# 一种细粒度的属性证书出示方案

王 凯 张红旗 任志宇 姜皇勤

(信息工程大学 郑州 450004) (河南省信息安全重点实验室 郑州 450004)

**摘 要** 针对现有X.509v4属性证书在细粒度出示部分属性后无法验证合法性的情况,提出了一种支持属性细粒度出示的证书方案。该方案由属性权威对证书中所有属性进行预处理,并对预处理结果生成签名;证书拥有者能够根据不同的应用场合移除证书中不相关属性,并计算验证证书必需的额外信息;验证方根据这些额外信息及证书中的签名能有效地验证被出示部分属性的合法性。该方案与现有标准兼容,并具有灵活性好、安全性高及付出额外开销小等特点。

**关键词** 属性证书,细粒度,双重签名,隐秘特征属性,哈希树

**中图法分类号** TP393.08 **文献标识码** A

## Fine-grained Disclosure Scheme for Attribute Certificate

WANG Kai ZHANG Hong-qi REN Zhi-yu JIANG Huang-qin

(Information Engineering University, Zhengzhou 450004, China)

(Henan Province Key Laboratory of Information Security, Zhengzhou 450004, China)

**Abstract** In order to effectively verify X.509v4 attribute certificate after part of attributes is removed, a fine-grained disclosure scheme is proposed. In this scheme, every attribute in certificate is pretreated, and digital signature of the pretreated results was generated by attribute authority. In different scenarios, uncorrelated attributes is removed from certificate and essential validation information is calculated by certification owner. The validation information and digital signature in certificate can be used to validate legitimacy of the attributes disclosed. The scheme has some characteristics as follows: strong compatibility, good flexibility, high security and little additional cost.

**Keywords** Attribute certificate, Fine-grained, Dual signature, Dark feature attribute, Hash tree

## 1 引言

为了解决用户身份信息与权限信息更新频率不同造成的证书管理负担,美国国家标准局(ANSI)X9委员会将用户更新频率较快的权限信息从X.509v3公钥证书[1]扩展项中分离出来,并单独形成一个证书。2000年,在X.509v4中增加了属性证书[2]的概念,其主要用来存储用户的角色信息或其他与用户权限相关的属性信息。目前,属性证书已经被广泛应用于授权管理基础设施(PMI)[3]、信任管理[4]及自动信任协商[5]等相关领域。

属性证书是属性权威(Attribute Authority, AA)用其私钥对用户拥有的一个或多个属性签名后的断言,对属性证书的信任来源于对属性权威的信任。在现有的X.509v4标准中,AA通常将用户的多个属性封装在一本属性证书中,并通过为证书生成签名来保证用户拥有这些属性的合法性。然而,在一些只需要出示证书中部分属性的场合,如证书中其他属性与该次应用无关且为敏感属性时,由于现有的X.509v4标准对证书中的所有属性集直接生成签名,使得验证方无法根据出示的部分属性验证签名的合法性;如果AA临时为用户生成这部分属性相关的属性证书,势必会大大增加AA的管理负担。

针对上述问题,本文提出了一种细粒度的属性证书出示方案。该方案只需要属性权威AA对用户拥有的属性集合进行预处理,并对预处理结果生成签名,证书拥有者就能够根据不同应用需求将证书中不相关属性移除并向验证者提供少量额外验证信息,同时验证者能够利用这些额外信息及证书中的签名验证用户合法拥有证书中剩余的属性。

## 2 相关工作

为了实现证书中部分属性出示及合法性验证,Persiano等提出了加密证书[6]的概念,加密证书使用CA的公钥对证书中的每个属性进行加密处理,再用自己的私钥对整个证书签名。证书拥有者可以任选部分属性出示并将加密证书发送给验证者,验证者首先用CA的公钥验证证书的合法性,再用

到稿日期:2011-12-08 返修日期:2012-03-18 本文受国家863计划项目(2006AA01Z457,2009AA01Z438),国家973重点基础研究发展计划(2011CB311801),河南省科技创新人才计划(114200510001)资助。

