# 802.1x EAP-TLS 部署应用

## 中文摘要

为了保证网络资源的安全可控，网络接入控制已经成为当前主要的安全环节。其中，采用网络身份鉴别协议 EAP 同 PKI 技术相结合，成为了集通用、安全、高性价比于一身的做法。

目前业界的看法是，网络接入控制是保证网络安全的一个重要环节，而接入控制的关键是身份鉴别。在已有的各种身份鉴别方法中，PKI 是被公认为最安全有效的。而可扩展认证协议 EAP 同 PKI 的结合，能够实现安全的网络接入控制。

EAP-TLS 是以 PKI 公钥为基础的，PKI 公钥基础结构是目前比较成熟、完善的互联网络安全解决方案，可以说，EAP-TLS 认证方式是 FreeRADIUS 最安全的认证方式。但 PKI 实施起来比较繁琐，并且当前实现 PKI 的商用软件价格较为昂贵，完整的 PKI 应用一般都集中于商业领域，如银行，金融和军事领域，这都阻碍了 PKI 的广泛应用。

本文针对这一现状，对 PKI 的技术用于 FreeRADIUS 用户认证做一定的应用分析。重点介绍实现 PKI 技术的开源软件 EJBCA。通过使用 EJBCA 和 FreeRADIUS 配合使用，介绍了 FreeRADIUS 和 EJBCA 结合应用的配置和使用方法，组建的一套完整的 PKI 用户证书管理和认证系统。

本文的研究主要针对 802.1x 的接入认证，实践探索以 EAP-TLS 为核心的证书认证方式，以及与此相关 CA 用户证书管理，解决了 FreeRADIUS 和 EJBCA 联合工作所需的实时检查证书有效性的问题，为使用证书认证的 FreeRADIUS 提供了较为完整的开源软件应用参考。

最后，在前面的基础上，探索使用 USB Key 方式的证书存储和认证方式，并且用于 FreeRADIUS 认证方法。解释了 USB Key 和 FreeRADIUS 结合使用能够为用户认证起的飞跃性作用。

关键词：公钥体系，扩展认证协议-传输层安全，802.1x，ejbca，freeradius

作　　者：杨凌凤

指导老师：朱巧明

# Deployment of 802.1x EAP-TLS

## Abstract

In order to secure network resources, control access to network has become the main security stage at present. Extensible Authentication Protocol (EAP) works with integration of Public Key Infrastructure (PKI), which is a common, safe way, and contains high cost performance.

It is believed in this field that control of the access to network is important, and its key is to distinguish the identity. PKI is regarded as the most effective and safe way among the current methods of distinguishing the identity. The combination of EAP and PKI can enable an excellent control of the access to network.

EAP-TLS is based on PKI which is presently a mature and perfect solution to the security problem of network. It can be said that EAP-TLS is the most secure for user authentication. But it's not so convenient to put into practice, and the commercial software, which performs PKI, is expensive. Almost all complete applications of PKI are focused on bank, financial and military area, which hinder the progress of the extensive application.

Aiming at this situation, this paper makes a certain analysis on the application of PKI to the FreeRADIUS user authentication. It gives an important introduction of open source sofrware, EJBCA and of the configuration and usage of the combining application of FreeRADIUS and EJBCA. Through using EJBCA and FreeRADIUS, a complete system of the management of PKI certificate and user authentication method has been constructed.

This paper mainly aims at the authentication of the access of 802.1x, explores the method of certificate authentication which focuses on EAP-TLS, and related management of certificate of CA users. It also solves the problem of checking the validity of the users' certificate in real time, which is needed in the application of FreeRADIUS together with EJBCA, providing a comparatively complete application reference to the using of FreeRADIUS certificate authentication.

Finally, this paper studies the method of certificate store and authentication using

USB Key. It explains the substantial function of combining use of USB Key and FreeRADIUS EAP-TLS to user authentication.

**Keywords**: pki, eap-tls, 802.1x, ejbca, freeradius

**Written by: Yang Lingfeng**

**Supervised by: Zhu Qiaoming**

# 苏州大学学位论文独创性声明及使用授权的声明

## 学位论文独创性声明

本人郑重声明：所提交的学位论文是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含其他个人或集体已经发表或撰写过的研究成果，也不含为获得苏州大学或其它教育机构的学位证书而使用过的材料。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

研究生签名：[signature] 日 期：2007.11.25

## 学位论文使用授权声明

苏州大学、中国科学技术信息研究所、国家图书馆、清华大学论文合作部、中国社科院文献信息情报中心有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权苏州大学学位办办理。

研究生签名：[signature] 日 期：2007.11.25

导师签名：[signature] 日 期：2007.11.25

# 第一章 绪论

## 1.1 背景

随着网络技术的不断成熟以及网络应用的不断普及，现在各高校的网络除了要满足全校教职工的教学科研外，还要更多的面对学生群体。而随着网络用户数的急剧增加，网络的安全性问题日益突出。建立一个更加安全可行的、可运营、可管理的网络环境便摆在了面前。

由于传统认证方式存在着不少的问题，许多的认证系统架设在主干出口处，这样就不可避免导致许多计算机在没有预先经过同意，对网络设备及资源进行非正常使用，也就是我们通常所说的非授权访问。采用传统的网关身份认证的方式也很难对代理服务进行防范，在网络中如果私自搭建了代理服务器，那么很多非授权终端就可以通过同一个代理服务器取得对网络的访问权。而在服务器端看来，这些访问都来自正常的一个终端机器的访问。而且传统的认证方式对校园网中用户数据包繁琐的处理也造成了网络传输瓶颈，如果通过增加其他网络设备来解决传输瓶颈势必造成网络成本的提升，因此无法满足用户对网络安全性、高效性和低成本的要求。

目前 802.1x[1]认证已经得到了非常广泛的应用，其部署难度小，并且通过对认证方式和认证体系结构进行优化，有效地解决了传统认证方式带来的问题，消除了网络瓶颈，减轻了网络封装开销，降低了建网成本。由于 802.1x 客户端部署在客户终端上，它能在应用层识别代理行为，从而在终端客户机上阻止代理行为的发生。

但在一般的应用中，我们都会使用 EAP-MD5 密码认证，这个方法把用户密码储存在数据库中，认证的时候进行最基本的对比即可完成认证。这个方法最简单，同时也是最脆弱的，潜在多种被攻击风险。并且，由于密码都由用户自己保存，因此带来的一个帐号被多人使用，用户密码被窃取等后果，给管理工作带来很大的不便。因此，在实际应用中，需要寻找更加适合的认证方式，尽最大可能保护账户安全。

通过表[2]1-1 的比较可以看出，EAP-TLS 为最安全的 EAP 协议，但是在实用程度上欠缺一些，因为最安全所要花费的代价最高，需要建立 PKI 证书管理，每一个服务器端及客户端维护成本极高。目前比较让业界接受的是 EAP-TTLS 和 PEAP，两者都

有支持双向认证，同时也只需要具备 Radius Server 的证书即可实现认证信息的加密传送验证。

| | EAP-MD5 | LEAP | EAP-TTLS | PEAP | EAP-TLS |
|---|---|---|---|---|---|
| 服务器认证 | 否 | Hash 密码 | 公钥(证书) | 公钥(证书) | 公钥(证书) |
| 客户端认证 | Hash 密码 | Hash 密码 | 质询握手身份验证协议，密码认证协议，微软质询握手身份验证协议 V2，EAP | 任何 EAP，比如微软质询握手身份验证协议 V2，公钥 | 公钥(证书或智能卡) |
| 认证属性 | 单向认证 | 双向认证 | 双向认证 | 双向认证 | 双向认证 |
| 动态密钥传输 | 否 | 是 | 是 | 是 | 是 |
| 部署难度 | 简单 | 中等 | 中等 | 中等 | 难 |
| 安全风险 | 身份暴露，字典攻击，中间人攻击，会话劫持 | 身份暴露，字典攻击 | 中间人攻击 | 中间人攻击，第一阶段潜在身份暴露 | 身份暴露 |

表 1-1 各种 EAP 认证方法比较

## 1.2 接入认证系统的现状

### 1.2.1 WEB/PORTAL 技术

WEB/PORTAL 认证是基于业务类型的认证，不需要安装其他客户端软件，只需要浏览器就能完成，就用户来说较为方便。但是由于 WEB 认证走的是 7 层协议，从逻辑上来说为了达到网络 2 层的连接而跑到 7 层做认证，这首先不符合网络逻辑。其次由于认证走的是 7 层协议，对设备必然提出更高要求，增加了建网成本。WEB 是在认证前就为用户分配了 IP 地址，对目前网络珍贵的 IP 地址来说造成了浪费，而且分配 IP 地址的 DHCP 对用户而言是完全裸露的，容易造成被恶意攻击，一旦受攻击瘫痪，整个网络就没法认证；为了解决易受攻击问题，就必须加装一个防火墙，这样一来又大大增加了建网成本。WEB/PORTAL 认证用户连接性差，不容易检测用户离线，基于时间的计费较难实现；用户在访问网络前，不管是 TELNET、FTP 还是其它业务，必须使用浏览器进行 WEB 认证，易用性不够好；而且认证前后业务流和数

据流无法区分。

### 1.2.2 PPPOE 技术

PPPOE 是从基于 ATM 的窄带网引入到宽带以太网的，由此可以看出，PPPOE 并不是为宽带以太网量身定做的认证技术，将其应用于宽带以太网，必然会有其局限性。虽然其方式较灵活，在窄带网中有较丰富的应用经验，但是，它的封装方式，也造成了宽带以太网的种种等问题。在 PPPOE 认证中，认证系统必须将每个包进行拆解才能判断和识别用户是否合法，一旦用户增多或者数据包增大，封装速度必然跟不上，造成了网络瓶颈；其次这样大量的拆包解包过程必须由一个功能强劲同时价格昂贵的设备来完成，这个设备就是我们传统的 BAS，每个用户发出的每个数据包 BAS 必须进行拆包识别和封装转发；为了解决瓶颈问题，厂商想出了提高 BAS 性能，或者采用大量分布式 BAS 等方式来解决问题，但是 BAS 的功能就决定了它是一个昂贵的设备，这样一来建设成本就会越来越高。

### 1.2.3 IEEE802.1x 技术

802.1 x协议起源于无线局域网，它也是一项可在以太网上实现认证计费功能的新技术，它不同于传统的PPPOE、WEB认证，认证流和业务流不分离。802.1x认证基于逻辑端口把认证流和业务流进行分离。认证系统的端口分成两个逻辑端口：受控端口和不受控端口。不受控端口只能传送认证的协议报文，而不管此时受控端口的状态是已认证状态(Autherized)还是未认证状态(Unautherized)。受控端口传送业务报文。如果用户通过认证，则受控端口的状态为已认证状态，可以传送业务报文。如果用户未通过认证，则受控端口的状态为未认证状态，不能传送业务报文。它的认证方式就是通过认证前后打开/关闭受控端口来实现对用户接入的控制，从而实现对用户的物理端口的认证和控制。

802.1x认证的突出优点就是实现简单、认证效率高、安全可靠。无需多业务网管设备，就能保证IP网络的无逢相连。同时消除了网络认证瓶颈和单点故障。解决了采用多业务网关，不便于视频业务开展的难题。在二层网络上实现用户认证，大大降低了整个网络的建网成本。

### 1.2.4 针对普通密码认证的改进技术

由于 EAP-MD5 存在身份密码泄露、中间人攻击等问题，需要使用更好的认证方

式来取代，很多人对基于 802.1x 的其他认证方式进行了实践研究。如《EAP-TTLS 认证方式在 WLAN 中的应用研究》[3]，通过实践探索，总结出了该认证方式用于WLAN的应用前景，并且结合认证，开发了EAP-TTLS认证客户端。尽管实现EAP-TLS部署成本较高，仍然有很多基于 PKI 和 Radius 结合研究，如《PKI 在校园网中的应用研究》[4]，文中使用 PKI 技术和 Radius 结合，提出了校园网络单点认证，给出了设计框架、完整的 PKI 构建、证书管理和 PKI 客户端设计的思路。在《TinyCA et Freeradius en mode EAP-TLS》[5]一文中，使用 TinyCA 和 FreeRADIUS 结合，实现了单机版的证书管理软件和 FreeRADIUS 结合应用的实例。

在《How Secure Is Your Wireless Network? Safeguarding Your Wi-Fi LAN》[6]一文中，详细阐述了无线网络环境中如何使用802.1x和其他安全技术如VPN等PKI技术加强企业网络安全。

在对比分析了这些研究之后，并没有找到一个适合网络应用且部署难度适中的方案，对于具体的实现，也没有作更多说明，这对实际应用没有带来多大的促进作用。因此，我们需要摸索一个适合一般网络部署的EAP-TLS认证方案。

## 1.3 本论文主要研究的内容

本文研究内容就是在前人的理论分析和实践基础上，使用开源软件构架一个成本较低，并且容易管理的802.1x认证环境，在认证技术上使用目前最为可靠的EAP-TLS方式，为开源软件规模应用于PKI的认证技术提供一个切实可行的参考。

1、学习PKI的原理，选择合适的CA软件，学习构建成本较低的CA，为CA和802.1x联合应用做一定经验积累。

2、了解802.1x认证的基本原理和过程，学习使用FreeRADIUS，实现FreeRADIUS EAP-TLS证书认证配置的相关环节。

3、进行简单开发，实现CA和FreeRADIUS协同工作，达到管理简单化的目的。

## 1.4 本文的组织结构

第一章，绪论，介绍了目前802.1x使用和研究的现状，对本文要实现的目标进行定位。

第二章，802.1x协议介绍，对802.1x协议体系进行理解和分析。

第三章，讲述PKI和EAP-TLS相关技术，把EAP-TLS认证涉及的技术框架进行说明，并且对实现EAP-TLS认证之后，引入更加可靠方便的USB Key技术进行了说明。

第四章，EAP-TLS应用设计，对我们要实现的构架进行说明，对实现过程中需要的软件进行选择和说明。

第五章，设计中的问题和解决方法，对设计框架中需要解决的问题进行分析，提出EJBCA 和 FreeRADIUS 协作的解决方案。

第六章，系统的具体实现，对设计的实现做较为完整的说明，以具体例子给实际应用做出参考。

第七章，总结和展望，总结我们的设计和实践工作，并对这个设计构架可能面临的问题做出预见，对解决方法作出展望，给今后的工作提出方向。

# 第二章 802.1x 协议介绍

## 2.1 协议的开发背景

在 IEEE 802 LAN 所定义的局域网环境中，只要存在物理的连接口，未经授权的网络设备就可以接入局域网，或者是未经授权的用户可以通过连接到局域网的设备进入网络。例如：一个可以访问公共网络的大厦的办公网，或者是某个组织机构与其他组织连接的网络。在这样的网络环境中，往往不希望未经授权的设备或用户连接到网络，使用网络提供的服务[7]。

后来，随着局域网技术的广泛应用，特别是在运营网络中的应用，对其安全认证的要求已经提到了议事日程上。如何既能够利用局域网技术简单、廉价的组网特点，同时又能够对用户或设备访问网络的合法性提供认证，是目前业界讨论的焦点。IEEE 802.1x 协议正是在这样的背景下提出的。

