

# PMI 授权管理系统的接口设计

周彦萍

(河北省科学院 应用数学研究所, 河北 石家庄 050081)

**摘 要:**授权管理基础设施 PMI 是目前能够解决大规模网络安全问题的可行方案,但国内的 PMI 应用才刚起步。文中介绍了一种 PMI 授权管理系统,给出了系统架构,详细论述了系统授权管理的接口设计,为 PMI 应用提供了一个可借鉴的实例。该系统采用 PMI/PKI 技术、LDAP 数据库和 RBAC 模型等,实现了信息资源的访问控制。通过将访问控制机制从具体应用系统的开发和管理中分离出来,屏蔽了安全技术的复杂性,使访问控制机制与应用系统之间能灵活而方便地结合和使用。

**关键词:**PKI/PMI;授权管理;RBAC 访问控制模型;接口设计

**中图分类号:**TP39

**文献标识码:**A

**文章编号:**1673-629X(2010)03-0167-05

## Connection Design of Authorization Management System Based on PMI

ZHOU Yan-ping

(Institute of Applied Mathematics, Hebei Academy of Sciences, Shijiazhuang 050081, China)

**Abstract:** PMI is at present can solve the large-scale network security question feasible plan, but our country's PMI application just now starts. One kind of authorization management system based on PMI is introduced, the system construction is given, the connection design is described in this paper. This system has provided the example for PMI application. It uses the PMI/PKI technology, the LDAP database and the RBAC access control model and so on, has realized the information resource access control. The complexity of the safety technology is shielded in the system to connect and use flexibly and conveniently between the access control mechanism and the application system.

**Key words:** PKI/PMI; privilege management; RBAC access control model; connection design

### 0 引言

从20世纪90年代初期以来,美国、加拿大、英国、德国、日本和新加坡等国相继开展了可信第三方认证体系的研究和建设,因此到目前为止公钥基础设施 PKI(Public Key Infrastructure)体系得到了广泛部署和应用。但是,基于授权认证的 PMI(Privilege Management Infrastructure)应用还处于起步阶段,应用实例比较少。从1998年 PMI 首次被提出,到2000年颁布的完整地定义了属性证书和 PMI 模型的国际标准 ISO/IEC 9594-8(2000)<sup>[1]</sup>(又称 ITU X509(2000),目前最新为2005年第五版的 ISO/IEC 9594-8:2005<sup>[2]</sup>),及至目前,其应用权限管理的整体研究和权限的表达、权限的发布机制、应用权限管理原则方面仍

缺乏系统、全面的研究,其标准化和发展方向也还处于讨论中。

国内有关部门高度重视公钥基础设施 PKI 和授权管理基础设施 PMI 的发展,2002年4月国家标准化管理委员会批准成立全国信息安全标准化技术委员会,其下属的 PKI/PMI 工作组致力于分析国内外 PKI/PMI 标准体系;参与 PKI/PMI 国际标准化活动;研究制定国内的 PKI/PMI 标准体系。目前结合国内企业实际情况的标准体系正在修改完善中,与国际标准 ISO/IEC 9594-8:2001 对应的 GB/T 16264.8-2005 也已颁布<sup>[3]</sup>。

PMI 在国内发展不久,企业自身的 PMI 及属性证书的应用还刚刚起步,PMI 应用实例较少见,相信不远的未来,属性证书及 PMI 会在企业信息管理系统、办公系统和电子商务中得以广泛部署。

文中介绍了一种 PMI 授权管理系统。该系统利用授权管理基础设施 PMI 技术、LDAP 数据库和 RBAC 模型等,通过将权限信息存储在属性证书中,对

收稿日期:2009-06-29;修回日期:2009-09-21

基金项目:河北省财政计划项目(08926)

作者简介:周彦萍(1962-),女,副研究员,研究方向为网络信息安全、电子商务与政务及数据库应用等。

用户实现信息资源的访问控制;系统屏蔽了安全技术的复杂性,通过提供的接口函数就可构造高安全性的应用,而无须具备专业的安全知识背景。给出了系统架构,详细论述了系统授权管理的接口设计。

