南昌航空大学

硕士学位论文

基于公钥的Kerberos身份认证系统的设计与实现

姓名：吴喜兰

申请学位级别：硕士

专业：计算机应用技术

指导教师：舒远仲

20080601

摘 要

在当前分布式网络环境中，Kerberos 是应用最广泛的身份认证协议。Kerberos 是一种基于可信赖第三方的认证协议，使用对称密钥加密算法 DES 实现 KDC 的认证服务，提供了网络通信方之间相互的身份认证，在一定程度上保证了网络的安全。

由于 Kerberos 存在口令猜测、重放攻击、时间同步和密钥存储等方面的安全缺陷，近年来许多研究者提出了 Kerberos 与公钥系统结合的 Kerberos 公钥扩展方案，部分改善了 Kerberos 系统的安全性能，但是并没有结合 PKI 来管理证书和撤销列表，在系统性能瓶颈上的处理也还存在不足。

本文阐述了 Kerberos 协议、PKI 技术以及 Kerberos 的几种公钥扩展协议的认证方式，分析了 Kerberos 协议存在的几个问题。针对 Kerberos 协议存在的问题，通过把 PKI 技术、访问控制与 Kerberos 协议相结合，提出了一种基于公钥的 Kerberos 改进协议，阐述了其在认证服务交换、票据许可服务交换及客户端与应用服务器交换的思想。经与原 Kerberos 协议在性能方面的对比分析表明，在不改变其原有框架的前提下，采用了双因子认证方式的改进协议在密钥存储、时钟同步、口令猜测、票据的不可否认性等多项性能方面加强了 Kerberos 的安全性。

在该改进协议的基础上，建立了一个安全高效、易于扩展和具有实用性的身份认证系统模型，模型包括客户端、服务器端和证书管理中心三个部分。文章还阐述了认证模型的总体设计方案，认证系统的流程，并对认证系统模型的模块结构、功能模块以及在应用系统中的集成应用进行了设计和实现。

关键词：Kerberos，PKI，身份认证

ABSTRACT

Kerberos is the most popular protocol that used for identity authentication in distributed network environment. Kerberos is an authentication protocol based on trusted third-party and symmetry key cryptography. It provides the method for the two parties of the network commucations to authenticate each identity, and ensure the network security in a certain extent.

The limitations of Kerberos include password guessing attacks, replaying attacks, the synchronization of clock and memory of secret key. Now many researchers have given some extent scheme which integrates the Kerberos and PKI. Those schemes partly enhance the Kerberos system's security, but manage the certificates and CRL without using the PKI. So the problem of bottlenecks still exists.

The paper analyses the Kerberos protocol, PKI technology and some authentication method integrating public key cryprography in Kerberos, and picks out some limitations of Kerberos protocol. Just as those limitations, an improved Kerberos protocol with public key, which is over the PKI technology, access control and Kerberos protocol, is introduced and the idea about Authentication Service Exchange, Ticket Granting Exchange and Client/Server Exchange is given in this paper. Without changing its former frame, the new protocol enhanced its security performance. Compared with the KerberosV5，the improved protocol with double factors authentication technology has some improvement in many aspects, including memory of secret key, the synchronization of clock, password guessing attacks and the non-repudiation ticket.

A new model of authentication system is designed based on the improved Kerberos protocol. It has higher security and practicability, which is easier to extend,and consists of client、server and certificate management center. The paper analyses the whole scheme of the authentication model, the process of authentication system, meanwhile designed and realized the model structure of authentication model and the integrate application in system.

Keywords：Kerberos, PKI, Identity Authentication

第 1 章 绪论

1.1 研究背景

近年来，网络正以惊人的速度改变着人们的工作和生活方式，从机构到个人都越来越多地通过网络来发送邮件、互换资料及订购产品，这无疑给社会、企业乃至个人带来了前所未有的便利。然而，由于计算机网络的开放性、互联性、广义的多样性和终端分别不均匀性，致使网络信息系统易受黑客和病毒等不轨行为攻击。所以，设置网络信息安全和保密是十分重要的事情[1]。

身份认证技术是能够对信息收发方进行真实身份鉴别的技术，是保护网络信息资源安全的第一道大门，也是最重要的一道防线。尤其在互不谋面的网络中，用户身份一旦被冒充，不仅可能损害用户自身的利益，也可能损害其他用户和整个系统。网络中的各种应用和计算机系统都需要通过身份认证来确认用户的合法性，然后确定这个用户的个人数据和特定权限。身份认证，其实质是对参与通信的实体的身份加以鉴别和确认，从而证实是否名副其实或者有效的过程，目的是验证通信双方的真实身份，防止非法用户假冒合法用户窃取机密信息。

1.2 身份认证技术发展状况

身份认证技术从是否使用硬件来看可以分为软件认证和硬件认证，从认证需要验证的条件来看可以分为单因子认证和双因子认证，从认证信息来看可以分为静态认证和动态认证。当今常用的身份认证方式主要有以下几种：基于口令的认证方式、基于物理证件的认证方式、基于动态口令的认证方式和基于生物特征的认证方式。

(1)基于口令的认证方式[2]

基于口令的认证方式采用“用户所知”的认证方法，是最简单、最基本也是最常用的方法，系统事先保存每个用户的用户名和密码。用户进入系统时输入用户名和密码，系统根据保存的用户信息和用户输入的信息相比较，来判断用户身份的合法性。但是这种基于口令的认证方式是最原始、最不安全的身份确认方式，

非常容易泄露，非法用户可以通过口令猜测、线路窃听、重放攻击等手段来仿冒合法用户身份。

(2)基于物理证件的认证方式

基于物理证件的认证方式是一种基于“ 用户所有 ”的认证方式，可以实现“ 用户所知 ”和“ 用户所有 ”双因子身份认证。目前，主要的物理证件有智能卡和 USB Key 等。

a.基于智能卡的认证方式

智能卡具有嵌入卡片内部的 CPU 和存储器，同时还有一系列的安全机制来保证内部数据的安全。利用智能卡上 CPU 的计算能力，可以在卡上进行密钥对的生成和进行卡上的签名和验证运算；同时，利用智能卡出色的安全机制，能够对存储在其中的数据提供强有力的安全保证，这样在用户私钥的整个生命周期内，都处在智能卡的保护之下；而且，智能卡还具有方便的可携带性和灵活性。

b.基于 USB Key 的认证方式

基于智能卡的认证方式，具有较高的安全性。可是普通的智能卡的使用，需要智能卡读卡器，严重影响了其使用的方便性。基于 USB Key 的身份认证方式是一种方便、安全、经济的身份认证技术，它采用了软硬件相结合的一次一密的强双因子认证模式，很好地解决了安全性与易用性之间的矛盾。通过 USB Key 内置的智能卡芯片，可以存储用户的密钥或数字证书，利用内置的密码学算法实现对用户身份的认证。在本认证系统中应用 USB Key 技术来存储用户的私钥，使得系统的安全性得到了很大的提高。

(3)基于动态口令的认证方式

动态口令认证也称为一次性口令（One-Time Password）认证[3]。动态口令技术是一种让用户的密码按照时间或使用次数不断动态变化，每个密码只使用一次的技术，采用一次一密的方法，有效地保证了用户身份的安全性。由于采用了口令动态生成的方式，用户每次使用的密码都不相同，即使非法用户截获了一次密码，也无法利用这个密码来仿冒合法用户的身份，因此可以解决基于口令认证中的不能抵御口令猜测攻击的问题。但是动态口令认证需要客户端与服务器端保持时间或次数的良好同步，否则就可能发生合法用户无法登录的问题。

(4)基于生物特征的认证方式

基于生物特征的认证方式[4]以人体惟一的、可靠的、稳定的生物特征（如指纹、虹膜、脸部、掌纹等）为依据，采用计算机的强大功能和网络技术进行图像处理和模式识别。该技术具有很好的安全性、可靠性和有效性。近几年来，全球的生物识别技术取得了较大的发展，已从研究阶段转向应用阶段，前景十分广阔。

以密码学技术为基础的身份认证协议需要具有足够的强度来抵御常见的网络

攻击手段。协议规定了通信双方之间为了进行身份认证所需要进行交换的信息格式和次序。目前主要有以下几种身份认证协议：一次一密机制、Kerberos 认证协议和公钥认证体系。

1. 一次一密机制

一次一密机制主要有两种实现方式。第一种采用挑战/应答方式(challenge/response)，用户登录时系统随机提示一条信息，用户根据这一信息连同其个人数据共同产生一个口令字，用户输入这个口令字，完成一次登录过程，或者用户对这一条信息进行数字签名并发送给认证服务器(AS)进行鉴别。第二种方法采用时钟同步机制，即根据这个同步时钟信息连同其个人数据共同产生一个口令字。这两种方案均需要应用服务器端产生与用户端相同的口令字(或检验用户签名)用于验证用户身份。

2. Kerberos 认证协议[5]

Kerberos 身份认证协议是目前应用广泛，也相对较为成熟的一种身份认证机制[6]。Kerberos 是一种基于可信赖第三方的 TCP/IP 网络安全认证协议[7]，提供了一种在开放型网络中进行身份认证的方法，认证的实体可以是用户或应用服务，这种认证不依赖于主机的操作系统，也不依赖于主机的 IP 地址，不需要保证网络上所有主机的物理安全性，可以有效解决分布式网络环境下用户访问系统资源的安全性问题。

3. 公钥认证体系

公开密钥体系(Public Key Infrastructure)，就是利用公钥理论和技术建立的提供安全服务的基础设施。公钥认证体系[8]的原理是用户向认证机构提供用户所拥有的数字证书来实现用户的身份认证。数字证书是由可信赖的第三方-认证中心 CA 颁发的，含有用户的特征信息的数据文件，并包含认证中心的数字签名。在认证中心的私钥不被泄密的前提下，认证中心的数字签名可以确保数字证书不能被伪造和篡改，这样就可以通过对数字证书的验证来确认用户的身份。

1.3 Kerberos 研究现状

自 Kerberos 身份认证系统开发以来，Kerberos 系统就一直在 Unix 系统中被广泛地采用，常用的有两个版本[9]：第 4 版和第 5 版，其中第 5 版更正了第 4 版中的一些不足，并作为提议的 Internet 标准发布在 RFC 1510[10]中。

目前许多远程访问认证服务系统都支持 Kerberos 认证协议，如 Microsoft 的 ERP(可扩展认证协议)、Cisco 的终端访问控制器访问控制系统(TACACS/XTACACS)以及远程认证拨号业务(RADIUS)等。当今主要的操作系统也

都支持 Kerberos 系统，例如 SUN Microsystems 公司的 Solaris、IBM 公司的 AIX、HP 公司的 HP-UX 等，Microsoft 公司在其推出的 Windows 2000 中也实现了这一认证系统，并作为它的默认的认证系统。此外 Linux Redhat 环境下也可以使用 Kerberos 提供的 Ktelneted，Krlogind，Krshd 来代替传统的 telnet，rlogind，rshd 服务。Java 安全解决方案中的 Java 认证和权限服务(JAAS)也提供了对 Kerberos 的支持。目前，Kerberos 协议已成为业界的标准网络身份验证协议。

在目前分布式网络计算环境中，Kerberos 协议应用最为广泛，但其局限性和缺陷也是比较明显的，主要有以下几个方面存在不足：口令猜测攻击、重放攻击、时间同步问题、密钥的存储问题、认证域之间的信任问题、系统程序的安全性与完整性问题、原有应用系统的修改问题等[11][12]。

由于 Kerberos 存在上述缺陷，国内外许多学者都对 Kerberos 协议进行了深入研究并提出了一些安全改进措施。Kerberos 的公钥扩展研究较多，人们提出了很多 Kerberos 的公钥扩展方案，IETF 的通用认证技术工作组提出了若干方案，如：

①利用公钥进行预认证 PKINIT[13]，是用户在初始认证步骤中使用公钥机制，采用公钥技术获取票据许可票据 TGT。Kerberos 基本认证协议相比 PKINIT 只对 Kerberos 前两条认证消息中用 PA 数据类型进行公钥信息交换，在实现上对 Kerberos 改动很少。

②利用公钥跨区域认证 PKCROSS[14]

PKINIT 能够很好的将公钥机制与 Kerberos 在初始认证阶段很好的结合，但是在区域间需要交叉认证处理的却不好。PKCROSS 指在 Kerberos 不同域中使用公钥密码体制进行安全协议的认证，采用公钥技术进行域间鉴别。

③基于公钥技术的分布式的鉴别技术 PKDA[15]

所有使用 Kerberos 服务的程序都依赖 KDC，如果 KDC 产生问题，或者处理大量请求时 KDC 可能是一个“瓶颈”。通过引入基于公钥技术的分布式的鉴别技术(PKDA)，可以有效地解决这个问题。PKDA 增强了客户端的“隐密”和 Kerberos 框架的使用范围和安全性。

除了公钥扩展之外，还有很多人提出了对 Kerberos 协议的改进方案，如参考文献[16]中提出在 Kerberos 中引入序列号循环机制，来解决时间同步问题；文献[17]中提出将 Smart Cards 集成到 Kerberos，来解决口令猜测攻击问题；文献[18]中提出一种高效的结合 X.509 和域名系统 DNS 的 Kerberos 认证机制，使用了两种不同的密钥管理系统，不对称和对称方法，使通信相对于 PKINIT 要更为简单。上述这些改进方案，改善了 Kerberos 某些方面的性能，但如何整体改善 Kerberos 安全性能仍须进一步研究。

1.4 论文工作

在本文撰写中，查阅了大量的文献，包括以下几个方面内容：身份认证技术、PKI 技术、Kerberos 协议原理与跨域认证、Kerberos 存在的问题和 Kerberos 的几种公钥扩展等。主要对 Kerberos 协议和其扩展协议作了研究。

本文的主要工作如下：

1. 认真分析和研究了Kerberos协议和它的几种公钥扩展协议，并阐述了其在安全方面存在的某些不足之处。

2. 把PKI技术[19]、访问控制[20]与Kerberos协议相结合，提出一个基于公钥密码体制的 Kerberos 改进协议，并对改进协议的性能进行了对比分析。

3. 针对该改进协议，建立了身份认证系统模型，阐述了认证模型的总体设计方案和认证系统的流程。对认证系统模型的模块结构、功能模块以及在应用系统的集成应用进行了设计和实现。

1.5 论文组织结构

论文的整体结构如下：

第 1 章 绪论

简述了本文的研究背景和意义，对目前常用的身份认证的方法和协议进行了分析，并介绍了本文的研究内容。

第 2 章 Kerberos 协议和 PKI 技术

阐述了 Kerberos 协议的认证过程和跨域认证，并提出存在的问题。对 PKI 技术进行了分析。

第 3 章 基于公钥的 Kerberos 改进协议

对 Kerberos 的几种公钥扩展协议进行了研究，并提出了对 Kerberos 的改进，并对改进协议的性能进行了分析和比较。

第 4 章 基于改进协议的认证系统的设计

阐述认证模型的总体设计方案，详细介绍认证系统的流程，并对客户端和服务器端做了详细的设计与说明。

第 5 章 基于改进协议的认证系统的实现

详细叙述了认证系统的开发平台和系统模块结构，对各个功能模块和在应用系统中的集成应用的实现做了详细的分析。

第 6 章 结论

总结本文，概括了文章的主要贡献，并对进一步的研究方向进行了展望。

第 2 章 Kerberos 协议和 PKI 技术

2.1 Kerberos 协议简介

Kerberos 是一项鉴别服务[21]，是麻省理工学院(MIT)为 Athena[22]项目开发的，它的名字取自一个希腊神话，原义为凶猛的 3 头狗，意喻该协议具有极其强大的安全性能。Kerberos 的总体方案是使用一个协议来提供可信的第三方鉴别服务，客户和服务器信任 Kerberos 能仲裁它们之间的相互鉴别。Kerberos 协议的设计目标是，通过网络通信的实体可以互相证明彼此的身份，可以抵抗旁听和重放等方式的攻击，而且还可以保证通信数据的完整性和保密性。Kerberos 认证过程的实现不需要依赖于主机操作系统的认证、主机地址的确认以及整个网络的安全保证。

Kerberos 的设计在安全上要求具有以下特点：

①安全：网络窃听者不能获得必要的信息来假扮成另一个用户。

②可靠：Kerberos 应该是高可靠性的，应该使用分布式的服务器结构，一个系统能够对另一个系统进行备份。

③透明：用户应该意识不到鉴别服务的发生。

④可扩展：系统应该拥有支持大量客户和服务器的能力。

2.2 Kerberos 协议

Kerberos 协议是一种三路处理方法[23]，根据称为 KDC（Key Distribution Center，密钥分发中心）的第三方服务来验证计算机相互的身份，并建立密钥以保证计算机间安全连接。KDC 由两个部件认证服务器(AS)和授权服务器(TGS)组成。Kerberos 认证系统总共包括三个服务器：认证服务器(AS)，用于在登录时验证用户的身份并发放票据许可票据(TGT)；授权服务器(TGS)，用于发放服务许可票据(TS)；应用服务器(V)，客户端请求工作的实际执行者。

