**信息安全基础简介**

# 概述

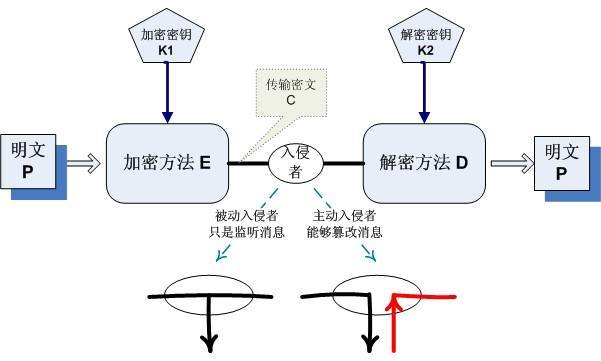
## 编写目的

主要介绍密码学相关的基本概念包括：

* 密码学简介
* 消息摘要简介
* 常用的哈希算法简介
* 数字证书简介
* 数字签名简介
* 证书认证简介
* PKI相关的基本概念简介
* 授权简介

# 密码学简介

## 什么是加密？为什么需要加密？



信息通过明文传输的过程中，**容易被第三方入侵者监听或篡改**，所以需要把语义明确的文字转换成语义无序或是不可读的密文，这个过程就叫做加密。

## 密码学历史简介

古希腊人当时据说已经对加密方法有了认识，斯巴达军队曾使用过密码棒进行换位加密，如下图所示，将皮革缠在一根木棒上，然后，横着写下需要加密的信息。写完后，皮革上的信息就成了无序的密码文。解密码的时候，用同样粗细木棒就可以读取相应的明文。这个就是比较古老的移位加密算法。

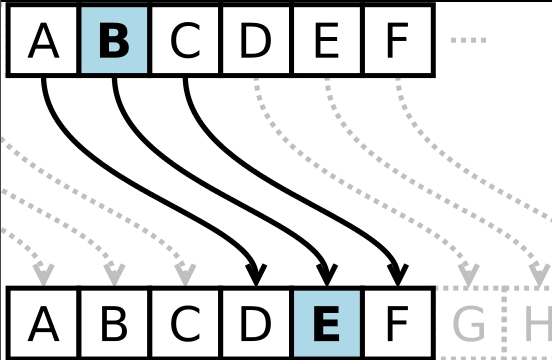
 

图2-1 密码棒与加密原理

二战时期著名的加密设备恩尼格玛密码机被德国军方用来加密通信。在机器上打字会点亮一个备用字母，然后在密码信息中使用它的位置。有26个设置决定了机器使用时哪个字母会亮起。一个插板通过将字母与操作者选择的任何字母交换来增加额外的安全性。



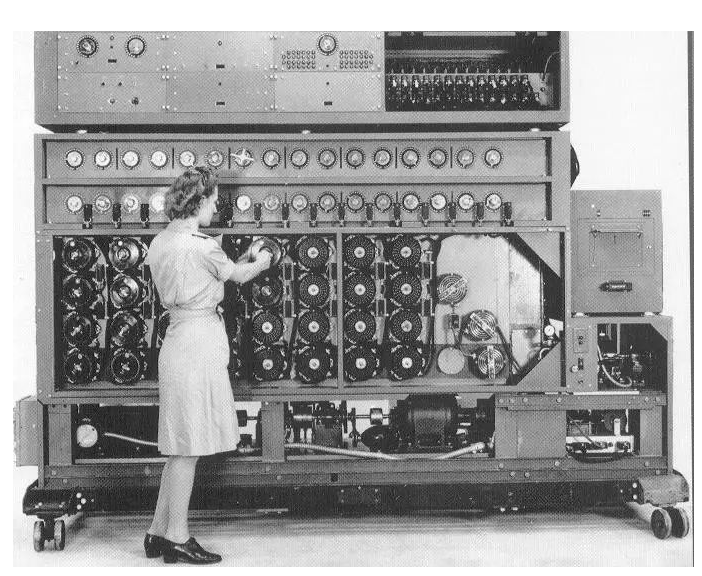
图2-2 恩尼格玛密码机

图2-3 图灵机（bomba）

解密的方式是穷举，即遍历所有可能的对应关系，直到找出有意义的关键词，而这恰恰是机器最擅长的事。英国的同盟国波兰在战前就成功研制了破解谜机的炸弹机（bomba）

## 现代加密解密算法

二战极大促进了密码学的发展。一方面，暗潮汹涌的情报战推着密码算法不断升级，越来越多的古老算法被证明不再适用；另一方，数学的发展与电子计算机的出现给密码学带来了全新的工具和自由。以香农的《保密系统的通信理论》为标志，密码学正式进入了科学的大门

现代密码学区别与古代密码学的一个重要的标志就是提出**密钥与算法分离，算法公开**，而密钥只能由加密解密双方掌握。

现在所使用的加密算法分为对称加密与非对称加密

### **对称加密**

对称加密算法 是应用较早的加密算法，又称为 共享密钥加密算法。在 对称加密算法 中，使用的密钥只有一个，发送 和 接收 双方都使用同一个密钥对数据进行 加密 和 解密。这就要求加密和解密方事先都必须知道加密的密钥。