王 凯(1987-),男,硕士生,主要研究方向为信任协商、访问控制,E-mail:wklwzy057@163.com;张红旗(1962-),男,博士,博士生导师,主要研究方向为等级保护、信任管理、网络安全;任志宇(1974-),女,博士生,讲师,主要研究方向为授权管理、访问控制;姜皇勤(1987-),男,硕士生,主要研究方向为身份认证、访问控制。

CA 的公钥加密被出示部分属性并与证书中的加密属性进行比较即可验证属性的有效性。然而，该方案由于对属性加密时没有引入随机因子，因此容易受到字典攻击；方案中 CA 为用户的每个属性分别进行非对称加密运算，这与为用户的每个属性都生成属性证书的开销是一样的，效率较低。

廖俊国等提出一种保护证书中敏感属性的方案 SDSA[7]，其对证书中每一敏感属性均采用对称加密算法进行加密，验证方通过密钥交换协议获得密钥并解密被出示的属性。然而，其密钥的生成基于离散对数非对称密码体制，并且需要一个同时被证书颁发者和申请者信任的管理员生成并维护系统参数，接收者在接收和解密证书前也需要获得对应的系统参数。此外，在颁发者生成证书和接收者解密敏感属性时均需要进行大量非对称加解密计算，拥有者在每次将证书发送给对方之前也需要一定量的计算。因此，SDSA 在系统实用方面具有较大缺陷。

为了提高该方案的实用性，廖俊国在文献[8]中提出了一种灵活实用的数字证书中敏感属性保密方案。该方案利用主密钥和哈希函数生成了对证书中各属性加密的对称密钥，较 SDSA 方案的性能有了极大的提高，然而其仍然需要证书颁发者和申请者信任的管理员生成并维护系统参数，存在管理上的不便。

肖淑婷等借鉴内容抽取签名[9]的思想，给出了一种支持属性选择性披露的 ATN 证书描述方案[10]。该方案使接收方在收到经过属性加密或移除处理的证书时，仍能对其完整性和数字签名进行验证，但同样存在受字典攻击威胁的问题。

现有的方案在一定程度上解决了属性证书中属性的细粒度出示问题。然而，从性能、安全性、管理复杂性的角度看，它们均在其中的某些方面考虑不足。

## 3 一种细粒度的属性证书出示方案

针对现有研究中存在的问题，本章首先给出细粒度属性证书出示方案必须满足的一些基本特征：(1)标准化。能够容易地将细粒度的属性证书出示方案与现有的 X.509v4 标准和系统集成。(2)灵活性。证书拥有者可以根据不同应用场合需求，任意选择证书中的部分属性信息进行出示。(3)可验证性。对于被出示的部分属性集合，证书拥有者必须有能力向验证方证明这部分属性是属性权威 AA 用私钥签名的合法属性。(4)安全性。对于未被出示的属性集合，利用证书拥有者向验证方发送的证书和额外信息得到未被出示的属性在计算上是不可行的。(5)高效性。与传统的 X.509v4 标准相比，证书在签名的生成、合法性验证及通信开销等方面所付出的额外代价是能够承受的。

根据上述特征，本文给出了针对性的解决方案。为了便于描述问题，用 AA 表示签发属性证书的属性权威；用 Alice 表示证书拥有者，Bob 为验证者；用 $PE_k(m)$ 和 $PD_k(m)$ 表示签名验证函数和签名生成函数；用 $H(m)$ 表示散列函数；用 $AC$ 表示属性证书；用 $(P_a, S_a)$ 表示实体 $a$ 的公私钥对，其中 $P_a$ 为公钥，$S_a$ 为私钥；用“$\|$”表示级联。