## 1 PMI 授权管理系统的体系结构

### 1.1 系统目标

授权管理系统应该达到如下目标:

①可为不同类型的应用提供授权管理和访问控制的平台支持;

②平台策略的定制应该灵活,以适应各类业务应用的安全需求;

③管理功能的操作应该简单,因为管理人员可能属于不同领域;

④应具有很好的扩展能力,如在不用改动程序的情况下可以随时增加功能模块,能够随时增加策略标准,可针对不同的应用定制实施模块;

⑤应具有较高的效率,避免决策过程对访问速度的明显影响;

⑥应该独立于任何应用。

### 1.2 系统架构

系统的授权访问控制依托于授权管理基础设施 PMI,遵循 X.509 协议和 GB/T 16264.8-2005 标准,采用基于角色的访问控制 RBAC(Role-based Access Control)模型<sup>[4,5]</sup>。RBAC 与传统的自主访问控制和强制访问控制相比,代表了在灵活性和控制粒度上的一个重大进步,并正逐渐取代传统的访问控制模型而成为主流<sup>[6]</sup>,它也是 PMI 中采用较多的一种访问控制机制。

系统主要有访问控制策略定制模块、权限分配模块、属性证书管理、访问控制决策单元 ADF(Access Control Decision Function)及系统管理五个模块组成,访问控制执行单元 AEF(Access Control Enforcement Function)是提供给应用系统的调用接口。系统架构如图 1 所示。

## 2 PMI 授权管理系统的接口设计

根据系统架构,按功能将接口分为 LDAP 服务、PKI 服务、存储身份证书的 USB 智能卡服务及 PMI 服务共四类。LDAP 服务遵守轻量级目录访问协议的 RFC 2251(V3)技术规范<sup>[7]</sup>,PKI 和 PMI 服务遵守 X.509、GB/T 16264.8-2005 及相应的 PKI 规范<sup>[8]</sup>。限于篇幅,下文只列出了 LDAP、PKI 和 PMI 三类服务的接口设计。

### 2.1 LDAP 的 API 接口

#### 2.1.1 连接与认证类函数

①连接 LDAP\_Connect(char \* hostname, int port-no, char \* cred, char \* passwd, int method);

②解除绑定并关闭连接 LDAP\_UnConnect()。

#### 2.1.2 更新类函数

①添加用户、角色、目标资源、操作行为、权限和属性证书条目。

LDAP\_AddUser(char \* dn, LDAPMod \* attrs []);

LDAP\_AddRole(char \* dn, LDAPMod \* attrs []);

LDAP\_AddObject(char \* dn, LDAPMod \* attrs []);