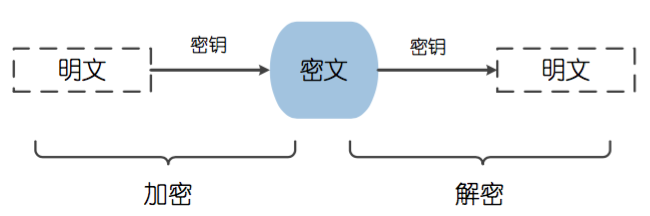


图2-2 对称加密与解密

常见的对称算法有：DES、3DES、AES。国产对称加密算法有SM1，SM4等

**对称加密的特点**是计算量小、速度快，因此适合对大量数据进行加密的场景。适用于大数据的加密。

最后就要谈到就是对称加密的安全性了。其实对算法本身来讲，对称加密已经足够安全了。

**对称加密的致命缺点**在于密钥本身的传输管理安全性，因为加密解密需要使用同一个密钥，也就是说得双方事先都得有这个密钥。

### **非对称加密**

非对称加密算法，又称为 公开密钥加密算法。它需要两个密钥，一个称为 公开密钥 (public key)，即 公钥，另一个称为 私有密钥 (private key)，即 私钥。

因为 加密 和 解密 使用的是两个不同的密钥，所以这种算法称为 非对称加密算法。

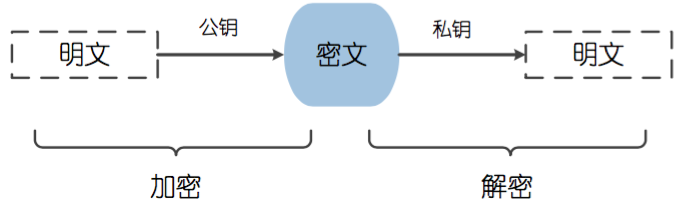


图2-3 非对称加密与解密

非对称加密的运算复杂，速度慢，常用于数字签名。或是只加密码小量数据。

常见的非对称加密算法有：RSA、DSA。

# 消息摘要简介

## 什么是消息摘要？为什么需要消息摘要？

对消息内容，进行一个**单向**的hash函数计算，得出一个固定长度的hash值，这个值就是摘要，也称为指纹。

消息摘要**旨在保护数据的完整性**，不同的消息内容得到的hash值不一样，所以**可防止内容被篡改**。

消息摘要能够保证内容完整性，但是不能确认发送方的身份是可信任的。

## 常用的摘要算法

### MD5

MD5 用的是 哈希函数，它的典型应用是对一段信息产生 信息摘要，以 防止被篡改。严格来说，MD5 不是一种 加密算法 而是 摘要算法。无论是多长的输入，MD5 都会输出长度为 128bits 的一个串 (通常用 16 进制 表示为 32 个字符)。

### SHA1算法

SHA1 是和 MD5 一样流行的 消息摘要算法，然而 SHA1 比 MD5 的 安全性更强。对于长度小于 2 ^ 64 位的消息，SHA1 会产生一个 160 位的 消息摘要。基于 MD5、SHA1 的信息摘要特性以及不可逆 (一般而言)，可以被应用在检查文件完整性以及数字签名等场景。

### HMAC算法

HMAC 是密钥相关的哈希运算消息认证码（Hash-based Message Authentication Code），HMAC 运算利用 哈希算法 (MD5、SHA1 等)，以 一个密钥 和 一个消息 为输入，生成一个 消息摘要 作为 输出。