IEEE 802.1x 称为基于端口的访问控制协议(Port based network access control protocol)。基于端口的访问控制(Port based network access control)能够在利用 IEEE 802 LAN 的优势基础上提供一种对连接到局域网(LAN)设备或用户进行认证和授权的手段。通过这种方式的认证，能够在 LAN 这种多点访问环境中提供一种点对点的识别用户的方式。这里端口是指连接到 LAN 的一个单点结构，可以是被认证系统的 MAC 地址，也可以是服务器或网络设备连接 LAN 的物理端口，或者是在 IEEE 802.11 无线 LAN 环境中定义的工作站和访问点。

## 2.2 几个名词的定义

以下的名词为 802.1x 协议中的重要组成部分[7]：

(1) Supplicant 客户端

认证客户端指网络中所连接的一端的实体(entity)，它向认证系统(Authenticator 如下)发起请求，对其身份的合法性进行检验。认证客户端是需要接入 LAN 及享受交换机提供服务的设备，如 PC 机客户端需要支持 EAPOL 协议客户端必须运行 802.1x 客户端软件，如 802.1x-complain Microsoft Windows XP。

(2) Authenticator 认证系统

认证系统指在 LAN 连接的一端用于认证另一端设备的实体(entity)。边缘交换机或无线接入设备是根据客户的认证状态控制物理接入的设备，交换机在客户和认证服务器间充当代理角色，代理交换机与客户端之间通过 EAPOL 协议进行通讯，交换机与认证服务器间通过 EAP over Radius 或 EAP 承载在其他高层协议上，以便穿越复杂的网络到达认证服务。中继交换机要求客户端提供身份，接收到后将 EAP 报文承载在 Radius 格式的报文中再发送到认证服务器，然后根据认证结果返回给交换机控制端口是否可用。需要指出的是我们的 802.1x 协议在设备内终结并转换成标准的 RADIUS 协议报文加密算法采用 PPP 的 CHAP 认证算法。

(3) Authentication Server 认证服务器

认证服务器指为认证系统提供认证服务的实体。这里认证服务器所提供的服务是指通过检验客户端发送来的身份标识，来判断该请求者是否有权使用认证系统所提供的网络服务。认证服务器对客户进行实际认证，认证服务器核实客户的身份，通知交换机是否允许客户端访问 LAN 和交换机提供的服务，认证服务器接受认证者传递过来的认证需求，认证完成后将认证结果下发给认证者，完成对端口的管理。由于 EAP 协议较为灵活，除了 IEEE 802.1x 定义的端口状态外认，证服务器实际上也可以用于认证和下发更多用户相关的信息。如 VLAN QOS 加密认证密钥 DHCP 响应等。

(4) Port Access Entity (PAE) 端口访问实体

指一个端口的相关协议实体。端口访问实体PAE负责响应来自认证方和申请方之间的请求信息，这些信息用来建立信任关系，在认证交换过程中执行申请方的角色称作申请方PAE。负责于申请方通信，并把申请方的信任信息交给相对应的认证服务器去检查以决定随后的认证状态被称为认证方PAE。认证方PAE依据认证过程的结构控制受控端口的授权和非授权状态。PAE 能够支持的功能包括：客户端(Supplicant)完成的功能、认证系统(Authenticator)完成的功能或者两者功能同时具备。

(5) Network Access Port 网络访问端口

网络访问端口指用户系统连接到 LAN 的访问端口。访问端口可以是物理端口，例如连接到用户的网络设备端口；也可以是逻辑端口，例如用户设备的 MAC 地址。

(6) System 系统

系统是指通过一个或更多端口连接到 LAN 的设备，例如：终端、服务器、交换

机或路由器等设备都称为系统。

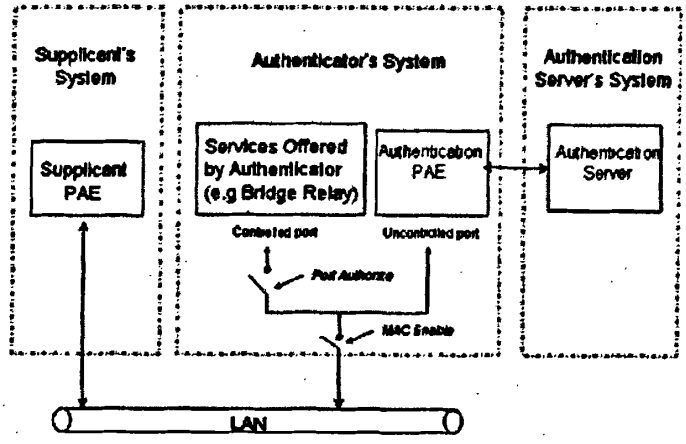


图 2-1 IEEE 802.1x 认证体系组成部分

## 2.3 工作机制

IEEE 802.1x认证过程的操作是利用EAP作为申请方和认证服务器之间认证信息交互的手段。在这样一个认证过程中有很多重要的机制，下面作一些具体的说明[7]。

### 2.3.1 各组成部分的功能

为了便于描述基于端口的访问控制过程，一个系统的端口(确切的讲应该是协议实体PAE)可以在整个控制过程中充当多个不同的角色。如下为在基于端口访问控制过程中，端口所充当的角色：

(1) Authenticator：在允许用户访问之前执行认证的端口，该端口充当认证系统的角色；

(2) Supplicant：访问认证系统所提供服务的端口，该端口充当客户端的角色；

(3) Authentication Server：认证服务器代表认证系统执行对用户身份的合法性进行检验功能。

从以上描述可以看出，为了完成一个认证过程，所有的三个角色是必须的。一个特定系统可以承担一种或多种角色；例如：一个认证系统和认证服务器能集成在一个系统中，实现认证功能而无需设置专门的外置认证服务器。同样，一个端口在一个认证过程中充当客户端的角色，而在另一个认证过程中可能充当认证系统的角色，例如：在一个桥接LAN中，一个新的交换设备添加到网络中，这个设备要能够对连接到其端

口的设备实现认证，自身必须先通过其上端设备的认证(该新设备通过上端设备接入到LAN中)。

### 2.3.2 受控和非受控的访问

IEEE 802.1x 协议的精华就是关于受控和非受控的访问，以下将详细描述关于受控和非受控的访问。

(1) 端口的类型

如图2-2所示，认证系统的端口分成两个逻辑端口：受控端口和不受控端口。不受控端口只能传送认证的协议报文，它始终处于双向连通的状态，不管是否处于授权状态都允许申请者和局域网中的其他机器进行数据交换，主要用来传递EAPOL协议帧，可保证随时接受客户端发出的认证EAPOL报文，而受控制端口只有在认证通过的状态下才打开，用于传递网络资源和服务。

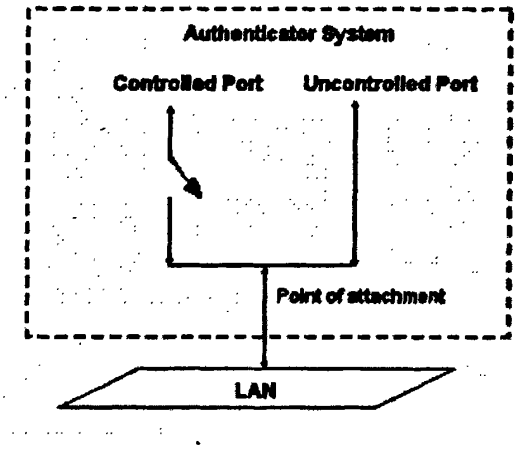


图 2-2 受控端口(Controlled Port)和不受控端口(Uncontrolled Port)

(2) 端口控制方式

对于端口的控制，可以有很多种方式。端口可以是物理的端口，也可以是用户设备的 MAC 地址，如果设备支持全程的 VLAN，也可以把 VLAN ID 看成是端口。

在 IEEE 802.1x 协议中的受控端口和不受控端口实际上是逻辑上的理解。端口的状态受相关的协议参数(AuthControlledPortStatus)控制，可设为强关状态(Unauthorized)，此时该端口处于断开状态；若端口状态设为 authorized，则此时该端口不需要要认证即可连通；若端口设为 Auto，则认证端的连接状态要取决于申请端，当申请通过认证时网络连通，否则断开。

如图 2-3 所示，当发起申请认证时，发起者通过非受控端口向认证服务器发送 EAPOL 的 802.1x 认证报文，当用户未通过认证时，受控端口处于开路，端口状态为未认证状态，此时交换机的交换功能是关闭的，也就是说交换机无法像传统的通过查找目标 MAC 地址来进行交换，如果用户有业务报文是无法通过的。

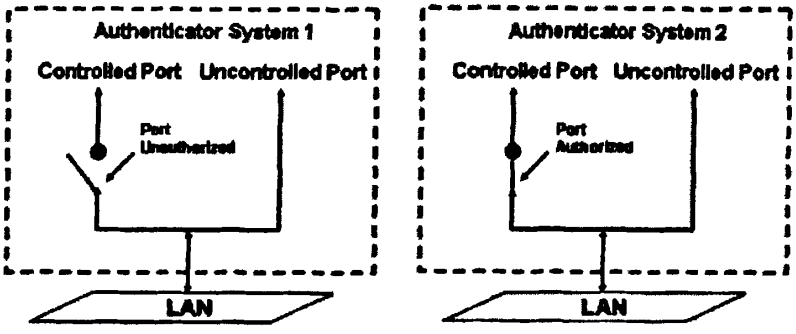


图 2-3 受控端口的状态变化

### 2.3.3 IEEE 802.1x 协议的体系结构

IEEE 802.1x 协议的体系结构(通过上文的说明可以知道)包括三个重要的部分：Supplicant System 客户端、Authenticator System 认证系统、Authentication Server System 认证服务器。图 2-4 描述了三者之间的关系以及互相之间的通信。

客户端系统一般为一个用户终端系统，该终端系统通常要安装一个客户端软件，用户通过启动这个客户端软件发起 802.1x 协议的认证过程。为支持基于端口的接入控制，客户端系统需支持 EAPOL(Extensible Authentication Protocol Over LAN)协议。

认证系统通常为支持 802.1x 协议的网络设备。该设备对应于不同用户的端口(可以是物理端口，也可以是用户设备的 MAC 地址)有两个逻辑端口：受控(controlled Port)端口和不受控端口(uncontrolled Port)。不受控端口始终处于双向连通状态，主要用来传递 EAPOL 协议帧，可保证客户端 始终可以发出或接受认证。受控端口只有在认证通过的状态下才打开，用于传递网络资源和服务。受控端口可配置为双向受控、仅输入受控两种方式，以适应不同的应用环境。如果用户未通过认证，则受控端口处于未认证状态，则用户无法访问认证系统提供的服务。

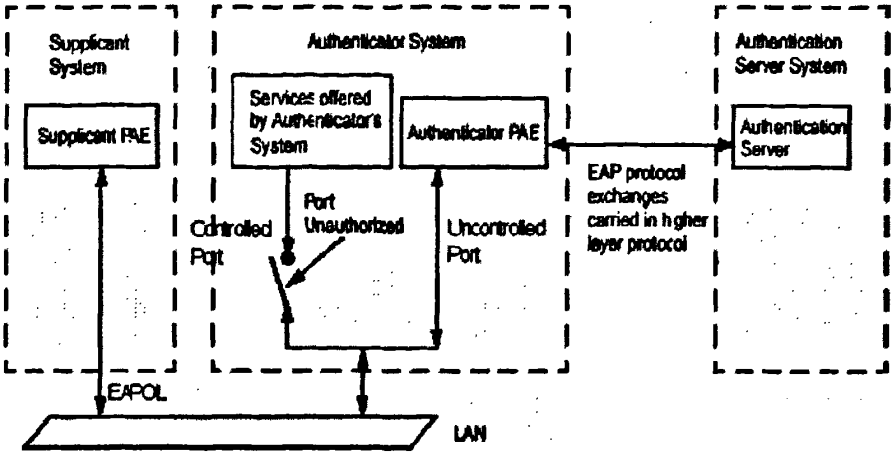


图 2-4 802.1x 协议的体系结构

认证系统的PAE通过不受控端口与Supplicant PAE进行通信，二者之间运行EAPOL协议。如果认证方服务器与认证方不在同一个系统时，认证方PAE接着对EAP协议重新打包并向上传到认证方服务器。RADIUS为后者的通信提供合适的手段，也可以用其他的协议去实现。认证系统的PAE与认证服务器之间运行EAP(Extensible Authentication Protocol)协议，认证方PAE控制受控端口的操作状态，但不干涉申请方PAE和认证服务器之间的认证交换。这种在认证方PAE和认证功能的分离允许后端的认证服务器用不同的认证机制来对申请方PAE进行验证。认证方PAE仅仅根据认证交换的结果来控制其受控端口的授权状态。

认证系统和认证服务器之间的通信可以通过网络进行，也可以使用其他的通信通道，例如如果认证系统和认证服务器集成在一起，二个实体之间的通信就可以不采用EAP 协议。

认证服务器通常为 RADIUS 服务器，该服务器可以存储有关用户的信息，比如用户所属的 VLAN、CAR 参数、优先级、用户的访问控制列表等等。当用户通过认证后，认证服务器会把用户的相关信息传递给认证系统，由认证系统构建动态的访问控制列表，用户的后续流量就将接受上述参数的监管。

对于终端用户的认证可以采用如上所示的机制进行，而对于网络设备之间的认证则采用如下的方式：

图 2-5 描述了这样一种情况：当一个网络设备 A 要求访问网络设备 B 所提供的

服务，则系统 A 的 PAE 就成为客户端(Suppliant)，系统 B 的 PAE 为认证系统(Authenticator)；如果 B 要求访问 A 所提供的服务，则 B 的 PAE 就成为客户端，A 的 PAE 就成为认证系统。

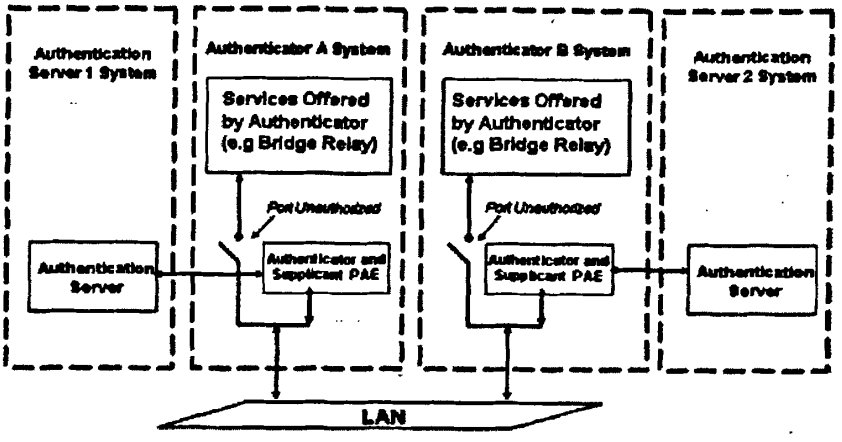


图 2-5 系统的两种工作模式

### 2.3.4 IEEE 802.1x 协议的工作机制

1、认证发起

认证的发起可以由用户主动发起，也可以由认证系统发起。当认证系统探测到未经过认证的用户使用网络，就会主动发起认证；用户端则可以通过客户端软件向认证系统发送EAPOL-Start报文发起认证。