该方案主要由 4 个阶段组成：第一阶段为准备阶段，由 AA 公布一些公共参数；第二阶段为证书生成阶段，主要包括对属性信息的预处理及数字签名的生成；第三阶段为证书交换阶段，Alice 将证书 $AC$ 中与应用场合无关的属性移除，并将 $AC$ 和验证签名合法性所必需的额外信息发送给 Bob；第四阶段为证书验证阶段，Bob 利用额外信息及证书中的签名对被出示部分属性的合法性进行验证。

### 3.1 准备阶段

由 AA 确定并公布以下参数：$H(m)$，$PE_k(m)$，$PD_k(m)$。一般情况下，这些参数公布后将不再发生变化，因而准备阶段工作只需进行一次。

### 3.2 证书生成阶段

#### 3.2.1 X.509v4 属性证书

为使本方案与现有的 X.509v4 中的属性证书标准[2]兼容，本节先对 X.509v4 属性证书的格式进行分析，然后给出适合本文的属性证书描述方式。X.509v4 中定义的属性证书结构如图 1 所示。

```
AttrbuteCertificate::=SIGNED{AttrbuteCertificateInfo}
AttrbuteCertificateInfo::=SEQUENCE
{
  version AttCertVersion DEFAULT v1,   //证书版本号
  holder Holder,              //证书持有者
  issuer AttCertIssuer        //属性证书颁发者
  signature AlgorithmIdentifier,    //属性证书签名时用的算法
  serialNumber CertificateSerialNumber  //属性证书序列号
  attrCertValidityPeriod AttCertValidityPeriod  //属性证书有效期
  attributes SEQUENCE OF Attribute  //证书拥有者的属性集
  issuerUniqueID UniqueIdentifier OPTIONAL  //标识颁发者的身份
  extensions Extensions OPTIONAL   //扩展项
}
```

图 1 X.509v4 标准中属性证书格式

由图 1 可知，属性证书主要由属性证书基本信息 *AttributeCertificateInfo* 和对这些基本信息的数字签名 *SIGNED*{*AttributeCertificateInfo*}两部分组成。其中属性证书的基本信息包括证书拥有者的属性集合 *attributes* 及其他基本信息，属性证书可被形式化定义如下。

**定义 1**(属性证书) 属性证书 $AC$ 可被描述成为一个四元组$\langle OtherInfos1, Attributes, OtherInfos2, Signature\rangle$，其中 $OtherInfos1$ 表示 $Attributes$ 之前的证书版本号、持有者、颁发者、签名算法、序列号和有效期，$Attributes$ 为证书拥有者拥有的属性全集，$OtherInfos2$ 表示 $Attributes$ 之后的颁发者身份标识和扩展项，$Signature$ 表示属性权威 $AA$ 用其私钥 $SAA$ 对证书生成的签名，其中 $Signature=PD_{S_{AA}}(OtherInfos1 \| Attributes \| OtherInfos2)$。

设 Alice 拥有的属性集个数为 $N(N>0)$，则 $Attributes=Attr_1 \| Attr_2 \| \cdots \| Attr_N$。若 Alice 随机选择 $K(0<K\leqslant N)$个属性 $Attrs'$向 Bob 出示，并将 $OtherInfos1$、$OtherInfos2$ 和 $Signature$ 发送给 Bob。Bob 首先利用 AA 的公钥 $P_{AA}$ 对 $Signature$ 进行签名验证，得到 $PE_{P_{AA}}(Signature)=OtherInfos1 \| Attributes \| OtherInfos2$。当 $K\neq N$ 时，Bob 不能根据 $OtherInfos1$、$OtherInfos2$ 和 $Attrs'$恢复出 $OtherInfos1 \| Attributes \| OtherInfos2$，因而不能验证签名的合法性。因此，按照现有 X.509v4 属性证书签名的生成方式，当细粒度出示属性证书中的部分属性后，验证方无法验证证书的合法性。

