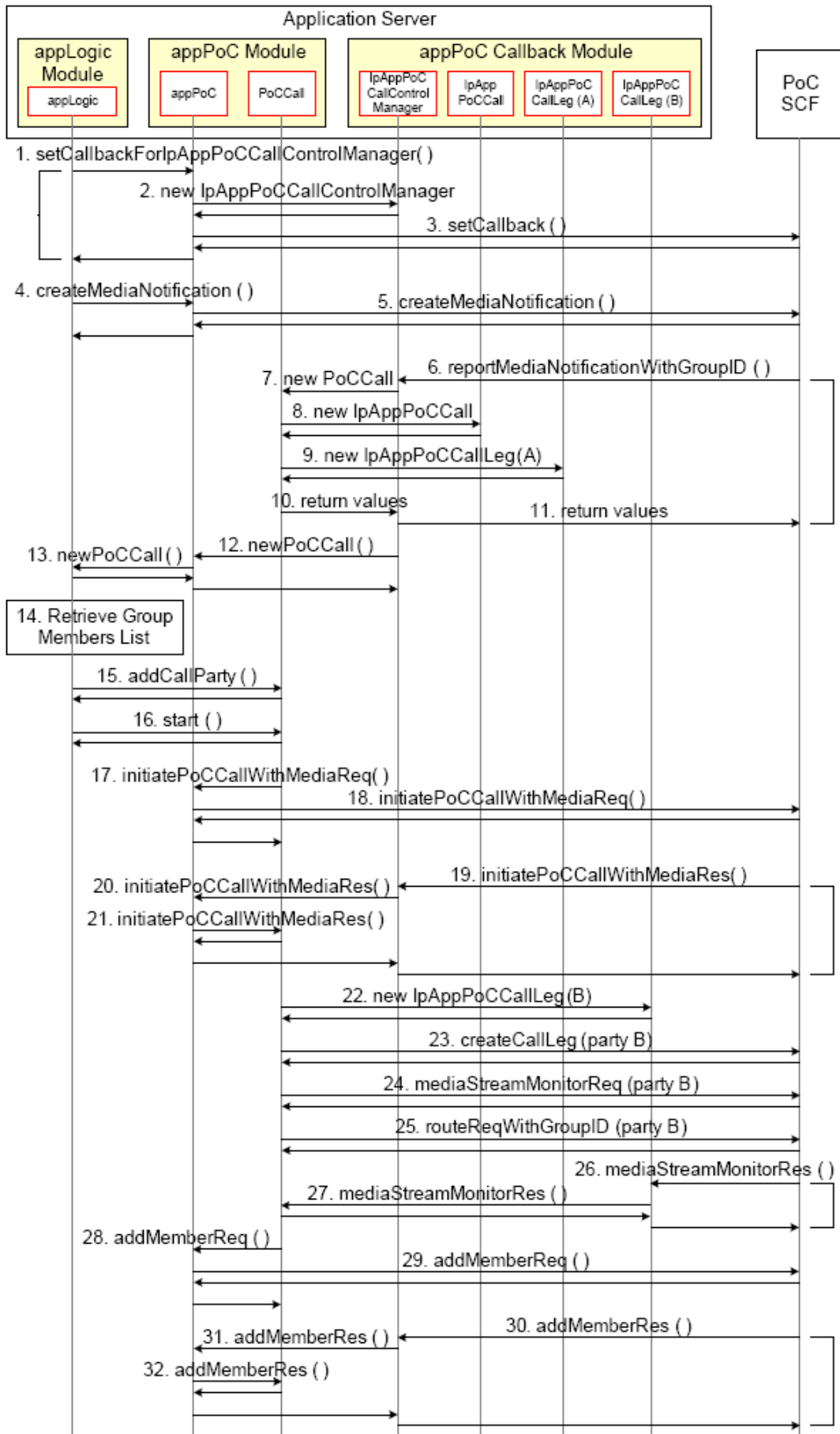


Figure A.2 Message flow for PoC session establishment

at Step 6, function `initiatePoCCallWithMediaRes` invoked at Step 19, and function `addMemberRes` invoked at Step 30).

**Step 2.** `appPoC` creates the `IpAppPoCCallControlManager` object to provide the callback functions to be called by the PoC SCF.

**Step 3.** `appPoC` invokes function `setCallback` to pass the reference of `IpAppPoCCallControlManager` to the PoC SCF. Through this reference, the PoC SCF invokes the callback functions at Steps 6, 19, and 30.