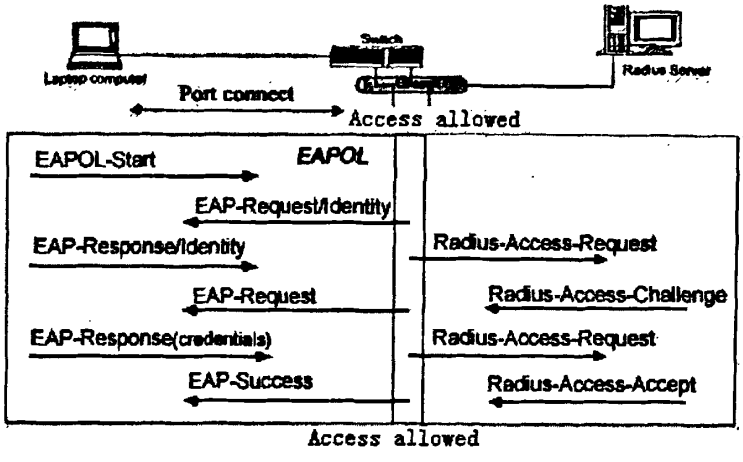


图 2-6 IEEE 802.1x 协议的工作机制

(1) 由认证系统发起的认证

当认证系统检测到有未经认证的用户使用网络时，就会发起认证。在认证开始之前，端口的状态被强制为未认证状态。

如果客户端的身份标识不可知，则认证系统会发送 EAP-Request/Identity 报文，请求客户端发送身份标识。这样，就开始了典型的认证过程。

客户端在收到来自认证系统的 EAP-Request 报文后，将发送 EAP-Response 报文响应认证系统的请求。

认证系统支持定期的重新认证，可以随时对一个端口发起重新认证的过程。如果端口状态为已认证状态，则当认证系统发起重新认证时，该端口通过认证，那么状态保持不便；如果未通过认证，则端口的状态改变为未认证状态

(2) 由客户端发起认证

如果用户要上网，则可以通过客户端软件主动发起认证。客户端软件会向认证系统发送 EAPOL-Start 报文主动发起认证。

认证系统在收到客户端发送的 EAPOL-Start 报文后，会发送 EAP-Request/Identity 报文响应用户请求，要求用户发送身份标识，这样就启动了一个认证过程。

2、退出已认证态

有几种方式可以造成认证系统把端口状态从已认证状态改变成未认证状态：

(1) 客户端未通过认证服务器的认证；

(2) 由于管理性的控制端口始终处于未认证状态，而不管是否通过认证；

(3) 与端口对应的 MAC 地址出现故障(管理性禁止或硬件故障)；

(4) 客户端与认证系统之间的连接失败，造成认证超时；

(5) 重新认证超时；

(6) 客户端未响应认证系统发起的认证请求；

(7) 客户端发送 EAPOL-Logoff 报文，主动下线；

退出已认证状态的直接结果就是导致用户下线，如果用户要继续上网则要再发起一个认证过程。

为什么要专门提供一个 EAPOL-Logoff 机制，是处于如下安全的考虑：

当一个用户从一台终端退出后，很可能其他用户不通过发起一个新的登录请求，

就可以利用该设备访问网络。提供专门的退出机制，以确保用户与认证系统专有的会话进程被中止，可以防止用户的访问权限被他人盗用。通过发送 EAPOL-Logoff 报文，可以使认证系统将对应的端口状态改变为未认证状态。

3、重新认证(根据时间)

为了保证用户和认证系统之间的链路处于激活状态，而不因为用户端设备发生故障造成异常死机，从而影响对用户计费的准确性，认证系统可以定期发起重新认证过程，该过程对于用户是透明的，也即用户无需再次输入用户名/密码。

重新认证由认证系统发起，时间是从最近一次成功认证后算起。重新认证可以激活或关闭，协议状态参数 reAuthEnabled 控制是否定期进行重新认证。重新认证的时间由参数 reAuthPeriod 控制，默认值为 3600 秒(一个小时)而且默认重新认证是关闭的。

重新认证的时间设定需要认真的规划，认证系统对端口进入的 MAC 地址的检测能力会影响到该时间的设定。如果对 MAC 地址的检测比较可靠，则重新认证时间可以设长一些。

4、认证报文丢失重传

对于认证系统和客户端之间通信的 EAP 报文，如果发生丢失，由认证系统负责进行报文的重传。在设定重传的时间时，考虑网络的实际环境，通常会认为认证系统和客户端之间报文丢失的几率比较低以及传送延迟低，因此一般通过一个超时计数器来设定，默认重传时间为 30 秒钟。

对于有些报文的丢失重传比较特殊，如 EAPOL-Star 报文的丢失,由客户端负责重传；而对于 EAP-Failure 和 EAP-Success 报文，由于客户端无法识别，认证系统不会重传。如果 EAP-Failure 或 EAP-Success 报文发生丢失，则客户端会在 auth-While 计数器超时后，自动转变为 CONNECTING 状态。

由于对用户身份合法性的认证最终由认证服务器执行，认证系统和认证服务器之间的报文丢失重传也很重要。

另外注意，对于用户的认证，在执行 802.1x 认证时，只有认证通过后，才有 DHCP 发起(如果配置为 DHCP 的自动获取) 和 IP 分配的过程。由于客户终端配置了 DHCP 自动获取，则可能在未启动 802.1x 客户端之前，就发起了 DHCP 的请求，而此时认

证系统处于禁止通行状态，这样认证系统会丢掉初始化的 DHCP 帧，同时会触发认证系统发起对用户的认证。

由于 DHCP 请求超时过程为 64 秒，所以如果 802.1x 认证过程能在这 64 秒内完成，则 DHCP 请求不会超时，能顺利完成地址请求；如果终端软件支持认证后再执行一次 DHCP，就不用考虑 64 秒的超时限制。

5、与不支持 802.1x 的设备的兼容

对于从一个不支持认证的系统过渡到认证系统，最理想的状态是希望能够平滑的进行过渡。由于 802.1x 协议是一个比较新的协议，如果应用在原有的旧网络中，则可能会存在与不支持设备的兼容性问题。

如果客户端支持 802.1x 协议，而网络设备不支持(也就是没有认证系统)，则客户端是不会收到认证系统响应的 EAP-Request/Identity 报文。在 802.1x 认证发起阶段，客户端首先发送 EAPOL-Star 报文到 802.1x 协议组申请的组播 MAC 地址，以查询网络上可以处理 802.1x 的设备(即认证系统)，由于网络中没有设备充当认证系统，所以客户端是得不到响应的。因此客户端在发起多次连接请求无响应后，自动认为已经通过认证。

如果客户端不支持 802.1x 协议，而网络中存在 802.1x 协议的认证系统，则客户端是不会响应认证系统发送的 EAP-Request/Identity 报文，因此端口会始终处于未认证状态。在这种情况下，客户端只能根据协议参数 OperControlledDirections 设定的值通过受控端口访问认证系统，通过未受控端口访问某些通过设置可以访问的服务。

6、EAP 帧的中继转发(Relay)

由于认证系统与客户端之间是采用 EAPOL 协议进行通信，而认证系统与认证服务器之间则是采用 EAP 协议进行通信，可以看出认证系统充当了把客户端的 EAPOL 协议帧转变为 EAP 协议帧的中继功能。

客户端将 EAPOL-Start 和 EAPOL-Logoff 帧发送给认证系统，而认证系统把 EAPOL-Key 帧发送给客户端。认证系统不能把这些帧中继给认证服务器。

EAP-Request/Identity 帧只能由认证系统发送给客户端，而不会发送给认证服务器。

除了以上的这些特殊帧外，客户端和认证系统之间通信的其他的帧都可以转发给认证服务器，不过帧格式要从 EAPOL 格式转变为 EAP 格式。相反，认证系统会把所有从认证服务器收到的 EAP 格式的帧转变为 EAPOL 格式发送给客户端。

7、加密 EAPOL 认证报文的传送

EAPOL 协议支持在客户端和认证系统之间加密传送认证报文。可以通过协议参数 KeyTransmissionEnabled 控制是否加密。如果该值为 TRUE，表示对认证报文进行加密；如果值为 FALSE，则表示不对认证报文加密。如果客户端或认证系统不支持加密，则 KeyTransmissionEnabled 参数设置为 FALSE。

### 2.3.5 802.1x 协议内容的实现

802.1x 协议在实现整个安全认证的过程中，其三个关键部分(客户端、认证系统、认证服务器)之间是通过通信协议进行交互的，因此有必要对其相关的通信协议做个介绍。

1、EAP 协议

802.1x 协议采用 EAP 协议在客户端、认证系统和认证服务器之间进行通信。EAP(Extensible Authentication Protocol 扩展的认证协议，RFC2284)是 PPP 认证的一个通用协议，支持多种认证机制，例如 smart cards, Kerberos, Public Key Encryption, One Time Passwords 等。EAP 在链路控制(LCP)阶段并不选择好一种认证机制，而把这一步推迟到认证阶段。这样就允许认证系统在确定某种特定认证机制之前请求更多的信息。

通过支持 EAP 协议，认证系统只需控制其受控端口的状态，但是并不干涉通过非受控端口在客户端和认证服务器之间传递的认证信息。这样，就实现了认证流和业务流的完全分离。可以使用一个“后端”服务器(认证服务器)来实际实现各种认证机制，认证系统仅仅需要传送认证信息，并根据认证返回的结果控制受控端口的状态。

2、EAP 的简单认证过程

在链路建立阶段完成后，认证系统发送一个或多个 Request 来对对方进行认证。Request 中有一个 type 域表明请求的类型。Request 中 type 的实例包括，Identity,

MD5-challenge, One-Time Passwords, Generic Token Card 等等。MD5-challenge 类型与 CHAP 认证协议紧紧对应。

典型情况下，认证系统将发送一个最初的 Identity 请求，然后是一个或多个请求认证信息的 Request。但是，最初的 Identity Request 并不是必需的，在 Identity 能被事先假定(如租用的链路，专用拨号线路等等)的情况下可以跳过(bypass)；

客户端发送一个 Response 数据包对每一个 Request 做出应答。对应于每一个 Request 数据包，Response 数据包包含一个 type 域，与 Request 中的 type 域对应；

认证系统发送一个 Success 或 Failure 数据包结束认证阶段。

3、EAP 帧结构

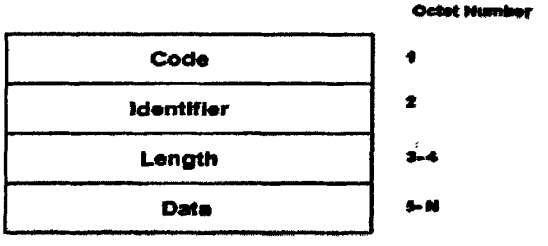


图 2-7 EAP 的帧结构

Code：1 个字节，表示 EAP 帧类型。EAP 代码分配如下：

1 – Request；

2 – Response；

3 – Success；

4 - Failure

Identifier：1 个字节，该值用于匹配 requests 的请求。Identifier 区域和系统端口一起单独标识一个认证过程。

Length：2 个字节，该值表示 EAP 帧的总长度

Data：0 或更多字节，表示数据

4、EAPOL 协议

在 802.1x 协议中定义了一种封装技术称为 EAPOL(EAP over LANs)，主要在客户

端和认证系统之间传送 EAP 协议报文，可以允许 EAP 协议报文在 LAN 上传送。

5、EAPOL 帧结构

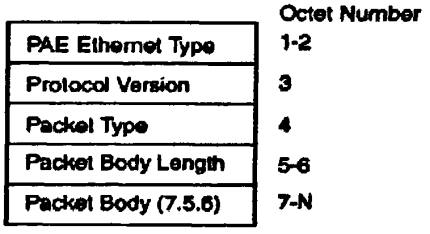


图 2-8 EAPOL 的帧结构

PAE Ethernet Type：2 个字节，表示协议类型，802.1x 分配的协议类型为 888E

Protocol Version：1 个字节，表示 EAPOL 帧的发送方所支持的协议版本号。

Packet Type：1 个字节，表示传送的帧类型。有如下几种帧类型。

(1) EAP-Packet. 值为 0000 0000，表示为 EAP 帧

(2) EAPOL-Start.值为 0000 0001，表示为 EAPOL-Start 帧

(3) EAPOL-Logoff. 值为 0000 0010，表示为 EAPOL-Logoff 请求帧

(4) EAPOL-Key. 值为 0000 0011，表示为 EAPOL-Key 帧.

(5) EAPOL-Encapsulated-ASF-Alert. 值为 0000 0100

Packet Body Length：2 个字节，表示 Packet Body 的长度

Packet Body：当 Packet 类型为 EAP-Packet、EAPOL-Key、EAPOL-Encapsulated-ASF-Alert，则 Packet Body 对应相应的值；对于其他帧类型，则该值为空。

6、EAPOL 帧的标记

在 EAPOL 帧传送过程中，不带 802.1q 的 VLAN 标记，但是可以带 802.1p 的优先级标记。所有的 PAE 都能够接收带或不带优先级标记的 EAPOL 帧。

7、EAPOL 帧发送的目标地址

当一个二层帧发送时，必须要有目标 MAC 地址。EAPOL 帧在发送时也不例外，

当客户端和认证系统互相之间不知道发送的目标时，其目标 MAC 地址为 802.1x 协议中分配的组播地址 01-80-c2-00-00-03。

### 2.3.6 基本的认证过程

(1) 用户开机后，通过 802.1x 客户端软件发起请求，查询网络上能处理 EAPOL(EAP Over LAN)数据包的设备。如果某台验证设备能处理 EAPOL 数据包，就会向客户端发送响应包并要求用户提供合法的身份标识，如用户名/密码；

(2)客户端收到验证设备的响应后，会提供身份标识给验证设备。由于此时客户端还未经过验证，因此认证流只能从验证设备的未受控的逻辑端口经过。验证设备通过 EAP 协议将认证流转发到 AAA 服务器，进行认证；

(3) 如果认证通过，则认证系统的受控逻辑端口打开，认证系统记录用户的相关信息，如 MAC 等信息，并建立动态的 ACL 访问列表，以限制用户的权限；

(4) 客户端软件发起 DHCP 请求，经认证设备转发到 DHCP Server；

(5) DHCP Server 为用户分配 IP 地址；

(6) DHCP Server 分配的地址信息返回给用户；

(7) 用户得到 IP 地址后可以上网；

(8) 如果用户要下网，可以通过客户端软件发起 LogOff 过程，认证设备检测到该数据后，会通知 AAA 服务器停止计费，并删除用户的相关信息(MAC 等)，受控逻辑端口关闭。用户进入再认证状态；