2.2.1Kerberos 协议认证过程

Kerberos 的认证过程包括三个阶段[24]：认证服务交换、票据许可服务交换和客户机与应用服务器的认证交换，分别由三组消息交换来完成。

在讨论协议之前，需要定义一些符号和概念：

ID_c：用户名称；

IP_c：用户的 IP；

TS：时间戳；

$Realm_c$：表示用户 C 所在的 Kerberos 领域；

Options：用户要求在回送票据中的附加标志；

Times：有效期限，包括开始时间、终止时间、重新更新的时间；

Nounce：一个随机数，用以说明发给的信息是新的，而不是重发的；

Seq# ：服务器送信息给客户端的附加的序号，以防止重发攻击。

Kerberos 认证过程如下图 2-1 所示：

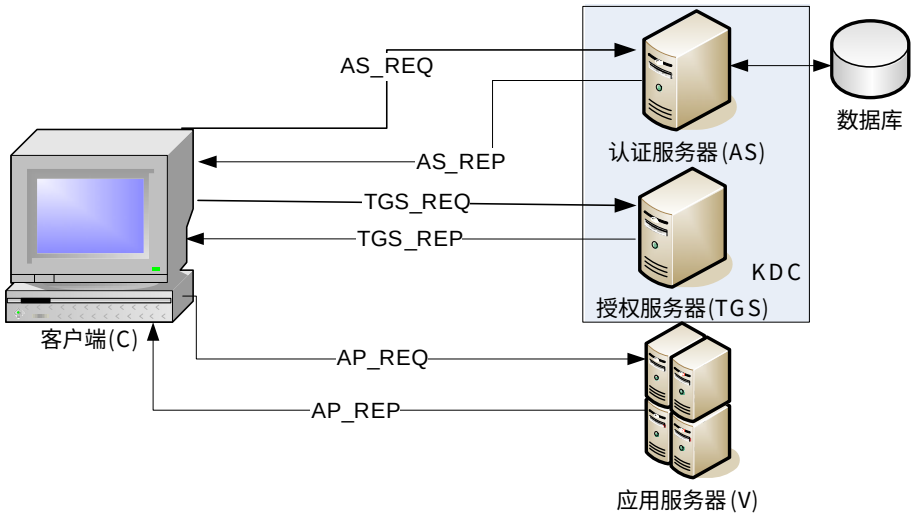

图 2-1 Kerberos 认证过程

(1)（认证服务交换）认证服务器(AS) 向用户发放票据许可票据(TGT)

C→AS：AS_REQ= $\{ID_c,\ ID_{tgs},\ Times,\ Nounce1,\ Realm_c,\ Options\}$

AS→C：AS_REP= $\{Realm_c, ID_c, Ticket_{tgs}, EK_c(K_{c,tgs}, Times, Nounce1, Realm_{tgs}, ID_{tgs})\}$

$Ticket_{tgs} = EK_{tgs}(Realm_c,\ ID_c,\ ID_{tgs},\ IP_c,\ Times,\ K_{c,tgs},\ Flags)$

客户端向认证服务器(AS)发送请求，请求获得票据许可票据 TGT。先以明文方式向 AS 发出请求，请求报文包括用户名称、TGS 名称、有效生存期限、随机数 1 和用户所在的 Kerberos 领域等信息。

AS 收到客户端的请求报文后，先根据用户名称在本地数据库中查找用户的密

钥 K_c，如果查找成功，则认证继续。AS 生成随机会话密钥 $K_{c,tgs}$，该会话密钥用于客户端和 TGS 之间的加密通信。然后 AS 产生用户请求的 TGS 的票据许可票据(TGT)，该票据包括用户名称、TGS 名称、用户 IP、随机数、有效生存期限以及会话密钥 $K_{c,tgs}$ 等，并用 TGS 的密钥 K_{tgs} 进行加密，以保证只有 TGS 才能解密。最后，AS 发送应答报文，该报文包括票据许可票据 TGT 和用户密钥 K_c 加密的信息。

(2)（票据许可服务交换）授权服务器(TGS) 向用户发放服务许可票据(TS)：

C→TGS：TGS_REQ= {ID_v，Times，Nounce2，$Ticket_{tgs}$，Authenticator1，Options}

Authenticator1 = $EK_{c,tgs}(ID_c，Realm_c，TS1)$

TGS→C：TGS_REP= {$Realm_c$，ID_c，$Ticket_v$，$EK_{c,tgs}(K_{c,v}$，Times，Nounce2，$Realm_v$，$ID_v)$}

$Ticket_v = EK_v(ID_c，ID_v，IP_c，Times，K_{c,v}，Realm_c，Flags)$

当客户端收到 AS 返回的应答报文后，使用密钥 K_c 进行解密，得到会话密钥 $K_{c,tgso}$。用户向授权服务器(TGS)发送访问应用服务器 V 的请求报文，请求获得服务许可票据(TS)。报文内容包括要访问的应用服务器 V 的名称、有效生存期限、随机数 2、TGT、认证符 1 等。认证符 1 用会话密钥 $K_{c,tgs}$ 进行加密，包括用户名称、用户 C 所在的 Kerberos 领域以及时间戳。

TGS 收到客户端发来的请求报文后，用自己的密钥 K_{tgs} 对票据 TGT 进行解密处理，该票据的含义为“使用 $K_{c,tgs}$ 的客户是 C”。 TGS 用从 TGT 中取出的会话密钥 $K_{c,tgs}$ 解密认证符 1，并将认证符 1 中的数据与 TGT 中的数据进行比较，从而可以相信 TGT 的发送者用户 C 就是 TGT 的实际持有者。TGS 验证用户的合法身份之后，生成随机会话密钥 $K_{c,v}$，该密钥用于客户端和应用服务器之间的加密通信。然后 TGS 产生用于访问应用服务器 V 的票据 TS，TS 包括用户名称、应用服务器名称、用户 IP、有效生存期限和会话密钥 $K_{c,v}$ 等，并用应用服务器 V 的密钥 K_v 来加密，以保证只有应用服务器 V 才能解得开。最后 TGS 发送应答报文，该报文包括服务许可票据 TS 和会话密钥 $K_{c,tgs}$ 加密的信息。

(3)（客户机与应用服务器的认证交换）用户向应用服务器获取服务：

C→V：AP_REQ= {Options，$Ticket_v$，Authenticator2}

Authenticator2 = $EK_{c,v}(ID_c，Realm_c，TS2，Subkey，Seq\#)$

V→C：AP_REP= {$EK_{c,v}(ID_c)$，$Realm_c$，TS2，Subkey，Seq# }

当客户端收到 TGS 的应答报文 TGS_REP 后，用会话密钥 $K_{c,tgs}$ 解密得到会话密钥 $K_{c,v}$。用户向应用服务器 V 发送请求报文 AP_REQ，报文内容包括应用服务器名称、用于访问应用服务器 V 的票据 TS 以及用 $K_{c,v}$ 加密的认证符 2。

应用服务器 V 收到客户端发来的请求报文 TGS_REP 后，用自己的密钥对票据 TS 进行解密，得到 $K_{c,v}$ 来解密认证符 2，并将解密后的认证符 2 中的数据与票

据 TS 中的数据进行比较，用户 C 的身份得到验证。应用服务器向客户端发送应答报文，包括用 $K_{c,v}$ 加密的用户名称等信息。当客户端收到信息后，解密信息并确认应用服务器的身份后，认证结束。

2.2.2 跨域认证

Kerberos 协议的认证模式分为域内认证和跨域认证两种[25]。上节中介绍的是域内认证的认证过程。

在互联网中可能存在多个组织结构，而所有用户都到一个 Kerberos 服务器上注册是不适宜的。这样在不同组织下的由客户和服务器组成的网络组成不同的领域。Kerberos 提供了不同域之间的认证机制，即跨域认证。这样，一个域中的用户就可以访问另一个域中的应用服务器。对于两个支持领域间鉴别的领域来说，需要共享一个域间密钥（inter-realm-key）和两个 Kerberos 服务器都必须相互注册。通过交换域间密钥，用户在本地域内的 Kerberos 服务器就可以获得远程 Kerberos 服务器的票据，持有该票据的用户就能够向远程 Kerberos 服务器申请所需的应用服务。

这个方案需要一个领域中的 Kerberos 服务器信任另一个领域中 Kerberos 服务器鉴别的用户，并且第二个领域中的参与服务器也愿意信任第一个领域中的 Kerberos 服务器。

一个用户要得到另一个领域内一服务器的服务，就需要获得那个服务器的许可票据。用户的客户程序遵循通常的过程来获得对本地 TGS 的访问，然后请求一个远程的 TGS（在另一领域内的 TGS）的票据许可票据。接着用户向远程 TGS 申请一张位于该远程 TGS 领域范围内特定服务器的服务许可票据[26]。向远程服务器出示的票据（V_{rem}）在该领域内的用户已经经过鉴别。服务器选择是否允许远程的请求。

跨域认证的认证过程如图 2-2 所示：

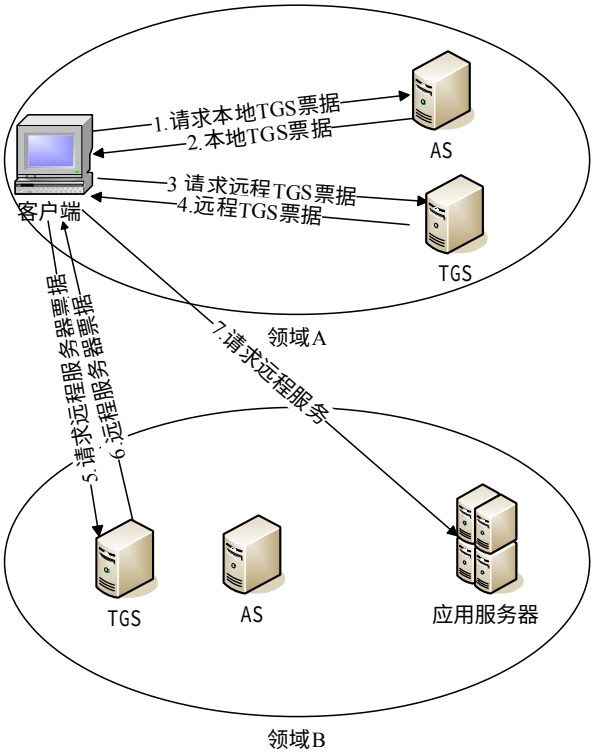

图 2-2 Kerberos 域间认证

交换细节如下：

(1)C→AS：{ID_c, ID_{tgs}, TS1}

(2)AS→C：{$EK_c(K_{c,tgs}$, ID_{tgs}, TS2, Times, $Ticket_{tgs})$}

(3)C→TGS：{ID_{tgsrem}, $Ticket_{tgs}$, Authenticatorc}

(4)TGS→C：{$EK_{c,tgs}(K_{c,tgsrem}$, ID_{tgsrem}, TS4, $Ticket_{tgsrem})$}

(5)C→TGS_{rem}：{ID_{vrem}, $Ticket_{tgsrem}$, Authenticatorc}

(6)TGS→C：{$EK_{c,tgsrem}(K_{c,vrem}$, ID_{vrem}, TSb, $Ticket_{vrem})$}

(7)C→V_{rem}：{$Ticket_{vrem}$, Authenticatorc}

2.3 Kerberos 协议存在的问题

Kerberos 协议主要在以下几个方面存在不足：口令猜测攻击、重放攻击、时间同步问题、密钥的存储问题、认证域之间的信任问题、系统程序的安全性与完整性问题、原有应用系统的修改问题等。

(1)口令猜测攻击：在 Kerberos 协议中，用户向 AS 请求获取访问 TGS 的票据 TGT 时，AS 发往客户端的报文是由用户的密钥 K_c 加密的。由于 K_c 是通过用户输入的口令按单向 Hash 算法导出的，容易被猜测和窃听。攻击者可以收集大量的 TGT，通过技术和密钥分析来进行口令猜测。当用户选择的口令不够强时，就更

不能有效的防止口令猜测攻击。

(2)重放攻击：虽然时间戳是专门用于防止重放攻击的，但票据的有效时间内仍然可能奏效，假设在一个 Kerberos 域内的全部时钟保持同步，收到消息的时间在规定的范围内，就认为该消息是新的。而事实上，攻击者可以事先把伪造的消息准备好，一旦得到票据就马上发出，这在生存期限内是难以检查出来的。

(3)时间同步问题：Kerberos 的认证是建立在时钟同步的基础上的，要保证客户端、AS、TGS 和应用服务器的机器时间保持一致，这在当前环境下是很难达到的。如果能够欺骗主机，改变主机的正确时间，就可以将过期的认证符重放，实施重放攻击。

(4)密钥的存储问题：Kerberos 采用对称密钥体制，认证中心要求保存大量的共享密钥，无论是管理还是更新都有很大的困难，特别需要细致的安全保护措施，为此将付出极大的系统代价。

(5)认证域之间的信任问题：Kerberos 信任域之间的多级跳跃过程复杂而且不明确，相互信任和协调不方便。若各 Kerberos 区域形成复杂或不规则的网状结构，则要求方便的域间服务，将付出极大的代价，即系统的可扩充性不强，针对这种无序的状况，应有规划有目的地建立起一个大的相互信任的多域网络结构。

(6)系统程序的安全性、完整性问题：实际上，最严重的攻击是恶意软件攻击。Kerberos 认证协议依赖于 Kerberos 软件的绝对可信。攻击者可以用执行 Kerberos 协议和记录用户口令的软件来代替所有用户的 Kerberos 软件，达到攻击目的。

(7)原有应用系统的修改问题：Kerberos 作为第三方的用户认证系统，必须和其他应用系统相结合才能发挥作用。但原有的应用系统并不能直接适用于 Kerberos 环境，其客户端和服务器端的软件都要作一定的修改，使之在建立连接后能够交换加密信息（如票据）。这样的应用被称之为“Kerberized”。如果原来的协议没有提供用户认证部分，其修改工作是比较复杂的。

2.4 PKI 的组成

PKI(Public Key Infrastructure)即公开密钥体系，是一种遵循既定标准的密钥管理平台，能够为所有网络应用透明地提供采用加密和数字签名等密码服务所需要的密钥和证书管理体系。

PKI 提供了四种主要的安全功能：保密性，保证信息的私有性；完整性，保证信息没有被篡改；真实性，证明一个人或一个应用的身份；不可否认性，保证信息不能被否认。PKI 最主要的任务就是确立可信任的数字身份，而这些身份可以被用来和密码机制相结合，提供认证、授权或者数字签名验证等服务，这些服务的

用户可以在相当程度上确信自己没有被误导。这个可信任的数字身份通过公钥证书来实现，公钥证书（X.509 证书）是用户身份和他所持有的公钥的结合。首先，由一个可信任的权威机构 CA 来证实用户的身份，然后该 CA 对用户身份和公钥的散列值进行数字签名，以证明该公钥的确属于该用户。

一个实用的 PKI 体系必须充分考虑互操作性和可扩展性。它所包含的认证机构（Certificate Authority，CA）、注册机构（Registration Authority，RA）、密钥（Key）与证书（Certificate）管理、密钥备份及恢复、撤销系统等功能模块应该有机地结合在一起。

2.4.1 认证中心 CA

CA 中心，又称为数字证书认证中心，在 PKI 体系中，认证中心 CA 是整个 PKI 体系中各方都承认的一个值得信赖的公正的第三方机构，专门解决公钥体系中公钥的合法性问题。CA 是证书的签发机构，是保证电子商务、电子政务、网上银行、网上证券等交易的权威性、可信任性和公正性的第三方机构[27]，它负责生成、分发和撤销数字证书，通过认证过程将一个公共密钥值与一个人、一台计算机或一个主体联系起来，并对它们的身份及与公钥的匹配关系进行认证和证明。CA 的主要功能是签发证书和管理证书。具体包括用户注册、证书签发、证书撤销、密钥恢复、密钥更新、证书使用和安全管理等。

2.4.2 注册中心 RA

注册中心 RA 是数字证书注册审批机构，是认证中心 CA 的证书发放、管理的延伸部分。RA 按照特定的政策和管理规范对用户的资格进行审查，并执行是否同意给该申请者发放证书、撤销证书等操作，承担因审核错误而引起的一切后果。如果审查通过，便可实时或批量向 CA 提出申请，要求为用户签发证书。

