國立交通大學

資訊科學與工程研究所

碩 士 論 文

在無線區域網路 802.1X 認證下之快速換手機制

Fast Handoff Mechanism for WLAN with 802.1X Authentication

研 究 生：許詩賢
指導教授：張明峰　教授

中 華 民 國　九 十 五　年 六 月

在無線區域網路 802.1X 認證下之快速換手機制
Fast Handoff Mechanism for WLAN with 802.1X Authentication

研 究 生：許詩賢　　Student：Shih-Hsien Shu

指導教授：張明峰　　Advisor：Ming-Feng Chang

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

Computer Science A Thesis

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

Master

in

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

在無線區域網路 802.1X 認證下之快速換手機制

學生：許詩賢　　指導教授：張明峰 教授

國立交通大學資訊工程學系〈研究所〉碩士班

摘　要

IEEE 802.11 無線區域網路〈Wireless LAN, WLAN〉已經日漸普及，在連結層〈link layer〉為使用者提供了移動性〈user mobility〉為其主要的優勢。在其應用上，如果結合了 VoIP〈Voice over IP〉而成為無線語音服務〈VoIP over WLAN, VoWLAN〉，我們相信此服務將成為在無線區域網路上的一個非常重要的殺手級應用。

VoIP 是及時性的應用服務〈real-time application〉，而且語音封包〈voice packets〉是延遲敏感的〈delay-sensitive〉。無線區域網路提供一個無線寬頻的資料存取途徑，然而其無線電波涵蓋範圍較小，切換無線存取點〈access points, APs〉是很常發生的情況。IEEE 802.1X 為無線區域網路提供了對使用者認證〈user authentication〉的機制。由於是只有在換手後才開始進行 802.1X 的認證，再加上認證的過程非常耗時，無法滿足 VoIP 對於語音封包低延遲性的需求。

本篇論文檢視無線語音服務，在換手後再進行 802.1X 認證所遭遇的問題，提出一個新的認證方法來解決此問題。此認證方法主要目的在於：減少認證訊息的交換與重覆使用已授權的權仗〈authorized access token〉直到它的使用期限到期為止。本篇論文針對在無線區域網路 802.1X 認證下，所造成的驗證延遲〈authentication latency〉，發展一個快速換手機制，以期縮短此驗證延遲，儘可能達到 VoIP 對於語音封包低延遲性的需求。

Fast Handoff Mechanism for WLAN with 802.1X Authentication

Student: Shih-Hsien Shu Advisor: Prof. Ming-Feng Chang

Department of Computer Science and Information Engineering

National Chiao Tung University

Abstract

The IEEE 802.11 Wireless LAN (WLAN) has become increasingly popular nowadays. It provides user mobility at link layer as its major advantage. If we take VoIP (Voice over IP) into consideration, VoWLAN (Voice over WLAN) becomes a cross-layer integrated application services above the WLAN. We believe that it should become a killer application of the WLAN.

VoIP is one kind of real-time application services and it has a feature that voice packets are delay-sensitive. Although the WLAN provides a wide bandwidth wireless link to access data networks, but due to its limited radio coverage, it is common that roaming user device changes wireless access points (APs) frequently. IEEE 802.1X provides WLAN with an user authentication framework. Because 802.1X starts only after handoff completes and its authentication process is time-consuming, it does not satisfy the low delay requirement of VoIP.

In this thesis, we investigate the VoWLAN handoff problem under 802.1X authentication, and propose a new authentication method to solve this problem. The goal of this new authentication method is to reduce the number of authentication message exchange and to reuse the authorized token until its lifetime is expired. Our goal is to develop a fast handoff

mechanism for VoWLAN under 802.1X authentication to minimize the 802.1X authentication latency and meet the low delay requirement of VoIP as possible as we can.

誌　　謝

首先感謝我最敬重的導師 張明峰教授。在就讀碩士班的這兩年，在導師的耐心教誨下，讓我得以順利完成研究，而導師對指導學生們的細心呵護，更是讓我們感動萬分。亦感謝 曹孝櫟老師及 逄愛君老師對於本篇論文，所給予寶貴的意見與指教。

在這兩年學習過程，感謝博士班的孟達與仲雍及弘鑫學長不辭辛勞的給予建議和提供相關的研究素材；感謝榮泰學長的承先，讓我可以啟後；感謝在實驗室的好伙伴至鴻同學，以及後偉、名均、聖全、博今學弟們的支持和激勵下，提供我在研究過程中源源不斷奮鬥下去的動力；也感謝其他實驗室的雅聯與宜榮同學，一起鑽研在無線網路的領域；感謝室友整賢同學，提供技術上的支援；再感謝之仁同學，提供書籍的資源；同時再一次感謝導師提供良好的研究環境與實驗儀器，使我可以在研究的過程中，沒有受到任何的阻礙與限制。

最後將此論文獻給我最親愛的父母與家人。感謝您們在我求學期間全心全意的支持，讓我得以順利的完成研究。

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 VoWLAN and 802.1X Authentication

VoIP (Voice over IP) over Wireless LAN (WLAN), also referred to as VoWLAN, has become one of the important applications of the 802.11 WLAN. Due to the advantages of WLANs, such as ease of deployment, low costs, and user mobility, the IEEE 802.11 WLAN has been increasingly popular up to now. Mobility is the most important feature provided by wireless networks. Conventional applications used to run on wired networks, move to wireless networks and may not work as well on wireless networks. If we take VoIP into consideration, voice over WLAN (VoWLAN) is expected to become one of the most attractive services used in mobile devices over wireless networks.

However, the combination of WLAN and VoIP, is not perfect and has some problems. WLAN and VoIP both have their attributes, some of which are opposite. When a mobile user is exercising VoWLAN and moving from the radio coverage of a serving 802.11-defined Access Point (AP) to another, the change of AP association according to current signal strength and quality is called handoff. According to the IEEE 802.11 specification, a mobile station can be associated with only one AP at any given instant. The 802.11 WLAN is designed to provide data access of wide bandwidth, but it has limited radio coverage. It overlooks the problem that radio disconnection of seconds may affect real-time interactive user applications. VoIP is one kind of real-time application services and it has some requirements that voice packets are delay-sensitive but tolerable of few packets lost. Consequently, one of problems of VoWLAN is that it lacks an efficient handoff procedure for supporting on-going VoIP communications when a mobile user is roaming between APs. The handoff latency contributes to great quality degradation of a VoIP communication over the 802.11 WLAN.

The IEEE 802.1X provides a user authentication framework for the 802.11 WLAN, but has side effects against VoWLAN operations. 802.1X is based on EAP (Extensible Authentication Protocol) which was initially developed as an authentication extension for PPP (Point-to-Point Protocol). 802.1X is port-based network access control which means that the ports to which authenticated users are connected are authorized or valid and any authenticated users' packets can pass through them. The ports of unauthenticated users are invalid or unauthorized and 802.1X drops packets through them except EAP authentication packets, i.e., only EAP authentication packets can pass through the unauthorized ports during authentication process. However, 802.1X was used to user authentication on wired networks and a port used to be a physical and static connection. It does not consider if the ports are mobile, and the problems resulted from user mobility. Now, take 802.1X for 802.11 WLAN user authentication for example, one 802.11 association is defined as a logical connection, i.e., a logical port. The port for 802.11 WLAN is dynamically and logically created after the 802.11 association completes. Only after the 802.11 handoff completes does the 802.1X start authentication process. Unfortunately, the 802.1X authentication process is time-consuming because of many round-trip transaction packets for authentication. We call this "the 802.1X authentication latency" and take it into consideration of the 802.11 handoff latency. During the 802.1X authentication process, any non-EAP authentication packets, especially including voice packets, are dropped. The VoIP communication is interrupted until the 802.1X authentication success. Obviously, the 802.1X authentication latency does not meet the requirement of low-latency feature of VoIP. As a result, this is the major problem of VoWLAN under 802.1X authentication. In this thesis, we propose an efficient handoff scheme for VoWLAN under 802.1X authentication to reduce the 802.1X authentication latency.

1.2 Related work

There are many researches investigating the topic of handoff process of the 802.11

WLAN at the data link layer. [1] reported an experimental analysis of the IEEE 802.11 handoff process at the MAC layer. They concluded that the delay of searching available APs on every channel from 1 to 11 accounts for more than 90% of the overall handoff delay. [2] analyzed that the event that three consecutive collisions occurs when transmitting a packet rarely happens, then suggested reducing the time for detecting lack of radio link by quickly reacting to packet losses. [3] suggested reducing the handoff delay by scanning only the channels selected in neighbor graph (NG), which is an undirected graph with each edge representing a mobility path between APs. [4] developed a handoff procedure to reduce the MAC layer handoff latency. They proposed a fast handoff algorithm including the selective scanning procedure and the caching procedure.

The researches above improved the handoff procedure only at the link layer, few of them concerned the 802.1X authentication. The 802.1X authentication ensures the access right to user application or services. It is a time-consuming procedure and starts after the completion of 802.11 association. Few studies lay stress on an efficient 802.1X handoff for the 802.11 WLAN such that the overall handoff delay can still satisfy the requirements of VoIP communications. The 802.1X is still an open issue, having different authentication methods and options. The more authentication messages exchange, the longer it takes to complete the 802.1X authentication. Original 802.1X authentication takes from 400 to 1000 ms to accomplish 802.1X handoff by using from EAP-MD5 to EAP-PEAP authentication methods. But it is too long for VoIP applications. These result stall takes long time to perform 802.1X handoff.

1.3 The purpose of this thesis and its overview

In this thesis, we propose a new 802.1X authentication method to reduce the 802.1X authentication latency as well as handoff delay. Our method aims to reduce the number of

802.1X authentication messages exchange during handoff and make only minimum changes to the existing wireless LAN deployments. The fewer 802.1X messages exchanges, the less opportunity of hack it leads. Also, the faster it takes to complete the 802.1X authentication, the fewer packets it drops, especially including the voice packets. We assume that each 802.11-defined AP supports the 802.1X authentication and an additional authentication server, typically a Radius server, exists. Our method reduces both the handoff latency and the packet loss.

The remaining of this thesis is organized as follows. In Chapter 2, we briefly discuss the related parts of the IEEE 802.11 specification, the IEEE 802.1X specification and the Kerberos. In Chapter 3, we present our solution that is a new authentication method for VoWLAN under 802.1X authentication. In Chapter 4, we implement the design by using open source software projects, including the Xsupplicant, Hostapd, as well as FreeRADIUS, and then evaluate the 802.1X authentication latency of our mechansim. In Chapter 5, we summarize and conclude our work.

Chapter 2 Preliminaries

First, we briefly describe some important parts of the IEEE 802.11 specification [6], including the network architecture and the handoff process. Then, we illustrate the authentication architecture of the 802.1X specification [7] as well as the typical 802.1X authentication message exchange and introduce the RADIUS protocol and some EAP authentication methods. Finally, we introduce the Kerberos [11] authentication architecture which is referenced by our mechanism.