(9) 验证设备会通过定期的检测来保证链路的激活，如果用户异常死机，则验证设备在发起多次检测后，自动认为用户已经下线，于是向认证服务器发送终止计费的信息。

整个认证过程如图 2-9 所示

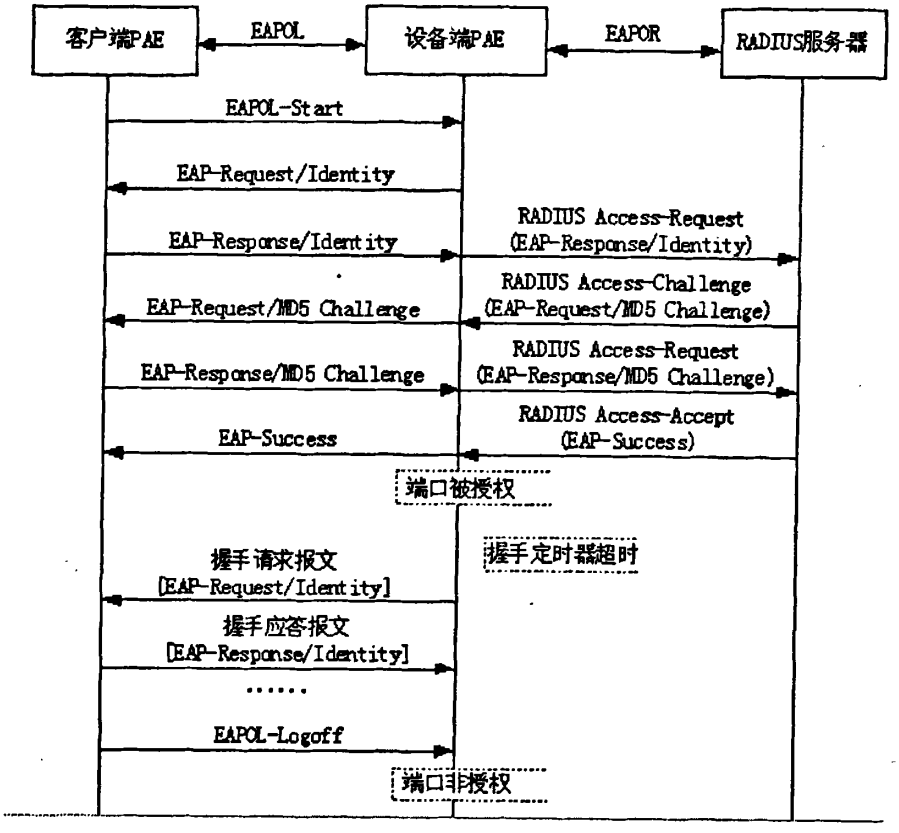
客户端PAE
EAPOL
设备端PAE
EAPOR
RADIUS服务器
EAPOL-Start
EAP-Request/Identity
EAP-Response/Identity
RADIUS Access-Request
(EAP-Response/Identity)
EAP-Request/MD5 Challenge
RADIUS Access-Challenge
(EAP-Request/MD5 Challenge)
EAP-Response/MD5 Challenge
RADIUS Access-Request
(EAP-Response/MD5 Challenge)
EAP-Success
RADIUS Access-Accept
(EAP-Success)
端口被授权
握手请求报文
[EAP-Request/Identity]
握手定时器超时
握手应答报文
[EAP-Response/Identity]
.....
EAPOL-Logoff
端口非授权

图 2-9 认证的步骤

# 第三章 PKI 和 EAP-TLS 相关技术

## 3.1 EAP-TLS 的基础 PKI

### 3.1.1 PKI 简介

随着Internet的普及，人们通过因特网进行沟通越来越多，相应的通过网络进行活动得到广泛的发展，但随之引发出一系列的网络安全问题。因此如何保证网上信息传输的安全，已成为网络通信的一个重要环节。为解决这些网络安全问题，世界各国对其进行了多年的研究，初步形成了一套完整的网络安全解决方案，即目前被广泛采用的PKI[8][9]技术(Public Key Infrastructure-公钥基础设施)，PKI(公钥基础设施)技术采用证书管理公钥，通过第三方的可信任机构--认证中心CA(Certificate Authority)，把用户的公钥和用户的其他标识信息(如名称、e-mail、身份证号等)捆绑在一起，在网上验证用户的身份。目前，通用的办法是采用基于PKI结构结合数字证书，通过把要传输的数字信息进行加密，保证信息传输的保密性、完整性、签名保证身份的真实性和抗抵赖性。

### 3.1.2 PKI 的基本定义与组成

PKI的基本定义十分简单，所谓PKI就是一个用公钥概念和技术实施和提供安全服务的具有普适性的安全基础设施。PKI是一种新的安全技术，它由公开密钥密码技术、数字证书、证书发放机构(CA)和关于公开密钥的安全策略等基本成分共同组成的。PKI 是利用公钥技术实现网络通信安全的一种体系，是一种基础设施，网络通讯、网上交易是利用它来保证安全的。从某种意义上讲，PKI包含了安全认证系统，即安全认证系统-CA系统是PKI不可缺的组成部分。

PKI公钥基础设施是提供公钥加密和数字签名服务的系统或平台，目的是为了管理密钥和证书。一个机构通过采用PKI框架管理密钥和证书可以建立一个安全的网络环境。PKI 主要包括四个部分：X.509格式的证书(X.509 V3)和证书撤销列表CRL(X.509 V2)；CA操作协议；CA管理协议；CA政策制定。

完整的PKI系统必须具有权威认证机构(CA)、数字证书库、密钥备份及恢复系统、

证书作废系统、应用接口(API)等基本构成部分，构建PKI也将围绕着这五大系统来着手构建。

(1) 认证机构(CA)：即数字证书的申请及签发机关，CA必须具备权威性的特征；

(2) 数字证书库：用于存储已签发的数字证书及公钥，用户可由此获得所需的其他用户的证书及公钥；

(3) 密钥备份及恢复系统：如果用户丢失了用于解密数据的密钥，则数据将无法被解密，这将造成合法数据丢失。为避免这种情况，PKI提供备份与恢复密钥的机制。但须注意，密钥的备份与恢复必须由可信的机构来完成。并且，密钥备份与恢复只能针对解密密钥，签名私钥为确保其唯一性而不能够作备份。

(4) 证书撤销系统：证书作废处理系统是PKI的一个必备的组件。与日常生活中的各种身份证件一样,证书有效期以内也可能需要作废，原因可能是密钥介质丢失或用户身份变更等。为实现这一点,PKI必须提供作废证书的一系列机制。

(5) 应用接口(API)：PKI的价值在于使用户能够方便地使用加密、数字签名等安全服务，因此一个完整的PKI必须提供良好的应用接口系统，使得各种各样的应用能够以安全、一致、可信的方式与PKI交互，确保安全网络环境的完整性和易用性。

通常来说，CA 是证书的签发机构,它是 PKI 的核心。众所周知，构建密码服务系统的核心内容是如何实现密钥管理。公钥体制涉及到一对密钥(即私钥和公钥)，私钥只由用户独立掌握，无须在网上传输，而公钥则是公开的，需要在网上传送，故公钥体制的密钥管理主要是针对公钥的管理问题，目前较好的解决方案是数字证书机制。

### 3.1.3 PKI 的原理

PKI是基于公钥密码技术的，要想深刻理解PKI的原理，就一定要对PKI涉及到的密码学知识有比较透彻的理解。下面简单介绍一下密码学知识。

对于普通的对称密码学，加密运算与解密运算使用同样的密钥。通常,使用的加密算法比较简便高效，密钥简短，破译极其困难，由于系统的保密性主要取决于密钥的安全性，所以，在公开的计算机网络上安全地传送和保管密钥是一个严峻的问题。正是由于对称密码学中双方都使用相同的密钥，因此无法实现数据签名和不可否认性等功能。

而与此不同的非对称密码学，具有两个密钥，一个是公钥一个是私钥，它们具有

这种性质：用公钥加密的文件只能用私钥解密，而私钥加密的文件只能用公钥解密。

公钥顾名思义是公开的，所有的人都可以得到它；私钥也顾名思义是私有的，不应被其他人得到，具有唯一性。这样就可以满足电子商务中需要的一些安全要求。比如说要证明某个文件是特定人的，这个人就可以用他的私钥对文件进行签名，别人如果能用他的公钥核对此签名，说明此文件就是这个人的，这就可以说是一种认证的实现。还有如果只想让某个人看到一个文件，就可以用此人的公钥加密文件然后传给他，这时只有他自己可以用私钥解密，这可以说是保密性的实现。

基于这种原理还可以实现完整性。这就是PKI所依赖的核心思想，这部分对于深刻把握PKI是很重要的，而恰恰这部分是最有意思的。

比如在现实生活中，我们想给某个人在网上传送一个机密文件，该文件我们只想让那个人看到，我们设想了很多方法，首先我们想到了用对称密码将文件加密，而在我们把加密后的文件传送给他后，我们又必须得让他知道解密用的密钥，这样就又出现了一个新的问题，就是我们如何保密的传输该密钥，此时我们发现传输对称密钥也不可靠。

后来我们可以改用非对称密码的技术加密，此时发现问题逐渐解决了。然而又有了一个新的问题产生，那就是如何才能确定这个公钥就是某个人的，假如我们得到了一个虚假的公钥，比如说我们想传给A一个文件，于是开始查找A的公钥，但是这时B从中捣乱，他把自己的公钥替换了A的公钥，让我们错误的认为B的公钥就是A的公钥，导致我们最终使用B的公钥加密文件，结果A无法打开文件，而B可以打开文件，这样B实现了对保密信息的窃取行为。

因此就算是采用非对称密码技术，我们仍旧无法保证保密性的实现，那我们如何才能确切的得到我们想要的人的公钥呢？这时我们很自然的想到需要一个仲裁机构，或者说是一个权威的机构，它能为我准确无误的提供我们需要的人的公钥，这就是CA。

这实际上也是应用公钥技术的关键，即如何确认某个人真正拥有公钥(及对应的私钥)。在PKI中，为了确保用户的身份及他所持有密钥的正确匹配，公开密钥系统需要一个值得信赖而且独立的第三方机构充当认证中心(CA)，来确认公钥拥有人的真正身份。就象公安局发放的身份证一样，认证中心发放一个叫“数字证书”的身份证明。这个数字证书包含了用户身份的部分信息及用户所持有的公钥。象公安局对身份证盖

章一样，认证中心利用本身的私钥为数字证书加上数字签名。任何想发放自己公钥的用户，可以去认证中心申请自己的证书。认证中心在鉴定该人的真实身份后，颁发包含用户公钥的数字证书。其他用户只要能验证证书是真实的，并且信任颁发证书的认证中心，就可以确认用户的公钥。认证中心是公钥基础设施的核心，有了大家信任的认证中心，用户才能放心方便的使用公钥技术带来的安全服务。

### 3.1.4 PKI 的核心部分 CA

认证中心CA 作为PKI 的核心部分，CA实现了PKI中一些很重要的功能，概括地说，认证中心(CA)的功能有：证书发放、证书更新、证书撤销和证书验证。CA的核心功能就是发放和管理数字证书，具体描述如下：

(1) 接收验证最终用户数字证书的申请；

(2) 确定是否接受最终用户数字证书的申请-证书的审批；

(3) 向申请者颁发、拒绝颁发数字证书-证书的发放；

(4) 接收、处理最终用户的数字证书更新请求-证书的更新；

(5) 接收最终用户数字证书的查询、撤销；

(6) 产生和发布证书废止列表(CRL)；

(7) 数字证书的归档；

(8) 密钥归档；

(9) 历史数据归档。

认证中心CA为了实现其功能，主要由以下三部分组成：

(1) 注册服务器：通过网站建立的站点，可为客户提供24x7不间断的服务。客户在网上提出证书申请和填写相应的证书申请表。

(2) 证书申请受理和审核机构：负责证书的申请和审核。它的主要功能是接受客户证书申请并进行审核。

(3)认证中心服务器：是数字证书生成、发放的运行实体，同时提供发放证书的管理、证书废止列表(CRL)的生成和处理等服务。

在具体实施时，CA 的必须做到以下几点：

(1) 验证并标识证书申请者的身份；

(2) 确保CA 用于签名证书的非对称密钥的质量；

(3) 确保整个签证过程的安全性，确保签名私钥的安全性；

(4) 证书资料信息(包括公钥证书序列号，CA 标识等)的管理；

(5) 确定并检查证书的有效期限；

(6) 确保证书主体标识的唯一性，防止重名；

(7) 发布并维护作废证书列表；

(8) 对整个证书签发过程做日志记录；

(9) 向申请人发出通知。

在这其中最重要的是CA自己的一对密钥的管理，它必须确保其高度的机密性，防止他方伪造证书。CA的公钥在网上公开，因此整个网络系统必须保证完整性。CA的数字签名保证了证书(实质是持有者的公钥)的合法性和权威性。

用户的公钥有两种产生的方式：

(1) 用户自己生成密钥对，然后将公钥以安全的方式传送给CA，该过程必须保证用户公钥的验证性和完整性。

(2) CA替用户生成密钥对，然后将其以安全的方式传送给用户，该过程必须确保密钥对的机密性，完整性和可验证性。该方式下由于用户的私钥为CA所产生，所以对CA的可信性有更高的要求。CA必须在事后销毁用户的私钥。

一般而言公钥有两大类用途，就像本文前面所述，一个是用于验证数字签名，一个是用于加密信息。相应的在CA系统中也需要配置用于数字签名/验证签名的密钥对和用于数据加密/解密的密钥对，分别称为签名密钥对和加密密钥对。由于两种密钥对的功能不同，管理起来也不大相同，所以在CA中为一个用户配置两对密钥，两张证书。

CA中比较重要的几个概念点有：

(1) 证书库：证书库是CA颁发证书和撤销证书的集中存放地，它像网上的“白页”一样，是网上的一种公共信息库，供广大公众进行开放式查询。这是非常关键的一点，因为我们构建CA的最根本目的就是获得他人的公钥，目前通常的做法是将证书和证书撤消信息发布到一个数据库中。

(2) 证书的撤消：由于现实生活中的一些原因，比如说私钥的泄漏，职员离开单位等情况的发生，应当对其证书进行撤消。这种撤消应该是及时的，因为如果撤消延迟的话，会使得不再有效的证书仍被使用，将造成一定的损失。在CA中，证书的撤

消使用的手段是证书撤消列表或称为CRL。即将作废的证书放入CRL中，并及时的公布于众，根据实际情况不同可以采取周期性发布机制和在线查询机制两种方式。

(3) 密钥的备份和恢复：如果用户由于某种原因丢失了解密数据的密钥，那么被加密的密文将无法解开，这将造成数据丢失。为了避免这种情况的发生，PKI提供了密钥备份于解密密钥的恢复机制。这一工作也是应该由可信的机构CA来完成的，而且，密钥的备份与恢复只能针对解密密钥，而签名密钥不能做备份，因为签名密钥匙用于不不可否认性的证明的，如果存有备份的话，将会不利于保证不可否认性。

(4) 一个证书的有效期是有限的：这样规定既有理论上的原因，又有实际操作的因素。在理论上诸如关于当前非对称算法和密钥长度的可破译性分析，同时在实际应用中，证明密钥必须有一定的更换频度，才能得到密钥使用的安全性。因此一个已颁发的证书需要有过期的措施，以便更换新的证书。为了解决密钥更新的复杂性和人工干预的麻烦，应由PKI本身自动完成密钥或证书的更新，完全不需要用户的干预。它的指导思想是：无论用户的证书用于何种目的，在认证时，都会在线自动检查有效期，当失效日期到来之前的某时间间隔内，自动启动更新程序，生成一个新的证书来替代旧证书。

## 3.2. EAP-TLS 协议介绍

### 3.2.1 概要介绍

为了保证网络资源的安全可控，网络接入控制已经成为当前主要的安全环节。其中，采用网络身份鉴别协议 EAP 同 PKI 技术相结合，成为了集通用、安全、高性价比于一身的做法。

网络，无论是局域网、城域网还是广域网，都是一种资源，而资源就需要保护，不能任何人、任何设备都随意接入。随着 WLAN 的迅速发展，网络接入变得更简单，但无线在带来方便性的同时，也使得网络接入的安全问题显得日益突出。