**Steps 4 and 5.** `appLogic` invokes the PoC SCF function `createMediaNotification` through `appPoC` to request notifications for specific PoC-related events (e.g., the PoC SCF receives a session request for a PoC group).

**Step 6.** Suppose that PoC party A attempts to establish a PoC session with group `Lab117` that has the SIP URI “`sip:poc_Lab117@pcs1.csie.nctu.edu.tw`”. The PoC SCF invokes the callback function `reportMediaNotificationWithGroupID` to notify the AS of this PoC session request. This notification includes the media stream information for party A (e.g., codec, the algorithm for compressing and decompressing voice information, such as G.711, G.723, etc) and the SIP URIs of both party A and group `Lab117`.

**Step 7.** `IpAppPoCCallControlManager` creates the `PoCCall` object to store information associated with this new PoC session (e.g., SIP URIs of the PoC group and group members, media stream information, the interface object references related to this session in both the PoC SCF and the AS, etc).

**Steps 8 and 9.** `PoCCall` generates the `IpAppPoCCall` object and the `IpAppPoCCallLeg` object for party A (i.e., `IpAppPoCCallLeg (A)` in Figures A.2 and A.3). `IpAppPoCCall` provides the PoC SCF with the session control management functions (e.g., function `callEnded` invoked at Step 4 in Figure A.3). `IpAppPoCCallLeg (A)` is used to receive specific events of party A from the PoC SCF.

**Steps 10 and 11.** Both interface object references generated at Steps 8 and 9 are stored in

PoCCall and passed back to the PoC SCF as the return values of function reportMediaNotificationWithGroupID invoked at Step 6.

**Steps 12 and 13.** IpAppPoCCallControlManager invokes function newPoCCall through appPoC to notify appLogic of the PoC session request.

**Step 14.** appLogic retrieves the group member list of the requested group Lab117. The list can be obtained from the GLMS server through the GLMS SCF or from the AS database.

**Step 15.** After the group member list (i.e., SIP URIs of party A and party B) is obtained, appLogic stores it in PoCCall through function addCallParty.

**Step 16.** appLogic invokes the PoCCall function start to establish the PoC session.

**Steps 17 and 18.** PoCCall invokes the PoC SCF function initiatePoCCallWith- MediaReq through appPoC. This function asks the PoC SCF to reserve resources for the PoC session and party A; e.g., reserve a RTP port to transmit media streams to party A.

**Steps 19 - 21.** When the resources for the PoC session and party A are reserved, PoC SCF invokes the callback function initiatePoCCallWithMediaRes through the IpAppPoC- CallControlManager and appPoC to inform PoCCall that the resources are reserved successfully.

**Step 22.** PoCCall generates the IpAppPoCCallLeg object for party B (i.e., IpAppPoC- CallLeg (B) in Figures A.2 and A.3). This object is used to receive party B related events from the PoC SCF (e.g., function callLegEnded invoked at Step 1 in Figure A.3).

**Step 23.** PoCCall invokes function createCallLeg and passes the callback reference of IpAppPoCCallLeg (B) to the PoC SCF.

**Step 24.** PoCCall invokes the PoC SCF function mediaStreamMonitorReq. This function requests the PoC SCF to report the media stream information of party B to the AS when party B joins in the PoC session (see Steps 26 and 27).

**Step 25.** To invite party B to join in the session, PoCCall invokes the PoC SCF function routeReqWithGroupID. This function requests the PoC SCF to deliver the invite



message to party B.

**Steps 26 and 27.** When party B joins in the session, the media stream information of party B is passed to the PoC SCF. The PoC SCF invokes the callback function `mediaStream-MonitorRes` through `IpAppPoCCallLeg (B)` to notify `PoCCall` about the media stream information for party B.

**Steps 28 and 29.** `PoCCall` invokes the PoC SCF function `addMemberReq` through `appPoC` to reserve resources for party B; e.g., a RTP port to transmit media streams to party B.

**Steps 30 - 32.** The PoC SCF invokes the callback function `addMemberRes` through `IpAppPoCCallControlManager` and `appPoC` to inform `PoCCall` that the resources have been reserved for party B.

At this moment, the PoC session is successfully established for parties A and B. To end a PoC session, the PoC session termination procedure is executed (see Figure A.3). The details are given below.