2.1 The IEEE 802.11 Architecture

According to the IEEE 802.11 specification, the basic service set (BSS) is the basic building block of WLAN and there are two types of network architectures, as shown in Figure 1, the ad-hoc mode (independent BSS) and the infrastructure mode (infrastructure BSS). In the ad-hoc mode, each station can directly communicate with each other unless they are close enough. In the infrastructure mode, stations must be connected with a central serving unit, access point (AP), to communicate with others and only through the AP can they communicate with each other no matter how close they are. As shown in Figure 2, multiple BSSs can be combined to form an extended service set (ESS) to provide a larger radio coverage area, which contributes to roaming convenience. In infrastructure BSS, the BSSID (Basic Service Set Identity) is the MAC address of wireless interface of an AP. The SSID (Service Set Identity) is a string identifier for service set(s) and can be viewed as the wireless network name of an AP or APs.

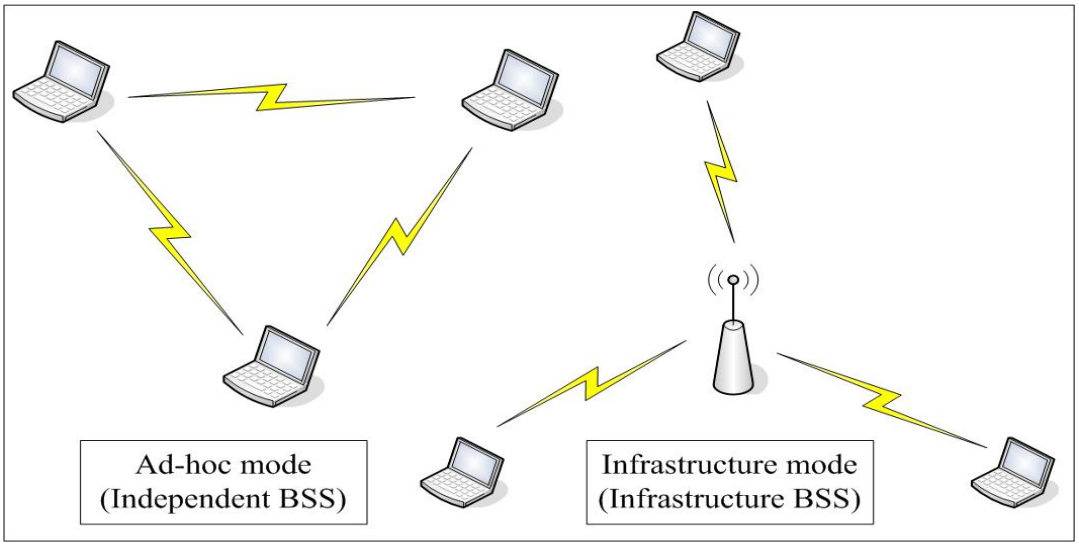


Figure 1: The IEEE 802.11 Network Architectures

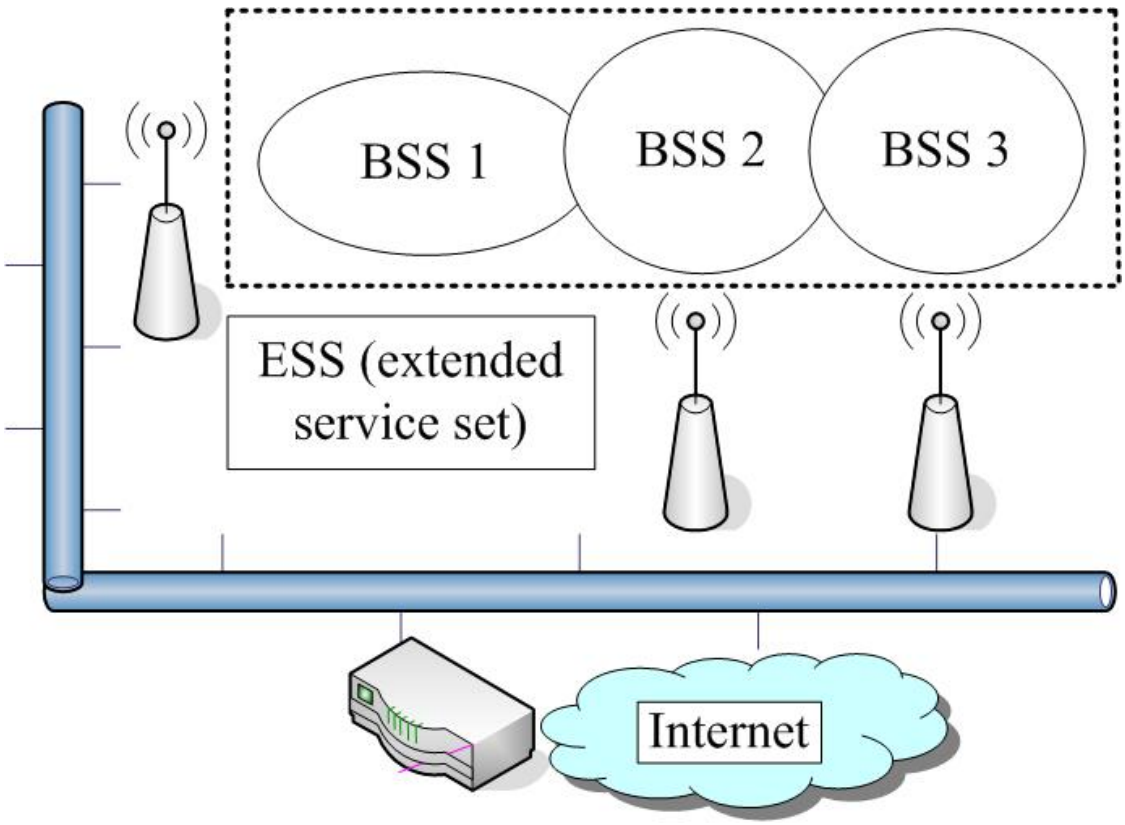


Figure 2: The extended service set (ESS)

2.2 The IEEE 802.11 Handoff Process

The handoff could be defined as the change of APs. For example, when a mobile station moves from BSS1 to BSS2, as shown in Figure 2, it may change the associated AP from AP1

to AP2. One wireless interface can be associated with only one AP at any time. The handoff procedure could be composed of three phases orderly: detection, search, and execution phases.

In detection phase, the handoff may be triggered due to the scenario that a mobile station is moving from one BSS to another such that it detects the fading of radio signal strength, too many errors of frame transmissions or a high packet loss rate, then it decides to search a better link quality of another AP and makes the decision to handoff to that target AP.

In search phase, the mobile station starts searching for available wireless networks which can provide better quality of radio link. According to the IEEE specification, there are two kind of scanning functions at MAC layer to locate the existence of APs. One is the active scanning mode; the other is the passive scanning mode. In the passive scanning mode, the mobile station listens for beacon frames transmitted by APs. Every AP transmits beacon frames at a regular interval on one specific channel. Different APs may operate on different channels, so mobile stations should switch to one channel to collect beacon frames, repeat switch to next channel until all channels are passively scanned. In the active scanning mode, the mobile station sends probe request frames to every channel and waits for responses. If some AP on its channel receives probe request frames, it should respond with probe response frames. If no AP on some channel, mobile stations will wait until timer expires, then switch to another channel.

In execution phase, the mobile station has got enough AP information, then it may choose a better link quality among them as the target AP to associate with. The execution phase could be composed of two steps orderly: authentication and (re)association. As shown in Figure 3, there are two authentication algorithms in the IEEE specification. Both of authentication algorithms are authentication to device, not to user.

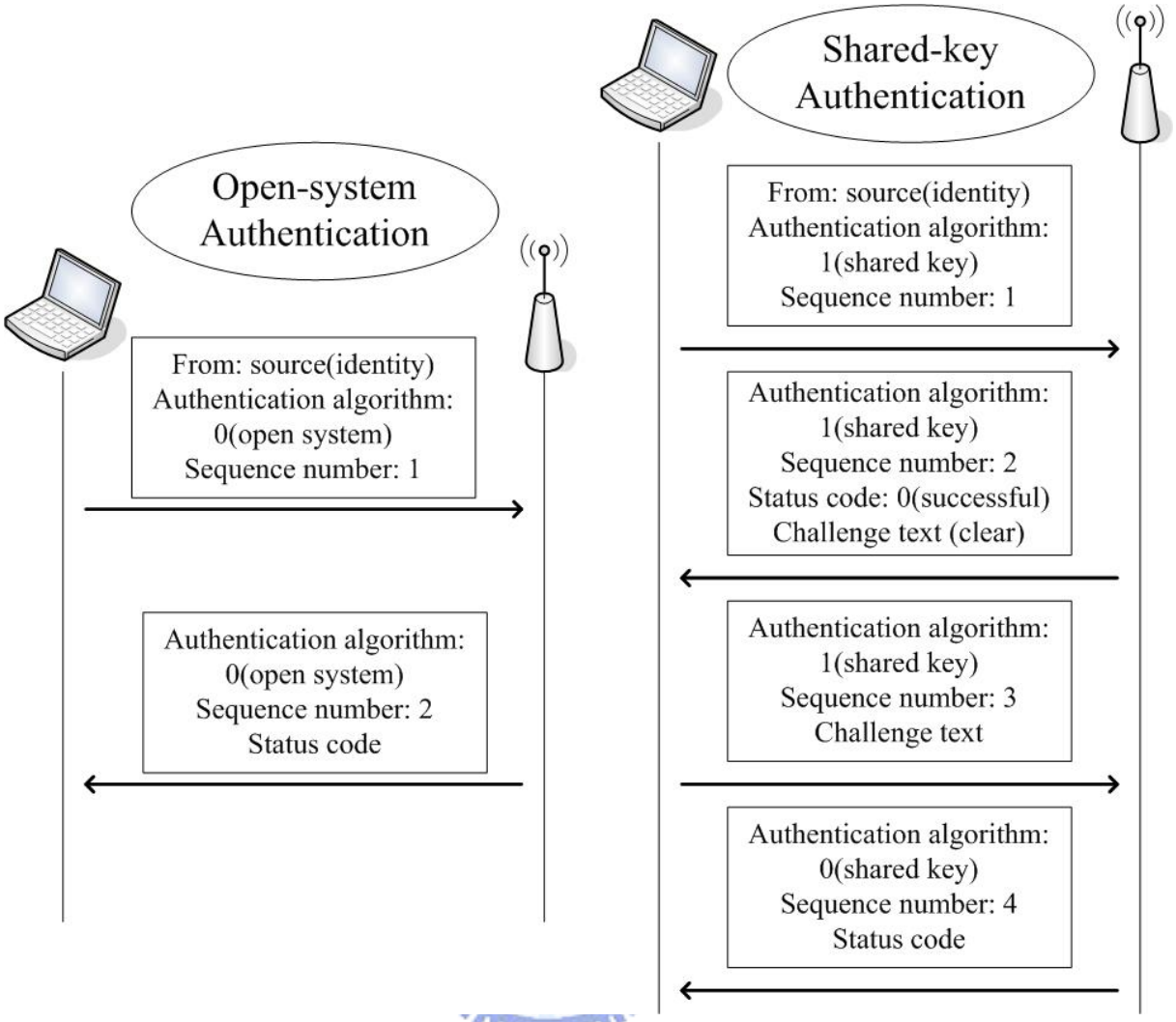


Figure 3: The IEEE 802.11 Authentication Procedures

One is Open-system authentication, which is simple and involves only two transaction messages. First, the mobile station sends authentication request frames, including its identity and which authentication algorithm it prefers. Second, the target AP replies with authentication response frames to indicate whether authentication is successful or not. The other is Shared-key authentication, which involves four transaction messages and uses Wired Equivalent Privacy (WEP) encryption algorithm. It verifies whether a mobile station is in possession of a secret key shared between them to grant access or not. It uses a 128-byte-length challenge text generated by the WEP pseudo-random number generator (PRNG). A mobile station must use its WEP key to encrypt this text to authenticate itself. After authentication is successful, the mobile station can send (re)association request frames

to gain access of wireless networks.