RA 负责对证书申请进行资格审核，RA 主要具有以下几种功能：

(1)填写用户注册信息：替用户填写有关用户证书申请信息。

(2)提交用户注册信息：核对用户申请信息，决定是否提交审核。

(3)审核：对用户的申请进行审核，以决定批准还是拒绝用户的证书申请。

(4)发送生成证书申请：向 CA 提交生成证书请求。

(5)发放证书：将用户证书和私钥发放给用户。

(6)登记黑名单：对过期的证书和因各种原因而撤销的证书及时登记，并向 CA 发送。

(7)CRL 管理：保证证书撤销列表 CRL 的及时性，并对 CRL 进行管理。

(8)日志审计：维护 RA 的操作日志。

(9)自身安全保证：保证服务器自身密钥、数据库信息、相关配置文件的安全。

2.4.3 数字证书库与 LDAP 目录服务器

证书库是 CA 颁发的证书和证书撤销列表（Certificate Revocation List，简称 CRL）的集中存放地，它就像网上的“白页”一样，是网上的一种公共信息库，可供广大公众进行开放式查询。查询的目的有两个：一个是想得到与之通信的实体的公钥；一个是要验证通信对方的证书是否已进入“黑名单”。LDAP 目录服务器用于存取证书以及证书撤销列表 CRL，用户可通过 LDAP 目录服务器将整个 CRL 下载到本地的机器上，再根据下载的 CRL 查询用户的证书状态。

2.4.4 密钥备份及恢复

用户由于某种原因丢失了解密数据的密钥，就会使被加密的密文无法解密，造成数据丢失。PKI 提供了密钥备份与恢复机制来避免这种情况发生，当用户证书生成时，加密密钥即被 CA 备份存储，当需要恢复时，用户只需向 CA 提出申请，CA 就会为用户自动进行恢复。但有一点必须强调：密钥备份和恢复只能针对加/解密密钥，而对签名密钥不能做备份。因为数字签名是用于支持不可否认服务的，有时间性要求，因此不能备份和恢复。密钥备份发生在用户申请证书阶段，如果注册声明公/私钥对是用于数据加密，那么 CA 即可对该用户的私钥进行备份。当用户丢失密钥后，就可以通过可信任的密钥恢复中心或者 CA 来完成密钥恢复。

2.4.5 证书撤销

证书由于某些原因需要作废，比如用户身份姓名的改变，私钥被窃或泄露，用户与所属企业关系变更等，PKI 需要使用一种方法警告其他用户不要再使用该用户的公钥证书，这种警告机制被称为证书撤销。

证书撤销的主要实现方法有两种，一种是利用周期性的发布机制，如证书撤销列表 CRL。证书撤销信息的更新和发布频率非常重要，两次证书撤销信息发布之间的间隔称为撤销延迟，在某些情况下，撤销延迟可以为几个小时或者几天，但在某些场合，几分钟的撤销延迟都是不可接受的。另一种是在线查询机制，如在线证书状态协议（Online Certificate Status Protocol，OCSP）。

2.4.6 客户端软件

为方便客户操作，解决 PKI 的应用问题，在客户装有客户端软件，以实现数字签名、加密传输数据等功能。此外，客户端软件还负责在认证过程中，查询证书和相关证书的撤消信息以及进行证书路径处理、对特定文档提供时间戳请求等。

2.5 证书和密钥管理

2.5.1 X.509 证书

数字证书[28]是网络通信中标志通信各方身份信息的一系列数据，其作用如同日常生活中使用的身份证，由 CA 发行，是持有者在网络上证明自己身份的凭证。数字证书是一个经证书授权中心数字签名的包含公开密钥拥有者信息以及公开密钥的文件。证书一方面可以用来向系统中的其他实体证明自己的身份，另一方面由于每份证书都携带证书持有者的公钥，所有证书也可以向接收者证明某人或某个机构对公开密钥的拥有，同时也起着公钥分发的作用。

X.509 证书[29]，包括三部分：证书内容、签名算法和使用签名算法对证书内容所作的签名。证书内容如下图 2-3 所示：

版本
证书序列号
签名算法（算法标识符、参数）
发证者名称
有效期（起始时间、结束时间）
主体名称
主体的公钥信息（算法、参数、密钥）
发行者唯一的标志符
主体唯一的标志符
扩展项
签名（算法标识符、参数、前面信息）

图 2-3 X.509 证书

1）版本：用来区分 X.509 不同年份的版本，目前常用的是 V3。

2）序列号：由 CA 分配给证书的唯一的数字型标志符。当证书被取消时，将

此证书的序列号放入由 CA 签发的 CRL 中。

3）签名算法标志符：用来标志对证书进行签名的算法和算法所需的参数。协议规定这个算法同证书格式中出现的签名算法必须是同一个算法。在 X.509 证书中，签名算法实际上是两个算法的组合：哈希算法（如 MD5）和公钥算法（如 RSA）。

4）发证者名称：是 X.509 证书的一个唯一标志符，用来标志签署和发行该证书的 CA 的身份，即签发证书的 CA 名称。

5）有效期：定义证书有效的起始时间和终止时间。

6）主体名称：证书中公钥对应的个人或者实体的名字。

7）主体公钥信息：包括主体的公钥、使用此密钥的算法标志符和相关的参数。

8）发行者唯一标志符：证书发行机构的名称存在重用的可能，该字段用于唯一标志证书颁发机构。

9）主体唯一标志符：主体名称存在重用的可能，该字段用于唯一标志主体。

10）扩展：在不改变证书格式的前提下，允许证书编码中加入额外的信息。

11）签名：包含证书发行者对该证书的签名、用于证书签名的算法标志符和所有相关参数。

2.5.2 证书管理

证书管理主要包括以下几个方面：证书的申请与颁发、证书的发放、证书的撤销、证书的更新和证书的归档。

a.证书的申请与颁发

数字证书的申请一般有两种方式：离线申请方式和在线申请方式。

(1)离线申请方式

用户持有有关证件到注册中心 RA 进行书面申请，填写按一定标准制定的表格。

(2)在线申请方式

用户通过网络，到认证中心的相关网站下载申请表格，按内容提示进行填写；也可以通过电子邮件和电话呼叫中心传递申请表格的信息，但有些信息仍需要人工录入，以便进行审核。

RA 对用户身份进行审核，如果审核通过，则向 CA 提交证书申请请求。CA 为用户产生证书后，将证书返回给 RA。如果密钥由 CA 中心产生，则同时将用户私钥也返回给 RA，然后由 RA 将证书和密钥返回给用户。

b.证书的发放

当 CA 颁发了一份证书，就需要在信息系统内部公布用户的公钥证书，让所有

人方便地获取到该证书，证书发放方式有三种：私下分发、资料库发布和协议发布。

(1)私下分发

这是最简单的分发方式，这种情况下，个人用户将自己的证书直接传送给其他用户。例如：直接分发给其他用户或通过 E-mail 附件方式传递。

(2)资料库发布

资料库发布，指用户的证书或者证书撤销信息存储在用户可以方便访问的数据库中或其他形式的资料库中。例如：LDAP 轻量级目录访问服务器、X.500 目录访问服务器、Web 服务器、FTP 服务器、数据有效性和验证服务器。

(3)协议发布

证书和证书撤销信息也可以作为其他通信交换协议的一部分。例如：TLS 协议中证书的交换、IPSec 中的密钥交换协议。

c.证书的撤销

在证书的有效期内，由于被泄密的私钥所对应的公钥证书应被作废；或者证书中包含的证书持有者和某组织的关系已经终止，则相应的公钥证书也应该作废，此时证书的持有者要提出证书撤销申请。CA 一旦收到证书撤销请求，立即执行证书撤销，并同时通知用户，使之知道特定证书已经撤销。从安全角度来说，每次使用证书时，系统都要检查用户的证书是否已被撤销。

证书撤销的主要实现方法有两种，一种是利用周期性的发布机制，如证书撤销列表 CRL；另一种是在线查询机制，如在线证书状态协议（OCSP）。

(1)证书撤销列表 CRL

CRL 是一种包含撤销证书列表的签名数据结构，CRL 的完整性和可靠性由它本身的数字签名来保证，CRL 的签名者一般也是颁发证书的签名者。

证书撤销列表 CRL 的格式如下图 2-4 所示：

版本号
签名算法（算法标志符、参数）
发行者名称
本次更新时间
下一次更新时间
撤销的证书（用户证书的序列号、撤销时间）
…
撤销的证书(用户证书的序列号、撤销时间)
签名（算法标志符、参数、签名信息）

图 2-4 证书撤销列表 CRL

证书撤销列表格式包括以下部分：

1）版本号：指出 CRL 的版本号。

2）签名算法标志符：用于 CRL 签名的算法以及所有的相关参数。

3）发行者名称：CRL 颁发者的名称。

4）本次更新时间：本次 CRL 的发布时间。

5）下一次更新时间：下一个 CRL 的发布时间。

6）撤销的证书：被撤销证书的列表，每一项包括证书的序列号和撤销时间。

7）CRL 扩展：在不改变 CRL 格式的前提下，允许加入额外的信息。

8）签名：包括对 CRL 的签名、用于签名的算法和所有相关参数。

在 PKI 系统中，证书撤销列表 CRL 是自动完成的，对用户是透明的。CRL 中并不存放撤销证书的全部内容，只是存放证书的序列号，以便提高检索速度。CRL 产生的主要步骤是：RA 建立与 CA 的连接，提出撤销申请（包括撤销证书的序列号及撤销理由），CA 将撤销证书的序列号签发到 CRL 中，然后系统通过数据库或 LDAP 目录等方式发放新的 CRL，并且提供用户在线查询。

(2)在线查询机制

目前，最普遍的在线撤销机制就是在线证书状态协议（OCSP），这是一种相对简单的请求/响应协议。在 OCSP 之前，用户没有一种方便的途径来检查证书的有效性。由证书机构签发的 CRL 是一张无效证书及其持有者的名单，这种 CRL 处理方式要求用户配置客户机来处理来自证书机构的 CRL。而且由于撤销证书的数量很大，CRL 常常会越变越大，当它们体积过于庞大变得难以使用时就带来了另一个问题，即每次 CRL 分发会大量消耗网络带宽和降低客户机处理能力。

OCSP 实时在线地向用户提供证书状态，比 CRL 处理快的多，并避免了逻辑问题和处理开销。

为立即检查证书是否被撤销，用户的客户机必须形成请求，并将该请求转发到一个 OCSP 应答器，应答器回答下列 3 个有关证书有效性信息中的一个：“ 有效 ”、“ 撤销 ”和“ 不知道 ”。响应信息必须经过数字签名，以保证响应是源于可信任方并且在传输过程中没有被篡改。签名密钥一般属于颁发证书的 CA，是一个可信任的第三方。

d.证书的更新

进行证书更新有几个原因：与证书相关的密钥可能达到它有效生命的终点；证书可能已经“ 接近 ”过期；证书中已经证明的一些属性可能已经改变，并且这些新属性值必须重新证明。在这些情况下，必须颁发一个新的公/私钥和相关证书，这被称为证书更新或者密钥更新。并且，只要证书没有撤销，先前的密钥和证书仍然能够用来完成认证过程。

证书更新包括两种情况：最终实体证书更新和 CA 证书更新。

对于最终实体证书更新，有两种方式：一种就是执行人工密钥更新，用户向 RA 提出更新证书的申请，RA 根据用户的申请信息更新用户的证书；另一种就是实现自动密钥更新，PKI 系统采用对管理员和用户透明的方式，对快要过期的证书进行自动更新，生成新的密钥对。

CA 证书更新的同时，使用它的旧密钥签名新公钥，新密钥签名旧公钥（CA 证书更新前的密钥称为旧公钥和旧私钥）。因此，一次 CA 证书更新将拥有四个证书：旧用旧证书，用旧私钥签名的旧公钥，是原始自签名证书；旧用新证书，用新私钥签名的旧公钥；新用旧证书，用旧私钥签名的新公钥；新用新证书，用新私钥签名的新公钥，这是自签名 CA 证书，所有 CA 用户使用这个 CA 证书作为新的信任第三方。

e.证书的归档

由于证书更新的原因，一段时间后，每个用户都会形成多个“ 旧 ”证书和至少一个“ 当前 ”证书，这一系列旧证书和相应的私钥就组成了用户密钥和证书的历史档案，简称密钥历史档案。

记录密钥历史是非常重要的。用户以前自己加密的或者其他人用他的旧证书加密的数据，无法用现在的公/私钥解密，这时，就需要到用户的密钥历史档案中查找正确的解密密钥来解密数据。同理，也可能需要到密钥历史档案中查找合适的证书来验证他自己以前的数字签名。同证书更新一样，管理密钥历史档案最好由 PKI 自动完成。除了用户的签名密钥外，证书的所有数据信息都要进行归档，保存在 CA 中心的数据库服务器中。

2.5.3 密钥管理

密钥管理主要包括以下几个方面：密钥产生、密钥安全保护、密钥恢复和密钥档案。

a.密钥产生

用户公/私钥密钥对的产生有两种方式：用户自己产生密钥对和 CA 为用户产生密钥对。

(1)用户自己产生密钥对

用户自己选择产生密钥对的方法和长度，负责私钥的存放；然后向 CA 提交自己的公钥和身份证明。CA 对提交者的身份进行认证，对密钥强度和持有者进行审查。如果审查通过，CA 将用户身份信息和公钥捆绑封装并进行签名产生数字证书，然后发放给用户。

(2)CA 为用户产生密钥对

用户提出证书申请请求，经过 RA 审查通过后，向 CA 提交该请求，CA 负责产生并获得密钥对，同时生成公钥证书和私钥，公钥证书发布到目录服务器，私钥交给用户。CA 对公钥证书进行存档，如果用户私钥注明不是用于签名，则 CA 对用户私钥也进行存档，从而保证当用户要求恢复密钥时可以提供密钥恢复服务。

以上两种方法各有利弊。采取 CA 为用户产生密钥对的方式将使密钥归档和恢复之类的服务简单化，但此时 CA 必须具有高度的可信性、可用性和安全性，私钥如何安全传输给用户也是个非常重要的问题，并要确保私钥在其生命期中不被暴露。如果由用户自己产生密钥对，用户私钥传送给 CA 中心也同样是一个问题。

b.密钥安全保护

密钥安全保护是 PKI 系统安全的核心部分，主要是指私钥的保护。它包括两个方面的内容：CA 认证中心私钥的管理和用户密钥的安全管理。

(1)CA 私钥管理

作为一个安全认证中心，CA 自身的密钥非常重要。CA 的密钥安全，主要是指 CA 的签名私钥的安全，如果签名私钥泄密，则 CA 所签发的全部证书都要及时作废，否则后果不堪设想。

(2)用户密钥管理

用户可以自己产生公/私密钥对，也可由 CA 中心为其产生。如果由 CA 中心产生，CA 必须保证将用户的私钥安全地传送给用户。CA 中心可为用户保存加密私钥，作为备份以供将来为用户提供密钥恢复服务，但签名私钥绝对不能备份存储。CA 中心为用户产生公/私密钥对后，将用户身份信息和公钥封装签名生成公钥证书，然后将公钥证书及其相关联的私钥通过加密封装在一起，形成 PKCS12 证书传送给用户。这使得用户可以通过 PKCS12 证书获取自己的非对称密钥对和 X.509 证书。

c.密钥恢复

用户由于某种原因丢失解密数据的密钥，则被密钥加密的密文无法解开，造成数据丢失。PKI 提供了密钥备份与恢复机制来避免这种情况的发生。密钥恢复分为两种情况：CA 密钥恢复和用户解密密钥恢复。

(1)CA 密钥恢复

CA 密钥的产生是由密钥加密模块产生的，因此密钥备份由加密模块系统管理员启动加密管理程序执行。它将 CA 密钥分割成多块，为每一块生成一个随机口令，使用该口令加密对应的密钥块，每人保存一块。恢复密钥时，必须由各密钥备份持有人员都到场，共同恢复 CA 密钥。

(2)用户密钥恢复

如果用户申请证书时注明是用于加/解密，则 CA 签发用户证书时，可以对用

户私钥进行解密，把用户私钥进行备份。若用户私钥丢失或者其他原因，但用户又不愿意撤销原证书和密钥，希望能对原私钥进行恢复，CA 就可以根据用户的密钥历史对密钥进行恢复。

d.密钥档案

密钥档案一般是由第三方提供的服务，并且涉及许多终端实体的密钥资料的存储，包括终端实体签名私钥的保存，主要在于审计和交易争端时使用。密钥档案提供的服务包括：公证和时间戳服务、跟踪审计和终端实体的密钥历史恢复。

2.6 信任模型

实际网络环境中不可能只有一个 CA，多个认证机构之间的信任关系必须保证 PKI 用户不必依赖和信任专一的 CA，否则将无法进行扩展、管理和包含。信任模型建立的目的是确保一个认证机构签发的证书能够被另一个认证机构的用户所信任。