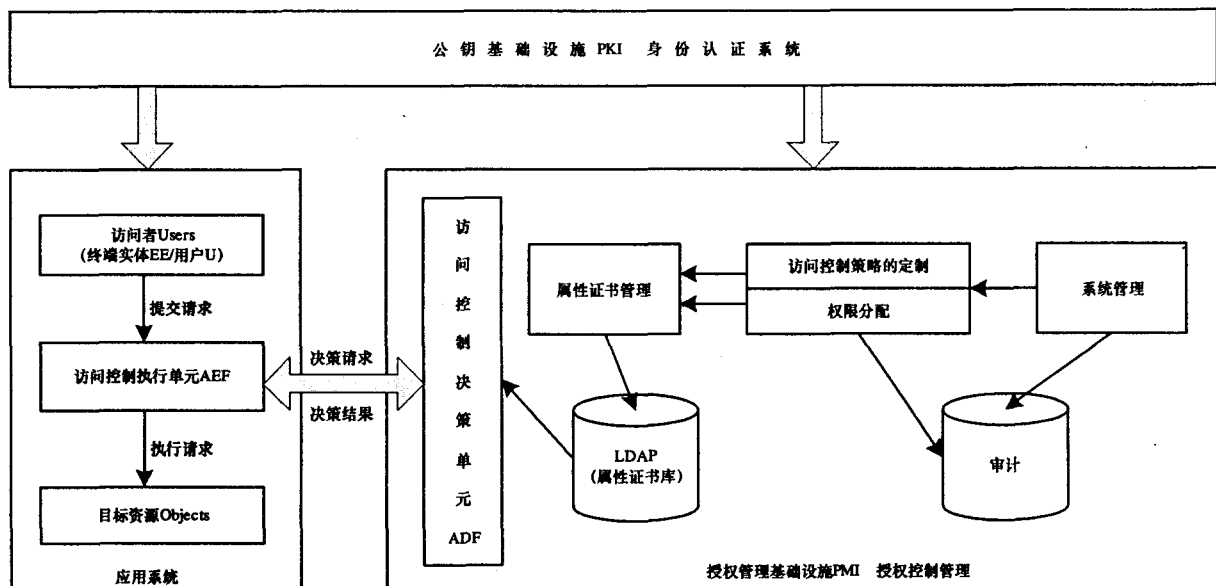


图 1 系统架构

```
LDAP_AddOperation(char * dn, LDAPMod * attr-
trs[]);
```

```
LDAP_AddPermit(char * dn, LDAPMod * attr-
s[]);
```

```
LDAP_AddAC(char * dn, LDAPMod * attr[]);
```

②删除某条用户、角色、目标资源、操作行为、权限和属性证书条目。

```
LDAP_DeleteUser(char * dn);
```

```
LDAP_DeleteRole(char * dn);
```

```
LDAP_DeleteObject(char * dn);
```

```
LDAP_DeleteOperation(char * dn);
```

```
LDAP_DeletePermit(char * dn);
```

```
LDAP_DeleteAC(char * dn);
```

③修改某条用户、角色、目标资源、操作行为、权限和属性证书条目的相对名。

```
LDAP_ModrUser(char * dn, char * rdn, int dele-
teoldrdn);
```

```
LDAP_ModrRole(char * dn, char * rdn, int dele-
teoldrdn);
```

```
LDAP_ModrObject(char * dn, char * rdn, int
deleteoldrdn);
```

```
LDAP_ModrOperation(char * dn, char * rdn, int
deleteoldrdn);
```

```
LDAP_ModrPermit char * dn, char * rdn, int
deleteoldrdn);
```

```
LDAP_ModrAC(char * dn, char * rdn, int delete-
oldrdn);
```

④修改某条用户、角色、目标资源、操作行为、权限和属性证书条目。

```
LDAP_ModifyUser(char * dn, LDAPMod * mods
[]);
```

```
LDAP_ModifyRole(char * dn, LDAPMod * mods
[]);
```

```
LDAP_ModifyObject(char * dn, LDAPMod *
mods[]);
```

```
LDAP_ModifyOperation(char * dn, LDAPMod *
mods[]);
```

```
LDAP_ModifyPermit(char * dn, LDAPMod *
mods[]);
```

```
LDAP_ModifyAC(char * dn, LDAPMod * mods
[]);
```

### 2.1.3 查询类函数

①检索某用户、角色、目标资源、操作行为、权限和

属性证书条目。

```
LDAP_SearchUser(char * base, int scope, char *
filter, char * attr[], int attrsonly, struct timeval *
timeout, LDAPMessage ** res);
```

```
LDAP_SearchRole(char * base, int scope, char *
filter, char * attr[], int attrsonly, struct timeval *
timeout, LDAPMessage ** res);
```

```
LDAP_SearchObject(char * base, int scope, char
* filter, char * attr[], int attrsonly, struct timeval *
timeout, LDAPMessage ** res);
```

```
LDAP_SearchOperation(char * base, int scope,
char * filter, char * attr[], int attrsonly, struct timeval
* timeout, LDAPMessage ** res);
```

```
LDAP_SearchPermit(char * base, int scope, char
* filter, char * attr[], int attrsonly, struct timeval *
timeout, LDAPMessage ** res);
```

```
LDAP_SearchAC(char * base, int scope, char *
filter, char * attr[], int attrsonly, struct timeval *
timeout, LDAPMessage ** res);
```