The Shared-key authentication has some authentication weakness as describe in [10]. As result, most people take the IEEE 802.1X as a user authentication solution for the 802.11 WLAN

2.3 The IEEE 802.1X Authentication Architecture

The IEEE 802.1X provides a port-based network access control mechanism for user authentication, unlike the IEEE 802.11 authentication algorithms are to device authentication. 802.1X is a framework and based on extensible authentication protocol (EAP) [8]. 802.1X architecture has defined three components, as shown in Figure 4.

The PAEs (Port Authentication Entities) includes the supplicant and the authenticator. The supplicant is the user machine requesting for network resource access. The authenticator plays the role of gatekeeper by controlling network access and acts as a bridge like a switch or an AP. It doesn't maintain user information and just passes authentication messages to and from someone. There needs an authentication server, such as the RADIUS server, implementing various authentication mechanisms. The authentication transaction messages exchange between the supplicant and the authentication server. EAPoL (EAP over LAN) is used between the PAEs, while RADIUS (Remote Authentication Dial-In User Service) [9] is used between the authenticator and the RADIUS sever.

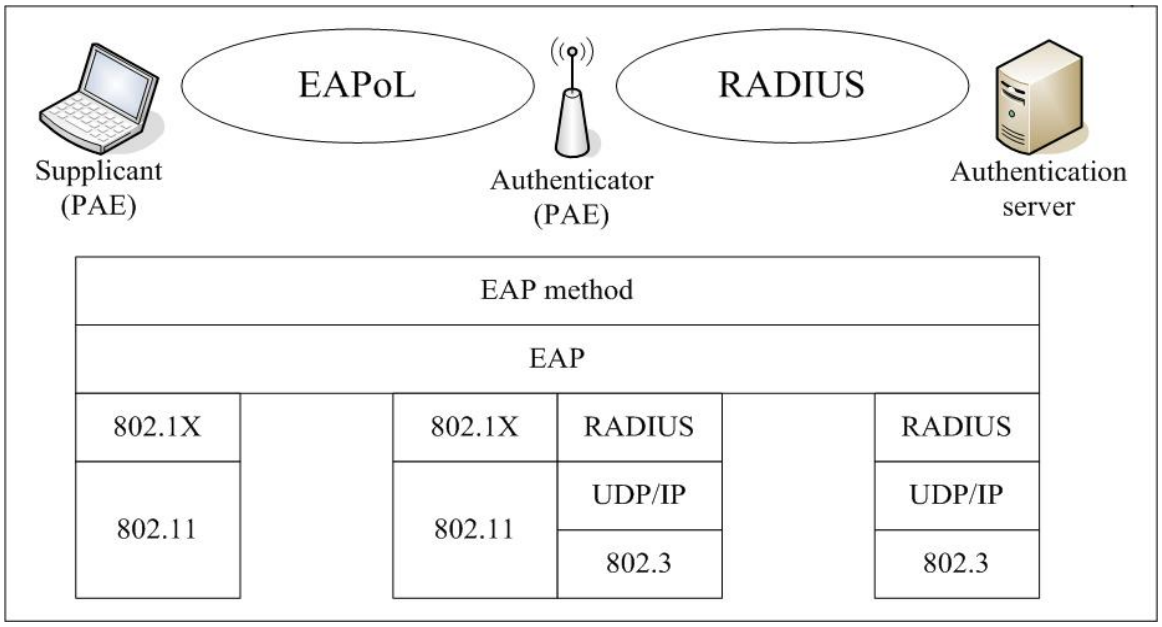


Figure 4: The IEEE 802.1X Architecture

2.4 Introduction of RADIUS

RADIUS is an AAA (Authentication, Authorization, and Accounting) based access control protocol. It provides a client / server protocol enabling remote access servers to communicate with a central authentication server. It uses UDP instead of TCP as transport protocol with port number 1812. Next, we describe some of the RADIUS attributes we used in 802.1X authentication for our mechanism.

User-Name: in IEEE 802.1X, the supplicant typically provides its identity by an EAP-Response / Identity frame. **NAS-IP-Address**: the IP address of the bridge or AP acting as an authenticator. **NAS-Port**: because an AP does not have physical ports, it assigns a unique “association ID” to each mobile station whenever a successful association completes. **Called-Station-Id**: the MAC address of the bridge or AP. **Calling-Station-Id**: the MAC address of the wireless interface of the supplicant. **NAS-Identifier**: a string identifier for the authenticator to originate the Access- Request packet. **NAS-Port-Type**: values for indicating types, such as Ethernet, wireless-IEEE 802.11, Token Ring, and FDDI. **EAP-Message**: used to encapsulate EAP packets between the authenticator and the authentication server.

2.5 The EAP Authentication Methods

As shown in Figure 5, EAP can be used to run on any kind of link layer and has many authentication methods above it. The following are some current EAP authentication methods, which have a set of rules to authenticate a user.

MD5 Challenge: The MD5 Challenge is analogical to the CHAP protocol. It requires that the challenge should be successfully encoded with a shared secret. All EAP implementations must support the MD5 Challenge.

Generic Token Card (GTC): Token cards offer the security of random one-time password. EAP-GTC can be used for “username+password” authentication.

EAP-TLS: Transport Layer Security (TLS) can be used to establish a trusted communication tunnel over a distrusted network subject to eavesdropping. It provides mutual authentication through certificate exchange. However, it needs to have a well-built public key infrastructure to generate and distribute certificates.

EAP-TTLS and EAP-PEAP: Both of them work similarly. In first step, they establish a TLS tunnel similar to EAP-TLS. In second step, this TLS tunnel is used to encrypt an older authentication protocol that authenticates users to the network. Certificates are required only for outer authentication. The difference between TTLS and PEAP is how they handle the inner authentication. TTLS uses the encrypted channel to exchange AVPs (attribute-value pairs) while PEAP uses the encrypted channel to start a second EAP authentication method.

EAP-MSCHAPv2: Microsoft CHAP version 2 (MSCHAPv2) was initially introduced with Windows 2000. It was designed to address the shortcomings of MSCHAP and provided mutual authentication. It can be used as inner authentication method of TTLS or PEAP.

EAP-SIM and EAP-AKA: Both of them can be used for authentication against mobile telephone databases for providing integrated billing with mobile telephone accounts. EAP-SIM provides an interface to the Subscriber Identity Module (SIM) database on GSM telephone networks. EAP-AKA is based on the authentication system in third-generation mobile telephone networks, called Authentication and Key Agreement (AKA).

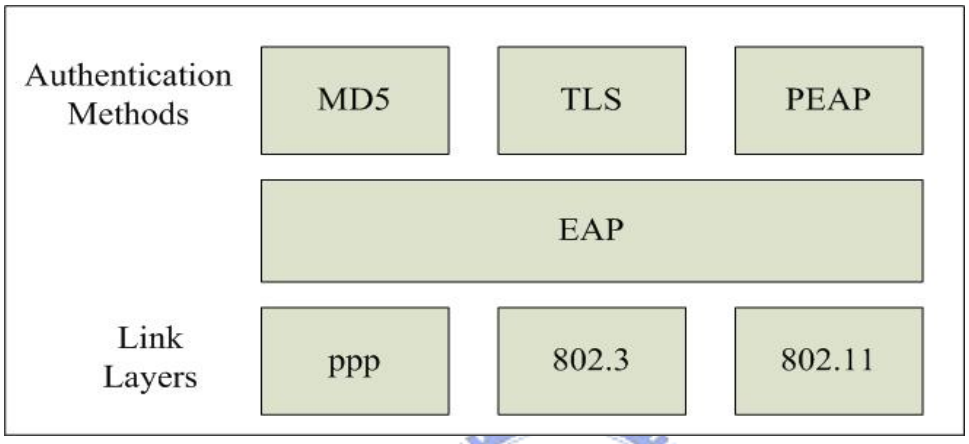


Figure 5: The EAP Architecture

2.6 Typical 802.1X Authentication Message Exchange

There is an example of typical 802.1X authentication message exchange on 802.11 WLAN, as shown in Figure 6. Once the 802.11 association is complete, the supplicant can issue the EAPoL-Start frame to trigger the EAP transaction. The authenticator, i.e., the AP issues the EAP-Request / Identity frame and the supplicant replies with the EAP-Response / Identity frame. That frame is passed by the AP to the RADIUS server as a Radius-Access-Request packet. According to the type of authentication required, the RADIUS server determines to send an EAP request for that method type and encapsulates it in a Radius-Access-Challenge packet to the AP. When it reaches the AP, the EAP request is extracted and passed to the supplicant. Whenever the AP receives a frame from the supplicant, it passes that frame to the RADIUS server as a Radius-Access-Request packet. Whenever the AP receives a Radius-Access-Challenge packet from the RADIUS server, it extracts and passes the EAP request frame to the supplicant. According various authentication methods,

there are different numbers of authentication message exchange.

If the RADIUS server grants access by the Radius-Access-Accept packet, then the AP issues the EAP-Success frame and authorizes the port. Otherwise, the RADIUS server sends the Radius-Access-Reject packet, then the AP issues the EAP-Failure frame and sets the port unauthorized. When the supplicant has finished accessing the network resources, it can send the EAP-Logoff frame to put the port back into the unauthorized state.