(1)严格层次信任模型

在严格层次信任模型中，上层 CA 为下层 CA 颁发证书。这种信任模型中有且只有一个根 CA，每个证书使用者都知道根 CA 的公钥。只要找到一条从根 CA 到一个证书的认证路径，就可实现对这个证书的验证，建立对该证书主体的信任。

严格层次结构可以被描绘为一个倒树，根在顶上，树枝向下伸展，树叶在最下层。根 CA 充当信任的根，也就是认证的起点或终点。在根 CA 的下面是零层或多个子 CA，它们是从属于根 CA，最下层的叶节点对应于终端实体。在这个模型中，层次结构中的所有实体都信任根 CA。在严格层次模型中，每一个实体（包括子 CA 和终端实体）都必须拥有根 CA 的公钥。根 CA 的签名密钥是非常重要的，一旦泄漏，对整个信任模型都将产生严重的后果。

(2)分布式信任模型

与严格层次结构中的所有实体都信任唯一一个 CA 相反，分布式信任结构把信任分散在两个或多个 CA 上。

(3)以用户为中心的信任模型

在以用户为中心的信任模型中，每个用户自己决定信任哪些证书。因为要依赖于用户自身的行为和决策能力，因此以用户为中心的模型在技术水平较高和利害关系高度一致的群体中是可行的，但是在一般的群体中是不现实的。

(4)交叉认证

交叉认证是一种把以前无关的 CA 连接在一起的机制，从而使得它们各自终端用户之间的安全通信成为可能。一个认证机构可以是另一个认证机构颁发的证书

的主体，该证书称为交叉证书。交叉证书的实际构成除了证书中主体和颁发者都是 CA 外，与普通用户证书一致。

交叉认证有两种类型，一种是域内交叉认证，如果两个 CA 属于相同的域，这一认证被称作域内交叉认证，另一种是域间交叉认证，就是两个 CA 属于不同的域的认证过程。

第 3 章　基于公钥的 Kerberos 改进协议

3.1 Kerberos 的几种公钥扩展协议

Kerberos 是为 TCP/IP 网络(Internet)设计的基于 C/S 模式的三方验证协议，也是 Internet 访问控制技术的一个代表，广泛应用于 Internet 服务的安全访问控制[30]。但由于 Kerberos 协议基于对称密钥密码体制，这样在客户与 KDC 之间，应用服务器与 KDC 之间都必须建立和维护大量的共享密钥，因此密钥的产生、分配和管理都受到很大限制。Kerberos 协议的应用体制上也存在很大问题，特别是目前大部分系统的 Kerberos 协议仍采用 DES 算法，DES 算法的保密强度比较低，这样给系统带来很大隐患。

公钥密码体制使用两个相关联的密钥将加密、解密分开，其中一个密钥是公开的，称之为公钥，用于加密和验证数字签名；另一个密钥是保密的，称之为私钥，用于解密和数字签名。这样避免了通信过程中由于加密密钥泄露造成的威胁，又可以通过数字签名来保证信息的不可否认。PKI（公钥基础设施）就是提供公钥加密和数字签名服务的系统，目的是为了管理密钥和证书，保证网络信息的安全性。PKI 适用在事先无法确定用户的场合下，Kerberos 则适用于用户和应用事先都已经知道的 Internet 的环境。因此由于彼此的需要，PKI 与 Kerberos 经常配合使用，通过 Kerberos 向网络认证用户，然后获得用于数字签名等功能的公钥证书或者私钥数据。

随着公钥系统的不断发展，人们提出了很多 Kerberos 的公钥扩展，如利用公钥进行预认证(PKINIT)、利用公钥跨区域认证(PKCROSS)、基于公钥技术的分布式认证(PKDA)等。

3.1.1 PKINIT

在 Kerberos 协议中，当用户向 AS 请求 TGT 时，AS 发送的应答报文是通过用户的密钥 K_c 来加密的，而密钥 K_c 是通过用户输入的口令导出的，由于用户趋于选择低质量容易记忆的口令，这样 K_c 很容易被猜测和窃听。

PKINIT[31]协议试图利用引入基于 X.509 的公钥证书来克服这个弱点。PKINIT

一般指在 Kerberos 域内最初使用公钥密码体制进行安全协议的认证。在初始认证阶段 AS_REQ 和 AS_REP 消息中使用公钥机制，后续的消息操作继续使用原来的 Kerberos 的协议。PKINIT 协议 AS 交换流程如下图 3-1 所示：

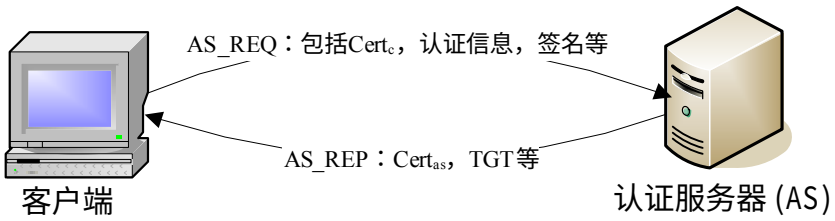

图 3-1 PKINIT 的 AS 交换

传统的 Kerberos 的 AS 交换中，AS 只需要查看本地数据库是否存在该用户便会返回相应的 TGT。而 PKINIT 扩展修改了 TGT 的申请，客户端需要把自己的公钥证书、认证信息和签名等信息一起发送给 AS。AS 收到 AS_REQ 报文后，首先验证用户的证书的有效性，然后验证用户的数字签名。然后 AS 产生临时会话密钥 K_t，并用 K_t 加密 $K_{c,tgs}$、时间戳和生存期限等信息。这个临时密钥 K_t 通过用户的公钥加密，而且用 AS 的私钥进行签名，确保其安全性和不可伪造。接着将返回应答报文，报文中包含 AS 的证书、TGT、加密并签名的密钥 K_t 和使用 K_t 加密的信息。

3.1.2 PKCROSS

PKCROSS 一般指在 Kerberos 不同域中使用公钥密码体制进行安全协议的认证[32]。在 KDC 之间采用 PKCROSS 进行鉴别，不仅可以降低密钥分发的复杂度，还可以提供不可否认服务，提高系统的安全性。PKCROSS 增强了 Kerberos 在跨域网络环境中的使用范围，其中应用服务器参与了认证过程。其具体协议认证过程如下图 3-2 所示：

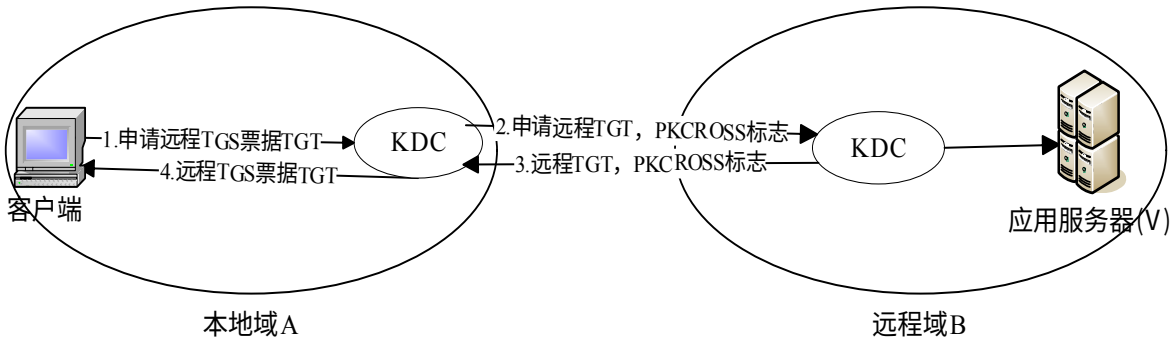

图3-2基于公钥的跨域认证（PKCROSS）

(1)用户向本地 KDC 请求远程 TGS 票据 TGT，这个过程和传统的 Kerberos 请求一致。

(2)本地 KDC 收到请求后，就向远程 KDC 请求远程域内的 TGT 票据，其中

选项 Options 的设置为 PKCROSS 标志，表示将采用公钥鉴别技术。

(3)远程 KDC 收到信息后，先验证证书的有效性，并验证签名。验证成功后为该用户生成 TGT，并且将 Options 设置为 PKCROSS，然后把会话密钥等信息一起传送给本地的 KDC。

(4)本地 KDC 接收远程 KDC 返回的 TGT 和一些验证信息，并且把 TGT 和采用本地 KDC 和用户 C 的共享密钥加密的会话密钥等参数传回。

通过上述四个步骤，就可以得到了远程 TGS 的 TGT，就可以进一步地访问远程域中的服务了。PKCROSS 票据用来实现本地 KDC 和远程 KDC 之间的相互认证，在两个 KDC 中传递的信息严格遵守 PKINIT 规范，可以把本地 KDC 视为客户端。当远程 KDC 向本地 KDC 发送 PKCROSS TGS 时，远程 KDC 相信本地 KDC 能够代表它把远程 TGT 发送给本地客户端。

3.1.3 PKDA

在 Kerberos 协议中，如果大规模的应用都通过一个 KDC 来完成的话，会产生两个问题：①所有使用 Kerberos 服务的程序都依赖 KDC，如果 KDC 产生问题，这些程序都无法正常运行；②在处理大量请求时，KDC 可能是一个“瓶颈”。

通过引入基于公钥技术的分布式的鉴别技术(PKDA)，可以有效地解决这两个问题。与以上两种对 Kerberos 的公钥扩展不同的是，PKDA 去除了 Kerberos 框架中的 KDC，是通过改变 Kerberos 框架来完成分布式鉴别的。把认证过程从 KDC 移动到客户与应用服务器交互双方来增加应用范围和安全性，因为这种移动减少了由于认证请求数量太多时而在 KDC 处产生的瓶颈。同时也消除了在 KDC 密钥汇集处因某一密钥的失败而对整个系统造成不安全的影响[33]。PKDA 和传统 Kerberos 不同的是直接向服务器端请求，不需要和 KDC 之间传递对称会话密钥，甚至可以进行跨域间的鉴别。事实上，PKDA 不需要向 KDC 做初始的身份验证，即使跨域进行也不需要，鉴别只在客户端和应用服务器上进行。

下面是 PKDA 协议的交换过程如下图 3-3 所示：

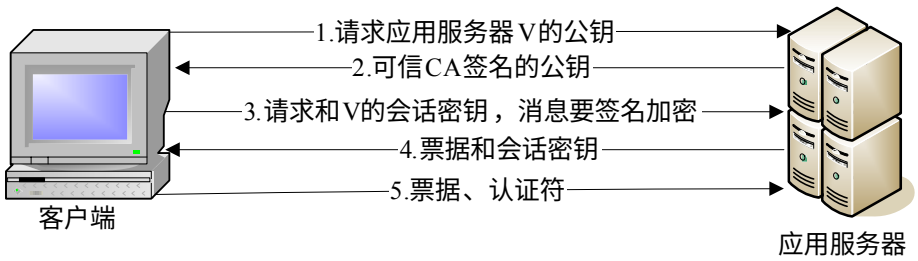

图3-3基于公钥的分布认证（PKDA）

(1)客户端向应用服务器请求其公钥证书。

(2)服务器端收到信息后，将自己的证书或证书链发给客户端。

(3)客户端收到消息后，先验证证书的有效性。产生服务票据 Ticket 请求。消息都用服务器端的公钥进行了加密，这样可以确保只有服务器才能解密请求报文。加密的部分包括用户名称、证书和用户的私钥签名的认证符。认证信息包括服务器名称、公钥、认证时间和一个用于加密下一个报文的随机对称密钥。服务器名称和公钥用来鉴别用户，支持双向认证，并有效地防御了中间人攻击。

(4)服务器端收到信息后，解密消息，验证用户的证书的有效性，生成一个 Ticket 和一个会话密钥，返回给客户端。这里的 Ticket 的使用和传统的 Ticket 一样，但是在 Ticket 的组成有所不同，Ticket 的加密采用的对称密钥仅有服务器端知道，传统的 Kerberos 中是和 KDC 共享的。在票据返回时，传统的 Kerberos 是以明文的方式返回用户名称，而这里是以加密的形式返回，保护了用户的身份信息。

(5)客户端收到信息后，向服务器端发送 Ticket 和鉴别符来获取相应的服务。鉴别符为用会话密钥加密的用户名和一个当前的时间戳 TS1。

这样就完成了从请求 Ticket 到请求服务的全过程。值得注意的是第三步，其中传输的信息采用了公钥的加密和私钥的签名，保证了双方的相互鉴别。但是要进行 4 次加解密的计算，开销很大，相对速度较慢。在跨域交换时，消息数目的降低带来的好处可能被公钥计算的开销所抵消。

3.2 Kerberos 协议的改进

以上的公钥结合方案虽然都提出了怎样结合公钥的机制，并部分改善了 Kerberos 系统的安全性能，但是都只描述了标准 Kerberos 认证协议中使用公钥密码的消息语法，所有的私钥和公钥都由 KDC 直接管理，没有结合 PKI 来管理证书或者撤销列表，有的方案则通过改变 Kerberos 框架，把认证过程限制在客户端与应用服务器之间，使得应用服务器的压力较大，限制了这些方案的应用范围。

本文在对 KerberosV5 及其多个扩展协议的研究基础上，在不改变 Kerberos 原有框架的前提下，把 PKI 技术、访问控制与 Kerberos 协议相结合，提出一个基于公钥密码体制的 Kerberos 改进协议。在与上文所研究的几种 Kerberos 公钥扩展协议相比，该改进协议主要有以下两个不同之处：

(1)不改变原有框架的前提下，改进协议结合了 PKI 来管理证书，并应用 USB Key 技术来保存用户的私钥而不是直接由 KDC 来管理。

(2)通过票据来加载基于角色的访问控制信息，在改进协议中添加了授权的功能，在实现用户认证的同时实现了授权服务。

下面是改进协议中定义的一些符号和概念：

$Cert_c$、$Cert_{as}$ 、$Cert_{tgs}$、$Cert_v$ ：C、AS、TGS、V 的证书；

$Kpri_c$、$Kpri_{as}$、$Kpri_{tgs}$、$Kpri_v$ ： C、AS 、TGS、V 的私钥；

$Kpub_c$、$Kpub_{as}$、$Kpub_{tgs}$、$Kpub_v$ ：C、AS、TGS、V 的公钥；

改进协议认证过程如下图 3-4 所示：

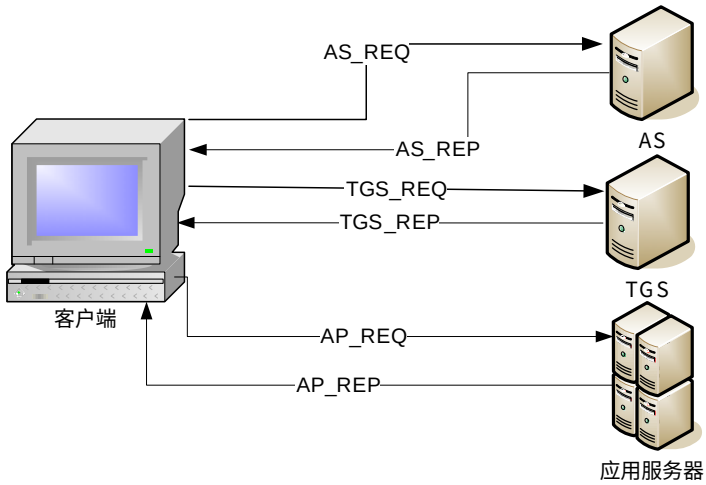

图3-4 Kerberos 改进协议认证过程

3.2.1 认证服务交换

认证服务交换过程如下图 3-5 所示：

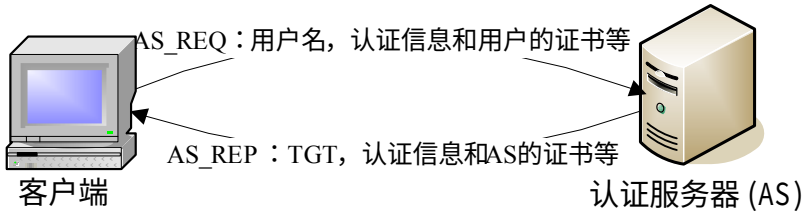

图3-5 认证服务交换

C→AS：AS_REQ={ID_c, ID_{tgs}, $EKpri_c$ ($EKpub_{as}$ (ID_c, ID_{tgs}, Nounce1, Times)) , $Cert_c$}

客户端向认证服务器 AS 发送请求报文，请求获得票据许可票据 TGT，在 Kerberos 原协议中，该报文是明文发送的。改进后请求报文 AS_REQ 包括用户名称、TGS 名称、认证信息和用户的证书。认证信息使用 AS 的公钥 $Kpub_{as}$ 加密，确保只有 AS 才能解开认证信息，使用用户私钥 $Kpri_c$ 签名，来保证信息是由该用户发出的。认证信息内容包括用户名称、TGS 名称、随机数 1 和当前时间等信息。