②条目结果的处理。

LDAP\_FirstEntry(LDAPMessage \* res)获取查询结果中的第一个条目;

LDAP\_NextEntry(LDAPMessage \* entry)获取查询结果中的下一个条目;

LDAP\_CountEntries(LDAPMessage \* res)计算查询结果的条目个数。

③对属性结果的处理。

LDAP\_FirstAttribute(LDAPMessage \* entry, void \*\* ptr)返回指向包含第一个属性名缓冲区的指针;

LDAP\_NextAttribute(LDAPMessage \* entry, void \*\* ptr)返回指向包含下一个属性名缓冲区的指针。

④获得属性值。

LDAP\_GetValues(LDAPMessage \* entry, char \* attr)和 LDAP\_GetValuesLen(LDAPMessage \* entry, berval \* attr)获得条目的属性值;

LDAP\_CountValues(char \*\* vals)和 LDAP\_CountValuesLen(berval \*\* vals)计算返回值的个数。

⑤目录项 DN 分析处理。

LDAP\_GetDN(LDAPMessage \* entry)获得条目的 DN;

LDAP\_ExplodeDN(char \* dn, int notypes)将 DN 中的各字段分离开;

LDAP\_DNtoUFN(char \* dn)将 DN 的名字转换成较易读取的名字。

## 2.2 PKI 的 API 接口

### 2.2.1 公钥证书函数

①打开公钥证书 PKI\_OpenCert()。

②从打开的公钥证书中获取主题名 PKI\_GetSubjectDN(char \* pNameStr, DWORD \* pdwDNStr)。

③从打开的公钥证书中获取公钥块或公钥信息。

PKI\_GetPubKeyBlob(DWORD dwKeySpec, BYTE \* pbPubKeyBlob, DWORD \* pdwPubKeyBlobLen);

PKI\_GetKeyInfo ( KEY\_ PROV\_ INFO \* pKeyProvInfo)。

④从打开的公钥证书中获取证书块 PKI\_GetCertBlob ( BYTE \* pbCertBlob, DWORD \* pdwCertBlobLen)。

⑤验证公钥证书有效性 PKI\_VerifyCert()。

⑥关闭打开的证书 PKI\_CloseCert()。

### 2.2.2 加解密与签名

①对信息或文件内容加密。

PKI\_EncryptMsg ( const BYTE \* pbToBeEncrypted, DWORD dwToBeEncryptedLen, BYTE \* pbPubKeyByt, DWORD dwPubKeyBytLen, BYTE \* pbEncryptedBlob, DWORD \* pdwEncryptedBlobLen);

PKI\_EncryptFile ( const LPTSTR pszToBeEncryptedFileName, const LPTSTR pszEncryptedFileName, BYTE \* pbPubKeyByt, DWORD dwPubKeyBytLen)。

②对信息或文件内容解密。

PKI\_Decrypt PKI\_DecryptMsg ( const BYTE \* pbEncryptedBlob, DWORD dwEncryptedBlobLen, BYTE \* pbDecryptedMessage, DWORD \* pdwDecryptedMessageLen);

PKI\_DecryptFile ( const LPTSTR pszToBeDecryptedFileName, const LPTSTR pszDecryptedFileName)。

③对信息或文件签名。

PKI\_PKI\_SignMsg(const BYTE \* pbToBeSigned, DWORD dwToBeSignedLen, BYTE \* pbSignedBlob, DWORD \* pdwSignedBlobLen);

PKI\_SignFile ( const LPTSTR pszToBeSignedFileName, const LPTSTR pszSignedFileName)。

④对信息或文件的签名进行验证。

PKI\_Verify PKI\_VerifySignMsg (const BYTE \* pbSignedBlob, DWORD dwSignedBlobLen, const BYTE \* pbToBeSigned, DWORD dwToBeSignedLen, BYTE \* pbPubKeyByt, DWORD dwPubKeyBytLen);