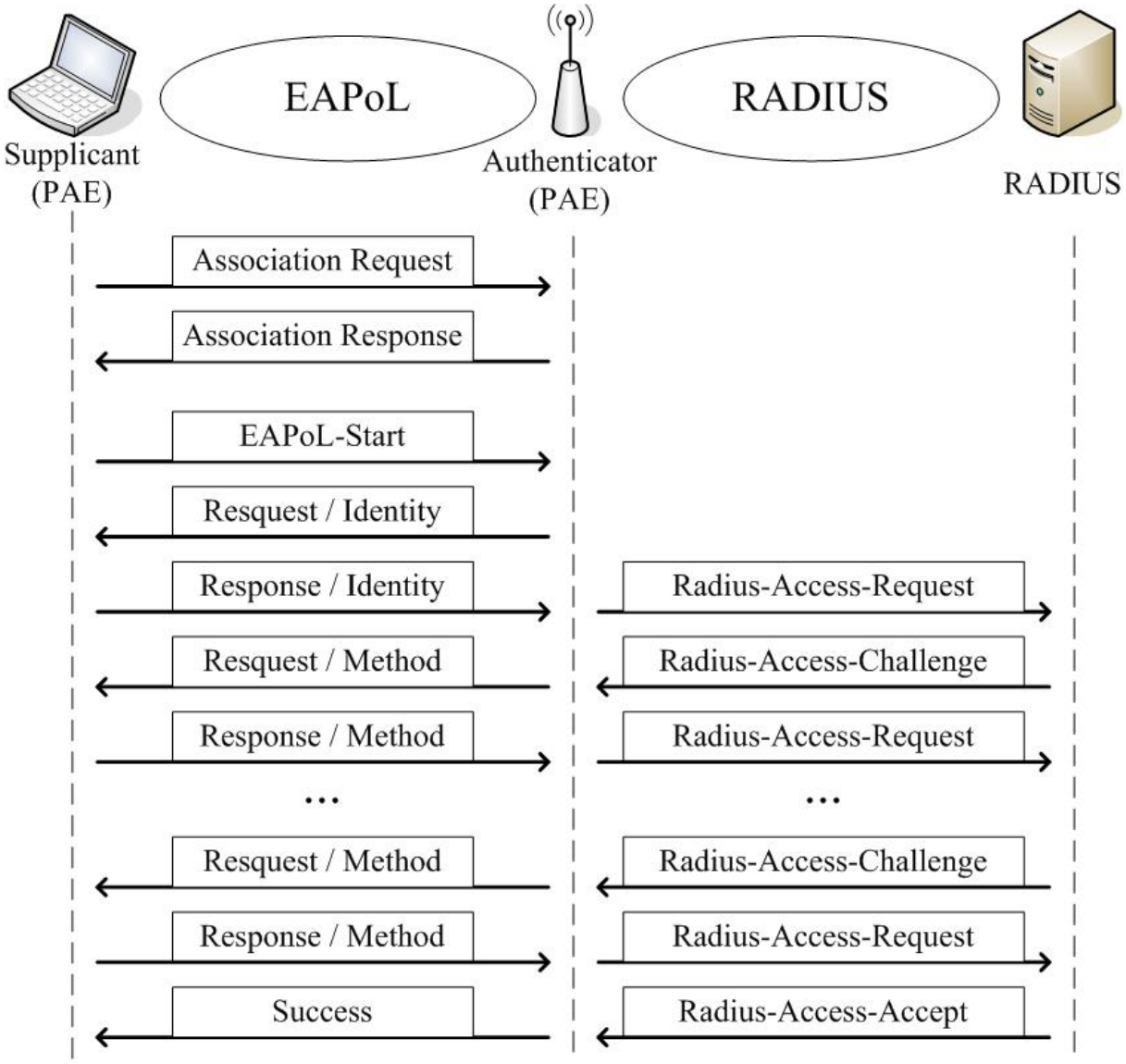


Figure 6: The typical 802.1X authentication message exchange on 802.11

We have measured the 802.1X authentication latency by selecting some current authentication methods, such as EAP-MD5 and EAP-PEAP. Figure 7 shows the typical

EAP-MD5 authentication message exchange on 802.11. It takes about 400 ms to complete the 802.1X authentication process by using EAP-MD5, which is the basic authentication method. Figure 8 shows the typical EAP-PEAP authentication message exchange on 802.11. It takes about 1000 ms to complete the 802.1X authentication process by using EAP-PEAP, which is the popular authentication method built in Windows XP OS.

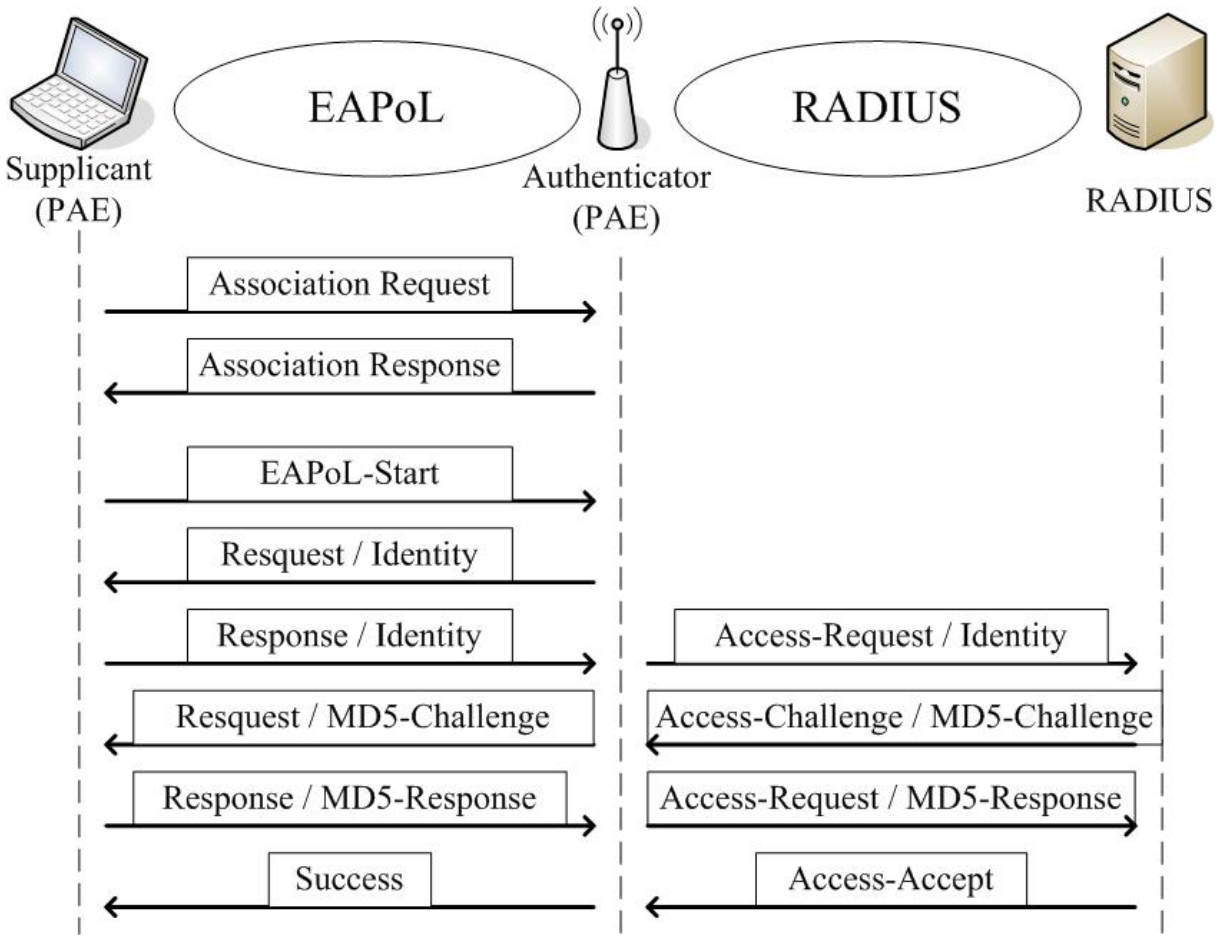


Figure 7: The typical EAP-MD5 authentication message exchange on 802.11

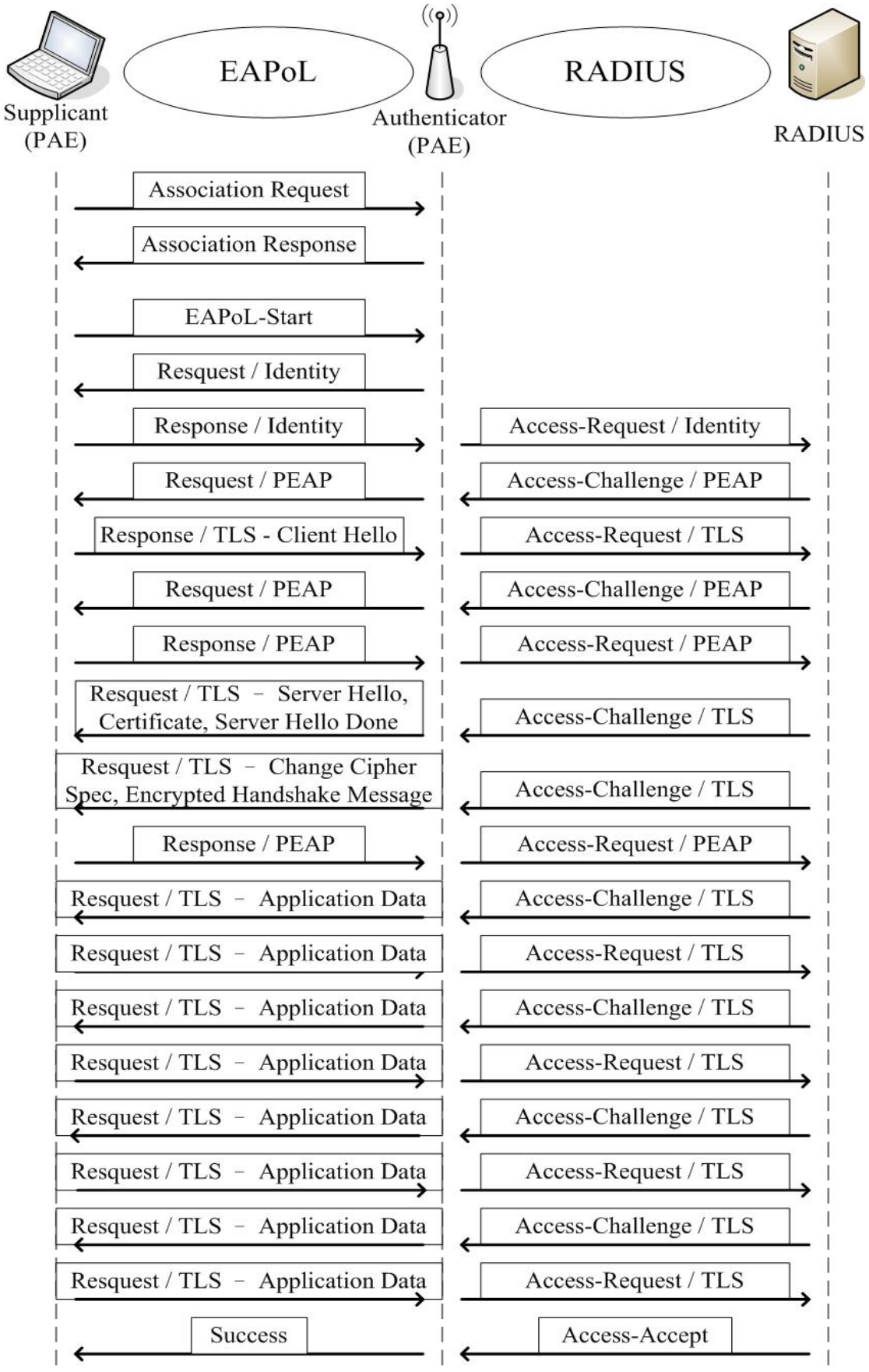


Figure 8: The typical EAP-PEAP authentication message exchange on 802.11

2.7 Introduction of Kerberos

Kerberos [11] is an authentication service developed as part of Project Athena at MIT. It addresses the problem that users wish to access services on servers distributed in an open distributed network environment. Servers should be able to restrict access to authorized users and to authenticate requests for services.

Kerberos has a three-layered authentication architecture as shown in Figure 9. Table 1 describes the Kerberos version 4 authentication message exchange. The AS of Kerberos is one kind of Key Distribution Center (KDC) which is a database storing, managing and centralizing secret keys of users and servers.