HMAC 发送方和接收方都有的 key 进行计算，而没有这把 key 的第三方，则是无法计算出正确的散列值的，这样就可以防止数据被篡改。

# 数字签名

## 什么是数字签名？为什么需要数字签名？

数字签名 = 用私钥加密的过程就是签名

因为公钥和私钥是一一对应的，所以当一个公钥能解密某个密文时，说明这个密文一定来自于私钥持有者。

## 数字签名过程

以下以一个简单的例子说明数字签名的过程：

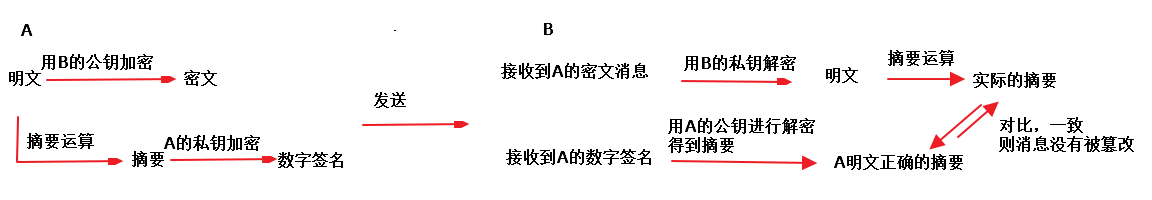


图6-1 数字签名的过程

以上过程简单说明如下：

A：将明文进行摘要运算后得到摘要（消息完整性），再将摘要用A的私钥加密（身份认证），得到数字签名，将密文和数字签名一块发给B。

B：收到A的消息后，先将密文用自己的私钥解密，得到明文。将数字签名用A的公钥进行解密后，得到正确的摘要（解密成功说明A的身份被认证了）。

对明文进行摘要运算，得到实际收到的摘要，将两份摘要进行对比，如果一致，说明消息没有被篡改（消息完整性）。

疑问：

摘要使用A的私钥加密，如果被拥有A的公钥的第三者截获，不就可以获取到摘要了么？会不会对安全造成威胁。

不会。因为摘要是不可逆推出原文的。

## 更高效数字签名过程

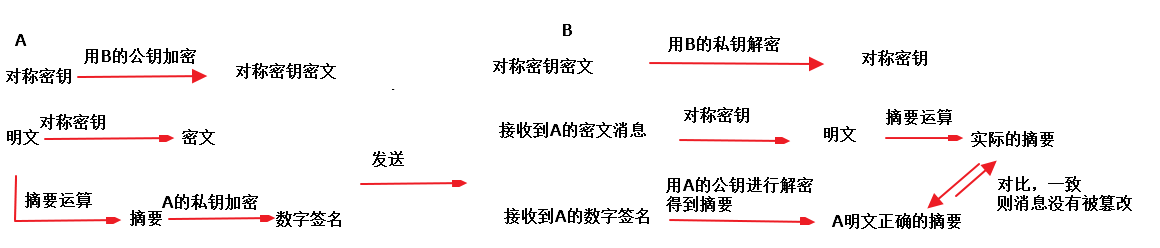


图6-2 使用数字签名进行安全通信的过程

这里有一个前提，首先A B双方要互相信任对方证书。然后就可以进行通信了，与上面的数字签名相似。不同的是，使用了对称加密。这是因为，非对称加密在解密过程中，消耗的时间远远超过对称加密。如果密文很长，那么效率就比较低下了。但密钥一般不会特别长，对对称加密的密钥的加解密可以提高效率。

## 常用签名算法

**RS256**

RS256 (采用SHA-256 的 RSA 签名) 是一种非对称算法, 它使用公共/私钥对: 标识提供方采用私钥生成签名, JWT 的使用方获取公钥以验证签名。由于公钥 (与私钥相比) 不需要保护, 因此大多数标识提供方使其易于使用方获取和使用 (通常通过一个元数据URL)。更加安全。

**HS256**

HS256 (带有 SHA-256 的 HMAC 是一种对称算法, 双方之间仅共享一个 密钥。由于使用相同的密钥生成签名和验证签名, 因此必须注意确保密钥不被泄密。

# 数字证书简介

## 什么是数字证书？为什么需要数字证书？

数字证书从本质上来说是一种电子文档，是由电子商务认证中心（以下简称为CA中心）所颁发的一种较为权威与公正的证书。

证书可以也称为公钥证书，证书包含一个实体及其公钥，并且能够保护和证明该公钥属于该实体。

如果将公钥直接发送，则有可能被另一个实体拦截并篡改，这种称为中间人攻击，所以需要通过受信任第三方来交换公钥，来**保证该公钥确实属于您正在通信的实体**。

## 数字证书的构成

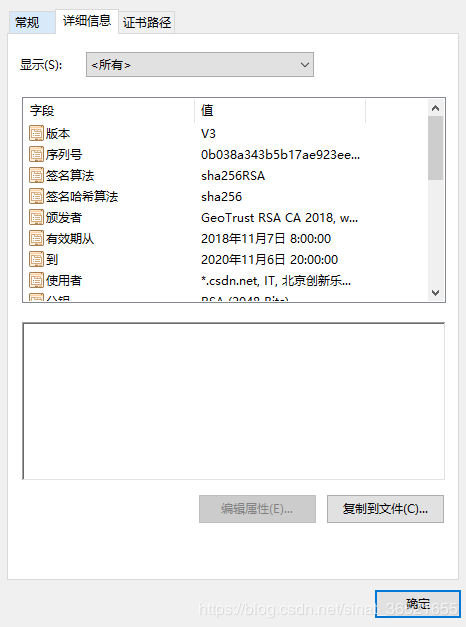


图4-1 数字证书

证书的构成内容比较多，包含颁发者、使用者(持有者)、有效期、公钥等，下面讲一讲重要的信息

### 颁发者(证书的发布机构)

表示该证书是哪个机构发布的，这个机构是证书链非常重要的一环。对于图中的证书而言，"GeoTrust RSA CA 2018"就是这个机构。一般而言这些机构都是极具信誉的大机构。

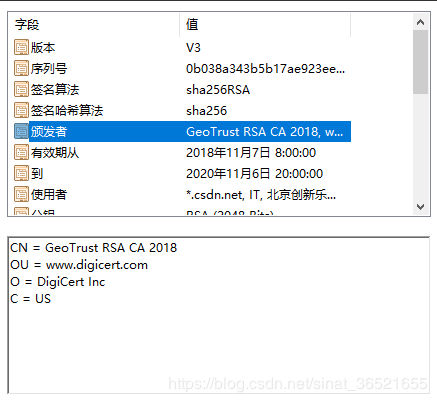


图4-2 证书颁布机构

### 有效期(证书的有效期)

表示该证书的有效期。比如该证书就是‎2018‎年‎11‎月‎7‎日 8:00:00至‎2020‎年‎11‎月‎6‎日 20:00:00，超过期限证书就会作废，需重新申请