目前业界的看法是，网络接入控制是保证网络安全的一个重要环节，而接入控制的关键是身份鉴别。在已有的各种身份鉴别方法中，PKI 是被公认为最安全有效的。而网络身份鉴别协议 EAP 同 PKI 的结合，能够实现安全的网络接入控制。

EAP 是为点对点协议(PPP)设计的身份鉴别协议，它采用 Request/Response 方式，这点同 RADIUS 非常类似。EAP 涉及三个实体： Peer、Authenticator 及 Authentication

Server。这里的 Peer 即待接入的终端设备，如计算机；Authenticator 是网络接入设备，如接入服务器、交换机、无线接入点 AP 等；Authentication Server 用于完成用户的身份鉴别。

在采用 EAP 协议后，Peer 就会连接到 Authenticator，而 Authenticator 会向 Peer 发出 EAP 请求，要求 Peer 提交身份信息，Peer 响应身份信息后，Peer 与 Authentication Server 之间交换信息，完成身份鉴别。身份鉴别成功后，Peer 即可接入到 Authenticator，并连到网络上。

在这个过程中，Authenticator 与 Authentication Server 是从功能上划分为二个实体，实际上 Authentication Server 与 Authenticator 可在一个设备上实现。当这两个功能在不同设备上实现时，Authenticator 相当于 Peer 和 Authentication Server 之间鉴别信息交换的桥梁。如果需要，Authenticator 将实现 EAP 与其他身份鉴别协议间的转换。当这两个功能分开实现时， Authenticator 与 Authentication Server 之间可走 RADIUS 协议。

PKI 是目前实际身份鉴别的最好技术，这是因为 PKI 不但能够实现身份鉴别，而且能够同密钥协商机制有机地结合，实现数据链路层的信息加密。目前国际上基于 PKI 的 EAP 身份鉴别方法有许多种，对于国内用户来说，比较有现实意义的主要集中在 EAP-TLS 和 PEAP 两种技术上(目前也有 EAP-TTLS 技术)。

EAP-TLS[10]是一个 IETF 标准。TLS 即传输层安全(Transport Layer Security)，也称为 SSL。它原本是一种传输层的安全协议，主要实现两个功能：身份鉴别与信息加密。TLS 可实现基于证书的单向身份鉴别(只鉴别服务器)和双向身份鉴别(同时鉴别客户端与服务器)。TLS 在完成身份鉴别的同时，还交换密钥信息，通过密钥信息可导出会话密钥用于信息加密。

需要指出的是，在这里 TLS 并不是作为一个安全传输层协议跑在 TCP/IP 层之上，而是将 TLS 的 Handshake Record 直接嵌套在 EAP 数据包中，作为 EAP Request / Response 的数据来传送，通过 TLS 的握手记录完成单向或双向的身份鉴别。EAP-TLS 只利用了 TLS 的身份鉴别功能，并没有利用 TLS 建立的加密通道。

为了能够进一步利用 TLS 建立的安全通道交换 EAP 身份鉴别信息，IETF 随后出台了 PEAP(Protected EAP Protocol)标准。PEAP 不但通过 EAP Request / Response 数据包传送 TLS 的 Handshake Record 完成身份鉴别，并且完成身份鉴别后进一步通过 TLS

的 Data Record 再传送 EAP 身份鉴别协议。

这就是说，在使用 PKI 数字证书进行的身份鉴别协议中，EAP-TLS 必须使用客户端证书完成客户端的身份鉴别，而 PEAP 则可以使用客户端证书，也可以不使用客户端证书，这是因为该协议可在建立起来的 TLS 加密通道的基础上，进一步采用其他的身份鉴别协议，如口令身份验证、动态口令身份验证等。这种做法既利用了 PKI 安全的优点，又兼顾了目前口令鉴别应用广泛、简单的特点。

对于 EAP 结合 PKI 实现网络的接入控制，主要可以根据控制环境分为三类。

(1) PPP 网络连接，这是目前国内用户应用非常广泛的接入方式，大部分家庭和很多中小企业的互联网接入均采用这种模式。在这种接入环境下，客户端身份鉴别的对象可以是终端设备(比如 Modem)，使用设备数字证书进行身份鉴别，只有授权的设备才能接入，身份鉴别的对象也可以是人，使用个人数字证书进行身份鉴别，只有授权的用户才能接入。

(2) PPPoE 的接入，如果网络接入设备，如交换机、接入服务器，支持 PPPoE(PPP over Ethernet)协议，那么这种网络也可以使用基于 PKI 的 EAP 身份鉴别。同样地，数字证书可以是与用户绑定，只有授权的用户才能接入，也可以是同终端设备绑定，只有授权的设备才能接入。

(3) 802.1x 基于端口的访问控制，无疑，802.1x 是针对点对点 LAN，或者逻辑上具有点对点链接的 LAN(WLAN)而设计的一种基于端口的访问控制协议。这里说的端口，主要是指设备与 LAN 的接入点。所谓基于端口的访问控制，即控制其他设备通过点对点链接连到要访问的端口，并通过端口交换数据，获得该端口提供的服务。这里的端口可以是一个交换机的端口，也可以是一个无线局域网接入点 AP 的逻辑端口，还可以是一个计算机、服务器与 LAN 的连接端口(有线或无线的)。

从技术上说，802.1x 可以把每个 LAN 端口进一步地在逻辑上分为不受控的端口(Uncontrolled Port)和受控的端口(Controlled Port)。不受控的端口用于交换控制与管理信息(如身份鉴别信息)，受控的端口用于交换数据，只有通过身份鉴别后，其他设备才能访问该端口交换数据。

那么，如果802.1x使用EAPOL(EAP Over LAN)进行身份鉴别，在802.1x中Peer称为Supplicant。与EAP相比，EAPOL允许待接入Supplicant(终端设备)发起身份鉴别请求。IEEE目前正在逐步完善无线局域网的安全标准802.11i，该标准采用802.1x进行身

份鉴别，而Wi-Fi联盟则推出了一个针对802.11i的过渡标准WPA。WPA支持的鉴别协议包括EAP-TLS、PEAP和EAP-TTLS等。所以说，对于支持802.1x的网络接入，都可以使用PKI数字证书技术实现安全身份鉴别。

WPA是Wi-Fi Protected Access(Wi-Fi保护接入)的简称，是无线网络的数据加密规格。它通过使用可扩展认证协议(EAP)提高了WEP的安全功能，通过一种加密方法提高了数据传送的安全性。

WPA使用802.1x认证服务器给每个用户分配不同的密钥，但是它在安全性比较差的“预先共享密钥(PSK)”模式中也有效。PSK用于家庭和小型办公室网络，每个用户都有同样的密码口令。TKIP是Temporal Key Integrity Protocol(临时密钥完整性协议)的简称，是一种加密方法。TKIP提供结合信息完整性检查和重新按键机制的信息包密钥。AES是Advanced Encryption Standard(高级加密标准)的简称，是Wi-Fi授权的高效加密标准。

从国内的情况看，PKI 作为保证信息安全的一种成熟技术，目前已在很多领域获得了广泛的应用，并日益受到人们的重视。基于 PKI 的身份鉴别具有其他身份鉴别技术无法替代的优点。可以预见的是，将来在各种网络接入控制中，PKI 无疑会获得越来越多的应用与发展机会。

### 3.2.2 EAP-TLS 认证过程

(1) 通过802.1x客户端软件发起EAPoL-Start请求，查询认证者AP；

(2) 认证者向客户端发送EAP-Request/Identity响应帧，要求用户提供合法的身份标识，如用户名/密码；

(3) 客户端收到认证方的请求后，向认证者发送EAP-Response/Identity帧，提供身份标识；

(4) 认证者将用户身份标识信息封装成RADIUS Access Request帧发给RADIUS服务器；

(5) RADIUS服务器向客户端发送EAP-Request开始TLS握手；

(6) 开始TLS握手第一阶段，确定一些相关的参数，包括协议版本、会话ID、加密套件、压缩算法和初始随机数；

(7) 进行TLS握手的第二阶段，服务器端发送证书、交换密钥、证书请求，最后

发送hello阶段的结束信号；

(8) 进行TLS握手的第三阶段，客户端发送证书(如果服务器端有请求)、密钥交换、证书验证；

(9) 进行TLS握手的第四阶段，改变加密套件，结束握手协议；

(10) 客户端向服务器端发送TLS ACK；

(11) RADIUS 服务器向客户端发送 EAP Success 消息(其中包含密钥信息)，认证成功。

## 3.3 智能卡, USB Key

智能卡[11][12] (Smart Card)是IC卡的一种，它具有价格便宜，可方便携带的特点。它内含有集成电路芯片，本身具有储存能力和计算能力。根据其中镶嵌的集成电路的不同，IC卡基本上可以分为3类：

一类是存取器卡，它的集成电路为EEPROM，即可擦除的可编程的只读存储器，它可以写入一定的数据，并在掉电的情况下保证数据不会丢失。

第二类是逻辑加密卡，卡中的集成电路具有加密逻辑和EEPROM，使卡中的数据受到一定的加密算法的保护。

第三类是CPU卡，CPU卡的集成电路不但包含EEPROM，而且还包含CPU(中央处理器)、随机存储器RAM以及固化在只读存储器中的操作系统COS(Chip Operating System)。严格地讲，CPU卡才是真正的智能卡，智能卡在卡内完成密码运算，保障了密钥的安全性，加上物理防篡改技术的采用，伪造的难度和成本很高，大大提高了其安全性。但是智能卡的缺点在于，它需要读卡器配合使用，增加了用户的成本，同时对于便携性有一定的影响。

USB Key 是智能加密钥匙，它的功能和智能卡一样，同样具有COS，但是USB Key 主要采用USB 接口直接同计算机通信，免去了对读卡器的需要，在保证了安全性的同时，提高了易用性和便携性，目前使用广泛。

USB Key普遍支持RSA 1024、DES/3DES和SHA-1等算法，支持RSA 1024非对称算法的签名/验证，加密/解密的运算，芯片内生成随机的RSA 1024密钥对。同时USB Key有PIN码保护，即使硬件丢失，在不知道PIN码的时候，不能运行获得证书签名信息。

USB Key可以由内部芯片生成证书，也可以从外部导入证书。在这里，我们选择使用外部导入证书，这样由CA生成的证书就可以导入USB Key，受USB Key管理且不能被导出，保证证书安全。

随着USB Key厂家越来越多，各厂家USB Key的硬件功能和中间件操作方式不尽相同，各CA中心为了支持多家的USB Key，不得不在自己的系统中使用和维护各厂家的初始化工具，给CA中心在颁发USB Key的过程中添加了巨大的工作量。

无驱型USB Key，通过在标准PKI接口和硬件驱动之间加入一个TSP(Token Service Provider)的方式来实现对硬件的操作，将与硬件有关的所有操作封装在TSP中，PKI应用程序无需了其使用的硬件的特性，硬件厂家只需要提供符合标准的TSP，就可以在用户原来的应用中使用。

## 3.4 JAVA PKCS#11 介绍

在我们的实践中，选用了EJBCA[13]配套的一个项目Hard Token Management Framework[14]，同样使用JAVA来编写，所以，下面先介绍一下JAVA PKCS#11[15][16]的一些内容。

PKCS是公开密钥密码系统标准(Public Key Cryptography Standards)，由RSA实验室开发。而PKCS#11则是PKCS的一部分，特指密码设备的应用编程接口，PKCS#11标准称之为“Cryptoki”，它是“Cryptography token interface”的缩写。Cryptoki是一个较底层的编程接口，它抽象了密码设备的细节，为应用程序提供一个通用的调用模板。

JAVA平台定义了一套进行加密操作的编程接口，这些接口集合被称作Java Cryptography Architecture(JCA)和Java Cryptography Extension (JCE)，这些加密接口是基于服务提供者的，特别是需要跟编程接口交互的程序，以及在配置的服务中进行切实的加密操作，跟一组服务提供接口(Service Provider Interfaces (SPIs))紧密相关。这个结构支持不同的服务提供的实现方法。有一些服务提供者可以在软件中进行密码操作，另一些则可以在硬件上进行操作，比如，在智能卡或者是硬件密码加速器。

这个密码硬件接口标准PKCS#11，由RSA安全公司开发，并且定义了操作密码硬件的本地编程接口，比如硬件密码加速器和智能卡。要实现本地PKCS#11硬件接口和JAVA平台的结合，一个新的服务提供者，Sun PKCS#11，在J2SE 5.0版本中被引入。

这个新的服务提供者允许现有使用JCA和JCE AP开发的应用，在不修改应用的情况下，能够访问PKCS#11硬件，仅有的要求，只是正确配置一下JAVA Runtime中的提供者。

使用现有的API，应用系统可以使用大多数PKCS#11功能，有些应用也许可以获得更多的可伸缩性和性能。比如一个应用需要能够更加轻松的处理智能卡的频繁的插入和取出。否则一个PKCS#11硬件需要为一个非关联的操作进行认证，结果，应用必须在不使用密钥库的情况下能够登录进硬件。在J2SE 5.0的版本中，JCA已经被加强，允许应用在处理不同提供者时，获得更大的伸缩性。

对比其他的服务提供者，Sun PKCS#11服务提供者，不是实现密码算法本身，而是在JAVA JCA、JCE API和本地PKCS#11加密接口之间扮演一个桥梁，在两者之间传递系统调用和约定。也就是说，JAVA应用程序调用标准的JCA和JCE API，能够完全不变地利用PKCS#11的底层实现。

J2SE只是实现了到本地PKCS#11的访问，本身并不包括PKCS#11实现。密码设备，包括智能卡，硬件加速器等，一般都会提供一套PKCS#11实现的软件，需要安装到操作系统上，并根据制造商的说明进行配置。

# 第四章 EAP-TLS 应用设计

## 4.1 设计框架

### 4.1.1 CA 和 FreeRADIUS 协作

本文的FreeRADIUS EAP-TLS设计的目标是使用CA软件和EAP-TLS进行协同工作，解决用户管理和认证的细节问题，证书核发和CRL的更新是两个软件合作的关键。

需要做的工作主要有以下三个方面

(1) CA软件的选择、部署和使用，进行证书管理；

(2) 安装FreeRADIUS，配置EAP-TLS，以使证书认证能够正常使用；

(3) CA和FreeRADIUS的关联，CRL的实时产生和更新。

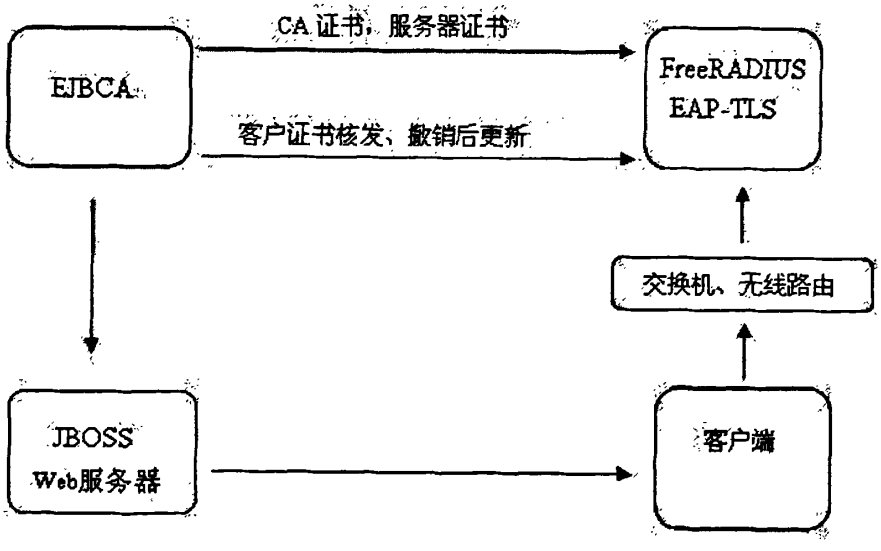


图4-1 完整的EAP-TLS应用框架

如图4-1所示，在这个简单框架中，包含了这个用户认证能够运行所需的所有条件。EJBCA，这是CA的核心，所有的用户证书产生、授权和撤销都在这里决定。JBOSS则是EJBCA运行的平台，证书请求和发布都通过这个web服务器。