AS→C ：AS_REP={$Ticket_{tgs}$, $Cert_{as}$, $EKpri_{as}$ ($EKpub_c$ ($K_{c,tgs}$, ID_{tgs}, Nounce1, Times)) }

$Ticket_{tgs}$= $EKpri_{as}$ ($EKpub_{tgs}$ (ID_c, IP_c, ID_{tgs}, Nounce1, Times, $K_{c,tgs}$))

认证服务器 AS 收到用户 C 的请求报文后，先检查用户的证书 $Cert_c$ 的可信性，

并根据用户名称在数据库中取出用户的公钥，验证用户签名并用自己的私钥解密后得到认证信息。AS 随机生成用户 C 和票据服务器 TGS 的共享密钥 $K_{c,tgs}$，并生成票据许可票据 TGT($Ticket_{tgs}$)，该 TGT 由 TGS 的公钥 $Kpub_{tgs}$ 加密并用 AS 的私钥 $Kpri_{as}$ 签名，包括会话密钥 $K_{c,tgs}$、TGS 名称、随机数 1 和当前时间等信息。然后 AS 生成认证信息，该认证信息通过用户的公钥来加密并用自己的私钥签名来确保其安全性和不可伪造性，包括会话密钥 $K_{c,tgs}$、TGS 名称、随机数 1 和当前时间等信息。然后加上 $Cert_{as}$ 一起发送应答报文给用户 C。

3.2.2 票据许可服务交换

票据许可服务交换过程如下图 3-6 所示：

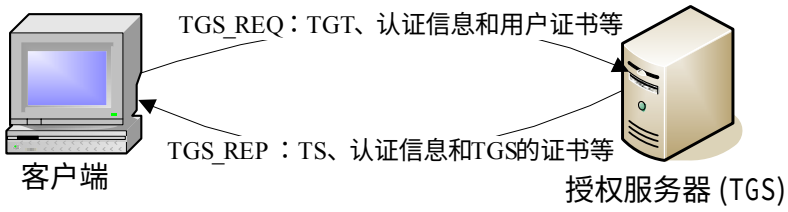

图3-6 票据许可服务交换

C→TGS：TGS_REQ= $\{ID_v, Times, Nounce2, Ticket_{tgs}, Authenticator1, Cert_c\}$

$Authenticator1= EK_{c,tgs}(ID_c, Times, Nounce2)$

客户端接收到 AS 的应答报文后，检查 AS 的证书的可信性。然后用 AS 的公钥来验证签名，利用用户自己的私钥解密认证信息，得到随机数 1 和与 TGS 通信用的会话密钥 $K_{c,tgs}$，并验证随机数 1 是否与自己发给 AS 的随机数相同，如果相同则确信这是新报文。最后向授权服务器 TGS 发送请求报文，请求获得访问应用服务器 V 的票据 TS($Ticket_v$)。该请求报文包括应用服务器名称、当前时间、随机数 2、AS 发送给用户的 TGT 票据、认证符 1 和用户的证书等信息。认证符 1 使用 C 和 TGS 的会话密钥 $K_{c,tgs}$ 加密，包括用户名称、随机数 2 等信息。

TGS→C：TGS_REP= $\{ID_c, Ticket_v, EK_{c,tgs}(K_{c,v}, Times, Nounce2, ID_v), Cert_{tgs}\}$

$Ticket_v= EKpri_{tgs}\ (EKpub_v\ (ID_c, IP_c, Rolevalue, Times, K_{c,v}))$

TGS 收到报文后，检查用户的证书 $Cert_c$ 的可信性，并从 $Ticket_{tgs}$ 中得到会话密钥 $K_{c,tgs}$，然后验证认证符中的信息是否正确，根据用户名称在数据库中取出角色值(根据角色值便可以确定该用户的访问权限)，产生随机会话密钥 $K_{c,v}$。TGS 向用户发送应答报文，应答报文包括用户名、票据 TS、认证信息和 TGS 的证书。认证信息采用 $K_{c,tgs}$ 加密，内容包括：会话密钥 $K_{c,v}$、当前时间、随机数 2 以及应用服务器的名称等。票据 TS 则采用应用服务器 V 的公钥加密，并用 TGS 的私钥签名，内容包括用户名称、用户 IP、角色值、当前时间和会话密钥 $K_{c,v}$ 等信息。

3.2.3 客户端与应用服务器的认证交换

客户端与应用服务器交换过程如下图 3-7 所示：

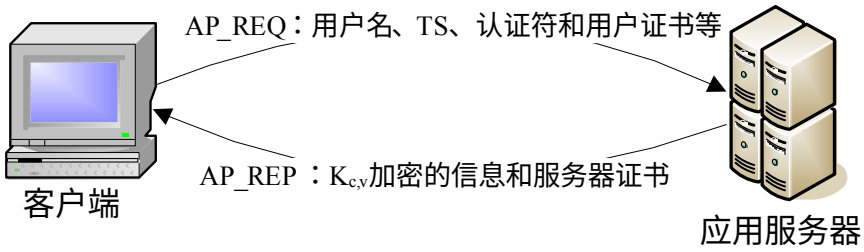

图3-7 客户端与应用服务器交换

C→V：AP_REQ= $\{ID_c, IP_c, Ticket_v, Cert_c, Authenticator2\}$

$Authenticator2 = EK_{c,v}(ID_c, IP_c, Nounce3)$

当用户收到应答报文后，检查证书 $Cert_{tgs}$ 的可信性，用 $K_{c,tgs}$ 解密得到 $K_{c,v}$ 和随机数 2，验证随机数 2 是否与自己发给 TGS 的随机数相同，如果相同则可以确信这是新报文。用户向应用服务器发送请求报文，希望获得应用服务器的服务。该报文包括用户名称、用户 IP、TS 票据、用户的证书和 $K_{c,v}$ 加密的认证符 2 等信息。认证符 2 内容包括用户名称、用户 IP 和随机数 3。

V→C：AP_REP= $\{EK_{c,v}(Nounce3, ID_c, IP_c, ID_v)\ , Cert_v\}$

应用服务器 V 先检查证书 $Cert_c$ 的可信性，从 TS 票据中得到角色值和 $K_{c,v}$，然后用 $K_{c,v}$ 来解密认证符 2。当用户的身份验证成功后，给用户发送应答报文。应答报文由应用服务器的证书和用 $K_{c,v}$ 加密的认证信息组成，该认证信息包括随机数 3、用户名称、用户 IP 和应用服务器名称等信息组成。当用户验证了应用服务器 V 的身份后，便可以获得其服务。当用户访问应用服务器时，应用服务器就可以根据该用户角色值来授予用户相应的服务。

3.3 改进协议的性能分析

从以上描述及交换过程来看，改进后的 Kerberos 协议与原有的协议相比较，在一定程度上克服了原有协议的一些局限性，其在安全性和实用性方面得到了很大的提高，主要表现在以下几个方面：

(1)在原协议中会话密钥 $K_{c,tgs}$ 是由用户的密钥 K_c 加密的，由于 K_c 是通过用户输入的口令导出的，容易被猜测和窃听。改进后的协议则是通过用户的公钥加密 $K_{c,tgs}$，并用 AS 的私钥来签名，通信都通过公钥体制进行加密解密，能够确保身份的真实性，这样对系统的安全性有了一定的增强，有效的解决了口令猜测攻击。

(2)在 Kerberos 协议中，数据库中存放了与用户通信用的密钥，一旦数据库被

非法访问，就会产生很大的问题。而在改进协议中，数据库中只保存公钥证书和证书撤销列表，私钥没有保存在数据库中，即使数据库被非法访问，也不会有很大的安全隐患。

(3)采用 USB Key 双因子认证，应用 USB Key 技术来保存用户的私钥，使得私钥泄露的可能性大大降低，用户只要携带 USB Key 便可以通过任何具有 USB 接口的计算机接入网络，向服务器认证身份。提高了认证系统的安全性，减少了私钥泄露的可能性，用户可以合法、安全地使用网络资源。

(4)原协议通过时间戳来防止重放攻击，但是要保证客户端、认证服务器、授权服务器和应用服务器的机器时间保持一致，这在当前环境下是很难达到的。在改进的协议里采用了随机数来替代时间戳，通过随机数来表示该报文是新的，避免了网络中时间难于同步带来的问题。

(5)添加了授权的功能，改进协议是通过票据来加载基于角色的访问控制信息实现，在实现用户认证的同时实现了授权服务。

(6)票据的不可否认性。改进协议中的票据是先用票据使用者的公钥加密再用票据颁发者的私钥签名，攻击者就无法解密也不能修改和伪造它。票据使用者收到票据后，用票据颁发者的公钥验证签名再用自己的私钥解密，就可以确认该票据是否由票据颁发者开出，实现了双向认证。

第 4 章 基于改进协议的认证系统的设计

本文前面几章主要研究了 PKI 技术、Kerberos 协议和其公钥扩展协议，并把 PKI 技术、访问控制与 Kerberos 协议相结合，提出了一种基于公钥的 Kerberos 改进协议。本章在该改进协议的基础上，建立一个安全高效、易于扩展和具有实用性的身份认证系统模型。模型设计目标如下：

(1)认证过程对用户来说是透明的，不需要用户的参与，是由各个功能模块共同完成的。

(2)实现身份的双向认证。

(3)采用 USB Key 双因子认证，应用 USB Key 技术来保存用户的私钥，减少用户信息泄露的可能性。

(4)传输的数据都应是密文，增强系统的安全性。

本章将首先阐述认证模型的总体设计方案，详细介绍认证系统的流程，其后分别就客户端和服务器端的设计给予必要的说明。

4.1 认证模型的总体设计方案

认证系统模型由客户端和服务器端和证书管理中心三个部分组成。服务器端为 AS 服务器、TGS 服务器和应用服务器组成。证书管理中心为可信第三方，负责给用户和服务器发放证书和密钥。系统模型如图 4-1 所示：

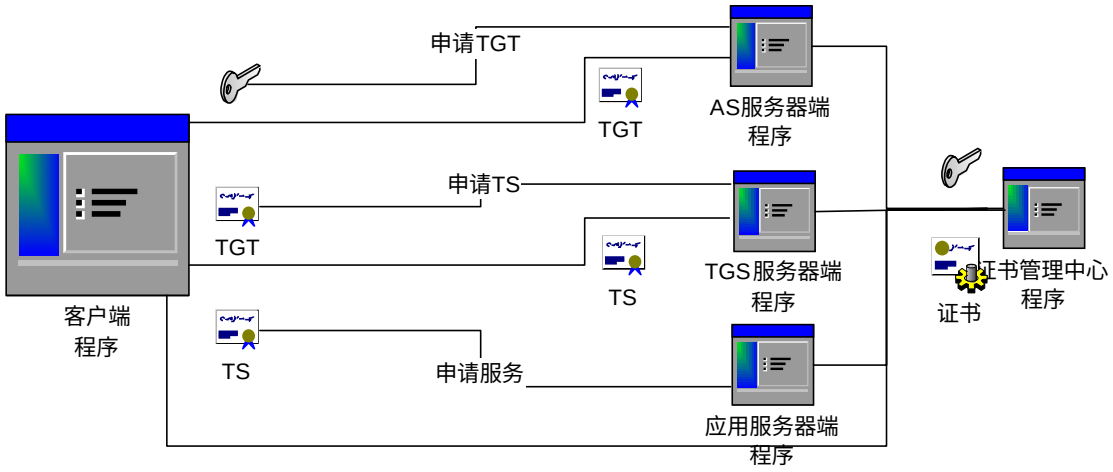

图 4-1 认证系统模型

本认证系统模型要求客户端与应用服务器、TGS 服务器、AS 服务器均有数据

交换，具体工作流程可以如下图 4-2 所示：

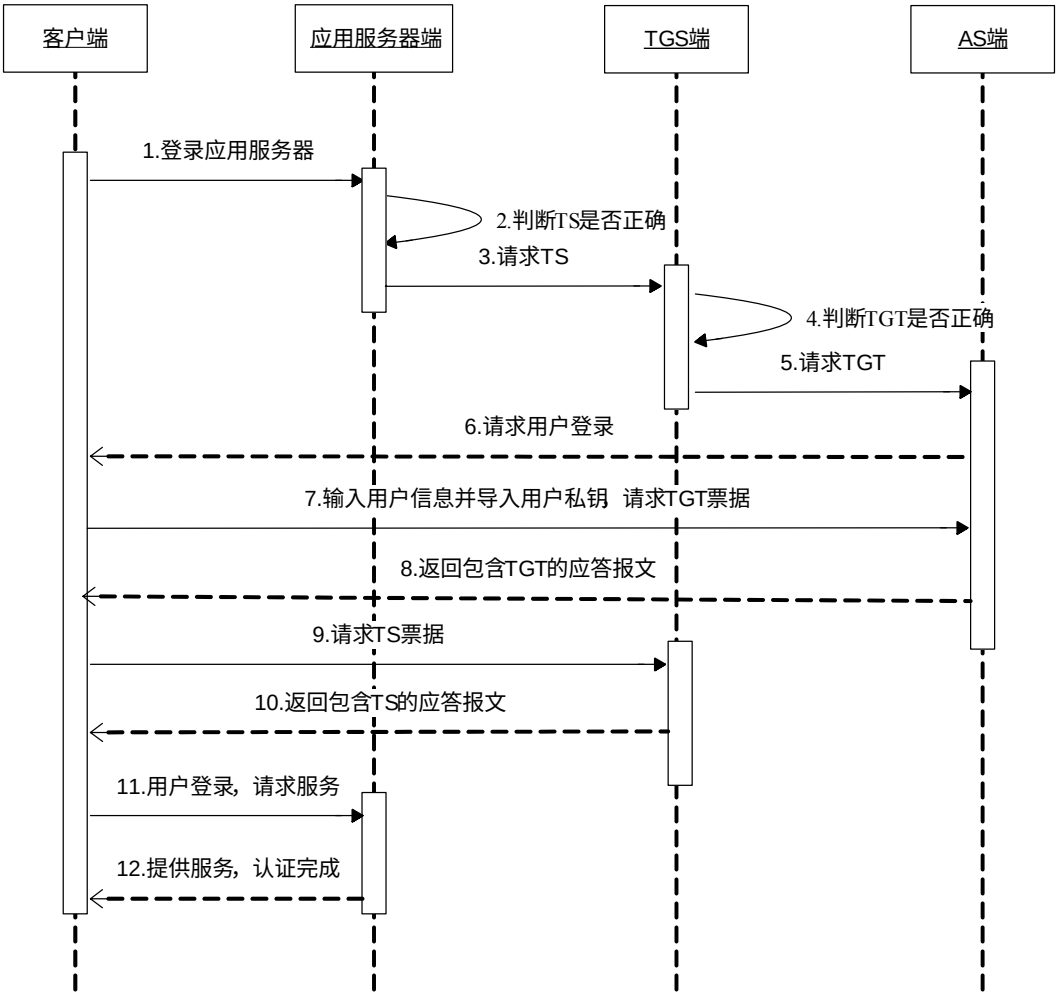

图 4-2 认证系统流程

(1)用户访问应用服务器；

(2)应用服务器判断 TS 票据，如果没有或票据错误，则定向到 TGS 请求 TS 票据，如果用户发送的是有效 TS 票据，则向用户提供服务，认证结束；

(3)TGS 判断 TGT 票据，如果没有或票据错误，则定向到 AS 端请求 TGT 票据；

(4)AS 服务器请求用户登录；

(5)用户在客户端输入服务器名称、用户名称和服务器 IP 并导入用户私钥，向 AS 服务器请求票据许可票据 TGT；

(6)AS 服务器验证用户信息，若信息正确，给用户发送包括票据许可票据 TGT 的应答报文，如用户信息不正确则跳出登录；

(7)客户端接收到 AS 的应答报文后，向 TGS 服务器请求服务许可票据 TS，请

求报文包括 TGT 和认证符等信息；

(8)TGS 服务器收到报文后，验证票据和用户信息，若正确，则给用户发送包括 TS 和角色值的应答报文，如信息不正确则跳出登录；

(9)客户端接收到 TGS 的应答报文后，向应用服务器请求服务；

(10)应用服务器验证信息，提供服务，认证结束。

4.2 客户端的设计

客户端主要包括两方面：证书模块和认证模块。客户端的证书模块至少应具有以下两个功能：申请证书、下载证书和私钥。客户端的认证模块要完成与 AS、TGS 及应用服务器的票据请求和票据接收。

申请证书就是用户向证书管理中心申请注册证书。用户首先要登记自己的基本信息，证书管理中心对用户的信息进行审核，确认信息无误后，就接受用户的证书申请。用户除了要提出申请外，还需到证书管理中心办理证件审核，以备证书的发放。当用户确认申请证书成功之后，客户端的用户就可以下载证书管理中心生成的证书和私钥。