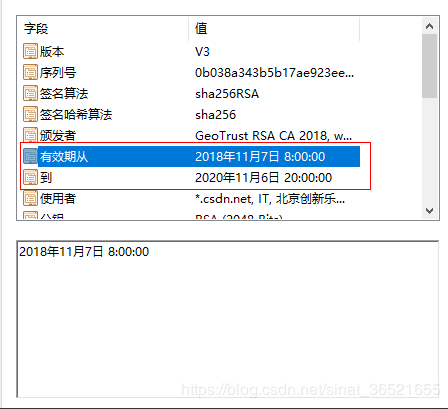


图4-3 证书有效期

### 公钥(证书持有者的公钥)

用于加解密传输的消息，从图中可以看到这个证书的公钥是2048位的，就是一长串16进制的数字

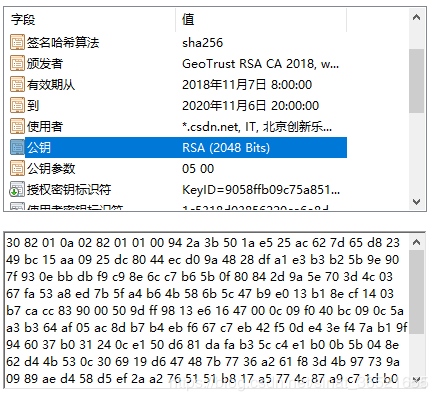


图4-4 证书公钥

### 使用者(证书持有者)

表示证书的持有者，或者说证书颁发的对象，一般包括公司、机构或者网站名等。图中证书使用者就是北京创新乐知信息技术有限公司旗下的\*.csdn.net网址。

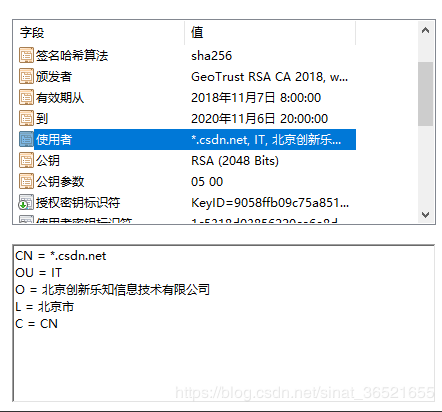


图4-5 证书使用者

### 签名哈希算法(摘要算法)

对证书主体内容进行哈希的算法，可以获得证书主体内容的摘要。这个摘要就是证书的指纹，最终用于证书数字签名和证书认证。

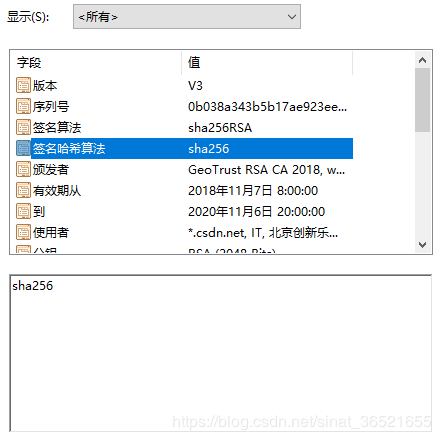


图4-6 证书签名哈希算法

### 指纹

即摘要，一个散列值，主要功能是确认证书完整性，即确认证书内容未被篡改。

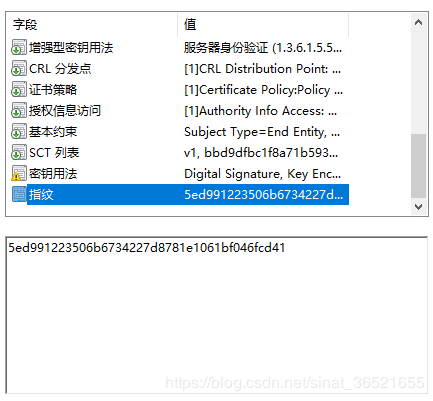


图4-7 证书指纹

### 签名算法

数字签名所使用的加密算法，使用CA的私钥加密

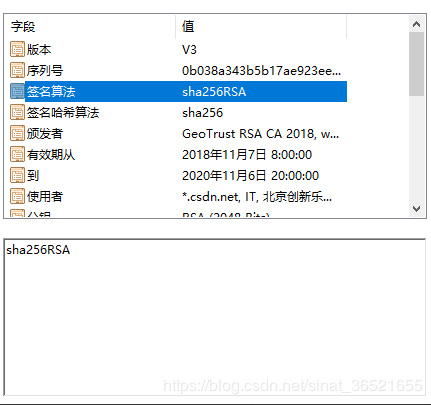
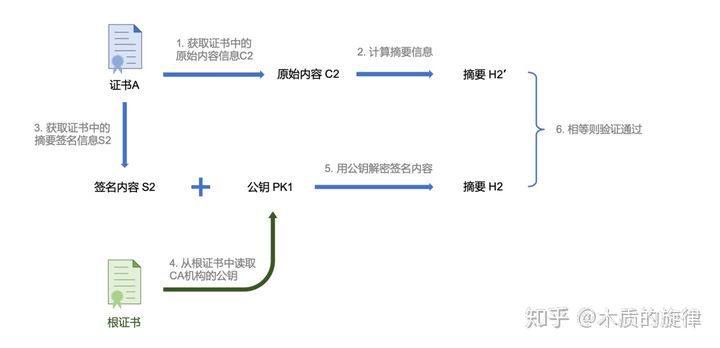