对于关系最紧密的两个部分，EJBCA和FreeRADIUS，可以安装于同一服务器硬件，也可以在不同的服务器硬件上。如果处于不同的服务器，那么，根证书、服务器证书和CRL证书的安装可以通过移动介质进行，如U盘；也可通过网络获取，采用ftp等方式。通过网络传输的情况下，需要确保各种证书安全传输，特别是经常需要更新的CRL证书，在到达后，可以使用根证书对CRL证书的有效性进行核实。

### 4.1.2 使用USB Key硬件证书

1、证书的缺点

前面讨论了基于FreeRADIUS的认证，研究了EAP-TLS证书认证方式和证书管理的各种方式。但是，到目前为止，我们虽然通过使用证书，解决了密码被盗用的问题，仍然有一个迫切的问题需要我们解决。直到现在，我们的证书都是以文本方式存在，也就是说，仍然具有可复制性，在知道了证书的解密密码之后，同一证书还是可以被多人同时使用。怎么使得证书不具有复制功能呢？这就需要引入硬件，把证书保存在硬件中，这样就能保证证书的唯一性。

2、Hard Token Management Framework 密码硬件管理框架

Hard Token Management Framework(www.hardtokenmgmt.org)是一个用Java编写的EJBCA扩展，用于管理智能卡或者USB专用加密设备。HTMF通过PKCS#11协议接口和硬件进行通信，只要硬件完全按照PKCS#11接口进行设计，HTMF软件就能适应这些硬件。HTMF有丰富的应用模组，可以有效组合以满足组织使用的要求。

目前使用HTMF框架的应用组件模组叫做ToLiMa，具有以下的特性：

(1) 签署硬件证书，包括正常的、临时的和企业应用；

(2) 在不把PUK暴露给用户或者管理员的情况下解锁PIN；

(3) 撤销丢失的硬件证书；

(4) 更新过期的硬件证书；

(5) 在系统中激活硬件证书。

(6) 同样，在没有硬件证书管理员的情况下，比如24x7的工作环境中，也可以根据批准的原则来签署和解锁硬件证书。这样，就可以让终端用户的的同事生成一个行为请求，并且提交给中心支持部门进行审查和批准。

一个HTMF应用所需要的运行环境主要包括以下的部分：

(1) Hard Token Management Framework密码硬件管理框架

(2) Hard Token Management Framework安装在EJBCA 3.5或者更新版本之上

(3) Ant编译工具

(4) JAVA证书库，用于给jar文件签名

(5) 每个管理工作站需要两个读卡器，如果是USB Key，则需要有两个USB接口

(6) 最少两个配套密码硬件

(7) 一套完整的PKCS#11软件

(8) 关于USB Key使用的构想

USB Key带有智能卡芯片，完全支持智能卡的所有功能，如数字签名功能，而且还将智能卡和读卡器的功能合二为一，用户使用时只要通过计算机的USB端口即可实现即插即用的安全认证服务。

在EAP-TLS认证设置方面，没有任何需要特别改动的地方。在有线网络链接的属性中，认证方式选择使用IEEE 802.1x认证。无线网络则选择认证方式则选择WPA，此时认证选项中IEEE 802.1x为必选。在EAP类型中选择智能卡或者证书，进一步选择使用智能卡。至此，客户端配置即完成，已经可以使用。当出现系统提示时，将USB Key插入计算机上的，然后在系统提示时输入个人识别码 (PIN)，经FreeRADIUS服务器认证通过即可使用网络。可以看出使用USB Key方式来管理证书的时候，在使用便捷性，安全性上都有了进一步提高。

虽然Hard Token Management Framework支持标准的PKCS#11协议，但是它自带的PKCS#11本地实现很有限，而且它支持的硬件在国内也没有销售，所以，Hard Token Management Framework和USB Key硬件联合使用尝试这块只能暂时搁置。

受Hard Token Management Framework的限制，我们暂时不能使用USB Key，但是，我们除了使用这个框架软件，我们是否还有其他办法来操作USB Key？答案很显然，一般的USB Key供应商都会提供一套PKCS#11的接口，我们可以根据这个接口来编写程序，直接对USB Key进行操作。并且，由于USB Key的标准规范化，更多设备已经可以摆脱驱动的束缚，使用操作系统自带的标准智能卡访问接口就可以正确操作，这也就是无驱型USB Key的最大优点。选择无驱型USB Key，Windows XP以上操作系统无需安装驱动程序，免去编写额外的驱动程序编写工作，也为客户端使用增加了便捷。

由此，我们获得新的启发，首选无驱型USB Key,要进行编程操作，只需要使用PKCS#11标准接口或者制造商提供的编程接口来进行开发，从数据库获得证书，通过API写入到USB Key中，这个方法更加直接一些，具体实践留待以后再进行实践。

## 4.2 CA软件选择

尽管TinyCA给可以满足一般的证书管理，使用方便，然而，当CA业务规模扩大，工作需要分布在不同区域的时候，一个人的工作显然是不能应付的，单机版本的CA软件显然不能满足要求，我们需要一个适合在网络环境中使用的证书管理工具。目前比较完善的CA软件有OpenCA和EJBCA，这是两种不同风格的软件，怎么选择需要根据使用环境的大小来确定。

1、OpenCA简介

OpenCA是一个开源项目，用于构建一个功能强大的CA系统。OpenCA是构架在多项开源项目上，可同时使用OpenLDAP、OpenSSL、Apache Project、Apache mod_ssl、MySQL来实现一个完整的CA认证系统。

OpenCA最初的设计主要包含三部份：(1)以Perl为主的网页介面，(2)以OpenSSL为后端的密码技术，(3)利用Database储存使用者的所有信息。现在的OpenCA更加复杂，不过这三个部分仍是的基础，几乎所有的操作都能通过网页界面来操作OpenCA。

目前有六个预设的操作介面：(1)Node-后端管理，(2)CA-证书授权中心界面，(3)RA-证书管理中心，(4)LDAP-目录服务中心，(5)Public-使用者，(6)SCEP-由Cisco发展制定的协议，提供VPN、路由器和交换机之间的通讯，如何传送请求和认证的协议。OpenCA作为PKI系统建立的基础，而数据库存储用户的必要资料，如证书签署的请求、认证、证书的撤销和CRL。

但是，由于OpenCA安装和使用比较复杂，给部署带来了不便，维护要求比较高，不太适合在一般规模的应用中使用。所以，我们需要寻找较为简单的实现。

2、选择EJBCA

EJBCA是一个完整功能的证书认证程序，完全用Java编写，使用J2EE(Java2 Enterprise Edition，Java2企业版)技术。它建造在J2EE平台之上，是一个开放的、健壮的、高性能的、平台独立的、灵活的和基于组件的CA，可以被单独使用或集成到其

它J2EE应用程序中，并且它还提供了一个灵活强大的基于Web的用户图形界面。由于它实现了PKI中的几乎所有重要的部件，比如RA(Registration Authority，注册中心)、CA(Certification Autohrity，认证中心)、CRL(Certification Remove List，证书撤销列表)和证书存储数据库等，因此得到了广泛的应用。

EJBCA建立的CA安全性主要体现在两方面，一方面是EJBCA本身的体系结构，另一方面是JCE的安全服务提供者Boubcycastle提供的加密应用程序接口。EJBCA是一个开放源代码的项目，因此它具有公开性、开放性和灵活性，同样其对安全上的配置也很方便和灵活，可以有多种方法来保证EJBCA安全性，比如，可以设置对内部Bean的访问设置权，使得应用服务器上的管理员能访问到CA中心；另外还可以对运行EJBCA的服务器JBOSS进行配置，使得JBOSS只用SSL层与外部建立安全连接，从而来保证CA的安全性。

Boubcycastle是以Java实现的比较优秀加密算法接口，其提供的加密应用程序接口是EJBCA安全的核心组件，因为它提供了密钥和证书的生成及证书发布等功能。Boubcycastle提供的加密应用程序接口同样是开放源代码的，因此算法也是公开的，Bouncycastle集成了用于产生密钥对的RSA算法，多种消息摘要算法如MD算法、SHA-1算法、RIPEMD-160算法等，另外Bouncycastle还将目前安全性较高的椭圆曲线加密算法(ECC)也作为其的一个API，通过对其公开算法验证，可保证其理论上的安全性。

EJBCA是一个基于组件的结构，用户甚至可以自己开发一些私有的加密算法，并作为EJBCA的一个组件添加到EJBCA中，使得密钥对用自己的加密算法产生，同时也可以按照X509协议，开发自己的证书和证书列表等。另外，随着密码学的发展，可能会有更好的加密算法出现，根据EJBCA基于组件的体系，可以很容易将新算法集成到新的组件中，并把它嵌入到EJBCA体系中。EJBCA装配简单、灵活、易于管理，可以通过它建立的CA方便地管理网络的安全，EJBCA是一个很有价值的开源系统。

3、EJBCA框架

EJBCA系统模型包括注册机构RA、CA服务器、LDAP目录服务器(可选)和数据库服务器等。如图4-2所示。

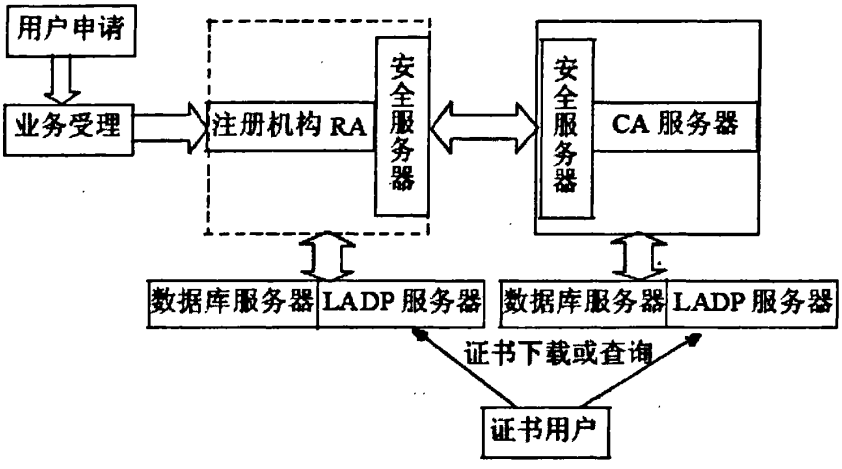


图4-2 EJBCA框架

安全服务器面向普通用户，用于提供证书申请、浏览、证书撤消列表以及证书下载等安全服务。安全服务器与用户的通信采取安全信道方式(如SSL的方式)。用户首先得到安全服务器的证书(该证书可由EJBCA中的CA颁发)，然后用户与服务器之间的所有通信，包括用户填写的申请信息以及浏览器生成的公钥均以安全服务器的密钥进行加密传输，只有安全服务器利用自己的私钥解密才能得到明文，这样可以防止其他人通过窃听得到明文，从而保证了证书申请和传输过程中的信息安全性。

CA服务器：EJBCA的CA服务器是整个证书机构的核心，EJBCA的CA服务器可建立根CA密钥对、存储证书、发布证书列表、建立子CA、发送证书请求、负责证书的签发等功能。建立CA服务器首先产生自身的私钥和公钥(密钥长度至少为1024 位)，然后生成数字证书，并且将数字证书传输给安全服务器。CA还负责为操作员、安全服务器、子CA及注册机构服务器生成数字证书。安全服务器的数字证书和私钥也需要传输给安全服务器。CA服务器是整个结构中最为重要的部分。

注册机构RA：登记中心服务器面向登记中心操作员，在EJBCA体系中，用户注册操作简单，RA管理员可直接通过命令行注册，另外还通过浏览器方便地管理注册用户的添加、删除、显示、注销等操作。RA在CA体系结构中起承上启下的作用，一方面向CA转发安全服务器传输过来的证书申请请求，另一方面向LDAP服务器和安全服务器转发CA颁发的数字证书和证书撤销列表。

LDAP服务器(可选)：LDAP服务器提供目录浏览服务，负责将注册机构服务器传输过来的用户信息以及数字证书加入到服务器上。这样其他用户通过访问LDAP服务器就能够得到其他用户的数字证书。在EJBCA体系里，LDAP服务器的配置是可选的，通过在配置文件的修改可使证书、证书列表放在相匹配的LDAP服务器上。

数据库服务器：数据库服务器是认证机构中的重要部分，用于认证机构中数据(如密钥和用户信息等)和日志统计信息的存储和管理。

## 4.3 FreeRADIUS 版本选择

FreeRADIUS[17]是应用最广泛的 RADIUS 服务器，具有运行快速、可扩展性高、可配置性好的特点，支持的协议超过了大多数商业服务器，支持包括 SQL、LDAP、RADIUS 代理、负载均衡和几乎 100 家厂商的字典文件。

FreeRADIUS 支持的数据库类型：Oracle、MySQL、PostgreSQL、Sybase、IBM DB2、MS SQLSERVER。

FreeRADIUS支持的认证类型：本地配置文件中的明文密码(PAP)、本地配置文件中的加密密码、CHAP、MS-CHAP、MS-CHAPv2、Windows域控制器认证、代理到其他RADIUS服务器、系统认证(通常通过/etc/passwd)、PAM(可插拔认证模块)、LDAP(只支持PAP)、CRAM、SIP Digest(Cisco VoIP，SER)、Netscape-MTA-MD5加密的密码、Kerberos 认证、X9.9认证环。

EAP的嵌入式认证方法：EAP-MD5、CISCO LEAP、EAP-MSCHAP-V2、EAP-GTC、EAP-SIM、EAP-TLS、EAP-TTLS、EAP-PEAP。

FreeRADIUS 2.0版本修补了一些错误，选用这个高版本会对我们使用EAP-TLS提供一些方便。

## 4.4 数据库选择 PostgreSQL

PostgreSQL[18]是由分布在全球的数百名开发者(包含非营利组织团体，学术研究机构及国际企业体)志愿贡献与共同开发的成果，历今 22 年来持续发展。长期以来被用于要求极端严谨的商业应用和科学研究环境以及政府组织中。

PostgreSQL 按照 BSD 版权协议发布，允许在商业和非商业应用的两种环境下均能享有自由取得而且不受限制的使用权。PostgreSQL 具有高度扩展性，且完全遵循