#### 3.2.2 基于“双重签名”的细粒度属性出示证书

为了解决传统 X.509v4 属性证书在细粒度出示属性时不够灵活的问题，借鉴“双重签名”[11]的思想，首先对属性证书中的属性进行预处理。

如图 2 所示，对于属性集 $Attributes$ 中的每一个属性 $Attr_i$，分别进行哈希运算得到其消息摘要为 $AttrMD_i$。将 $N$ 个消息摘要按序级联并再次哈希，得到整个属性集合的消息摘

要 $AttrsMD$。不同于 X.509v4 标准中签名的生成，基于“双重签名”的属性证书 $AC$ 的签名为 $Signature=PD_{S_{AA}}(OtherInfos1 \| AttrsMD \| OtherInfos2)$。属性权威 AA 将属性证书 $AC=\langle OtherInfos1, Attributes, OtherInfos2, Signature\rangle$ 颁发给 Alice。

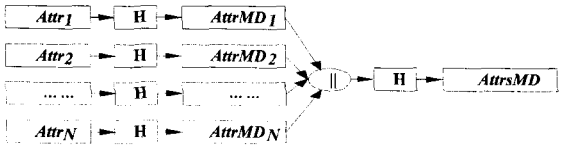


图 2　基于双重签名的属性证书预处理

对于 Alice 而言，她若想任选证书中的 $K$ 个属性出示，则需先将其余 $N-K$ 个属性从证书中移除，然后将证书和 $N-K$ 个被移除属性的哈希值发送给 Bob。Bob 首先计算 $K$ 个被出示属性的哈希值，并与其余 $N-K$ 个属性的哈希值按在证书中的先后顺序级联并哈希得到 $AttrsMD'$；然后将证书中的 $OtherInfos1$ 和 $OtherInfos2$ 与 $AttrsMD'$ 级联得到 $OtherInfos1 \| AttrsMD' \| OtherInfos2$；用 AA 的公钥 $P_{AA}$ 对 $Signature$ 进行验证签名处理得到 $OtherInfos1 \| AttrsMD \| OtherInfos2$ 并与 $OtherInfos1 \| AttrsMD' \| OtherInfos2$ 匹配，若相等则验证成功。由于哈希函数为单向函数，因此 Bob 无法获得其余 $N-K$ 个属性的具体内容。

### 3.2.3　基于“隐秘特征属性”的属性机密性保护

对于证书中未被出示的 $N-K$ 个属性，如果仅用哈希函数进行预处理，在得知属性取值范围时容易受到字典攻击的威胁。例如，年龄属性的取值范围一般在 0 到 200 之间，Bob 最多只需要进行 200 次哈希运算，并与提供验证的哈希值进行比对，就能得到 Alice 的真实年龄。为了保护未被出示属性的机密性，本文通过计算单个属性的隐秘特征属性[12]来进行保护。

**定义 2**(隐秘特征属性)　对属性证书中的每个属性 $Attr_i$ 进行密码学变换 $f(Attr_i)$，使得验证者无法用 $f(Attr_i)$ 获得关于 $Attr_i$ 的任何信息，除非验证者知道生成 $f(Attr_i)$ 的秘密。

Pedersen 承诺[13]是构造隐秘特征属性的一个方法，但其由于计算是非线性的，不适合用于效率要求较高的场合。本文将给出一种更加简洁高效的隐秘特征属性构造方法。

如图 3 所示，设对于属性证书中的每个属性 $Attr_i$，由属性权威 AA 生成一随机位串 $r_i$ 并计算其隐秘特征属性 $AttrPF_i=H(Attr_i \| r_i)$。由于验证者在不知道秘密 $r_i$ 的情况下，即使通过字典攻击也无法根据 $AttrPF_i$ 获得 $Attr_i$ 的任何信息，因此 $AttrPF_i$ 是安全的。