图4-8 签名算法

## 证书真伪验证



## 密钥用法

仅仅加密 Encipher Only

仅仅解密 Decipher Only

数字签名 Digital Signature

认可签名 Non Repudiation

密钥加密 key Encipherment

数据加密 Data Encipherment

密钥协商 key Agreement

证书签名 Key CertSign

CRL 签名 Crl Sign

## 增强密钥用法

Microsoft 信任列表签名 (1.3.6.1.4.1.311.10.3.1)

合格的部属 (1.3.6.1.4.1.311.10.3.10)

密钥恢复 (1.3.6.1.4.1.311.10.3.11)

许可证服务器确认 (1.3.6.1.4.1.311.10.6.2)

生存时间签名 (1.3.6.1.4.1.311.10.3.13)

Microsoft 时间戳 (1.3.6.1.4.1.311.10.3.2)

加密文件系统 (1.3.6.1.4.1.311.10.3.4)

文件恢复 (1.3.6.1.4.1.311.10.3.4.1)

根列表签名者 (1.3.6.1.4.1.311.10.3.9)

数字权利 (1.3.6.1.4.1.311.10.5.1)

密钥数据包许可证 (1.3.6.1.4.1.311.10.6.1)

文档签名 (1.3.6.1.4.1.311.10.3.12)

证书申请代理 (1.3.6.1.4.1.311.20.2.1)

智能卡登录 (1.3.6.1.4.1.311.20.2.2)

私钥存档 (1.3.6.1.4.1.311.21.5)

Windows 硬件驱动程序验证 (1.3.6.1.4.1.311.10.3.5)

密钥恢复代理 (1.3.6.1.4.1.311.21.6)

代码签名 (1.3.6.1.5.5.7.3.3)

时间戳 (1.3.6.1.5.5.7.3.8)

客户端身份验证 (1.3.6.1.5.5.7.3.2)

安全电子邮件 (1.3.6.1.5.5.7.3.4)

服务器身份验证 (1.3.6.1.5.5.7.3.1)

IP 安全用户 (1.3.6.1.5.5.7.3.7)

IP 安全 IKE 中级 (1.3.6.1.5.5.8.2.2)

OEM Windows 系统组件验证 (1.3.6.1.4.1.311.10.3.7)

IP 安全终端系统 (1.3.6.1.5.5.7.3.5)

IP 安全隧道终止 (1.3.6.1.5.5.7.3.6)

目录服务电子邮件复制 (1.3.6.1.4.1.311.21.19)

内嵌 Windows 系统组件验证 (1.3.6.1.4.1.311.10.3.8)

Windows 系统组件验证 (1.3.6.1.4.1.311.10.3.6)

所有颁发的策略 (2.5.29.32.0)

## 证书类型

(1)根证书

密钥用法：认可签名，证书签名，CRL签名

keyUsage=nonRepudiation, keyCertSign,cRLSign

(2)代码签名

密钥用法：数字签名

增强密钥用法：代码签名

keyUsage=digitalSignature

extendedKeyUsage=codeSigning

(3)计算机

密钥用法：数字签名，密钥协商

增强密钥用法：服务器验证，客户端验证

keyUsage=digitalSignature,keyAgreement

extendedKeyUsage=serverAuth,clientAuth

(4)WEB服务器

密钥用法：数字签名，认可签名，密钥加密，数据加密，密钥协商

增强密钥用法：服务器验证

keyUsage=digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment,keyAgreement

extendedKeyUsage=serverAuth

(5)客户端

密钥用法：数字签名，认可签名，密钥加密，数据加密

增强密钥用法：客户端验证

keyUsage=digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment

extendedKeyUsage=clientAuth

(6)信任列表签名

密钥用法：数字签名

增强密钥用法：信任列表签名

keyUsage=digitalSignature

extendedKeyUsage=msCTLSign

(7)时间戳

密钥用法：数字签名，认可签名，密钥加密，数据加密

增强密钥用法：时间戳

keyUsage=digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment

extendedKeyUsage=timeStamping

(8)IPSEC

密钥用法：数字签名，认可签名，密钥加密，数据加密

增强密钥用法：IP安全IKE中级

keyUsage=digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment

extendedKeyUsage=1.3.6.1.5.5.8.2.2

(9)安全Email

密钥用法：数字签名，认可签名，密钥加密，数据加密

增强密钥用法：安全电子邮件

keyUsage=digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment

extendedKeyUsage=emailProtection

(10)智能卡登陆

密钥用法：数字签名，密钥协商，仅仅解密

增强密钥用法：密钥恢复，加密文件系统，智能卡登陆

keyUsage=digitalSignature,keyAgreement,decipherOnly

extendedKeyUsage=1.3.6.1.4.1.311.10.3.11,msEFS,1.3.6.1.4.1.311.20.2.2

# PKI简介

## 概念引入

Public Key Infrastructure（PKI)，中文叫做公开密钥基础设施，简单讲就是利用公开密钥机制建立起来的基础设施。这里以一个例子加以说明：假设Bob想向Alice发送一个安全的电子邮件消息。这可以通过以下方式实现：