国际 ISO-SQL 规范的开发方向。

PostgreSQL 可以说是最富特色的对象-关系数据库管理系统(ORDBMS)，特性覆盖了 SQL-2/SQL-92 和 SQL-3/SQL-99，首先，它包括了可以说是目前世界上最丰富的数据类型的支持，其中有些数据类型连商业数据库都不具备，比如 IP 类型和几何类型等；其次，PostgreSQL 是全功能的自由软件数据库，很长时间以来，PostgreSQL 是唯一支持事务、子查询、多版本并行控制系统、数据完整性检查等特性的唯一的一种自由软件的数据库管理系统。

## 4.5 操作系统 Fedora Linux 7

Fedora[19] Linux 由 RedHat 支持开发和发布，尽管 Fedora 发行版的目的用户是桌面用户，但是由于 Linux 的特性，完全可以作为一般应用和开发阶段使用的服务器。而且，Fedora 是 RedHat 企业版服务器的前瞻版本，里面集中了很多企业版本即将拥有的特性。

# 第五章 设计中的问题和解决方法

## 5.1 EAP-TLS 认证的配置问题

检索已有文献和网络资源后，我们发现现有的FreeRADIUS EAP-TLS使用说明都不完整，一般的文档只说明了如何使用证书进行认证的一般配置，对于用户证书有效性识别的这块，没有进行很好的阐述，特别是用户证书撤销之后如何管理没有进行阐述。即使是FreeRADIUS文档，对这个方面也一带而过，把问题交给了用户。

国内能找到的文章，如王浩的《EAP-TLS on FreeRadius 中文版》[20]，也没有阐述如何进行配置：check_crl=no。大量搜索网络，关于CRL配置这块问题都很多，即使配置了CRL检测，很多FreeRADIUS也不能正常运行。这里面很多原因在于FreeRADIUS功能的不完善，另外也有配置不正确的原因。

既然有这个，就肯定有解决的办法，在阅读了最新的FreeRADIUS发行注解和配置文件之后，通过反复尝试，找到了解决问题的方法。具体解决方法见第六章系统的具体实现。

## 5.2 定时生成 CRL

我们需要FreeRADIUS定时更新CRL，那么，CRL从哪里来，怎么产生，怎么获得？这是这个系统能够连续运行的一个前提条件。首先是EJBCA如何产生CRL，并且可以不间断的生成CRL，另外，如何把CRL配置到FreeRADIUS中，这是下面需要解决的问题。

一个新建的CA总是会生成一个空的CRL，可以在CA创建的时候生成，或者运行EJBCA的程序'ca.sh createcrl caname'。使用EJBCA时，则可以通过设定一些参数，让服务器自动完成这个过程。

当在证书策略中一个CRL发布点和CRL签署者时，可以选择在证书策略中设定设定参数，或者在CA的配置中设定参数。通过在CA设置中设定参数，可以对所有的子CA进行配置，或者，也可以对每一个子CA都来设定CRL发布点证书配置。

在CA的一般配置中，有三个途径来指定CRL的生存周期。

(1) CRL超时周期，以小时为单位，这个是强制标志，指定CRL的有效期。如果设定为24小时，那么，下次更新CRL时，会自动加上24小时。

(2) CRL签署间隔时间，以小时为单位，这是个可选的标志。这个间隔时间是固定的，如果设定了1小时，即使旧的CRL仍然在后面24小时有效，新的CRL仍旧会被签署。这里默认是0，意思是当旧的CRL过期时，签署新的CRL。保持这个值为0，和CRL超时周期效果是一样的。

(3) CRL交迭时间，以分钟为单位，是个可选标记。在检查CRL是否需要被重新生成的时候，如果旧的CRL即将过期，那么就会生成新的CRL，这个值默认是10分钟。比如，一个CRL如果在24小时后过期，那么在23小时50分之后，新的CRL会被自动签署。这个特性保证合法的CRL签署没有时间延迟，甚至是几秒钟。这也给客户端在旧证书过期之前能够有一定时间去下载新的CRL。

EJBCA可以通过最少两条途径来间歇性的创建更新的CRL。

(1) CRL更新服务程序

在EJBCA中，有一个定时服务程序框架。在web管理界面中，选择‘Edit Services’并且添加一个服务，编辑服务选择‘CRL Updater’工作程序，并且填上间隔运行时间，然后设置这个服务为激活(Active)。然后这个服务就会每隔一段时间运行一次，根据每个子CA产生CRL。

(2) JBOSS服务程序

另一中方法就是通过生成和发布JBOSS CRL服务程序，在ejbca.properties中设置‘createcrl.service.enabled’这个选项为true之后，需要重新发布EJBCA这个应用到JBOSS，并且重新启动JBOSS。这个服务会每隔几分钟运行一次去检测是否CRL需要重新生成，运行时间可以通过修改src/appserver/jboss/crlcreate-service.xml中‘Polltime’来实现。一个新的CRL在轮询时间加上10分钟之内就会生成，所以总是会在旧的CRL过期之前就完成。最差的情况下，也能在旧CRL到期之前获得新的CRL。

(3) 使用Cron定时作业

Cron仅在类Unix系统中能够使用。把‘bin/ejbca.sh ca createcrl’加入到cron的作业任务中，createcrl命令会检查所有的有效子CA，有需要就会去更新各自的CRL。如果要强制某一个CA的CRL更新，可以使用‘bin/ejbca.sh ca createcrl *caname*’。可参考cron设置：

```
PATH=$PATH:/usr/local/java/bin
@daily cd /usr/local/ejbca; /usr/local/ejbca/ca.sh createcrl;
```

## 5.3 自动更新 FreeRADIUS 的 CRL 证书

在信任一个证书之前，一定要确认这个证书是合法的。通常，需要执行4个步骤来检验证书是否合法，这4个步骤如下：

(1) 校验证书的签名，并且核实这个证书是否有信任的证书机构授权；

(2) 检查证书在当前日期是有效的，在它的有效日期中使用；

(3) 检查证书撤销列表CRL，确定证书没有被它的签署机构撤销；

(4) 检查证书提供的凭证，根据应用系统，执行额外的审核，根据访问控制列表(ACL)或者在线证书状态处理(OCSP-Online Certificate Status Processing)。

FreeRADIUS中校验CRL的代码片段如下：

```
/*
 * Check the certificates for revocation.
 */
#ifdef X509_V_FLAG_CRL_CHECK
    if (conf->check_crl) {
        certstore = SSL_CTX_get_cert_store(ctx);
        if (certstore == NULL) {
            radlog(L_ERR, "rlm_eap: SSL error %s",
                ERR_error_string(ERR_get_error(), NULL));
            radlog(L_ERR, "rlm_eap_tls: Error reading Certificate Store");
            return NULL;
        }
        X509_STORE_set_flags(certstore, X509_V_FLAG_CRL_CHECK);
    }
#endif
```

X509_STORE_set_flags这个函数在openssl中定义，并且也是调用了其他函数：

```
int X509_STORE_set_flags(X509_STORE *ctx, unsigned long flags)
{
    return X509_VERIFY_PARAM_set_flags(ctx->param, flags);
}

int X509_VERIFY_PARAM_set_flags(X509_VERIFY_PARAM *param,
unsigned long flags)
{
    param->flags |= flags;
    if (flags & X509_V_FLAG_POLICY_MASK)
        param->flags |= X509_V_FLAG_POLICY_CHECK;
    return 1;
}
```

函数X509_VERIFY_PARAM_set_flags的实现又使用OpenSSL中定义的参数，这样就使得直接修改FreeRADIUS代码，从数据库实时获取CRL变得复杂，需要借鉴相当多的OpenSSL代码，所以，放弃修改FreeRADIUS的想法，另外寻找解决方法。

从上面EJBCA生成CRL获得启发，可以编写一个程序，并且定时执行，读取数据库中的CRL，然后覆盖写入FreeRADIUS的证书撤销列表crl.pem。

在EJBCA的使用中，已经可以定期的生成新的CRL，而且这个CRL是保存在数据库中的，格式为X509。这个格式和FreeRADIUS使用的格式一致，所以需要做的就是从PostgreSQL中读取数据，并更新到FreeRADIUS的证书目录中。出于方便，这里使用java编写，实例代码如下：

```
import java.io.*;
import java.lang.*;
import java.sql.*;

public class UpdateCRL
{
    public static void main(String[] args)
    throws Exception
    {
```

```
        String url    = "jdbc:postgresql://localhost:5432/ejbca";
        String user   = "ejbca";
        String passwd = "ejbca";
        Class.forName("org.postgresql.Driver");
        Connection conn = DriverManager.getConnection(url, user, passwd);

        conn.setAutoCommit(false);
        CallableStatement proc = conn.prepareCall("{ ? = call get_crl() }");
        proc.registerOutParameter(1, Types.LONGVARCHAR);
        proc.execute();
        String crl = proc.getString(1);
        conn.commit();

        proc.close();
        conn.close();

        // CRL写入crl.pem
        StringBuilder sb = new StringBuilder();
        sb.append("-----BEGIN X509 CRL-----\n");
        sb.append(crl);
        sb.append("\n");
        sb.append("-----END X509 CRL-----\n");

        String fileName = "/usr/local/etc/raddb/certs/crl.pem";
        File crlFile = new File(fileName);
        if(!crlFile.exists())
        {
            crlFile.createNewFile();
        }
        FileWriter fw=new FileWriter(fileName);
        fw.write(sb.toString());
        fw.close();
    }
}
```

程序中使用的存储过程如下：

```
CREATE OR REPLACE FUNCTION get_crl () RETURNS varchar
AS '
    declare crl varchar;
    begin
```

```
        select base64crl into crl from ejbca.crldata
            where crlnumber = (select max(crlnumber) from ejbca.crldata);
        return crl;
    end;
    ' LANGUAGE plpgsql;
```

编译上面的java程序，并且加入cron任务或者使用at任务，使这个程序定时运行以更新CRL。也可以改进程序，是程序一直运行，每运行一次就sleep一个指定的时间，在时间到达后再次运行，不停地更新。需要注意的是，在更新CRL之后需要重新启动一下FreeRADIUS，重启过程很快，几分钟或者几十分钟一次的频率重新启动，对性能影响不大，不会引起停止服务的结果，在一般的应用中，可以做为一种合理的方法。

# 第六章 系统的具体实现

## 6.1 EJBCA 和 PostgreSQL 的安装

1、软件环境：

Fedora 7 Linux

JDK1.6 (java.sun.com)

Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy

apache-ant-1.7 (www.apache.org)

EJBCA-3.5.1 (www.ejbca.org)

JBOSS-4.2.1 (www.jboss.com)

PostgreSQL JDBC驱动postgresql-8.2-506.jdbc4.jar (jdbc.postgresql.org)

2、解压缩文件：

解压缩JDK1.6到/usr/local/java

解压缩apache-ant到/usr/local/ant

解压缩JBOSS到/usr/local/jboss

解压缩EJBCA到/usr/local/ejbca

复制PostgreSQL JDBC驱动/usr/local/jboss/server/default/lib/

解压jce，把里面的jar包，覆盖/usr/local/java/jre/lib/security/ jar包

3、配置环境变量

```
export JAVA_HOME=/usr/local/java
export ANT_HOME=/usr/local/ant
export JBOSS_HOME=/usr/local/jboss
export PATH=$ANT_HOME/bin:$JBOSS_HOME/bin:$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar: \
    $JAVA_HOME/lib/tools.jar:$JAVA_HOME/jre/lib/rt.jar: \
```

```
$JBOSS_HOME/client/jbossall-client.jar:$JBOSS_HOME/client/jboss-j2ee.jar: \
$JBOSS_HOME/lib/jboss-system.jar
```

4、配置PostgreSQL数据库

```
psql template1 -U postgres
create database ejbca;                    -- sql
create user ejbca with password 'ejbca';  -- sql
grant all on database ejbca to ejbca;     -- sql
psql ejbca -U ejbca
create schema ejbca; -- sql，创建schema，否则在ejbca安装的时候会提示出错信息。
```

5、安装Ejbca

进入Ejbca的解压目录下，

```
cd /usr/local/ejbca
cp conf/ejbca.properties.sample conf/ejbca.properties
cp conf/database.properties.sample conf/database.properties
```

修改databas.properties文件

```
# ---------- Database configuration ----------
datasource.jndi-name=EjbcaDS                  # 数据库jndi配置
datasource.jndi-name-prefix=java:/
database.name=postgres                        # 数据库选择
datasource.mapping=PostgreSQL 8.0             # 数据源映射
database.url=jdbc:postgresql://127.0.0.1/ejbca  # 数据库链接URL
database.driver=org.postgresql.Driver         # JDBC驱动名称
database.username=ejbca                       # 数据库用户名
database.password=ejbca                       # 数据库登录密码
```

执行ant bootstrap，等待结束，启动jboss，能看到JBOSS正在启动所有东西，并且

发布ear应用，初始化数据库，如果没有报错则表明安装成功。

执行ant install，这个过程会生成所有的证书和私钥。完成之后在${ejbca.home}/p12目录中会发现一个管理使用的证书superadmin.p12，千万不要删除这些文件。ant install只能运行一次，当CA安装过以后，会在数据库中生成很多东西，所以不能再一次运行，否则会给你提示错误信息。

亭止JBOSS，执行ant deploy。重新发布应用，配置servlet运行的证书库，如果要使用JBOSS特定的服务来自动创建CRL，需要在ejbca.properties中打开这个选项：

createcrl.service.enabled=true。

## 6.2 FreeRADIUS 安装配置

1、要使用 EAP-TLS 认证，首先系统要安装 OPENSSL 的开发库，确保 openssl-devel 已经安装进系统。编译时，不需要手工添加参数，编译时会自动搜索 openssl 开发库，只需要在 FreeRADIUS 目录下运行 configure 就可以了。

./configure --prefix=/usr/local

2、配置也很简单，只需要修改很少的地方，radiusd.conf 中确保没有被注释掉，这句配置打开 FreeRADIUS 的 eap 认证功能。

$INCLUDE ${confdir}/eap.conf

在 eap.conf 中有关 tls 的配置

```
eap {
    default_eap_type = tls                          # 默认使用tls认证
    tls {
        certdir = ${raddbdir}/certs                 # 服务器证书目录
        cadir = ${raddbdir}/certs                   # CA证书目录
        private_key_password = freeradius           # 私钥解密密码
        private_key_file = ${certdir}/server.pem    # 私钥文件
        certificate_file = ${certdir}/server.pem    # 如果私钥和证书在同一文
                                                    # 件中，私钥和证书必须是
```

```
                                                    # 同一文件名
        CA_file = ${cadir}/ca.pem                   # 信任的根CA
        dh_file = ${certdir}/dh
        random_file = ${certdir}/random
        CA_path= ${certdir}      # 要使用CRL检测，需要加入CA_path这个参数
        check_crl = yes                             # 打开CRL检测
        cipher_list = "DEFAULT"
    }
}
```

如果只使用 tls，可以把其他的认证方式都注释掉，如 PAP、CHAP、MSCHAP、TTLS、PEAP 等等。

在 client.conf 中，这里加入我们做测试使用的无线路由

```
client 192.168.1.1 {
    secret = freeradius    # 无线路由链接 FreeRADIUS 的密码
    shortname = wlan
}
```

## 6.3 无线路由的配置

在测试中，选用了小型家用无线路由 TP-LINK TL-WR541G 无线路由。具体配制如图 6-1。

在无线网络的基本参数里选择[开启安全选项]：

安全类型：[WPA/WPA2]