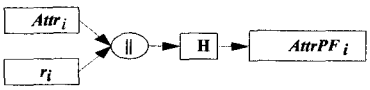


图 3　隐秘特征属性的生成

### 3.2.4　基于哈希树的高效签名生成方法

当属性证书中被签署的属性个数 $N$ 特别大，而 Alice 在某些应用场景下需要出示的属性个数 $K$ 较小时，基于“双重签名”的属性证书在验证过程中，Alice 需要向 Bob 发送大量与本次协商无关的 $N-K$ 个属性哈希值，这将造成在通信中不必要的浪费。

分析“双重签名”中属性证书的预处理过程，可将该过程抽象为一棵 $N$ 叉树，该树有 $N$ 个叶子结点和一个根结点。根结点 Root 的值等于所有叶子结点的值按序级联并经过哈希运算得到的哈希值。

图 4 给出了 $N=7$ 时的一个抽象示例。设 Alice 在某次应用场合下仅需要向 Bob 出示 $Attr_1$ 和 $Attr_2$，图中用“●”表示它们的哈希值，则在验证过程中需要向 Bob 发送后续 5 个属性的哈希值，这将导致效率低下。为了解决该问题，结合 Merkle哈希树[14]的定义，本文给出属性哈希树的定义如下。

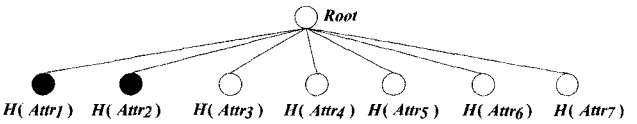


图 4　$N=7$ 的“双重签名”$N$ 叉树抽象

**定义 3**(属性哈希树)　属性哈希树是一棵二叉树，每个非叶子结点都有两个孩子结点，父结点的值等于左右孩子结点的值级联后生成的哈希值，叶子结点为属性证书中每个属性的哈希值。

**引理 1**[15]　对于任何一棵二叉树，如果其终端结点数为 $n_0$，度为 2 的结点数为 $n_2$，则有 $n_0=n_2+1$。

**定理 1**　叶子结点个数为 $N$ 的属性哈希树其非叶子结点总数为 $N-1$。

证明：由定义 3 知，哈希树的每个非叶子结点的度均为 2，其终端结点数 $n_0$ 等于叶子结点个数 $N$。由引理 1，可得非叶子结点的总数为 $N-1$。

**定理 2**　属性哈希树的深度 $L=\lceil \log_2 N \rceil+1$。其中，$N$ 为叶子结点的个数，“$\lceil x \rceil$”表示不小于 $x$ 的最小整数。

证明：当叶子结点个数 $N=2^{k-1}$ 时，哈希树为满二叉树，其深度 $L=k$；当叶子结点个数 $N=2^{k-1}+1$ 时，哈希树的深度 $L=k+1$。那么 $2^{k-1}$ 是深度为 $k$ 的哈希树叶子结点个数的上限，$2^{k-2}$ 是深度为 $k$ 的哈希树叶子结点个数的下限，且当 $2^{k-2}<N\leqslant 2^{k-1}$ 时，$k\geqslant 2$，哈希树的深度为均为 $k$。于是 $k-2<\log_2 N\leqslant k-1$，由于 $k$ 是整数，因此 $k=\lceil \log_2 N \rceil+1$。经验证，当 $k=1$ 时，上式仍然成立。定理 2 得证。

属性哈希树的生成过程如下：首先，计算属性证书中 $N$ 个属性的哈希值，并按照在证书中的先后顺序构成哈希树的叶子结点。然后，将叶子结点按序进行两两配对，并分别生成父结点，父结点的值等于两个子结点值的级联哈希值。当 $N$ 为奇数时，将最后一个未配对的结点与前面生成的 $\frac{N-1}{2}$ 个中间结点一起进行下一轮的两两配对运算。对上述过程进行递归，直到生成根结点为止。图 5 给出 $N=7$ 时的属性哈希树生成过程。