1. 鲍勃和爱丽丝都有自己的钥匙对。他们将自己的私钥安全地保存在自己的手中，并将自己的公钥直接发送给对方。
2. Bob使用Alice的公钥加密消息并将其发送给她。
3. 爱丽丝用她的私钥解密信息。

这个过程中Bob必须对他用于加密消息的公钥有一个明显的顾虑。也就是说，他无法确定他用于加密的密钥实际上是属于爱丽丝的。监视Bob和Alice之间通信信道的另一方可能替换了另一个密钥。公钥基础设施概念的发展有助于解决这个问题和其他问题。

## PKI的体系结构

完整的PKI/CA系统如图5-1：

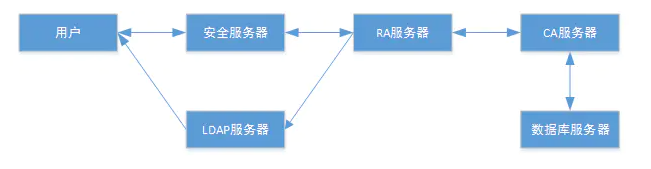


图5-1 PKI体系结构

* **安全服务器**：安全服务器面向普通用户，用于提供证书申请、浏览、证书撤销列表、证书下载等安全服务；用户需要首先得到安全服务器的证书（该证书由CA颁发）；
* **注册机构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（CRL）。
* **LDAP服务器**：Lightweight Directory Access Protocol（轻量目录访问协议），提供目录浏览服务，负责将注册机构服务器RA传输过来的用户信息以及数字证书加入到服务器上。用户通过访问LDAP服务器就能够得到其他用户的数字证书。
* **CA服务器**：证书颁发机构（CA, Certificate Authority）整个证书机构的核心，负责证书的签发。CA首先 产生自身的私钥和公钥，然后生成数字证书，并且将数字正常传输给安全服务器。CA还负责为安全服务器、RA服务器生成数字证书。
* **数据库服务器**：CA中的核心部分，用于CA中数据（如密钥和用户信息等）、日志、统计信息的存储和管理。

## PKI的工作过程

针对一个使用PKI的网络，配置PKI的目的就是为指定的实体向CA申请一个本地证书，并由设备对证书的有效性进行验证。下面是PKI的工作过程：



图5-2 PKI的工作过程

1. 实体向CA提出证书申请；
2. RA审核实体身份，将实体身份信息和公开密钥以数字签名的方式发送给CA；
3. CA验证数字签名，同意实体的申请，颁发证书；
4. RA接收CA返回的证书，发送到LDAP服务器以提供目录浏览服务，并通知实体证书发行成功；
5. 实体获取证书，利用该证书可以与其它实体使用加密、数字签名进行安全通信；
6. 实体希望撤消自己的证书时，向CA提交申请。CA批准实体撤消证书，并更新CRL，发布到LDAP服务器。

这里需要注意的是，如果要检查证书是否吊销，就需要访问LDAP服务器，以下载并更新自己的CRL。

## 证书撤销过程

用户申请：用户向RA发送一封签名加密邮件，申请撤销证书。

RA审核：注册机构同意证书撤销，并对申请签名。

CA更新CRL：CA验证证书撤销请求的RA签名，如果正确，则同意申请，并更新CRL，并输出。

RA转发CRL：注册中心收到CRL，以多种方式将CRL公布（包括LDAP服务器）。

用户告知：用户访问LDAP服务器，下载或浏览CRL。

## 证书的管理

认证中心CA负责维护和发布证书废除列表CRL（certificate revocation lists，又称为证书黑名单）。

当一个证书，特别是其中的公钥因为其他原因无效时（不是因为到期），CRL提供了一种通知用户和其他应用的中心管理方式。CA系统生成CRL以后，放到LDAP服务器中或Web服务器的合适位置，供用户查询或下载。

## 主要应用

PKI技术的广泛应用能满足人们对网络交易安全保障的需求。作为一种基础设施，PKI的应用范围非常广泛，并且在不断发展之中，下面给出几个应用实例。

1. 虚拟专用网络（VPN，Virtual Private Network）

VPN是一种构建在公用通信基础设施上的专用数据通信网络，利用网络层安全协议（如IPSec）和建立在PKI上的加密与数字签名技术来获得机密性保护。

2. 安全电子邮件

电子邮件的安全也要求机密、完整、认证和不可否认，而这些都可以利用PKI技术来实现。目前发展很快的安全电子邮件协议S/MIME(Secure/Multipurpose Internet Mail Extensions，安全/多用途Internet邮件扩充协议)，是一个允许发送加密和有签名邮件的协议。该协议的实现需要依赖于PKI技术。

3. Web安全

为了透明地解决Web的安全问题，在两个实体进行通信之前，先要建立SSL（Secure Sockets Layer，安全套接字层）连接，以此实现对应用层透明的安全通信。利用PKI技术，SSL协议允许在浏览器和服务器之间进行加密通信。此外，服务器端和浏览器端通信时双方可以通过数字证书确认对方的身份。