安全选项：[WPA]

加密方法：[自动选择]

Radius 服务器 IP：[192.168.1.2]

Radius 端口：[1812]

Radius 密码：[freeradius]

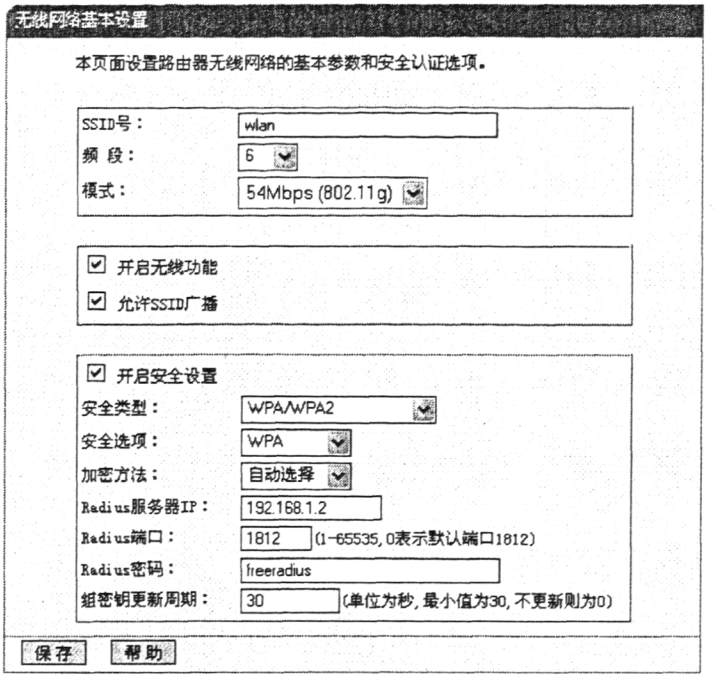


图 6-1 TL-WR541G 无线路由的 WPA 认证设置

## 6.4 生成证书

启动 JBOSS 服务器，把 superadmin.p12 导入浏览器，输入 https://localhost:8443/ejbca/ 可以访问 EJBCA 的管理页面，或者输入 http://localhost:8080/ejbca可以访问EJBCA的公共页面。

1、创建服务器证书

在EJBCA的管理界面中https://localhost:8443/ejbca/adminweb/，创建一个新用户配置，CA Functions->Edit Certification Profile，这个用户配置用途需具备如下两个特性，这将被用于服务器证书的授权。

KeyUsage: Digital signature, Key encipherment

Extended key usage: Server Authentication

创建服务器证书，登录https://localhost:8443/ejbca/，选择RA Functions -> Add End Entity，Username 选择 server，服务器证书的 Distinguished name (DN) 中，CommonName(CN)栏目可以为服务器的域名全称，如：

"C=CN,O=IT,CN=myca.org"。

Certificate Profile证书配置文件为刚才创建的用于服务器的配置。

然后登录https://localhost:8443/ejbca/，点击Create Server Certificate,输入用户名和密码，即可下载服务器证书，证书格式选择PEM。

2、创建用户证书

登录 https://localhost:8443/ejbca/，如上创建服务器证书过程，用户名选择 client，其他按照服务器证书生成过程，最后证书配置文件选择 ENDUSER，表示生成用户证书，证书格式选择 p12，因为我们的客户端使用 Windows。同样，生成一个 revoked 证书，仅被用于测试 CRL 是否有效，这个证书撤销之后可导出 CRL。

3、撤销用户证书

登录 https://localhost:8443/ejbca/adminweb/，选择 RA function->List/Edit End Entity，在使用用户名查找用户中输入用户 revoked，点击查找，在查找结果中选中这个用户，在 Revocation Reason 中随便选择一个撤销证书的理由，点击 Revoked Selected。这样用户证书就撤销了。

4、导出证书

上面的步骤做完后，最后导出根证书和 CRL，登录 https://localhost:8443/ejbca/，选择 Export CA 和 Export CRL，导出 CA 根证书为 PEM 格式的 ca.pem 和 Internet Explorer 格式的 ca.crt，CRL 证书导出为 crl.pem，用户证书导出成 p12 格式。

## 6.5 FreeRADIUS 服务器证书和 CRL 证书的安装

把之前导出的 ca.pem 和 server.pem 复制到 FreeRADIUS 证书目录就可以了。

```
cp ca.pem /usr/local/etc/raddb/certs/
cp server.pem /usr/local/etc/raddb/certs/
cp crl.pem /usr/local/etc/raddb/certs/
```

安装 CRL 证书后这里有一个我们需要的工具程序 c_rehash，包含在 openssl-perl 这个程序包里头，使用 yum install openssl-perl 来安装。c_rehash 是一个 perl 脚本，它扫描一个目录下的所有文件，并对其 hash 值添加符号链接。

c_rehash /usr/local/etc/raddb/certs/

## 6.6 客户端证书的安装

Windows XP 已经附带了 802.1x 客户端，所以不需要另外安装什么客户端程序。只要直接安装配置证书就可以。

从服务器上下载刚才导出的两个证书 ca.crt 和 client.p12，把这两个证书安装到客户端的证书管理器中。

(1) 安装 CA 证书，双击 ca.crt，会看到提示说明这个 CA 根证书不被信任，如要信任这个证书，把它安装到受信任的根证书库。

点击安装证书，接下来把证书放到 Trusted Root Certification Authorities，最后会提示 Windows 无法确认证书的合法性，忽略这个提示，确定安装就可以了。

(2) 安装 client.p12 证书，双击 client.p12，在提示为私钥输入密码的时候，输入证书导出时候用的密码，在下一步把这个证书放到 Personal 这个证书库，最后确认安装。

(3) 最后再查看一下我们刚才安装的证书：控制面板->网络选项->内容->证书。可以在 Personal 库里头看到 client 这个证书，双击能看到这个证书详细情况，最后证书路径在 myca.org 下面。

(4) 无线网络属性中新建一个网络，SSID 为 wlan，在认证选项里选择允许为这个网络使用 802.1x 认证方式，点击属性，继续选择使用这台电脑上的证书，选中校验服务器证书，在 Trusted Root Certification Authorities 选择 myca，最后确认即可完成配置。

## 6.7 测试结果

1、使用我们刚才安装的正常的 client 证书，没有被撤销的证书时，我们能看到 Access-Accept，客户端认证通过，被接收进入网络，在无线路由的客户端状态可以看到该终端的状态为连接(WPA)。

FreeRADIUS 调试结果输出片断如下：

auth: type "EAP"

+- entering group authenticate

rlm_eap: Request found, released from the list

rlm_eap: EAP/tls

rlm_eap: processing type tls

rlm_eap_tls: Authenticate

rlm_eap_tls: processing TLS

rlm_eap_tls: Received EAP-TLS ACK message

rlm_eap_tls: ack handshake is finished

eaptls_verify returned 3

eaptls_process returned 3

rlm_eap: Freeing handler

++[eap] returns ok

Sending **Access-Accept** of id 5 to 192.168.1.1 port 1060

MS-MPPE-Recv-Key =
0xe42e384404eabae85a2cf79654b3d6f78dcb6bc66e6560469825c1321d60ea48

MS-MPPE-Send-Key =
0x08182823b25c6307612699dfa35c21f485bd3995537601e5e5579b21b1732af6

EAP-Message = 0x03050004

Message-Authenticator = 0x00000000000000000000000000000000

User-Name = "client"

2、删除刚才安装的 client 证书，按照上面安装客户端证书的步骤，安装一个在服务器端已经被撤销的证书，双击 revoked.p12 按照提示安装。最后使用这个被撤销的证书在认证的时候，会发现 FreeRADIUS 提示证书被撤销，认证不会被通过。

FreeRADIUS 调试结果输出片断如下：

auth: type "EAP"

+- entering group authenticate

rlm_eap: Request found, released from the list

rlm_eap: EAP/tls

rlm_eap: processing type tls

rlm_eap_tls: Authenticate

rlm_eap_tls: processing TLS

rlm_eap_tls: Length Included

eaptls_verify returned 11

rlm_eap_tls: <<< TLS 1.0 Handshake [length 03b5], Certificate --> verify error:num=23:**certificate revoked**

rlm_eap_tls: >>> TLS 1.0 Alert [length 0002], fatal **certificate_revoked**

TLS Alert write:fatal:**certificate revoked**

TLS_accept:error in SSLv3 read client certificate B

rlm_eap: SSL error error:140890B2:SSL routines:SSL3_GET_CLIENT_CERTIFICATE:no certificate returned

rlm_eap_tls: SSL_read failed in a system call (-1), TLS session fails.

eaptls_process returned 13

rlm_eap: Freeing handler

++[eap] returns reject

auth: Failed to validate the user.

Found Post-Auth-Type Reject

+- entering group REJECT

expand: %{User-Name} -> revoked

attr_filter: Matched entry DEFAULT at line 11

++[attr_filter.access_reject] returns updated

Delaying reject of request 13 for 1 seconds

Going to the next request

Sending delayed reject for request 13

Sending **Access-Reject** of id 12 to 192.168.1.1 port 1060

EAP-Message = 0x040c0004

Message-Authenticator = 0x00000000000000000000000000000000

至此，使用EAP-TLS的FreeRADIUS配置已经完成，测试结果跟预期的一样，能

够对证书进行正确识别并给与相应的授权，让合法的证书用户获得认证通过，而不合法的证书用户则认证失败，不能获得合法的网络访问授权。

# 第七章 总结和展望

## 7.1 总结

在本文中实现的是基于802.1x协议的接入认证方法，使用安全的EAP-TLS方式。并且，结合EAP-TLS认证，探讨了CA证书管理中心的配套应用，结合两者应用，总结出适合低成本部署的PKI用户认证方式。

实现过程中，我们采用Linux服务器，使用基于J2EE的CA管理软件EJBCA，部署相对简单的CA管理构架。同时，和FreeRADIUS共同应用，来达到使用证书安全认证的目的。

在本文中，基本目的已经达到，经过实践，利用现有的开放源代码软件，有效的进行组合，并且编写少量工具程序，促成这个系统协同工作，组成了一个有效的FreeRADIUS认证解决方案。

最后，在解决了使用EAP-TLS认证的一般框架下，本文介绍性的引入了USB Key的使用，这是一种更加安全的证书使用方式，解决了文本证书带来的可复制问题，尽管本文没有最终实现这个目标，但是，USB Key已经在银行、金融等领域广泛使用，随着成本的降低，一定会被其他行业所接受。

## 7.2 展望

当然EAP-TLS认证方式并不完美，有着很大的缺点，由于认证过程中使用的是高强度的RSA算法，在大规模应用中，会因为大量加密解密计算导致FreeRADIUS服务器压力大增。解决这个问题的办法，可以使用运算能力更高的服务器，或者采用分布式认证方式，把用户认证分散到不同的服务器，这个环节所需要做的，也仅仅是让分布在不同地区的FreeRADIUS服务器保持相同的CRL。由于目前采购计算机硬件代价相对远低于软件开发代价，要实现分布式的应用，使用更多硬件变得相对容易些，因此，对今后部署EAP-TLS认证提供了有力的支持。

# 参考文献

[1] IEEE Std 802.1x-2001, "Port-Based Network Access Control", IEEE-SAStandards Board 2001

[2] 802.1x认证+无线AP, http://freebsd.ntut.idv.tw/document/freebsd_wireless_802.1x_freeradius.html

[3] 张志峰，EAP-TTLS认证方式在WLAN中的应用研究，武汉理工大学，2006

[4] 刘海慧，PKI在校园网中的应用研究，山东大学硕士论文，2006

[5] Par Thus0, SPIP/TinyCA-et-Freeradius-en-mode-EAP, http://www.pervasive-network.org/SPIP/TinyCA-et-Freeradius-en-mode-EAP, 2005

[6] Lee Barken, How Secure Is Your Wireless Network? Safeguarding Your Wi-Fi LAN, Prentice HallPub, August 26, 2003

[7] IEEE 802.1x技术白皮书v1.0，北京港湾网络有限公司产品部，2002年2月

[8] Andrew Nash & William Duane 等著，张玉清、陈建奇等译《公钥基础设施(PKI)实现和管理电子安全》，清华大学出版社，2002

[9] Carlisle Adams & Steve Lloyd, Understanding PKI: Concepts, Standards, and Deployment Considerations, Second Edition, Addison Wesley Professional, November 06, 2002

[10] Network Working Group, PPP EAP TLS Authentication Protocol, October 1999

[11] Timothy M. Jurgensen; & Scott B. Guthery, Smart Cards: The Developer's Toolkit, Prentice Hall, July 09, 2002

[12] 王爱英，智能卡技术，清华大学出版社，2000

[13] EJBCA, http://www.ejbca.org

[14] Hard Token Management Framework, http://www.hardtokenmgmt.org

[15] Shing Wai Chan, Key Management and PKCS#11 Tokens in Sun Java System Application Server 8.1, http://java.sun.com, May 19, 2005

[16] Java PKCS#11 Reference Guide, http://java.sun.com, 11 May 2004

[17] FreeRADIUS, http://www.freeradius.org

[18] PostgreSQL, http://www.postgresql.org

[19] Fedora, http://fedora.redhat.com

[20] 王浩, EAP-TLS on FreeRadius 中文版，2006，
http://blog.chinaunix.net/u/12/showart_22148.html

[21] 陆金山，基于 RADIUS 协议的校园网络接入服务器的研究和实现，
合肥工业大学，2006

[22] 夏谦，TLS 协议分析研究与软件实现，电子科技大学计算机学院，2001

[23] C.Adams & S.Lloyd 著，冯登国等译《公开密钥基础设施—概念、标准和实施》，
人民邮电出版社，2000

[24] 李兴国，基于 8021X 的宽带网认证计费系统设计与实现，
电子科技大学硕士论文，2004

[25] 吴世忠，应用密码学，机械工业出版社，2000

[26] 卢开澄，计算机密码学，北京清华大学出版社，1990

[27] OpenSSL project, http://www.openssl.org

[28] 李卫著，计算机网络安全与管理[M]，清华大学出版社，2004

[29] 高晓琦，802.1x 协议分析及 Windows 平台下客户端设计，
武汉大学硕士论文，2004

[30] 马建峰，朱建明编著，无线局域网安全—方法与技术，机械工业出版社，2005

[31] 江林，无线局域网安全与认证的研究和公用 WLAN 应用，
西安电子科技出版社，2003

# 攻读硕士学位期间发表的学术论文

[1] 杨凌凤，使用 USB Key 提高 FreeRadius 证书认证的安全性，计算机安全，已录用
[2] 杨凌凤，FreeRADIUS EAP-TLS 应用实践，电脑与电信，已录用

# 致谢

本文从选题到完成的整个过程中始终得到了导师朱巧明教授的悉心指导，他渊博的知识、严谨的治学态度、独到的见解，以及诲人不倦的精神都给了我莫大的帮助。在此献上我对朱老师最诚挚的谢意。

我还要感谢罗喜召老师，在本文的完成过程中，感谢他对本课题给予的一些看法和意见。

同时我还要感谢在苏大读书期间所有的任课老师所给予的帮助，感谢你们让我顺利完成了学业。

在此，我还要感谢我的同事在论文完成期间给予的支持和帮助，感谢我的师父给予的无私的指导和帮助。

最后要特别感谢我的家人，是你们的支持和鼓励才让我能顺利走下去。