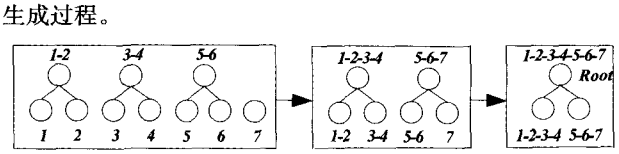


图 5　$N=7$ 属性哈希树生成过程

如果用属性哈希树取代“双重签名”中的 $N$ 叉树，在细粒度出示其中的 K 个属性后，只有当 Alice 向 Bob 发送的哈希值个数小于 $N-K$ 时，哈希树方案才比“双重签名”方案更有意义。如图 6 所示，仍以 $N=7$ 且 Alice 向 Bob 出示 $Attr_1$ 和

$Attr_2$ 为例，为了构造出根结点，Alice 只需要向 Bob 发送结点“3-4”和结点“5-6-7”两个中间结点。相比原来的“双重签名”，这将大大减少通信量。

通过对图 6 进行分析，为了恢复出根结点，必须找到离根结点最近且所有子孙结点均未被出示的中间结点以替代叶子结点的出示。由于一个中间结点至少可以代替两个叶子结点的出示(如结点“3-4”)，不同中间结点所代替的叶子结点集之间不存在交集(如结点“3-4”和结点“5-6-7”代表的叶子结点之间不存在交集)，并且中间结点的总个数比叶子结点总数少 1(定理 1)，因此，可以得出用离根结点最近且所有子孙结点均未被出示的中间结点替代叶子结点的出示是可行的，并且相对“双重签名”其是高效的。

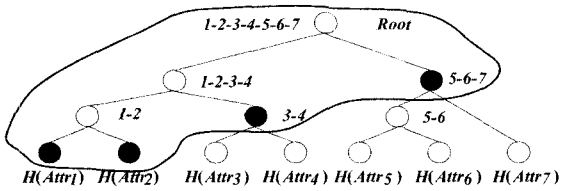


图 6　$N=7$ 属性哈希树细粒度出示图

### 3.2.5　证书生成阶段详细过程

综上所述，下面给出属性权威 AA 在为 Alice 生成证书的过程中需要进行的操作，如图 7 所示。

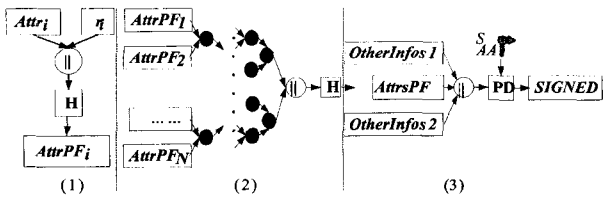


图 7　改造后的属性证书签名构造图

(1)AA 生成 $N$ 个随机位串 $r_i$，$1\leqslant i\leqslant N$，并对属性集合 *Attributes* 中的每个属性 $Attr_i$ 分别计算隐秘特征属性 $AttrPF_i=H(Attr_i \| r_i)$。

(2)将 $N$ 个隐秘特征属性 $AttrPF_i$ 作为属性哈希树的叶子结点，并生成属性哈希树，设属性哈希树根结点的值为 *AttrsPF*。

(3)将属性证书中的 *OtherInfos*1 和 *OtherInfos*2 与 *AttrsPF* 级联，并用 AA 的私钥对其签名得到 $SIGNED=PD_{S_{AA}}(OtherInfos1 \| AttrsPF \| OtherInfos2)$。实际应用中，可能需要先对签名内容进行哈希，本方案在这种情况下仍然是适用的。

生成签名后，AA 按照 X.509v4 属性证书格式将 *OtherInfos*1、*Attributes*、*OtherInfos*2 及生成的签名 *SIGNED* 封装在证书中，并将其颁发给 Alice；将各属性对应的秘密参数 $r_i$ 按在证书中的先后顺序发送给 Alice，这些信息必须通过安全信道秘密发送，实际使用中可通过“数字信封”[11]构建安全信道。