客户端的认证模块需要发出三种请求：AS 请求、TGS 请求和应用服务器请求。AS 请求需要实现的是向 AS 请求票据许可票据 TGT，当 AS 确认了用户的身份后就会向用户发放票据 TGT，否则认证结束。TGS 请求是向 TGS 请求服务许可票据 TS，如果用户取得了 AS 发放的 TGT 票据，同时用户的身份被确认的话，TGT 就会向用户发放服务许可票据 TS，否则认证结束。应用服务器请求是向应用服务器申请服务，当用户取得了 TGT 发放的服务许可票据 TS，同时用户身份再次确认的话，用户就会得到应用服务器提供的服务。

4.3 服务器端的设计

服务器端主要包括两方面：数据库模块和认证模块。服务器端的数据库模块主要是用来保存公钥证书和证书撤销列表，服务器端认证模块主要完成对用户的请求的处理。

在系统模型中，数据库中保存了公钥证书和证书撤销列表，当收到用户的请求服务时，服务器程序从数据库中取出相应用户的公钥进行验证。

服务器端的认证模块需要对三种请求进行处理：对 AS 请求的处理、对 TGS 请求的处理和对应用服务器请求的处理。对 AS 请求的处理是向用户发放票据许可

票据 TGT。当 AS 接收到用户的请求报文后，确认用户身份后就向用户发放 TGT。对 TGS 请求的处理是向用户发放请求服务票据 TS，当 TGS 接收到用户的请求报文，先检验 TGT 的正确性，而且用户的身份被确认的话，TGT 就会向用户发放服务许可票据 TS。对应用服务器请求的处理是向用户提供服务，应用服务器接收到用户的请求报文后，检验 TS 的正确性，并且用户身份确认后，应用服务器就会向用户提供服务。

4.4 证书管理的设计中心

证书管理中心模块主要包括这几个方面：证书申请、证书与私钥的发放、证书查询和证书撤销。证书申请模块的主要功能就是对用户的注册申请进行审核，当确认信息无误后，完成注册申请。证书与私钥的发放模块就是给用户和服务器发放证书和私钥。证书查询模块主要是检查证书撤销列表 CRL 中是否包含用户或服务器要查询的证书。证书撤销模块的作用是收到撤销请求后撤销证书并发布到证书撤销列表 CRL。

第 5 章 基于改进协议的认证系统的实现

本章将结合上一章提出的改进协议的认证模型及其设计思想，以此为基础，描述认证系统的实现过程，包括开发平台的选择、主要功能模块的实现以及在应用系统中的集成。

5.1 开发平台

软件平台：Windows NT/Windows XP/ Windows 2000 系列
硬件平台：通用服务器或者配置较高的 PC 机
开发工具：eclipse、WebLogic、j2sdk 、Microsoft SQL Server 2000
开发语言：java

5.1.1 javax.crypto

软件包 javax.crypto 为 cryptographic（加密）操作提供类和接口[34]。此包支持安全流和封装的对象，此包中定义的 cryptographic 操作包括加密、密钥生成、密钥协商以及消息验证代码 (MAC) 生成。

软件包 javax.crypto 为 cryptographic 操作提供以下重要的接口和类：

SecretKey：秘密（对称）密钥的接口。

Cipher类：提供了针对加密和解密的密码 cipher 功能。

CipherInputStream类：由一个 InputStream 和一个 Cipher 组成，这样 read()方法才能返回从基础 InputStream 读入但已经由该 Cipher 另外处理过的数据。

CipherOutputStream类：由一个 OutputStream 和一个 Cipher 组成，这样 write()方法才能在将数据写出到基础 OutputStream 之前先对该数据进行处理。

CipherSpi类：为 Cipher 类定义了服务提供程序接口(SPI)。

EncryptedPrivateKeyInfo类：实现 EncryptedPrivateKeyInfo 类型，如在 PKCS #8 中定义的那样。

ExemptionMechanism类：提供了豁免机制的功能，例如，密钥恢复、密钥唤醒和密钥托管。

ExemptionMechanismSpi类：为 ExemptionMechanism 类定义了服务提供程序接口

(SPI)。

KeyAgreement类：提供密钥一致性（或密钥交换）协议的功能。

KeyAgreementSpi类：为 KeyAgreement 类定义了服务提供程序接口 (SPI)。

KeyGenerator类：提供（对称）密钥生成器的功能。

KeyGeneratorSpi类：为 KeyGenerator 类定义了服务提供程序接口 (SPI)。

Mac类：提供“ 消息验证代码 ”(MAC) 算法的功能。

MacSpi类：为 Mac 类定义服务提供程序接口 (SPI)。

NullCipher类：是一个提供“标识密码”的类，其不转换纯文本。

SealedObject类：使程序员能够用加密算法创建对象并保护其机密性。

SecretKeyFactory类：表示秘密密钥的工厂。

SecretKeyFactorySpi类：定义 SecretKeyFactory 类的服务提供程序接口 (SPI)。

BadPaddingException异常：当输入数据期望特定的填充机制而数据又未正确填充时，抛出此异常。

ExemptionMechanismException异常：此为一般 ExemptionMechanism 异常。

IllegalBlockSizeException异常：如果提供给块密码的数据长度不正确（即与密码的块大小不匹配），则抛出此异常。

NoSuchPaddingException异常：当请求特定填充机制但该环境中未提供时，抛出此异常。

ShortBufferException异常：当用户提供的输出缓冲区太小而不能存储操作结果时，抛出此异常。

5.1.2 java.security

软件包 java.security 为安全框架提供类和接口，包括那些实现了可方便配置的、细粒度的访问控制安全架构的类[34]。此包也支持密码公钥对的生成和存储，以及包括信息摘要和签名生成在内的可输出密码操作。最后，此包提供支持 signed/guarded 对象和安全随机数生成的对象。

java.security 包为安全框架提供以下重要的接口和类：

Key：是所有密钥的顶层接口。

KeyStore.Entry：用于 KeyStore 项类型的标记接口。

PrivateKey：私钥接口。

PublicKey：公钥接口。

Principal接口：表示主体的抽象概念，它可以用来表示任何实体，例如，个人、公司或登录 ID。

AccessControlContext：AccessControlContext 用于基于它所封装的上下文作出系统资源访问决定。

AccessController：用于与访问控制相关的操作和决定。

AlgorithmParameterGenerator：用于生成要在某个特定算法中使用的参数集合。

DigestInputStream：使用通过流的位更新关联消息摘要的透明流。

DigestOutputStream：使用通过流的位更新关联消息摘要的透明流。

KeyFactory：密钥工厂是用来将 keys（Key 类型的不透明加密密钥）转换成 key 规范（基础密钥材料的透明表示），反之亦然。

KeyFactorySpi类：为 KeyFactory 类定义服务提供程序接口 (SPI)。

KeyPair类：是简单的密钥对（公钥和私钥）持有者。

KeyPairGenerator类：用于生成公钥和私钥对。

KeyPairGeneratorSpi类：为用来生成公钥和私钥的 KeyPairGenerator 类定义了服务提供程序接口 (SPI)。

KeyStore类：表示密钥和证书的存储设施。

KeyStoreSpi类：为 KeyStore 类定义服务提供程序接口 (SPI)。

MessageDigest类：为应用程序提供信息摘要算法的功能，如 MD5 或 SHA 算法。

MessageDigestSpi类：为 MessageDigest 类定义服务提供程序接口 (SPI)，MessageDigest 类提供信息摘要算法的功能，如 MD5 或 SHA。

Provider类：表示 Java 安全 API "provider"，这里 provider 实现了 Java 安全性的一部分或者全部。

SecureRandom类：提供加密的强随机数生成器 (RNG)。

SecureRandomSpi类：为 SecureRandom 类定义了服务提供程序接口 (SPI)。

Signature类：用来为应用程序提供数字签名算法功能。

SignatureSpi类：为 Signature 类定义了服务提供程序接口 (SPI)，可用来提供数字签名算法功能。

以上介绍的两个类包本身定义了应用程序可以写入的编程接口。从而实现本身可以由独立的第三方厂商来写，可以根据需要无缝的插入。

5.2 功能模块的实现

本认证系统主要解决域内的身份认证问题。认证系统的模块结构图可如图 5-1 所示：

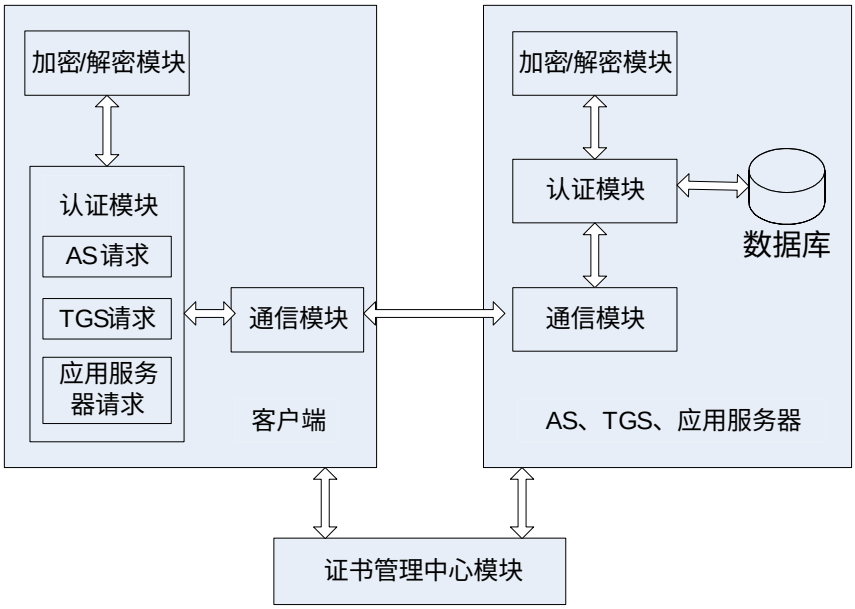

图5-1 系统模块结构图

系统由客户端和服务器端和证书管理中心三个部分组成。其中，客户端和服务器端都是由加解密模块、认证模块和通信模块构成。服务器端包括 AS 服务器、TGS 服务器和应用服务器等。认证系统中的证书管理中心模块是作为可信第三方，主要功能是：生成和管理数字证书，并负责给用户和服务器发放证书和密钥。

下面从模块的角度，阐述几个重要模块的实现过程。

5.2.1 加密/解密模块功能实现

认证系统的加密/解密模块主要包括有：非对称密钥模块、对称密钥模块和数字签名模块。

加密/解密模块功能结构如下图 5-2 所示：

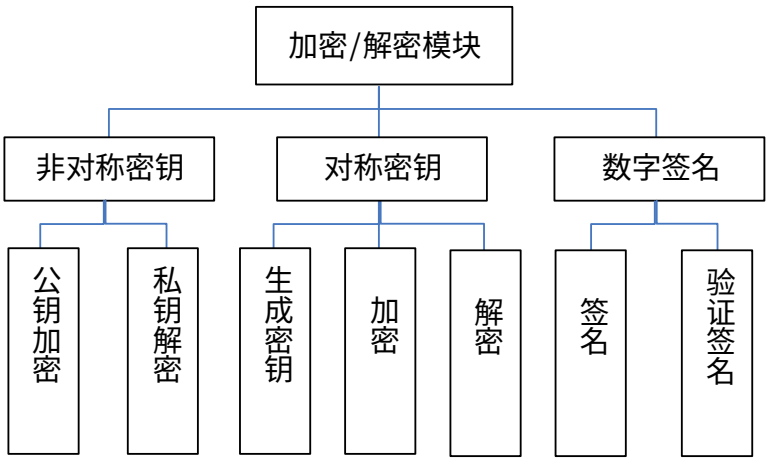

图5-2 加密/解密模块功能结构图

(1)非对称密钥模块

非对称密钥就是加密时使用一种密钥，解密时使用另一种密钥，就是使用公钥进行加密和使用私钥对加密的信息进行解密。在本认证系统中，每个用户都有一对非对称密钥，用户的公钥可对所有人公开，而私钥需用户自己持有，保存在 USB Key 中。

a.使用 RSA 公钥加密的执行过程如下：

①从后缀名为.dat 的文件中读取公钥并强制转换为 RSAPublicKey 类型，以便读取参数；

②通过 RSAPublicKey 类中的 getPublicExponent()和 getModulus()方法来分别获得公钥参数 e，n；

③获取明文 m；

④计算公式 m^e mod n 的值即为密文。

b.使用 RSA 私钥解密的执行过程如下：

①读取密文；

②从后缀名为.dat 的文件中读取私钥并强制转换为 RSAPrivateKey 类型，以便读取参数；

③通过 RSAPrivateKey 类中的 getPrivateExponent()和 getModulus()方法来分别获得私钥参数 d，n；

④计算公式 c^d mod n 的值即为明文。

下面介绍一个解密实例：

```
//从 Ek.dat 密文文件中读取密文
BufferedReader in=new BufferedReader(new InputStreamReader(new FileInputStream(“Ek.dat”)));
String ctext=in.readLine();
BigInteger c=new BigInteger(ctext);
//获取私钥
FileInputStream f=new FileInputStream(“Pri_user.dat”);
ObjectInputStream b=new ObjectInputStream(f);
RSAPrivateKey prk=(RSAPrivateKey)b.readObject();
//获取私钥的参数(d, n)
BigInteger d=prk.getPrivateExponent();
BigInteger n=prk.getModulus();
//执行计算
BigInteger m=c.modPow(d,n);
```

(2)对称密钥模块

对称密钥就是加密和解密使用相同的密钥。在改进协议中，我们生成了 $K_{c,tgs}$ 和 $K_{c,v}$ 两个会话密钥，并使用这两个对称密钥来加密和解密认证信息。下面我们来讲述对称密钥的生成和加解密。

a.对称密钥生成的执行过程如下：

①通过 Java 中 KeyGenerator 类中的 getInstance()方法获取密钥生成器；

②初始化密钥生成器，指定密钥的长度；

③通过 KeyGenerator 类中的 generateKey()方法来生成密钥；

④将密钥保存在文件中。

b.使用对称密钥加密的执行过程如下：

①在文件中获取密钥；

②通过 Java 中 Cipher 类中的 getInstance()方法创建密码器；

③初始化密码器；

④获取等待加密的明文；

⑤通过 Cipher 类中的 doFinal()方法执行加密。

c.使用对称密钥解密的执行过程如下：

①获取密文；

②在文件中获取密钥；

③通过 Java 中 Cipher 类中的 getInstance()方法创建密码器；

④初始化密码器；

⑤通过 Cipher 类中的 doFinal()方法执行解密。

下面介绍一个对称密钥生成实例：

```
//获取密钥生成器
KeyGenerator kg = KeyGenerator.getInstance("DESede");
//初始化密钥生成器
kg.init(168);
//生成密钥
SecretKey k = kg.generateKey();
//保存到文件
kb = k.getEncoded();
FileOutputStream f1=new FileOutputStream("Kc,tgs.dat");
f1.write(kb);
```

(3)数字签名模块

数字签名是非常重要的，它可以验证数据的身份。发送方用自己的私钥签名，验证签名是数字签名的反过程，在接到对方的签名信息后，要用对方的公钥才能

解开签名信息，就可以确认该数据是否由发送方开出。在改进协议中，采用了数字签名来确保数据的不可否认性。

a.使用私钥进行数字签名的执行过程如下：

①获取要签名的数据；

②获取私钥；

③使用 Signature 类中的 getInstance()方法来获取 Signature 对象；

④用私钥初始化 Signature 对象；

⑤传入要签名的数据；

⑥执行签名。

b.使用公钥验证数字签名的执行过程如下：

①获取要签名的数据；

②获取签名；

③读取公钥；

④使用 Signature 类中的 getInstance()方法来获取 Signature 对象；

⑤用公钥初始化 Signature 对象；

⑥传入要签名的数据；

⑦验证签名。

下面介绍一个使用私钥进行签名的实例：

```
//获取要签名的数据(msg.dat)
FileInputStream f=new FileInputStream(“msg.dat”);
Int num=f.available();
Byte[] data=new byte[num];
f.read(data);
//获取私钥 Pri_AS.dat
FileInputStream f2=new FileInputStream(“Pri_AS.dat”);
ObjectInputStream b=new ObjectInputStream(f2);
RSAPrivateKey prk=(RSAPrivateKey)b.readObject();
//获取 Signature 对象
Signature s=Signature.getInstance(“MD5WithRSA”);
//用私钥初始化 Signature
s.initSign(prk);
//传入要签名的数据
s.updata(data);
//执行签名
```

Byte[] signeddate=s.sign();

5.2.2 通信模块功能实现

客户端与服务器之间都需要进行数据通信，通信模块分为客户端的通信模块和服务器端的通信模块。在本系统中，客户端与服务器端之间的通信采用了套接字，用 Socket 类来实现客户端的套接字，利用 ServerSocket 类来实现服务器端套接字。