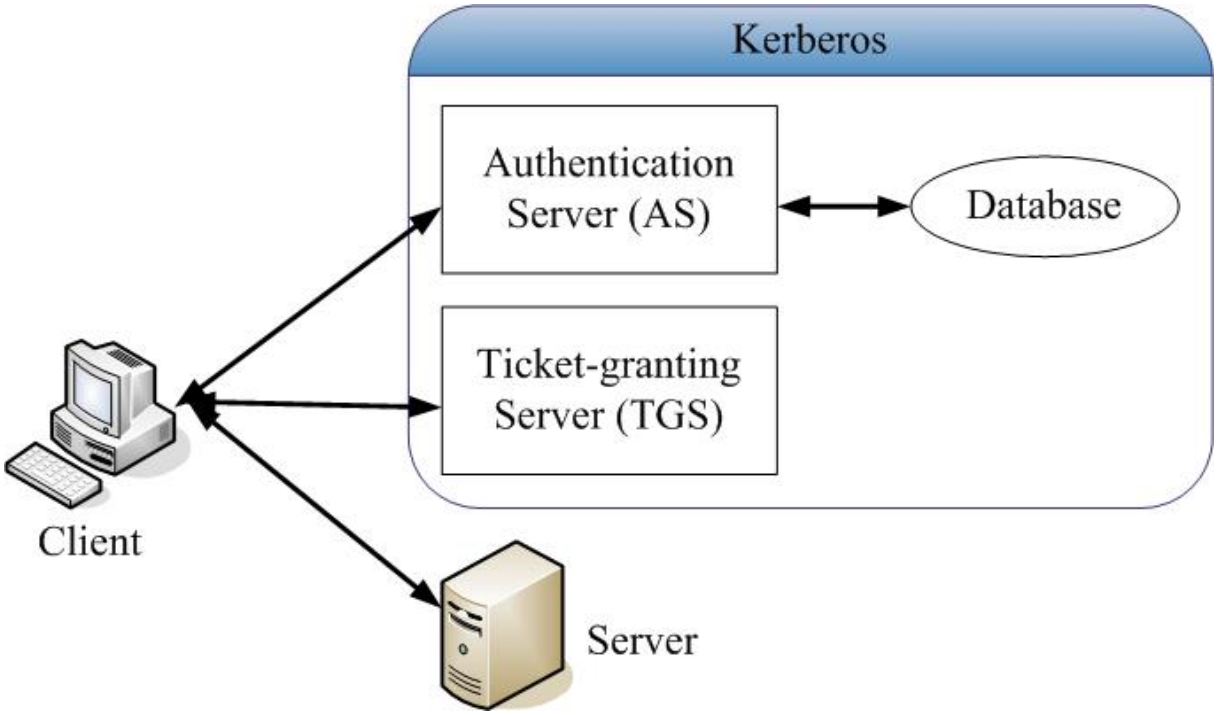


Figure 9: The Authentication Architecture of the Kerberos

Table 1: Summary of Kerberos Version 4 Authentication Message Exchanges

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) **Client → AS:** $ID_C \parallel ID_{TGS} \parallel TS_1$

(2) **AS → Client:** T_C

$T_C = Ek_C[K_{C,TGS} \parallel ID_{TGS} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{TGS}]$

$Ticket_{TGS} = Ek_{TGS}[K_{C,TGS} \parallel ID_C \parallel AD_C \parallel ID_{TGS} \parallel TS_2 \parallel Lifetime_2]$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) **Client → TGS:** $ID_V \parallel Ticket_{TGS} \parallel Authenticator_C$

(4) **TGS → Client:** $T_{C,TGS}$

$T_{C,TGS} = Ek_{C,TGS}[K_{C,V} \parallel ID_V \parallel TS_4 \parallel Ticket_V]$

$Ticket_{TGS} = Ek_{TGS}[K_{C,TGS} \parallel ID_C \parallel AD_C \parallel ID_{TGS} \parallel TS_2 \parallel Lifetime_2]$

$Ticket_V = Ek_V[K_{C,V} \parallel ID_C \parallel AD_C \parallel ID_V \parallel TS_4 \parallel Lifetime_4]$

$Authenticator_C = Ek_{C,TGS}[ID_C \parallel AD_C \parallel TS_3]$

(c) Client / Server Authentication Exchange: to obtain service

(5) **Client → Server_v:** $Ticket_V \parallel Authenticator_C$

(6) **Server_v → Client:** $Ek_{C,V}[TS_5 + 1]$ (for mutual authentication)

$Ticket_V = Ek_V[K_{C,V} \parallel ID_C \parallel AD_C \parallel ID_V \parallel TS_4 \parallel Lifetime_4]$

Kerberos is based on symmetric key encryption algorithm. It has a great advantage that is the signal sign-on idea. The ticket can viewed as an authorized access token to access servers. It can be reused many times until its lifetime is expired. The idea of reusing an authorized access taken is taken in our mechanism. It also helps to achieve the goal of fast handoff by reducing the number of authentication message exchange.

Chapter 3 Our Mechanism

First, we illustrate our system architecture which is integrated with the Kerberos authentication architecture. Then, we describe our mechanism in detail. Finally, we show our authentication message follow for fast handoff of VoWLAN under 802.1X authentication.

3.1 The System Architecture

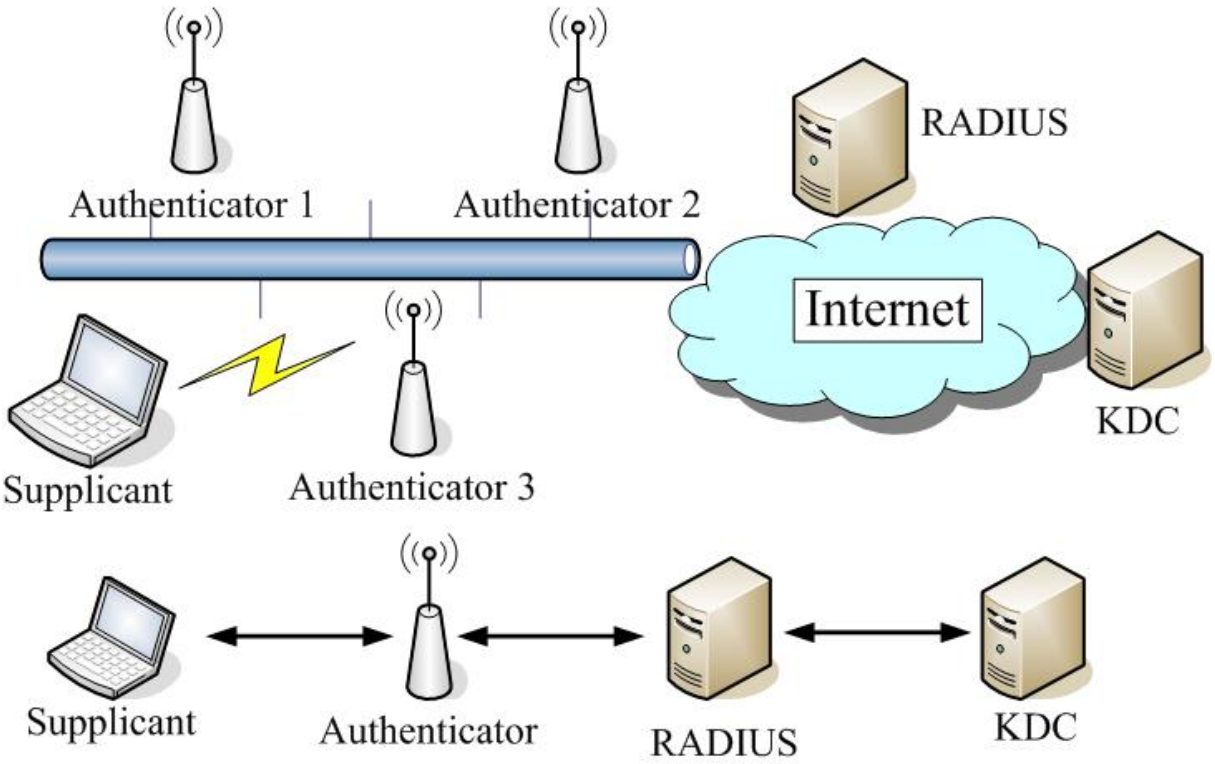


Figure 10: The System Architecture of the Fast Handoff Mechanism

Figure 10 depicts our system architecture for fast handoff on WLAN under 802.1X authentication. We assume that every AP (authenticator) in our environment is capable of supporting 802.1X function. Besides, we take the RADIUS server as our authentication server for authenticating supplicants seeking access wireless networks. Additionally, we need a trusted third-party, the Key Distribution Center (KDC), as described in section 2.7. We also assume that all secret keys of clients and servers are well configured in advance.

On the network-side, there are two servers. One is the RADIUS server which grants

access permission to the authenticators; the other is the KDC which knows secret keys of the RADIUS server and all users. The RADIUS server can act as the KDC proxy of the supplicant. The KDC server issues the ticket which can be viewed as the authorized access token for authentication as described in section 2.7. The secret key of one client is shared only between the client and the KDC server. The secret key of one RADIUS server is shared only between the RADIUS server and the KDC server. Once the client gets the ticket, it can reuse that ticket for 802.1X authentication until its lifetime is expired to archive our goal of fast handoff.

Kerberos has a three layered parallel authentication architecture. In Kerberos, the client can directly communicates with the AS, then with the TGS server, and finally with the servers in order. We integrate Kerberos authentication architecture with 802.1X authentication architecture. Our system architecture is a three layered vertical authentication architecture. The client requests to access APs, then APs pass requests to RADIUS server, finally RADIUS server pass them to the KDC server. The client indirectly communicates with the KDC server and the RADIUS server unlike Kerberos.

In our system architecture, the supplicant is viewed as the client of Kerberos, who requests to access wireless network resources. We take the authenticators as the servers of Kerberos, because the authenticators (APs) provide wireless network access services. The RADIUS server functions like the TGS server of Kerberos because it does the job of 802.1X authentication, but it does not issue the tickets for access servers (APs). The supplicant requests only one kind of service provide by APs. The supplicant has no tickets for access them because not only it does not need to do that but also APs can not recognize the tickets.

3.2 Our Authentication Method

EAP is so extensible and flexible that we can design our own authentication method to meet our requirement. We have referred to the Kerberos authentication mechanism [11] as shown in figure 9 and table 1. Then we propose our authentication method, as shown in table 2, for fast handoff of VoWLAN under 802.1X authentication. All authentication messages between the client and the RADIUS server are passed by the authenticators.

We take the RADIUS server as the TGS of Kerberos, which authenticates the clients. The RADIUS server does not need to issue tickets for accessing APs. We take the authenticators as the servers of Kerberos, which provide access service of wireless networks. When the client has a ticket accessing the RADIUS server, it can reuse that ticket to access wireless network until that ticket is expired.

Table 2: Summary of Our Authentication Message Exchanges

(1) **Client → RADIUS:** ID_C

(2) **RADIUS → KDC:** $ID_C \,||\, ID_R \,||\, TS_1 \,||\, Lifetime_1$

(3) **KDC → RADIUS → Client:** T_C

$T_C = Ek_C[K_{C,R} \,||\, ID_R \,||\, TS_1 \,||\, Lifetime_1 \,||\, Ticket_R]$

$Ticket_R = Ek_R[K_{C,R} \,||\, ID_C \,||\, ID_R \,||\, TS_1 \,||\, Lifetime_1]$

(4) **Client → RADIUS:** $Ticket_R \,||\, Authentica\ tor_C$

$Ticket_R = Ek_R[K_{C,R} \,||\, ID_C \,||\, ID_R \,||\, TS_1 \,||\, Lifetime_1]$

$Authentica\,tor_C = Ek_{C,R}[ID_C \,||\, MacAddr_C \,||\, TS_2]$

(5) **RADIUS → AP:** grant access permission or rejection to the authenticator by *Access-Accept* or *Access-Reject* message

(6) **AP → Client:** authentication success or failure by *EAP-Success* or *EAP-Failure* message

Now, we start to describe our authentication message in detail. In (1), the client initially transmits its identity to the RADIUS server. It does not need to know the identity of the RADIUS server in advance unlike Kerberos.