### 3.3　证书交换阶段

在证书交换阶段，Alice 根据应用场景向 Bob 出示属性证书 *AC* 中的部分或全部属性及验证证书有效性所必需的信息，主要包括以下步骤：

(1)Alice 确定证书 *AC* 中可向 Bob 出示的部分属性集，并将不需要出示的属性从 *Attributes* 中移除。设被出示的属性个数为 $K(0<K\leqslant N)$，它们分别为 $Attr_{i_1},\cdots,Attr_{i_K}$。

(2)Alice 需要告诉 Bob 被出示的 $K$ 个属性在证书中的位置及排列顺序，通过计算一个 $N$ 比特的二进制数 *SerNo* 来传递该信息，其计算方法如下：

① 按照 $N$ 个属性在证书 *AC* 中的先后顺序，将 $K$ 个被出示属性与二进制数 *SerNo* 的位建立对应关系。如图 8 所示，被出示属性 $Attr_{i_1}$ 对应属性证书中第 1 个属性，$Attr_{i_2}$ 对应证书中第 3 个属性，$Attr_{i_K}$ 对应证书中第 $j$ 个属性。

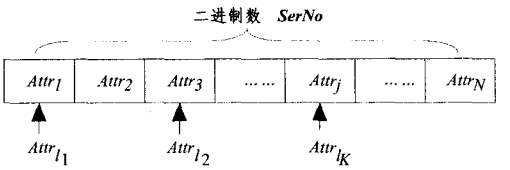


图 8　被出示属性与 SerNo 中位的对应关系

② 二进制数 *SerNo* 的生成过程如下。设 $x$ 表示 *SerNo* 的第 $x$ 位，$f(x)$ 表示第 $x$ 位的值，则有：

$$f(x)=\begin{cases}1, & \text{第 } x \text{ 个属性被出示}\\ 0, & \text{第 } x \text{ 个属性不被出示}\end{cases}$$

(3)Alice 将移除不相关属性后的证书 *AC* 发送给 Bob，其中包括 *OtherInfos*1、*OtherInfos*2、需要出示的部分属性集和改造后的签名 SIGNED；将二进制数 *SerNo* 发送给 Bob；并将被出示的 $K$ 个属性对应的秘密参数 $r_i$ 按照在证书 *AC* 中的先后顺序发送给 Bob；根据哈希树的生成过程将验证签名所必需的中间结点值及这些中间结点在属性哈希树中的位置信息发送给 Bob。为了防止中间人截获攻击，这些信息仍然需要通过安全信道发送。上述被发送的消息中，计算验证签名所必需的中间结点是该方案的技术难点，本文将在第 4 节进行深入讨论。

### 3.4　证书验证阶段

在证书验证阶段，Bob 需要验证 Alice 出示的属性证书是否为权威 AA 签名的合法证书，主要包括以下步骤：

(1)Bob 从属性证书中获取被出示的 $K$ 个属性，并根据其对应的秘密参数分别计算其对应的隐秘特征属性 $AttrPF_i=H(Attr_i \| r_i)$。

(2)Bob 利用二进制数 *SerNo* 确定 $K$ 个隐秘特征属性在属性哈希树叶子结点中的先后顺序，利用接收到的中间结点生成属性哈希树的根结点，并设其值为 $AttrsPF'$。图 6 中被圈住的部分结点为 Bob 可以恢复出的哈希树局部结点。

(3)Bob 从细粒度出示的属性证书中获取 *OtherInfos*1 和 *OtherInfos*2 并与 $AttrsPF'$ 级联得到 $OtherInfos1 \| AttrsPF' \| OtherInfos2$；利用属性权威 AA 的公钥对签名 *SIGNED* 进行加密处理得到 $OtherInfos1 \| AttrsPF \| OtherInfos2$，若两者相等，则验证成功，否则 Bob 不相信 Alice 出示属性证书的合法性。