(1)客户端的执行过程如下：

①打开通信通道，并连接到服务器在主机的特定端口；

②向服务器发出请求报文，等待服务器的应答报文，继续提出请求；

③收到服务器发来的应答信号，请求结束后关闭信道。

(2)服务器端的程序必须要比客户端的先启动，并根据请求提供相应服务，直到通信结束。服务器端的执行过程如下：

①打开通信通道，并通知本地主机在某端口接收客户端的请求；

②监听端口，直到客户端的请求到达指定端口；

③接收到请求，启动一个进程来处理用户请求并发送应答信号，服务完成后，关闭此进程与客户端的通信链路；

④继续监听端口，等待客户端的请求；

⑤关闭服务器。

下面通过一个服务器端套接字实例来说明：

```
//创建一个名为 server 的 ServerSocket 连接
private ServerSocket server;
server=new ServerSocket(5532);
//监听设定的端口，接受连接请求，建立 Socket 连接
private Socket client=null;
client=server.accept();
//定义一个输出流
out=new DataOutputStream(client.getOutputStream());
//定义一个输入流
in=new InputStreamReader(client.getInputStream);
//服务器停止
server.close();
```

5.2.3 认证模块功能实现

认证模块主要包括客户端认证请求模块、AS 服务器端模块、TGS 服务器端模块和应用服务器端模块，通过调用加密/解密模块，来分别完成认证的六个过程。客户端认证请求模块包括 AS 请求模块、TGS 请求模块和应用服务器请求模块。认证模块功能结构如下图 5-3 所示：

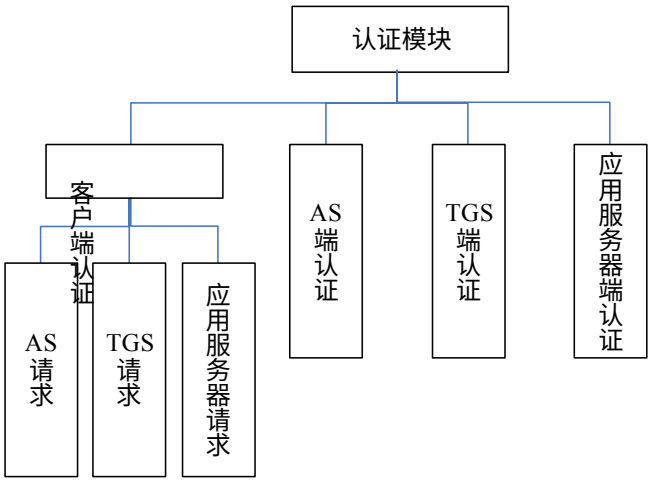

图5-3 认证模块功能结构图

(1)客户端认证请求模块的实现

用户通过客户端来进行系统的登录，用户登录只需通过输入信息和导入密钥，客户端与服务器端的认证信息和授权服务等交换对用户来说都是透明的，整个过程由认证客户端自动完成。客户端的认证请求模块由客户端的 AS 请求模块、TGS 请求模块和应用服务器请求模块组成。客户端认证模块工作流程如下图 5-4 所示：

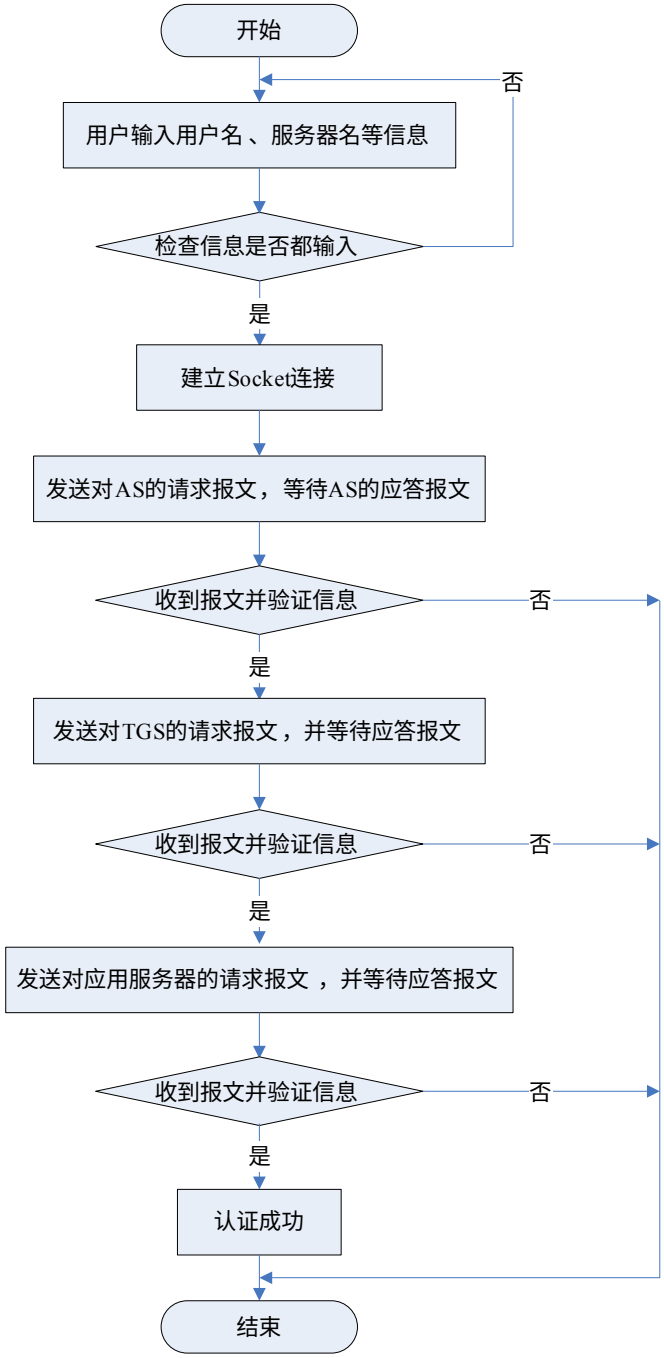

图5-4 客户端认证请求模块流程图

a.AS 请求模块：

客户端的 AS 请求模块的功能是向 AS 端认证模块发送票据请求，请求获得访问 TGS 的票据 TGT。AS 请求模块的执行过程如下：

①用户在客户端的登录界面输入服务器名称、用户名称和服务器 IP，并导入用户私钥；

②验证信息是否都输入，如果没有则需要重新登录；

③建立 Socket 连接；

④生成认证信息，向认证服务器 AS 发送请求报文，请求获得票据许可票据

TGT；

⑤收到 AS 的应答报文 AS_REP，验证信息；

⑥验证认证信息，信息正确则获得 TGT 和 $K_{c,tgs}$ 并运行 TGS 请求模块，否则认证失败。

b.TGS 请求模块：

客户端的 TGS 请求模块的功能是向授权服务器 TGS 发送票据请求，请求获得访问应用服务器的服务许可票据 TS。TGS 请求模块的执行过程如下：

①验证 AS 的应答报文后，生成请求报文；

②发送请求报文给 TGS 服务器，请求获得服务许可票据 TS；

③收到 TGS 的应答报文，验证信息；

④验证信息，信息正确则获得 TS 和 $K_{c,v}$ 并运行应用服务器请求模块，否则认证失败。

c.应用服务器请求模块：

客户端的应用服务器请求模块的功能是向应用服务器端发送请求服务，请求获得应用服务器提供的服务。应用服务器请求模块的执行过程如下：

①验证 TGS 的应答报文后，生成请求报文；

②向应用服务器发送请求报文，请求获得服务；

③收到应用服务器的应答报文，验证信息；

④验证信息正确则根据角色值来获得应用服务器提供的服务。

(2)AS 端认证模块的实现

AS 端认证模块用于初始认证用户是否合法，并为合法用户发放票据许可票据 TGT。

AS 端认证模块的工作流程如下图 5-5 所示：

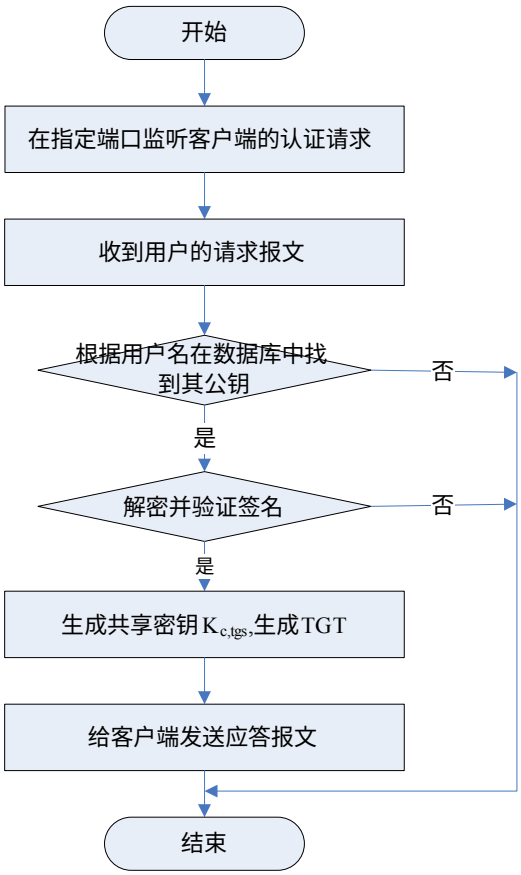

图5-5 AS端认证模块流程图

AS 端认证模块的执行过程如下：

①AS 服务器启动，在指定端口堵塞等待客户端的连接；

②当收到用户的请求报文 AS_REQ 后，解读报文；

③先检查用户的证书 $Cert_c$ 的可信性，再根据用户名称在数据库中取得用户的公钥，如用户证书不可信或数据库中没有该用户名则认证失败；

④用自己的私钥解密，用户的公钥来验证签名，得到认证信息内容，验证信息，如信息不正确则认证失败；

⑤生成用户和 TGS 的共享密钥和 TGT 票据；

⑥发送应答报文 AS_REP 给客户端。

票据许可票据 TGT 的数据结构为：

$Ticket_{tgs}$= $EKpri_{as}$ ($EKpub_{tgs}$ (ID_c, IP_c, ID_{tgs}, Nounce1, Times, $K_{c,tgs}$))

其中 ID_c 为用户名称，IP_c 为用户 IP，ID_{tgs} 为 TGS 名称， Nounce1 为随机数 1，Times 为当前时间（1970 年 1 月 1 日以后的毫秒数），$K_{c,tgs}$ 为用户和 TGS 服务器的会话密钥，$Kpri_{as}$ 为 AS 的私钥，$Kpub_{tgs}$ 为 TGS 的公钥。

下面是在 AS 给用户发送应答报文时截取到的 TGT 加密后的一组数据实例：

TGT=313437373636373833432363973137363636303736323730383931313035 3

2363938353230323038353636353032383937363431343939303533373631303032343638383738363735363734343839343437393433333233383239323830333530333539343636303032393932343736343434393631333837353932353537393439363234383230373033353139373836313932313039355633383735373735313235363634323939393132333534333935343038333738333533313639383036333830333833333037303334343935303438313631333236343131303834323438303536343934333733383930323630373738303139363235383635393035343537353431373734343632343033323730343138313339383836380343635393736373638323335393931333934363933313851ce3b8b8c6fb6e5397bba40953ab01b9e1d13c9c762e0847afe5f2c71c9cf8ca68f9dc3364e77597eaded134416fa7c0ea947c4dd3669c38205884130337548ce22e0b03044273059d0e06d63a89be07427cfbfb9bd840348e9ffd446a17d40ba133ebd7e18c537649f2e843f2ba8b12958cbd4b8902b51dbfee55223fcf312；

解密后，TGT= user, 10.1.25.59, tgs, 492, 1205598764906, 8083f446c438299bd5efe9ad156e 43130ba23720194040c2，在票据中以“，”分开内容，依次为用户名称、用户 IP、TGS 名称、随机数 1、当前时间和 $K_{c,tgs}$。

(3)TGS 端认证模块的实现

TGS 端认证模块给合法用户提供其申请的服务许可票据 TS。模块的执行过程如下：

①TGS 服务器启动，在指定端口堵塞等待客户端的连接；

②当收到用户的请求报文 TGS_REQ 后，解读报文；

③先检查用户的证书 $Cert_c$ 的可信性，如用户证书不可信，认证失败；

④取出 TGT 票据，用自己的私钥解密，AS 的公钥来验证签名，得到会话密钥 $K_{c,tgs}$。用 $K_{c,tgs}$ 解密认证符，验证信息是否正确，如信息不正确则认证失败；

⑤再根据用户名称在数据库中取出角色值(根据角色值便可以确定该用户的访问权限)；

⑥生成用户和应用服务器的共享密钥 $K_{c,v}$ 和 TS 票据；

⑦发送应答报文 TGS_REP 给客户端。

服务许可票据 TS 的数据结构为：

$Ticket_v = EKpri_{tgs}(EKpub_v(ID_c, IP_c, Rolevalue, Times, K_{c,v}))$

其中 ID_c 为用户名称，IP_c 为用户 IP，Rolevalue 为该用户的角色值，Times 为当前时间，$K_{c,v}$ 为用户和应用服务器的会话密钥，$Kpri_{tgs}$ 为 TGS 的私钥，$Kpub_v$ 为应用服务器的公钥。

下面是在 TGS 给用户发送应答报文时截取到的 TS 加密后的一组数据实例：

TS=3634303039323736323330303736323330313930323832313035363431323137303631383934383932393139343036353432303234393630393134363030353939343430323334383934353734373831333633303638323536353433383132323338393734337434373831373833383130343337343037393338343730303139363733373534333339363131333633313831303335333838393438383239353932353434333733313834323736313532393634313833333835303732333439323138373434353732363335385381303738323631303039373533393437353237323735343231363338353231313136333936373233313331333438333132313238353234393530353438393132383730333238383732353733353335343834383238323435333232383133333039393631303632343632e398e2d0660c82a6f8036582b652e411a78d6454b8739eff7c55be69d0b03f3e9dad75545f44b51b945b3c9c8f505e282cc8acece1416cbc530524d9604a17e20d6d0281d2c369e2c7c4d87e950def09cf882304c03d8b8f5ad074557051e216f46c5372f279e1f7c62846a67a625791137c42dfa6c30c90b1555c94830a3f9

解密后，TS=user, 10.1.25.59, 11, 1205598765406, 68e95129c710801ac83de35457622ab0fbb9dc1c80316ec2，依次为用户名称、用户 IP、角色值、当前时间和会话密钥 $K_{c,v}$。

(4)应用服务器端认证模块的实现

应用服务器端模块的执行过程如下：

①应用服务器启动，在指定端口堵塞等待客户端的连接；

②当收到用户的请求报文 AP_REQ 后，解读报文；

③先检查用户的证书 $Cert_c$ 的可信性，如用户证书不可信，认证失败；

④取出 TS 票据，用自己的私钥解密，AS 的公钥来验证签名，得到会话密钥 $K_{c,v}$ 和角色值。用 $K_{c,v}$ 解密认证符，验证信息是否正确，如信息不正确则认证失败；

⑤用 $K_{c,v}$ 加密认证信息；

⑥发送应答报文 AP_REP 给客户端；

⑦用户收到应答报文，解密，验证应用服务器的身份后，认证完成后，应用服务器就可以根据得到的角色值来授予用户相应的服务。

5.2.4 证书管理中心模块功能实现

证书管理中心模块包括申请证书、证书与私钥的发放、证书查询和证书撤销等。证书管理中心模块功能结构如下图 5-6 所示：

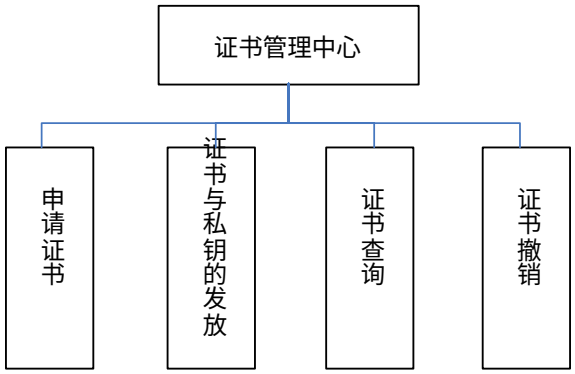

图5-6 证书管理中心模块功能结构图

证书管理中心的实现使用了 keytool，keytool 是个密钥和证书管理工具，它使用户能够管理自己的公钥/私钥对及相关证书。Keytool 创建的数字证书是以条目的形式存入密钥库，密钥库中的一个条目包含该条证书的私钥、公钥和对应的数字证书的信息。密钥库可以导出数字证书文件和私钥，数字证书文件只包括主体信息和对应的公钥。