PKI\_VerifySignFile (const LPTSTR pszToBeVerifyFileName, const LPTSTR pszSourceFileName, BYTE \* pbPubKeyByt, DWORD dwPubKeyBytLen)。

## 2.3 PMI 服务的接口

### 2.3.1 属性证书管理

①证书申请 PMI\_ACRequest (const DWORD dwType, const LPTSTR pszFileName)。

该函数提供一个输入证书申请信息的界面,并将结果存入指定的二进制文件中。

②证书生成(证书编码) PMI\_ACEncode (const LPTSTR pszACRequestFileName, AC\_ INFO \* pACEncode)。

该函数根据证书申请信息,按照 X.509v4 规范生成证书,并以 ANS.1 (Abstract Syntax Notation One) 编码格式对形成的证书进行编码。

③证书发布 PMI\_ACIssue (AC\_ INFO \* pACEncode)。

将生成的证书发布到 LDAP 证书库的服务器上,即将证书添加到 LDAP 中的属性证书子树里。

④证书检索 PMI\_ACSearch (const DWORD dwType, const char \* Filterdn, char \* ACdn)。

该函数从 LDAP 证书库中检索指定 SOA、角色或用户的所有属性证书,并将其列出以供选择。

⑤证书验证 PMI\_ACVerify (const char \* ACdn)。

属性证书的验证分为两步,第一步为验证属性证书的签名,即验证属性证书的合法性。第二步为验证属性证书的有效期。

⑥证书吊销 PMI\_ACRevoke (const char \* ACdn)。

⑦证书冻结 PMI\_ACFreeze (const char \* ACdn)。

⑧证书解冻 PMI\_ACMelt (const char \* ACdn)。

⑨证书延期 PMI\_ACDefer (const char \* ACdn)。

⑩列出赋予某用户的全部角色 PMI\_ListRoles (const char \* UserDN)。

⑪列出具备某角色的全部用户 PMI\_ListUsers (const char \* RoleDN)。

### 2.3.2 AEF 函数

①向 ADF 请求用户的访问凭证 PMI\_AEFGetCreds (const char \* UserDN, char \* UserCreds)。

该函数根据用户的 DN 名从 LDAP 服务器上检索用户的身份证书和公钥,验证其身份,通过后向 ADF 提交获取用户访问凭证的请求,最后返回从 ADF 处获取的用户访问凭证 UserCreds。

②向 ADF 请求用户的访问判决 PMI\_AEFPrivilegeVerify (const char \* UserCreds, REQUEST \* pRequest)。

该函数向 ADF 传递用户的访问请求,以便得到 ADF 对用户访问请求的判决结果。提交的访问请求包括用户要访问的目标、操作行为及应用系统相应的环境参数。函数首先要判断用户的访问凭证是否过期,若已过期则直接返回假,否则再向 ADF 提交请求。

③中断 AEF 函数 PMI\_AEFShutdown()。

### 2.3.3 ADF 函数

抽象成初始化、判决和结束三个函数。AEF 在启动时调用初始化函数完成 ADF 的初始准备,当接收到用户访问请求时,AEF 先验证用户身份,通过后调用 ADF 的判决函数完成权限验证,通过调用结束函数安全地停止 ADF 的权限判决服务。

①初始化 PMI-ADFInitialize(const LPTSTR pszFileName)。

该函数从 LDAP 服务器上获取策略的属性描述符证书和签发者的公钥证书,并利用签发者的公钥验证其在属性描述符证书上的签名,然后解析出访问控制策略并保存在本地文件 pszFileName 中。

②获取用户的访问凭证 PMI\_ADFGetCreds(const char \* UserDN, const byte \* SignValue, char \* UserCreds)。