In (2), the RADIUS server transmits client's identity appended with its identity and a timestamp as well as a lifetime to the KDC server. It functions like a KDC proxy to help the client to get the ticket. The information of timestamp and lifetime which determined by the RADIUS server is used for purpose of reusing ticket many times.

In (3), whenever the KDC server receives request messages, it retrieves identities of client and RADIUS server and finds corresponding secret keys of them. Then, it uses the secret key of RADIUS server to issue the ticket $Ticket_R$ for RADIUS server and the secret key of client to calculate the T_C for client. The ticket $Ticket_R$ and the T_C are encrypted by corresponding secret keys. The KDC server can dynamically provide a unique session key $K_{C,R}$ for one pair of client and RADIUS server. The pair of client and RADIUS must use their corresponding secret keys to decrypt the information contained in the corresponding tickets. The T_C contains the ticket $Ticket_R$. Both of them contain the same session key, identity of RADIUS server, timestamp and lifetime. The timestamp and lifetime are used to indicate the lifetime of this ticket from that timestamp. After producing the tickets, the KDC server transmits the T_C to the RADIUS server. Then, the RADIUS server just passes that to the client. Figure 11 shows the authentication procedure of KDC server for our mechanism.

In (4), after receiving the T_C, the client must use its secret key to decrypt the information encrypted in the T_C. After decrypting the T_C, it gets the session key $K_{C,R}$, identity of RADIUS server, timestamp and lifetime of the ticket $Ticket_R$. The client can not know the content of the ticket $Ticket_R$ because of not knowing the secret key of RADIUS server. Although the client get the ticket $Ticket_R$, it can not modify the content of that ticket.

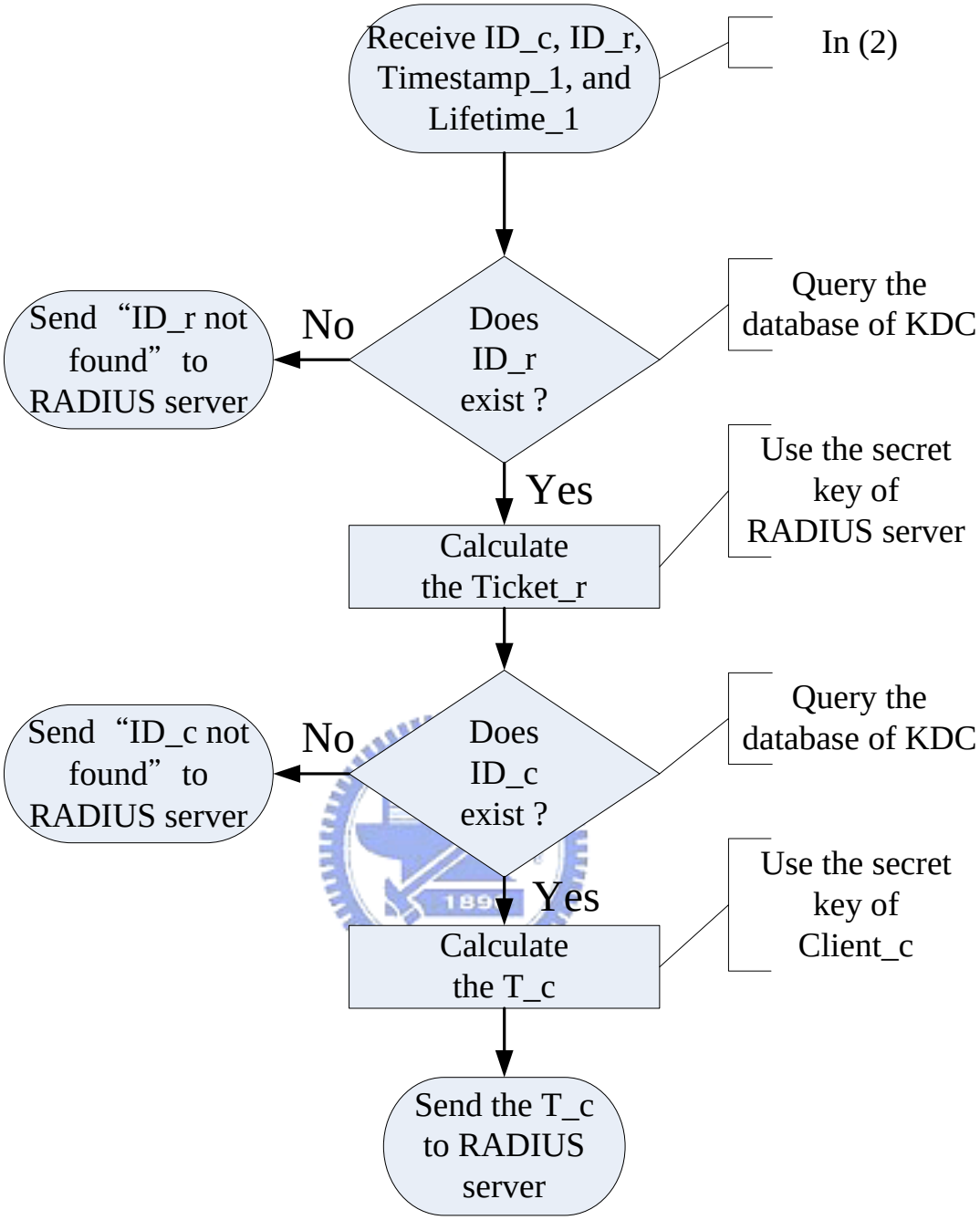


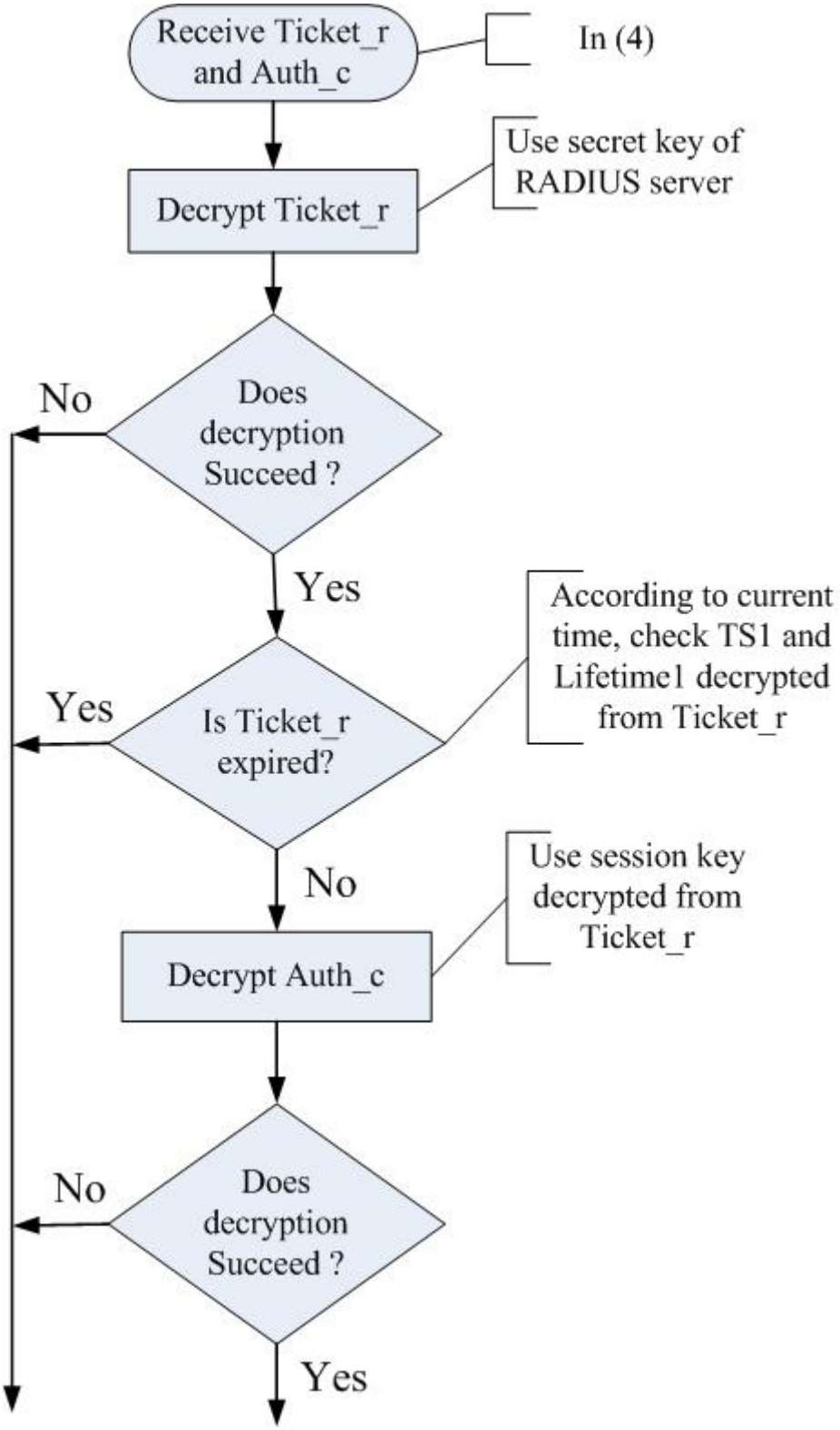
Figure 11: The Authentication Procedure of KDC server

The client calculates the $Authenticator_C$ by using the session key $K_{C,R}$ for encryption. The content of the $Authenticator_C$ includes the identity of client, the MAC address of wireless interface of client, and the timestamp which is the current timing information. The $Authenticator_C$ is used to prove who the person he declares to be. The timestamp in the $Authenticator_C$ can be used to prevent the reply attack. Because some bad guy may capture

the $Authenticator_C$ and the ticket $Ticket_R$, then uses and replays them for 802.1X authentication. However, that bad guy can not get the session key shared between the client and RADIUS server, so he can not modify the content of the $Authenticator_C$. After possessing the $Authenticator_C$ and the ticket $Ticket_R$, the client transmits them to the RADIUS server for 802.1X authentication. Whenever the client has a handoff, it can just pass and reuse them for 802.1X authentication until the lifetime of ticket $Ticket_R$ is expired.