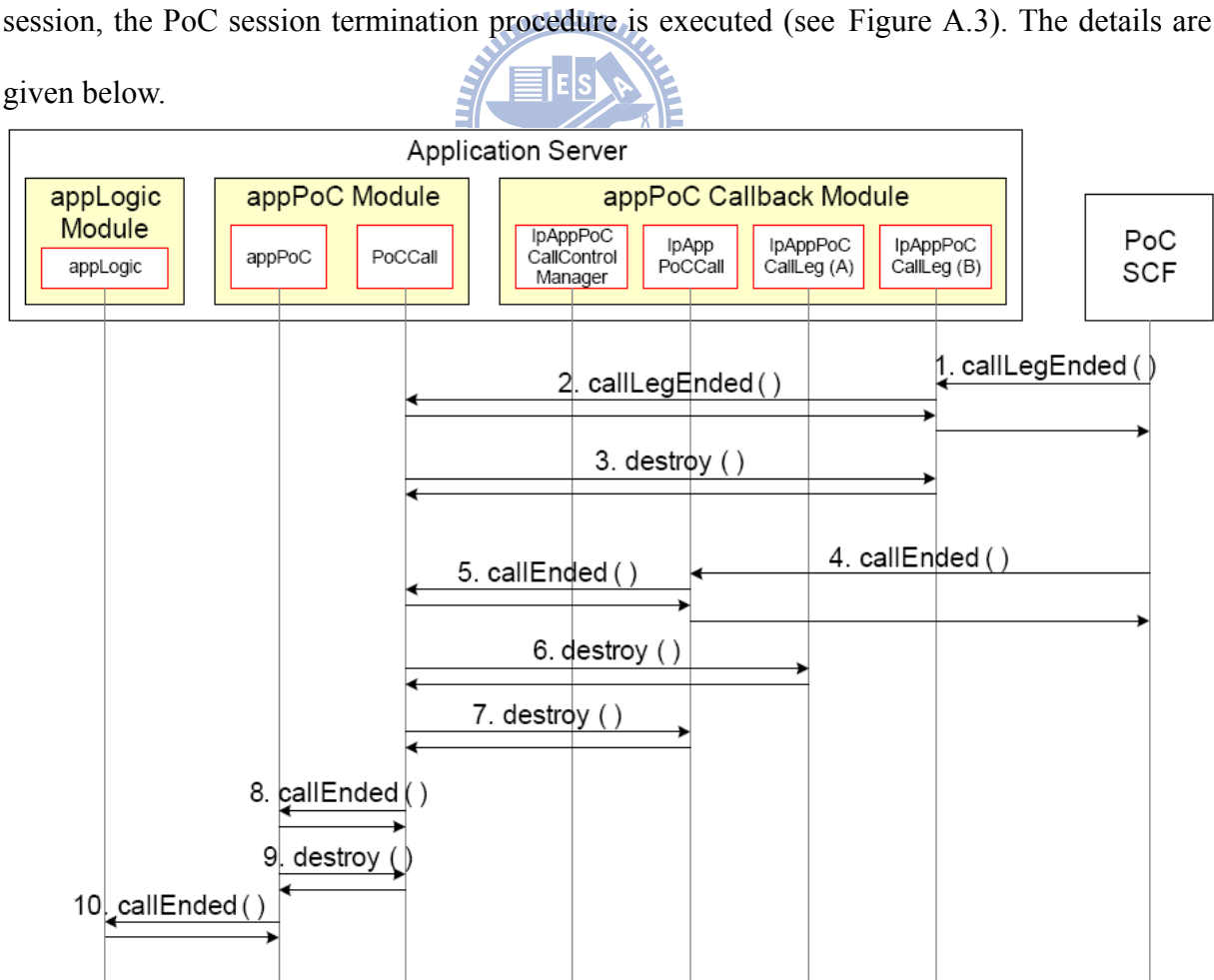


Figure A.3 Message flow for PoC session termination

**Steps 1 and 2.** When party B leaves the PoC session, the PoC SCF invokes callback function `callLegEnded` through `IpAppPoCCallLeg (B)` to notify `PoCCall` that party B has left.

**Step 3.** `PoCCall` invokes the function `destroy` of `IpAppPoCCallLeg (B)` to destroy this object.

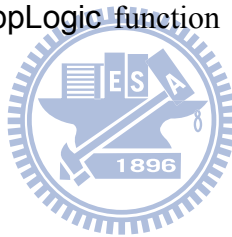
**Steps 4 and 5.** When the last party leaves the session (i.e., party A in this example), the PoC SCF invokes the callback function `callEnded` through `IpAppPoCCall` to notify `PoCCall` of the leaving for the last party.