# 身份认证与授权

认证(Authenticate)和授权(Authorize)，是两个容易混淆的概念。认证过程是证实某个用户是他所声明的那个人；授权又叫访问控制，即决定一个身份是否能够访问网站资源。由此可见，应用需要先认证用户身份，然后依据用户身份进行授权，二者需要联合使用。



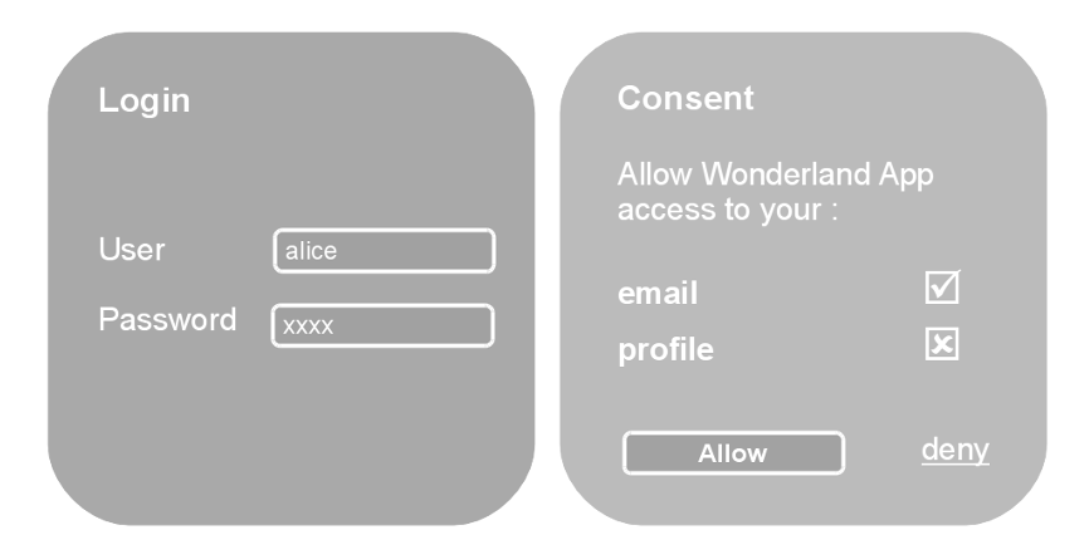


图7-1 认证与授权

对于微信这样的应用，用户在登录后会得到该账户的身份凭证。如果其他第三方应用信任并接受微信的身份凭证，就可以直接使用该凭证通过第三方的认证而登录。至于登陆之后用户能有权限去做什么，则是要第三方应用根据自己的政策进行授权。有些网站在第一次使用微信账户登录时，需要绑定已用账户，就是因为虽然网站能够通过微信账户的身份认证，但是对于这样的账户没有对应的授权。

目前，通用解决方案是单点登录(Single Sign-On,简称SSO)，用户在内网中只需登录一次，就可以访问所有相互信任的应用系统。单点登录和统一认证中主要的三个协议是OpenID、OAuth、SAML，被称为单点登录的“三驾马车”。

## Windows身份验证

### 域

Windows域为网络管理员提供了一种方法来管理大量的pc机，并从一个地方控制它们。一个或多个服务器(称为域控制器)控制域及其上的计算机。

当计算机连接到一个域时，它不会使用自己的本地用户帐户，用户帐户和密码都是在域控制器上统一管理。当您登录到该域中的计算机时，计算机将使用域控制器验证您的用户帐户名称和密码。这意味着您可以在任何连接到该域的计算机上使用相同的用户名和密码登录。

网络管理员可以更改“域控制器”上的组策略设置。域上的每台计算机将从“域控制器”获得这些设置，它们将覆盖用户在其pc上指定的任何本地设置。所有的设置都是从一个地方控制的。这也“锁定”了计算机。您可能不允许更改连接到域的计算机上的许多系统设置。

换句话说，当计算机是域的一部分时，提供该计算机的组织正在远程管理和配置它。是远程的那些人在控制着连接到域的计算机，而不是正在使用这台计算机的人；

因为域不是为家庭用户设计的，所以只有运行专业版或企业版Windows的计算机才能连接到域。



### 工作组

没有连接到域的Windows计算机都是工作组的一部分。工作组是同一本地网络上的一组计算机。与域不同的是，工 一起的。工作组也不需要密码。

工作组在以前的Windows版本被用于主文件和打印机共享。您现在可以使用homegroup在家里的pc机之间轻松地共享文件和打印机。工作组现在已经被推到后台，所以您不需要担心它们——只需要保留工作组的默认名称并设置homegroup文件共享。

### WinHTTP中提到了Windows支持的5种身份认证方式：

* Basic : 名文传输用户名和密码的。
* Digest：摘要式身份验证试图解决基本身份验证的许多弱点，特别是通过确保凭证从未以明文形式跨线发送来解决。摘要式身份验证通过网络以 MD5 哈希或消息摘要形式传输凭据。
* NTLM：集成 Windows 身份验证（以前称为 NTLM 或 Windows NT 质询/响应验证）是一种安全的验证形式，因为在通过网络发送用户名和密码之前，先将它们进行哈希计算。当启用集成 Windows 身份验证时，用户的浏览器通过与 Web 服务器进行密码交换（包括哈希）来证明其知晓密码。主要应用于工作组环境中；
* Passport：是基于 Cookie 的身份验证服务；
* Negotiate：要使协商身份验证正常工作，必须在同一连接上进行多次交换。因此，如果中间代理不支持保持活动连接，则不能使用协商身份验证。
* Kerberos认证：主要用于域环境中。