In (5), whenever the RADIUS server receives the $Authenticator_C$ and the ticket $Ticket_R$ from supplicant, it can use them for authentication. It first uses its secret key to decrypt the ticket $Ticket_R$ then gets the session key, identity of client, identity of RADIUS server, timestamp and lifetime of this ticket $Ticket_R$. Then, it use this session key to decrypt the $Authenticator_C$ to get the content including the identity of client, the MAC address of wireless interface of client, and the timestamp. It must find out if there occurs a replay attack by using the timestamp. Because when time goes by, the timestamp should be older and larger. A replay attacker can not modify the content of the $Authenticator_C$ because of not knowing the session key shared between the client and RADIUS server. Figure 12 shows the authentication procedure of RADIUS server for our mechanism. If authentication is successful, then RADIUS server sends the Access-Accept packet to the authenticator; otherwise, RADIUS server sends the Access-Reject packet to the authenticator

In (6), whenever the authenticator receives the Access-Accept packet from RADIUS server, then it sends the EAP-Success frame to the supplicant and authorizes the port the supplicant connected. Whenever the authenticator receives the Access-Reject packet from RADIUS server, then it sends the EAP-Failure frame to the supplicant and sets the port the supplicant connected unauthorized.


Receive Ticket_r and Auth_c
In (4)
Use secret key of RADIUS server
Decrypt Ticket_r
Does decryption Succeed ?
No
Yes
According to current time, check TS1 and Lifetime1 decrypted from Ticket_r
Is Ticket_r expired?
Yes
No
Use session key decrypted from Ticket_r
Decrypt Auth_c
Does decryption Succeed ?
No
Yes

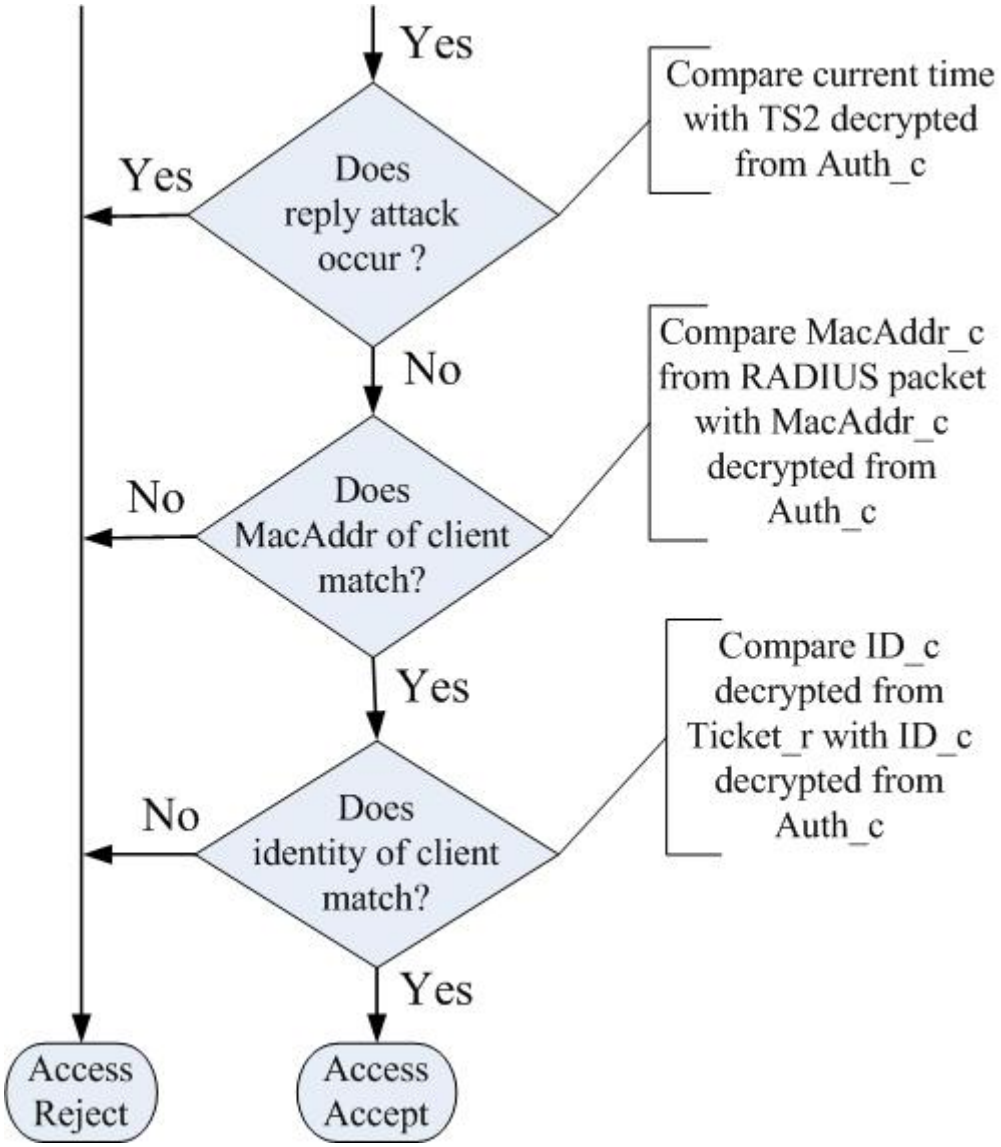


Figure 12: The Authentication Procedure of RADIUS server

3.3 Our Authentication Message Follow

Figure 13 shows our authentication message follow when the supplicant is first associated with authenticator 1.

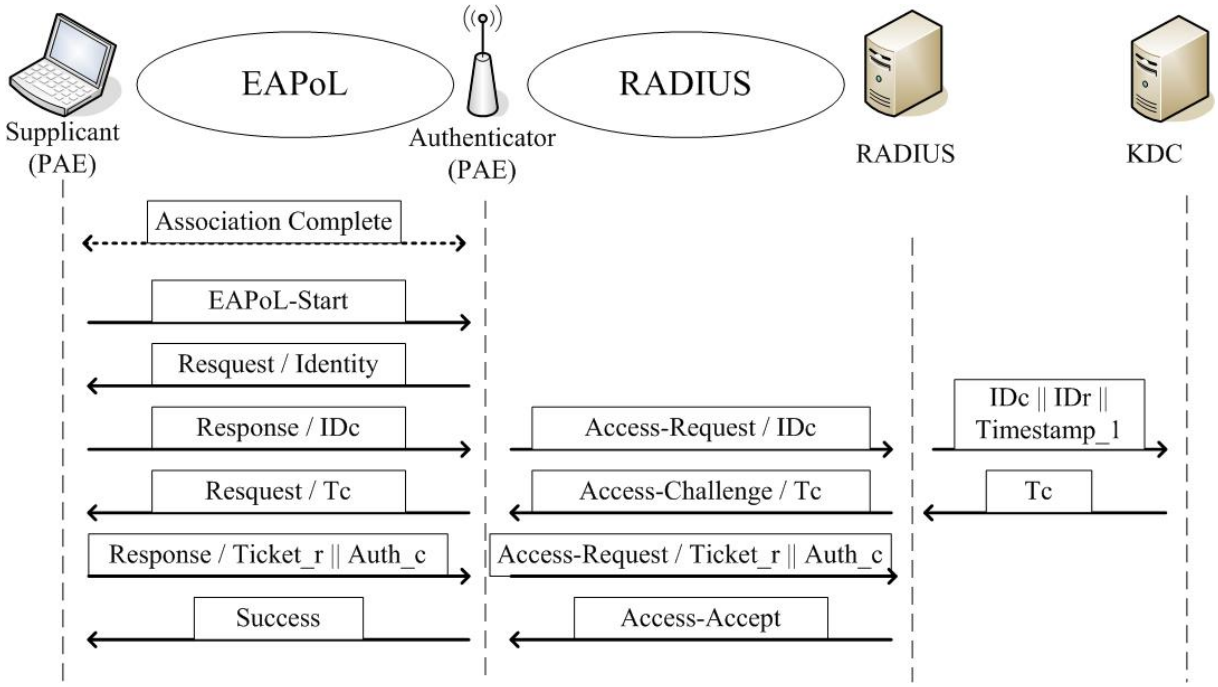


Figure 13: Our Authentication Message follow

Figure 14 shows our authentication message follow for fast handoff. Whenever the client has a handoff, it can just pass and reuse the $Authenticator_C$ and the ticket $Ticket_R$ for fast handoff until that ticket is expired.

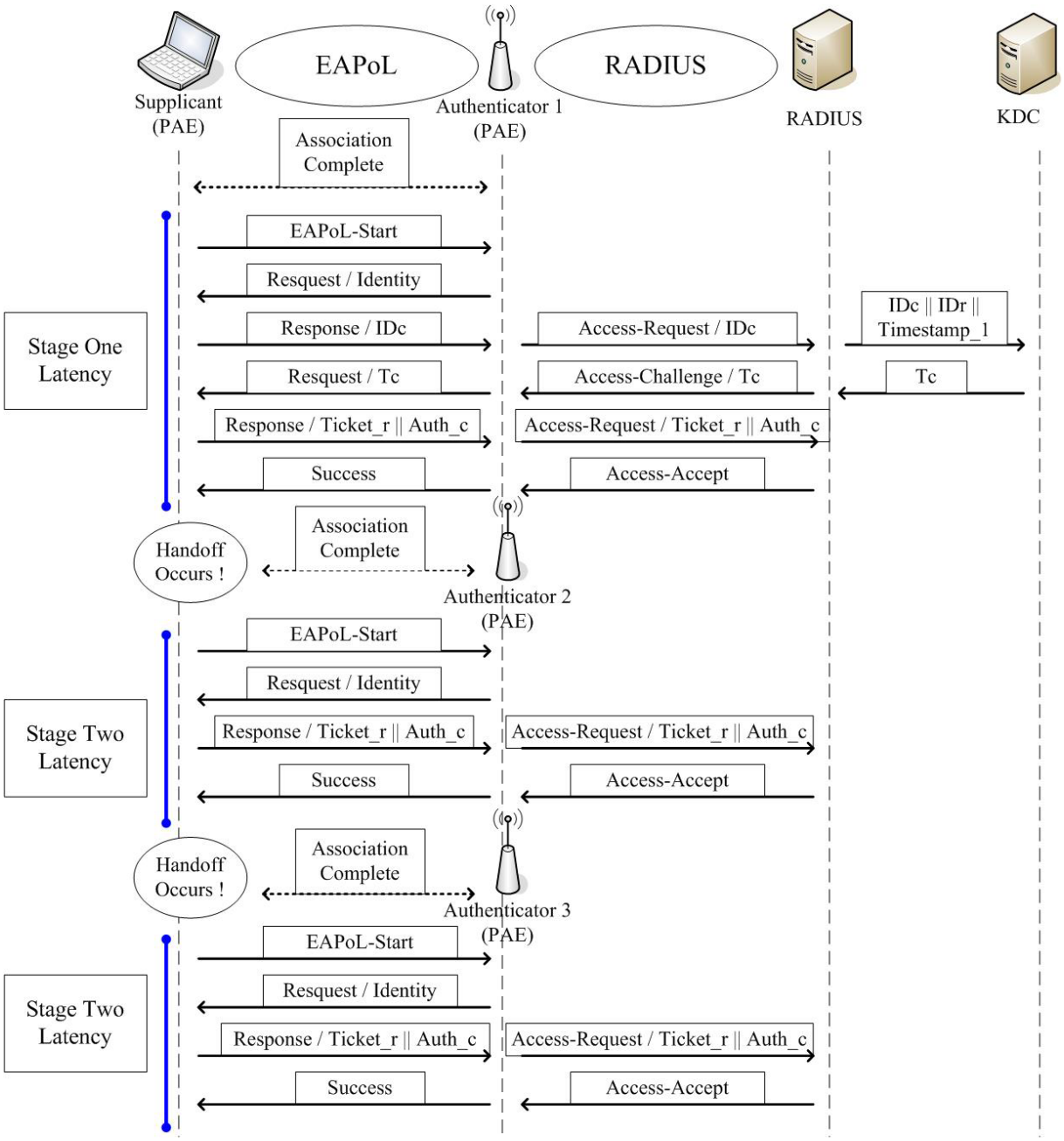


Figure 14: Our Authentication Message follow for fast handoff

Chapter 4 Implementation and Measurements

We used three laptop PCs and two desktop PCs. One of three laptop PCs is the user machine for 802.1X supplicant; two of three laptop PCs are used for simulating authenticators. One of two desktop PCs is the 802.1X authentication server; the other is used for measurements. We chose the Linux OS of Fedora Core 5 as our development platform. We used the Intel® PRO / Wireless 2200BG as our client's wireless interface card and its device driver for Linux [12].