**Steps 6 and 7.** Since all call parties have left, `PoCCall` invokes functions `destroy` of both `IpAppPoCCallLeg (A)` and `IpAppPoCCall` to destroy these objects.

**Step 8.** `PoCCall` invokes the `appPoC` function `callEnded` to notify the end of the PoC session.

**Step 9.** `appPoC` invokes the `PoCCall` function `destroy` to delete this `PoCCall` object.

**Step 10.** `appPoC` invokes the `appLogic` function `callEnded` to notify the end of the PoC session.



# Bibliography

- [1] 3GPP. Third Generation Partnership Project; Technical Specification Group Services and Systems Aspects; Virtual Home Environment/Open Service Access. 3G TS 23.127 v6.1.0, 2004.
- [2] 3GPP. Third Generation Partnership Project; Technical Specification Group Core Network; Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview; (Release 6). 3G TS 29.198-1 v6.3.1, 2004.
- [3] 3GPP. Third Generation Partnership Project; Technical Specification Group Core Network; Open Service Access (OSA); Application Programming Interface (API); Part 3: Framework. 3G TS 29.198-03 v6.4.0, 2005.
- [4] 3GPP. Third Generation Partnership Project; Technical Specification Group Services and Systems Aspects; Service Requirement for the Open Service Access (OSA); Stage 1. 3G TS 22.127 v7.0.0, 2005.
- [5] Bargh, M.S., Hulsebosch, R.J., Eertink, E.H., Prasad, A., Wang, H., and Schoo, P. Fast Authentication Methods for Handovers between IEEE 802.11 Wireless LANs. *ACM WMASH 2004*, 51-60, 2004.
- [6] Berndt, H., Hamada, T., and Graubmann, P. TINA: Its Achievements and Its Future Directions. *IEEE Communications Surveys and Tutorials*, 3(1): 2-11, 2000.
- [7] Boneh, D., and Franklin, M. Identity-based Encryption from the Weil Pairing. *Advances in Cryptology-CRYPTO'01*, 2139: 213–239, 2001.
- [8] Chang, M.-F., Wu, L.-Y., and Lin, Y.-B. Performance Evaluation of a Push Mechanism for WLAN and Mobile Network Integration. *IEEE Transactions on Vehicular Technology*, 55(1): 380-383, 2006.
- [9] Chen, J.-H., Pang, A.-C., Sheu, S.-T., and Tseng, H.-W. High Performance Wireless

- Switch Protocol for IEEE 802.11 Wireless Networks. *ACM Mobile Networking and Applications*, 10(5): 741-751, 2005.
- [10] Eastlake, D., and Jones, P. US Secure Hash Algorithm 1 (SHA1). IETF RFC 3174, 2001.
- [11] Hankerson, D., Menezes, A., and Vanstone, S. *Guide to Elliptic Curves Cryptography*. Springer-Verlag, 2004.
- [12] Haverinen, H., and Salowey, J. Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). RFC 4186, 2006.
- [13] Hung, H.-N., Lin, Y.-B., Lu, M.-K., and Peng, N.-F. A Statistic Approach for Deriving the Short Message Transmission Delay Distributions. *IEEE Transactions on Wireless Communications*, 3(6): 2345-2352, 2004.
- [14] Hwu, J.-S., Chen, R.-J., and Lin, Y.-B. An Efficient Identity-based Cryptosystem for End-to-end Mobile Security. *IEEE Transactions on Wireless Communications*, 5(9): 2586-2593, 2006.
- [15] IEEE. IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Fast Basic Service Set (BSS) Transition. IEEE Standard 802.11r, 2008.
- [16] ITU-T Recommendation X.509. Information Technology - Open Systems Interconnection - the Directory: Public-key and Attribute Certificate Frameworks. ITU-T, 2000.
- [17] Kassab, M., Belghith, A., Bonnin, J.-M., and Sassi, S. Fast Pre-Authentication based on Proactive Key Distribution for 802.11 Infrastructure Networks. *ACM Wireless Multimedia Networking and Performance Modeling*, October 2005.
- [18] Kleinrock, L. *Queueing System Volume 1: Theory*. John Wiley & Sons, 1976.