下面介绍一下 Keytool 的一些常用命令：

-genkey：在用户主目录中创建一个默认文件".keystore"，还会产生一个 mykey 的别名，mykey 中包含用户的公钥、私钥和证书；

-alias：产生别名；

-keystore：指定密钥库的名称(产生的各类信息将不在.keystore 文件中；

-keyalg：指定密钥的算法；

-validity：指定创建的证书有效期多少天；

-keysize：指定密钥长度；

-storepass：指定密钥库的密码；

-keypass：指定别名条目的密码；

-dname：指定证书拥有者信息；

-list：显示密钥库中的证书信息；

-v：显示密钥库中的证书详细信息；

-export：将别名指定的证书导出到文件；

-file：参数指定导出到文件的文件名；

-delete：删除密钥库中某条目；

-keypasswd：修改密钥库中指定条目口令；

-import：将已签名数字证书导入密钥库。

（1）申请证书

申请证书界面如下图 5-7 所示：

图 5-7 申请证书界面

用户向证书管理中心申请注册时，先登记用户的基本信息，例如：用户名、所在单位、所在组织名称和所在城市等等，其中用户名必不可少。证书管理中心对用户的信息进行审核，确认信息无误后，接受用户的证书申请。用户除了要网上申请注册外，还需到证书管理中心办理证件审核，证件审核成功后才完成注册。用户的公钥、私钥和证书是同时生成并保存在密钥库中。密钥库是存放密钥和证书的文件，一个密钥库是一个文件组成，在首次创建时，密钥库对应的文件如果不存在则自动创建。由于密钥库包含了私钥，需要保密，因此设立了一个密钥库密码。第一次使用该密钥库时输入的密码将成为该密钥库默认的密码，以后使用该密钥库时必须输入该口令才可以使用。使用该证书别名可以在密钥库中找到相应的公钥、私钥和证书。私钥的密码是对应于该用户的私钥的密码，密钥库中每个用户名可以使用不同的私钥密码。密钥库中的一个条目就包含了一个用户的私钥、公钥和对应的数字证书的信息。

(2)证书与私钥的发放

当用户申请证书成功之后，用户就可以向证书管理中心请求发放证书和私钥。使用 keytool 的-export 参数可以将条目名指定的证书导出到文件，文件名通过-file 参数指定。公钥可以使用 getPublicKey()方法来获得，并返回 PublicKey 类型的对象。私钥可以通过执行 KeyStore 对象的 getKey()方法，获取其参数对应的条目的私钥。

(3)证书的查询

当证书管理中心收到客户端或服务器端发来的证书查询信息后，检查证书撤销列表 CRL 是否包含该证书，如果包含此证书则返回该证书无效，否则返回证书

有效。

(4)证书的撤销

在本认证系统中，证书的指定默认有效期为 365 天。证书的撤销有两种，一种是正常撤销，就是当前时间超过了证书的有效期；一种是没有超过证书的有效期，但是由于私钥泄露或用户名称的修改等原因，要提前撤销证书。证书管理中心收到用户的证书撤销请求后，对请求信息进行确认，确认完成后就撤销证书并发布到证书撤销列表 CRL。

5.3 在应用系统中的集成

认证系统与应用系统集成的目标：

1. 使用认证系统的安全机制来为应用系统提供安全保障；
2. 可以保留应用系统自身的各种安全机制，尽量少改动应用系统。

为了达到这个目标，对于认证系统与应用系统的集成，除上文描述过的各种模块之外，还需在应用服务器配置一个代理服务器。

认证系统与应用系统的集成示意图如下图 5-8 所示：

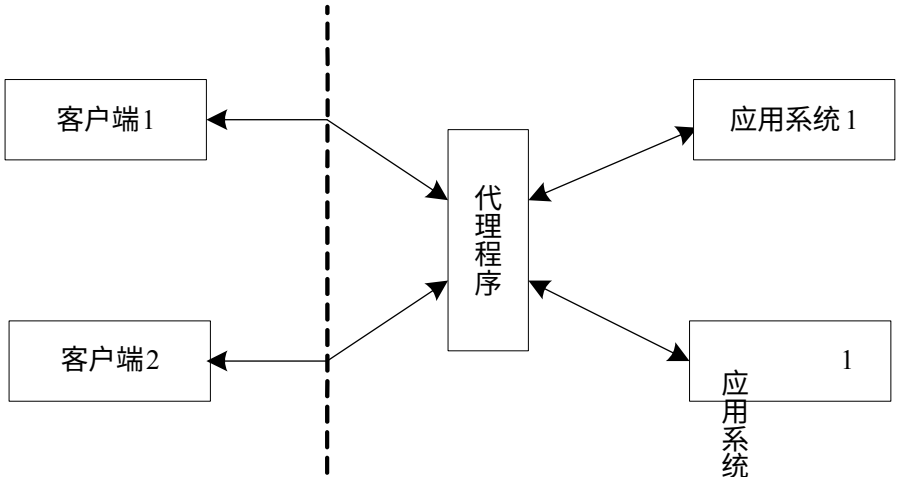

图5-8认证系统与应用系统集成示意图

设计思路：代理服务器上的代理程序给应用系统预留一个接口。该接口是个可供应用系统调用的方法或函数，对于用户来说是透明的。代理程序还要监听 TCP/IP 连接，如果发现通信端口的请求是访问应用系统的，代理程序就接收处理进行认证，如果认证成功，再通过接口给应用程序进行交互。代理程序同时要通过通信接口返回交互的响应信息。

用户端登录界面如下图 5-9 所示：

图5-9 客户端界面

用户首先输入信息并导入密钥后，单击“登录”按钮，开始身份认证过程。整个认证过程对用户来说是透明的，不需要用户的参与，是由各个功能模块共同完成的。身份认证成功后，应用服务器就可以根据该用户的角色值来为其提供服务。

第 6 章 结论

6.1 工作总结

随着计算机网络的发展和网络应用的迅速普及，网络安全越来越受到人们的重视。网络安全技术，特别是网络环境下的身份认证协议以及系统的安全性和可扩展性，已经成为了影响网络进一步发展的重要因素。本文以 Kerberos 协议为研究重点，围绕网络环境中身份认证协议的特点和应用环境，利用基于公钥体制的密码认证协议设计技术，对开放网络环境中的身份认证进行了全面的研究。研究的内容有以下方面：

1．对身份认证技术发展状况进行了研究，分析了目前几种常用的身份认证方式：基于口令的认证方式、基于物理证件的认证方式、基于动态口令的认证方式和基于生物特征的认证方式，并对几种身份认证协议：一次一密机制、Kerberos 认证系统和公钥认证体系进行了研究。

2．Kerberos 是 MIT 开发的基于 KDC 的认证系统，具有高安全性、可靠性、透明性、可伸缩性和很强的实用性等优点。本文对 Kerberos 的设计思想、认证过程进行了分析，指出了 Kerberos 协议的局限性，如：口令猜测攻击、重放攻击、时间同步问题、密钥的存储问题等。

3．PKI 即公开密钥基础设施，是一种遵循标准的利用公钥理论和技术建立的提供安全服务的基础设施。本文对 PKI 技术进行了深入的研究，并对传统 PKI 基本理论知识作了详细的分析。

4．在对 KerberosV5 及其多个扩展协议的研究基础上，把 PKI 技术、访问控制与 Kerberos 协议相结合，提出一个基于公钥密码体制的 Kerberos 改进协议，并对改进协议的性能进行分析。分析以及实验表明，在口令猜测攻击、重放攻击、时间同步、密钥的存储问题和票据的不可否认等性能方面有了很大的提高，改进协议采用了双因子认证的方式并添加了授权的功能。

5．在该改进协议的基础上，建立了一个安全高效、易于扩展和具有实用性的身份认证系统模型，阐述认证模型的总体设计方案和认证系统的流程。并对认证系统模型的模块结构和功能模块进行了设计和实现。该模型实现了对用户透明、双向认证、传输的数据为密文和使用双因子认证方式等目标，安全性和实用性得

到了较大的提高，能较好的满足网络的安全需求。

6.2 文章的主要贡献

在对 KerberosV5 及其多个扩展协议的研究基础上，提出一种基于公钥技术的 Kerberos 改进协议，并针对该改进协议，建立了身份认证系统模型。

(1)改进后的 Kerberos 协议在一点程度上克服了原有协议的一些局限性，主要表现在：口令猜测攻击、时钟同步问题、密钥的存储问题、票据的不可否认性等，改进后的 Kerberos 协议采用了双因子认证的方式和添加了授权的功能，安全性和实用性得到了很大的提高。

(2)针对该改进协议，建立身份认证系统模型，设计了认证模型的总体设计方案和流程，并对各个功能模块进行设计和实现。该模型实现了对用户透明、双向认证、传输的数据为密文和使用双因子认证方式等目标，安全性和实用性得到了较大的提高，能较好的满足网络的安全需求。

6.3 进一步的研究方向

尽管本文对 Kerberos 进行了一些改进，改进后的协议在一定程度上克服了原有协议的一些局限性，使得安全性和实用性得到了较大的提高，但是目前还是存在许多不足和未解决的问题，还有待进一步的提高和完善。

1．本系统采用了 RSA 公钥加密技术，使得在安全性得到了提高。但是 RSA 的基本运算是乘方取模，这种运算的特点是耗费的时间长度取决于乘方次数。算法强度越高，安全度越高，但是所耗费的系统资源和时间就越多。因此，在安全性的提高的同时，还要注意耗费时间和资源的问题。

2．数据库的安全问题，虽然服务器的数据库中没有保存用户的私钥，即使数据库被非法访问，也不会有很大的安全隐患，但是如果数据库数据出现问题的话，用户认证就不能继续进行，因此要确保数据库的安全。

3．本方案适用于企业网或校园网，还未能扩展到跨域等更大范围的网络环境。虽然 Kerberos 协议能实现跨域认证，但是由于时间关系，本认证系统还没有实现跨域认证。

参考文献

[1] 林柏钢.网络与信息安全教程[M].北京：机械工业出版社, 2004-07

[2] 郑纬民, 汤志忠.网络与信息安全概论(第二版)[M].清华大学出版社, 1998

[3] 叶锡君, 吴国新, 许勇等.一次性口令认证技术的分析与改进[J].计算机工程, 2000, (09)

[4] 包桂秋, 林喜荣, 苏晓生等.基于人体生物特征的身份鉴别技术发展概况[J].清华大学学报(自然科学版) , 2001, (Z1)

[5] Neuman B C, Ts'o T.Kerberos: an authentication service for computer networks [J]. IEEE Communications.1994, 32(9):33~38

[6] 陶翔.基于 ECC 和 USBKEY 的 Kerberos 安全改进方案[J].微计算机信息, 2007, 23(1)

[7] S. M. Bellovin, M. Merrit.Limitation of the authentication systems. Computer Communication Review, 1990, 20(5):119~132

[8] 关振胜.公钥基础设施 PKI 与认证机构 CA[M]. 北京：电子工业出版社, 2002

[9] Gary Tagg. Implementing a Kerberos Single Sign-on Infrastructure. IT Security Consultant, Tagg Consulting Ltd, 2002

[10] J. T. Kohl, B. C. Neuman.The Kerberos Network Authentication Service. Internet RFC 1510, 1993-09

[11] Bellovin S M, Merritt M. Limitation of Kerberos authentication system. Computer Magazine.1994, 32~38

[12] 李巍, 李伟琴.网络用户认证系统 Kerberos 的原理与应用[J].微型计算机, 1997, 17(3)

[13] Tung, B., et al. Publie Key Cryptography for Initial Authentieation in Kerberos. 2001, http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-12.txt

[14] Hur, M., et al. Publie Key Cryptography for Cross-Realm Authentication in Kerberos. 2000, http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-cross-06.txt

[15] Harbitter A, Menascé D.Performance of Public-Key-Enabled Kerberos Authentication in LargeNetworks [C]. In: 2001 IEEE Symposium on Security and Privacy Proceedings, IEEE Computer Society Press, 2001

[16] Kehne A, Schonwalder J, Langendorfer H.A nonce-based Protocol for multiple authentication[J].Operating Systems Review, 1992, 26(4), 84~89

[17] Itoi N, Honeyman P.Smart card integration with Kerberos v5 [J]. Lecture Notes in Computer Science, 2001, 2041, 73~78

[18] Cheolhyun KIM, Ilyong CHUNG．An Efficient Kerberos Authentication Mechanism Associated with X.509 and DNS (Domain Name System) [J]．IEICE Transactions on Information and Systems, 2002, 1384~1389

[19] 谢冬青, 冷健．PKI 原理与技术[M]．北京：清华大学出版社, 2004

[20] 黄益民, 平玲娣, 潘雪增．一种基于角色的访问控制扩展模型及其实现[J]．计算机研究与发展, 2003, 40(10), 1521~1528

[21] [美]William Stallings 著, 杨明, 胥光辉, 齐望东等译．密码编程学与网络安全[M]．北京：电子工业出版社, 2001, 256~257

[22] G. A. Champine, D. E. Geer, W. N. Ruh．Project Athena as a distributed computer system．IEEE Computer, 1990-09, 23(9),40~51

[23] 王曦, 杨健．网络安全技术与实务[M]．北京：电子工业出版社, 2006, 47~48

[24] Watanabe H, Fujiwara T, Kasami T．Improved Method for Formal Security Verification of Cryptographic Protocols．IEICE Transactions on Fundamentals of Electronics．Communications and Computer Sciences, 1996．E79-A(7)．1089~1096

[25] Ganesan．Yaksha．Augmenting Kerberos with the Public Key Cryptography[J]．Proceedings of the Internet Society Symposium on Network and Distributed System Security．IEEE Computer Society Press, 1995

[26] 顾文刚, 程朝辉, 荆金华等．基于 PKI 的 Kerberos 跨域认证协议的实现与分析[J]．计算机科学, 2001, 28(10), 78~80

[27] 师青红．电子商务的安全保证---认证中心[J] ．电子商务, 2002, (1), 47-49

[28] Charlie Kaufman, Radia Perlman, Mike Speciner, Network Security Private Communication in Public World, Prentice Hall PTR, 2002

[29] 汪雪莲．基于 X.509 标准的公钥证书管理系统研究[M]．计算机应用技术, 2003

[30] 文铁华, 谷士文．增强 Kerberos 协议安全性的改进方案[M]．通信学报, 2004-06

[31] Ian Downnard．Public-key crytography extensions into Kerberos．IEEE POTENTIALS, 2003-01

[32] Tung, M., et al．Public Key Cryptography for Cross-Realm Authentication in Kerberos．1998, http://www.internic.net/internet-drafts/draft-ietf-cat-kerberos -pk-cross-03.txt

[33] 刘壮, 郭荷清, 张娟娟．基于公钥的 Kerberos 分布式认证方法研究[M]．计算机工程与应用, 2006-04

[34] Java™ 2 Platform Standard Ed. 5.0．Sun Microsystems, Inc．2004

发表论文和参加科研情况说明

发表论文情况：

(1)吴喜兰，舒远仲，江泽涛.《一种结合 PKI 技术的 Kerberos 改进协议》.计算机应用与软件（已录用，待发表）

(2)梁旗军，吴喜兰，蔡轲.《J2ME 中 RMS 编程的研究》.计算机与现代化，2006-8，29~31

(3)易丽萍，叶水生，吴喜兰.《一种改进的汉语分词算法》.计算机与现代化，2007-2，13~15

参加的科研项目：

(1)2005 年江西省重大攻关招标项目（编号：2005A016）：基于多层安全代理的集成访问控制系统

致谢

在这三年的研究生生涯期间，许多人在各个方面给了我很多帮助。在论文结束之时，我非常真诚的向他们表达由衷的感谢之情。

首先，衷心感谢我的导师舒远仲教授。本文从选题、设计到定稿，都得到了舒老师悉心的指导，帮我解决了很多问题和困难。给我提供了良好的学习环境和科研氛围；悉心指导我的学术研究，开拓我的思路，鼓励并帮助我解决一个个困难。舒老师严谨的治学态度，深厚的学术功底，平易近人的待人风格使我受益匪浅。在此，我再次向他致上我最真诚的谢意。

同时，我还要感谢我的师姐邬志红，在我读研期间，她在学习和生活上给我很多帮助。其次，还要感谢我们项目组里的甘盛科老师和李克伟老师，多谢他们在课题上的帮助。还要感谢三年来，与我共同在一个项目组工作的同学王巍、刘勇、周巍和师弟胡硕，感谢他们的支持和帮助。感谢我的师妹石慧、师弟陶成功和李全，他们在生活和工作中的对我的支持，给我留下了深刻的印象。

感谢我的父母对我学业的关心和支持，默默地支持我的学习，才使我顺利完成学业。

最后，衷心感谢评阅论文和参与答辩的各位专家在百忙之中给予的悉心指导。

基于公钥的Kerberos身份认证系统的设计与实现

作者： 吴喜兰
学位授予单位： 南昌航空大学