4.1 Our Implementation

We have implemented our mechanism by using open source code projects on the Linux platform. We have used open source code projects such as the Xsupplicant [13], the Hostapd [14], and the FreeRADIUS [15].

4.1.1 Implementation of the supplicant program

We implemented and modified the 802.1X authentication program, the Xsupplicant, for client on Linux platform. The Xsupplicant is the supplicant program designed to work with Linux for 802.1X authentication.

In our implementation, when the Xsupplicant gets the T_C, it uses the secret key of client to decrypt the T_C. After decrypting the T_C successfully, it gets the ticket $Ticket_R$. When the client decides to handoff, it calculates the $Authenticator_C$ as soon as possible. Once it possesses the $Authenticator_C$ and the ticket $Ticket_R$, then it sends them to the RADIUS server for authentication. When it receives the EAP-request / Identity packet from the authenticator, it just replies with the $Authenticator_C$ and the ticket $Ticket_R$ in the content of EAP-response / Identity packet. Whenever it has a handoff, it can reuse them for fast handoff until the lifetime is expired.

4.1.2 Simulation of the authenticator program

We used the 802.1X authenticator program, the Hostapd, which is a user space daemon for AP and supports Linux. It implements IEEE 802.11 management function, IEEE 802.1X Authenticators, and RADIUS client. We used laptop PCs with wireless interface cards using the Prism chipset.

In our implementation, we modified the Hostapd source code such that we can dump 802.1X related information for testing and debugging.

4.1.3 Implementation of the KDC server program

We implemented the Key Distribution Center (KDC), which is a database managing and centralizing secret keys of users and servers as described in section 2.7. It can dynamically provide a unique session key for one pair of client and RADIUS server. It also has the responsibility of issuing tickets to clients and servers for authentication.

4.1.4 Implementation of the RADIUS server program

We implemented and modified the 802.1X authentication program, the FreeRADIUS, for 802.1X authentication server on Linux platform. It is the AAA server and supports request proxy, and access many types of back-end databases.

In our implementation, when the RADIUS server initially gets the identity of client, it helps the client to get the T_C and sends it back to the client. When it gets the $Authenticator_C$ and the ticket $Ticket_R$ from the client, it calculates them and decides to grant permission or not. After it makes a decision, it should tell the authenticator whether the authentication is successful. According to the decision of the RADIUS server, the authenticator should tell the supplicant the authentication results and setup the port the supplicant connected.

4.2 Our Measurements

To evaluate our implementation, we have conducted some experiments. We used the Ethereal [16] software to capture our 802.1X authentication packets and evaluate the authentication latency.

Figure 15 depicts our development environment for evaluating handoff performance. The client runs the Xsupplicant program for 802.1X authentication. We choose the FreeRADIUS as our authentication server program. To measure the authentication latency of our authentication method, we use a PC with one wireless interface capable of capturing 802.1X authentication message packets by means of the software Ethereal. We measure the authentication latency by calculating the duration from the EAP-request / identity frame to the EAP-success frame. Both of authenticator 1 and authenticator 2 are the machines running the Hostapd daemon programs.

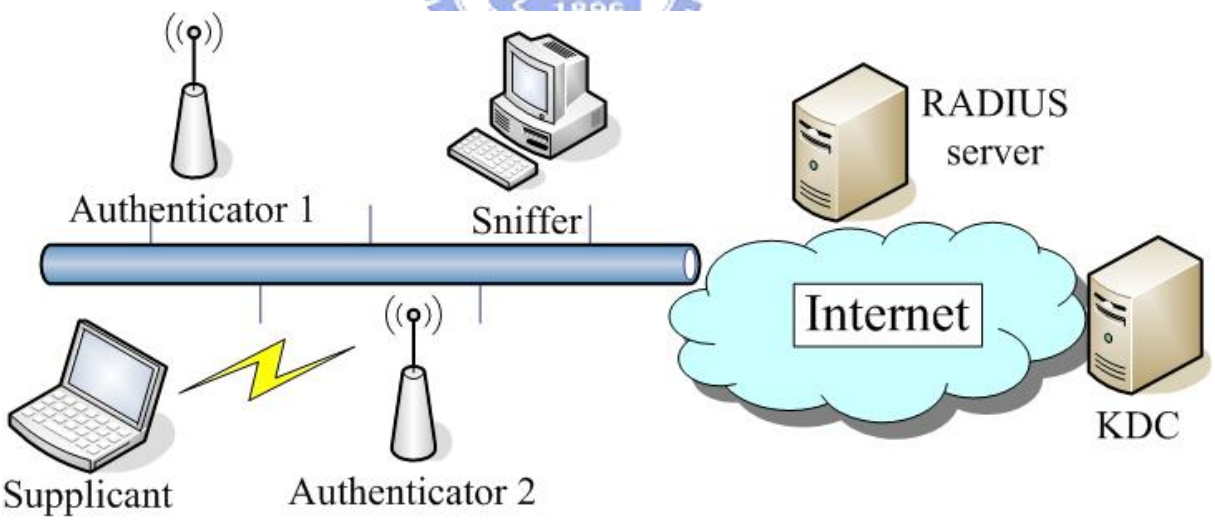


Figure 15: Our Development Environment

When the client is initially associated with authenticator 1, then it starts 802.1X authentication of our mechanism as shown in upper section of Figure 14. We measure and call this duration of 802.1X authentication the Stage One Latency of our mechanism. We have

estimated that the Stage One Latency is about 460 ms as shown in Table 3.

Afterwards, the supplicant of client has got authorized tokens which can be reused for fast handoff. If the client has a handoff from authenticator 1 to authenticator 2 or from authenticator 2 to authenticator 1, then it starts 802.1X authentication of our mechanism as shown in middle and lower section of Figure 14. We measure and call this duration of 802.1X authentication the Stage Two Latency of our mechanism. We have estimated that the Stage Two Latency is about 220 ms as shown in Table 4

Table 3: Stage One Latency of our mechanism (ms)

Experiment	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**average**
Stage One Latency	482	420	433	478	488	483	498	432	431	453	**460**

Table 4: Stage Two Latency of our mechanism (ms)

Experiment	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**average**
Stage Two Latency	214	222	208	235	209	207	246	211	216	228	**220**

Chapter 5 Conclusions

In this thesis, we propose a new EAP authentication method for fast handoff of VoWLAN under 802.1X authentication. We observe that the more authentication messages exchange, the longer it takes to complete the 802.1X authentication. We have referenced the ideas of the signal sign-on and the ticket from Kerberos. Our mechanism has minimized the number of round-trip authentication messages by means of reusing the authorized token until its lifetime is expired. Because we have minimized the number of authentication messages exchange, we reduce the 802.1X authentication latency efficiently. We also reduce the number of packets dropped during the 802.1X authentication procedure. We divide our authentication latency into two stages. Despite the performance of the Stage One Latency is not so good, the performance of the Stage Two Latency has improved a lot. In conclusion, we suggest that there need an EAP authentication method for real-time applications such as VoIP over WLAN to meet the low latency requirement.

In summary, our fast handoff mechanism needs to modify the programs of client and authentication server. Additionally, it needs a Key Distribution Center (KDC) like the authentication server of Kerberos system. The hardware and software of authenticators (APs) need not to be modified or changed. Our mechanism has the advantage that it can be integrated with current Kerberos authentication architecture.

There are still many challenges and problems in the wireless environment, such as data encryption and quality of service (QoS). To achieve a compatible solution for VoWLAN, we must take the above issues into consideration and integrate the fast handoff mechanism with them so that we can achieve a high security and good quality mobile telephony world in the future.

Kerberos has provided mutual authentication mechanism and our mechanism also

inherits its property. Our mechanism also provides mutual authentication for clients and servers, but we can not solve the problem of Rogue AP. IEEE 802.11i [17] has defined a pre-authentication mechanism for 802.1X authentication. Our mechanism can also be incorporated with IEEE 802.11i pre-authentication. Although we do not consider the encryption key distribution, there are already many researches on that issue and we can adopt them into our mechanism in the future.

Reference

[1] Arunesh Mishra, Minho Shin, and William Arbaugh, “An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process,” January 2003.

[2] Héctor Velayos and Gunnar Karlsson, “Techniques to Reduce IEEE 802.11b MAC Layer Handover Time,” April, 2003.

[3] Hye-Soo Kim, Sang-Hee Park, Chun-Su Park, Jae-Won Kim, and Sung-Jea Ko, “Selective Channel Scanning for Fast Handoff in Wireless LAN using Neighbor Graph,” International Technical Conference on Circuits/System, Computer and Communications, July 2004.

[4] Sangho Shin, Anshuman Singh Rawat, and Henning Schulzrinne, “Reducing MAC Layer Handoff Latency in IEEE 802.11 Wireless LANs”.

[5] Ishwar Ramani and Stefan Savage, “SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks”.

[6] LAN MAN Standards Committee of the IEEE Computer Society, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” ANSI/IEEE Std 802.11, 1999 Edition.

[7] LAN MAN Standards Committee of the IEEE Computer Society, “802.1X: Port-Based Network Access Control,” IEEE Std 802.1X-2004

[8] B. Aboba, L. Blunk, J. Vollbrecht, and J. Carlson, “Extensible Authentication Protocol (EAP),” IETF RFC 3748, June 2004.

[9] C. Rigney, S. Willens, A. Rubens, and W. Simpson, “Remote Authentication Dial In User Service (RADIUS),” IETF RFC 2865, June 2000.

[10] Mattbew S. Gast, “802.11 Wireless Networks: The Definitive Guide”, second edititon, book, Apirl 2005.

[11] William Stallings, “Cryptography and Network Security: Principles and Practice, third

edition,” book, 2003.

[12] Intel® PRO/Wireless 2200BG Driver for Linux, URL: http://ipw2200.sourceforge.net/.

[13] Open1x, Open Source Implementation of IEEE 802.1X, URL: http://open1x.sourceforge.net/.

[14] Hostapd, IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator, URL: http://hostap.epitest.fi/hostapd/.

[15] FreeRADIUS, the premiere open source RADIUS server, URL: http://www.freeradius.org/. .

[16] Ethereal, the network protocol analyzer, URL: http://www.ethereal.com/.

[17] LAN MAN Standards Committee of the IEEE Computer Society, “802.11i, Amendment 6: Medium Access Control (MAC) Security Enhancements,” IEEE Std 802.11i-2004