- [19] Lin, Y.-B. Performance Modeling for Mobile Telephone Networks. *IEEE Network*, 11(6): 63-68, 1997.
- [20] Lin, Y.-B., and Chlamtac, I. *Wireless and Mobile Network Architectures*. John Wiley and Sons, 2001.
- [21] Lin, Y.-B., and Pang, A.-C. *Wireless and Mobile All-IP Networks*. Wiley, 2005.
- [22] Ma, W., Fang, Y., and Lin, P. Mobility Management Strategy based on User Mobility Patterns in Wireless Networks. *IEEE Transactions on Vehicular Technology*, 56(1): 322-330, 2007.
- [23] Madson, C., and Glenn, R. The Use of HMAC-SHA-1-96 within ESP and AH. IETF RFC 2404, 1998.
- [24] Moerdijk, A.-J., and Klostermann, L. Opening the Networks with Parlay/OSA: Standards and Aspects behind the APIs. *IEEE Network*, 17(3):58-64, 2003.
- [25] Open Mobile Alliance. Push to Talk over Cellular (PoC) – Architecture. OMA-AD-PoC-V2\_0-20080507-C, 2008.
- [26] Open Mobile Alliance. PoC User Plane. OMA-TS-PoC\_UserPlane-V2\_0-20080507-C, 2008.
- [27] Pang, A.-C., and Chen, Y.-K. A Study on Availability of Mobility Databases. *International Conference on Information Networking (ICOIN)*, 195-200, 2004.
- [28] Pyarali, I., and Schmidt, D.C. An Overview of the CORBA Portable Object Adapter. *ACM StandardView Magazine*, 6(1):30-43, 1998.
- [29] Qian, Y., Hu, R.-Q., and Chen, H.-H. A Call Admission Control Framework for Voice over WLANs. *IEEE Wireless Communications*, 13(1): 44-50, 2006.
- [30] Revest, R., Shamir, A., and Aldeman, L. A Method for Obtaining Digital Signature and Public Key Cryptosystems. *Communication of the ACM*, 21(2): 120-126, 1978.
- [31] Rong, B., Qian, Y., Lu, K., Chen, H.-H., and Guizani, M. Call Admission Control Optimization in WiMAX Networks. *IEEE Transactions on Vehicular Technology*, 57(4):

2509-2522, 2008.

- [32] Ross, S.M. *Stochastic Processes*. John Wiley & Sons, 1996.
- [33] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E. SIP: Session Initiation Protocol. IETF RFC 3261, 2002.
- [34] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. RTP: A Transport Protocol for Real-Time Applications. IETF RFC 3550, 2003.
- [35] Shamir, A. Identity-based Cryptosystems and Signature Schemes. *Advances in Cryptology—CRYPTO'84*, 196: 47-53, 1984.
- [36] Stallings, W. *Cryptography and Network Security*. Prentice Hall, 1999.
- [37] Walkden, M., Edwards, N., Foster, D., Jankovic, M., Odadzic, B., Nygreen, G., Moiso, C., Tognon, S.M., de Bruijn, G. Open Service Access: Advantages and Opportunities in Service Provisioning on 3G Mobile Networks Definition and Solution of Proposed Parlay/OSA Specification Issues. *Project P1110 Technical Information EDIN 0266-1110*, 2002.
- [38] WiMAX Forum. WiMAX Forum Network Architecture, Stage 3: Detailed Protocols and Procedures. WiMAX Forum, Release 1.1.0, 2007.
- [39] Yang, S-R. Dynamic Power Saving Mechanism for 3G UMTS System. *ACM/Springer Mobile Networks and Applications (MONET)*, 12(1): 5-14, 2007.

# Publication List

## ● Journal Publications

1. Chou, C.-M., Hsu, S.-F., Lee, H.-Y., Lin, Y.-C., Lin, Y.-B., and R.S. Yang. CCL OSA: A CORBA-based Open Service Access System. *International Journal of Wireless and Mobile Computing*, 1(3/4): 289-295, 2006.
2. Hwu, J.-S., Hsu, S.-F., Lin, Y.-B., and Chen, R.-J. End-to-end Security Mechanisms for SMS. *International Journal of Security and Networks*, 1(3/4): 177-183, 2006.
3. Hsu, S.-F., Lin, Y.-C., and Lin, Y.-B. An OSA Application Server for Mobile Services. *International Journal of Pervasive Computing and Communications*, 3(1/2): 102-113, 2007.
4. Hsu, S.-F., and Lin, Y.-B. Selecting Transition Process for WLAN Security. *Wireless Communications and Mobile Computing*, 8(7): 921-925, 2008.
5. Hsu, S.-F., and Lin, Y.-B. A Key Caching Mechanism for Reducing WiMAX Authentication Cost in Handoff. Accepted and to appear in *IEEE Transactions on Vehicular Technology*.