## OpenID：去中心化认证的第三方登录经典模式

谈到单点登录和身份认证，就不得不谈OpenID。最典型的使用实例就是使用Google账户登录其他应用，这一经典的协议模式，为其他厂商的第三方登录起到了标杆的作用，被广泛参考和使用。

OpenID是一个去中心化的数字身份识别框架，属于认证(Authentication)协议。OpenID的创建基于这样一个概念：我们可以通过统一资源标识符(URI)来认证一个网站的唯一身份对于支持OpenID的网站。用户不需要记住像用户名和密码这样的传统验证标记。取而代之的是，他们只需要预先在一个作为OpenID身份提供者(Identity Provider, IdP)的网站上注册。

任何网站都可以使用OpenID来作为用户登录的一种方式，也都可以作为OpenID的身份提供者，它既解决了问题而又不需要依赖于中心化的网站来确认数字身份。

OpenID的第一部分是身份验证，即如何通过 URI来认证用户身份。目前的网站都是依靠用户名和密码来登录认证，这就意味着大家在每个网站都需要注册用户名和密码，即便你使用的是同样的密码。如果使用OpenID，你的网站地址(URI)就是你的用户名，而你的密码安全的存储在一个OpenID服务网站上。

## OAuth：不触碰用户信息即可授权的“黑匣”协议

OAuth协议为用户资源的授权提供了一个安全、开放且简易的标准。比起OpenID，OAuth更像是一个授权(Authorization)协议，其独特之处在于，它可以让第三方应用程序在不获取用户账号密码的情况下，获得属于用户的资源或是进行符合用户权限的操作。在这个过程中，令牌(Token)成为核心节点。OAuth2.0通过IdP给第三方应用颁发令牌来实现以上功能，第三方应用则通过使用令牌向资源服务换取对应的资源。

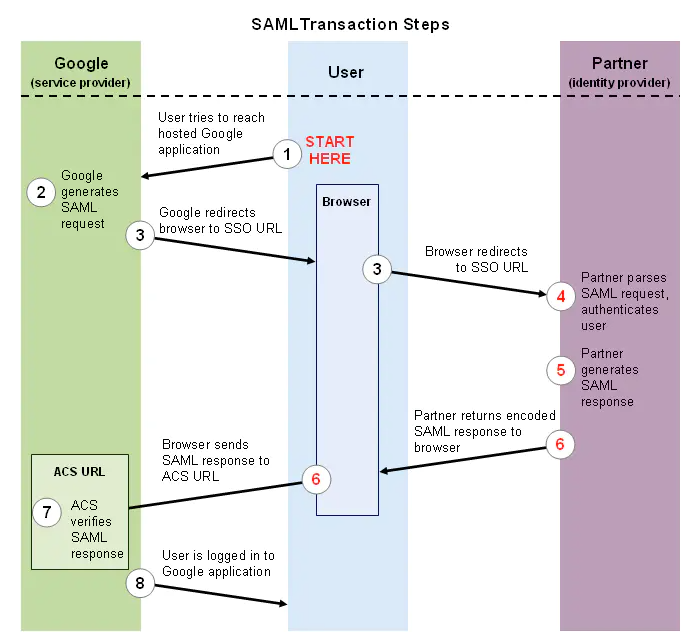
## SAML：兼顾认证与授权的身份信息共享通道

SAML协议(Security Assertion Markup Language,安全性断言标记语言)是三者中时间最长的协议，最初版本制定于2001年，并于2005年修改。作为一种安全性断言标记语言，SAML协议既可以用于认证也用于授权。

所谓的安全性断言，就是关于认证、授权以及用户属性(比如用户姓名、住址等信息)的声明集合，在SAML中，这些断言以XML格式进行传输。SAML的主要功能是在不同的安全域之间提供身份认证和授权信息交换，为用户跨平台进行身份认证和授权提供了方便，解决了多个系统共享认证、授权过程中的信息传递问题。

当要验证一个用户身份时，服务提供商(Service Provider,简称SP）会向IdP发出SAML认证请求，该请求中会以XML格式说明认证方式的设置，比如希望IdP以何种方式验证用户。IdP在认证通过用户身份之后，会返回SAML请求响应，同样以XML格式返回断言表明用户身份和相关属性，此外SAML安全性断言信息必须要使用数字签名以保证其完整性和不可抵赖性。SP接收到SAML断言之后，验证其消息来源是否费受信任的IdP，验证通过之后解析XML获得认证信息。

下面我们以用户登录SP，SP向IDP发起请求来确认用户身份为例子，看看SAML的工作流程。比如SP是Google的Apps，IDP是一所大学的身份服务器，Alice是该大学的一名学生。



**JWT令牌**

https://jwt.io/