该函数根据用户的 DN 名从 LDAP 服务器上检索用户的身份证书和公钥,验证其签名值,若正确则获取用户的访问凭证。

③判决 PMI\_ADFPrivilegeVerify(const char \* UserDN, REQUEST \* pRequest)。

该函数根据用户的 DN 名从 LDAP 服务器上检索用户的属性证书,验证证书签发者的签名,从中获取用户的角色,在访问策略的属性描述符证书中查找这些角色是否允许以 REQUEST 结构描述的访问请求,最后返回判决结果。

④结束 PMI\_ADFFinalize()。

## 3 结束语

该系统采用 PMI/PKI 技术、LDAP 数据库和 RBAC 模型等,实现了信息资源的访问控制。系统通过将访问控制机制从具体应用系统的开发和管理中分离出来,使访问控制机制与应用系统之间能灵活而方便地结合和使用。从系统的角度为企事业单位提出了一个信息安全服务平台,以保障企事业单位信息资源的安全。该系统可用于企业级的信息安全应用的开发,不论是企事业单位网络应用、信息管理,还是办公自动化,都可以利用本系统为其提供身份认证、信息加解密、数字签名与验证、授权管理与验证等安全服务。

### 参考文献:

- [1] ITU - T Rec. X509 (2000) | ISO/IEC 9594 - 8: 2000, The Directory: Public - key and attribute certificate framework[S/OL]. 2000. <http://www.iso.org/iso/store.htm>.
- [2] ITU - T Rec. X509 (2005) | ISO/IEC 9594 - 8: 2005, The Directory: Public - key and attribute certificate framework[S/OL]. 2005. <http://www.iso.org/iso/iso-catalogue/catalogue-tc/catalogue-detail.htm? csnumber=43793>.
- [3] 中华人民共和国信息产业部. GB/T 16264. 8 - 2005, 信息技术开放系统互连目录第 8 部分: 公钥和属性证书框架[S]. 北京: 中国标准出版社, 2005.
- [4] 刘宏月, 范久伦, 马建峰, 等. 访问控制技术的研究进展[J]. 小型微型计算机系统, 2004, 25(1): 56 - 59.
- [5] 李 辉, 王 芳. “一切皆角色”的访问控制策略[J]. 计算机科学, 2006(9A): 121 - 125.
- [6] 薛 伟, 怀进鹏. 基于角色的访问控制模型的扩充和实现机制研究[J]. 计算机研究与发展, 2003, 40(11): 1635 - 1642.
- [7] Yeong W, Howes T, Kille S. RFC 2251, Lightweight Directory Access Protocol (v3)[S]. [s. l.]: [s. n.], 1997.
- [8] Adame C, Lloyd S. 公钥基础设施 - 概念、标准和实施全[M]. 冯登国, 译. 北京: 人民邮电出版社, 2001.

(上接第 131 页)

- [4] Dorigo M. Optimization, learning and natural algorithms[D]. Politecnico di Milano, Italy: Department of Electronics, 1992.
- [5] Cheng H D, Chen Y H, Jiang X H. Thresholding using two - dimensional histogram and fuzzy entropy principle[J]. IEEE Transactions on Image Processing, 2000, 9(4): 732 - 735.
- [6] 杨卫莉, 郭 雷, 赵天云, 等. 基于分水岭变换和蚁群聚类的图像分割[J]. 量子电子学报, 2008, 25(1): 19 - 24.
- [7] 汤可宗, 江新姿, 高 尚. 蚁群模糊聚类的图像分割[J]. 计算机工程与设计, 2008, 29(7): 1770 - 1772.
- [8] 薛 琴, 陈 玮, 罗俊奇. 基于梯度算子的蚁群图像分割算法研究[J]. 计算机工程与设计, 2007, 28(23): 5660 - 5663.
- [9] 段海滨, 王道波, 于秀芬. 蚁群算法的研究现状及展望[J]. 中国工程科学, 2007, 9(2): 98 - 102.

中国计算机学会会刊、中国科技核心期刊  
《计算机技术与发展》欢迎订阅, 邮发代号: 52-127