## 4　中间结点出示算法及时间复杂度

根据不同应用场景需求，Alice 需要出示的属性集合有所不同，则其向 Bob 发送的中间结点也必然有所差异。为了恢复出根结点，必须找到离根结点最近且所有子孙结点均未被出示的中间结点。下面通过对属性哈希树进行 0-1 编码，给出需要被出示的中间结点计算过程。

对于属性哈希树而言，当叶子结点个数 $N$ 确定时，按照

第 3.2.4 节中的生成方法所生成的树必然是唯一的。属性哈希树的非叶子结点必有两个分支，从树的根结点开始，将左分支标记为 0，右分支标记为 1，则完成了属性哈希树的编码。这样编码的好处在于能够容易根据被出示叶子结点的编码迅速判断需要被出示的中间结点编码，而不需要对整棵树进行递归遍历。

如图 9 所示，设被出示的叶子结点编码为 000 和 001，由于被出示结点编码的第 1 位均为 0，说明根结点 Root 的 1 号分支没有被出示的叶子结点，则结点 1 为需要被出示的中间结点；再看被出示结点编码的前两位，由于只出现了 00 分支而未出现 01 分支，则说明中间结点 01 作为没有被出示的叶子结点应该被出示；再看编码的前三位，由于 000 分支和 001 分支均被出示，则不需要出示任何中间结点。因此，需要被出示的中间结点编码为 1 和 01。

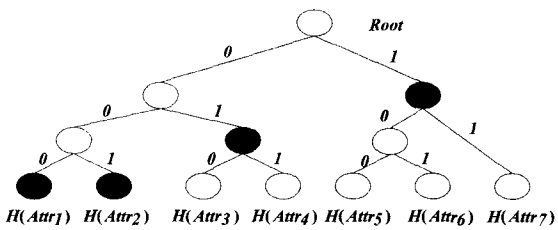


图 9　属性哈希树编码示意图

由定理 2 可知属性哈希树中叶子结点的编码长度 $CodeLength=\lceil \log_2 N \rceil$，下面给出叶子结点个数为 $N$ 的属性哈希树需要被出示的中间结点编码算法。

**算法 1**　GetMidCodes(N,ShowLeafCodes)

输入：N：属性哈希树叶子结点个数；ShowLeafCodes：所有被出示叶子结点编码集

输出：ShowMidCodes：被出示中间结点编码集合

```
1. ShowMidCodes=∅;
2. CodeLength=⌈log2N⌉;
3. For(int i=1; i<=CodeLength; i++)
4.    For(int j=1; j<=2^i; j++)
5.    {TempCode=GetBinaryCode(j-1,i);
6.     If(! IsExistPref(TempCode, ShowMidCodes) &&! IsExist
          (ShowLeafCodes,i,TempCode))
7.    ShowMidCodes+= TempCode;}
8. return ShowMidCodes;
```

上述算法包括 3 个子函数 *GetBinaryCode*($j-1$, $i$)，*IsExistPref*(*TempCode*, *ShowMidCodes*) 和 *IsExist*(*ShowLeafCodes*, $i$, *TempCode*)。其中函数 *GetBinaryCode* 的功能为将整数 $j-1$ 转化为长度为 $i$ 的二进制编码。*IsExistPref* 的功能为判断已被出示的中间结点编码集 *ShowMidCodes* 中是否存在一个中间结点作为 *TempCode* 的前缀出现，该函数主要用来过滤已被出示中间结点的所有子孙结点。函数 *IsExist* 用来判断 *TempCode* 是否在 *ShowLeafCodes* 的前 $i$ 比特前缀所组成的集合中出现。鉴于文章篇幅，这里不给出子函数的具体算法。

该算法从根结点的两个分支编码开始，按照树的层次结构从上往下、从左至右依次遍历属性哈希树的整个分支编码空间。对于每一个编码 *TempCode*，如果已被出示的中间结点编码集 *ShowMidCodes* 中没有一个编码作为 *TempCode* 的前缀出现，并且 *TempCode* 不在已出示叶子结点编码集 *ShowLeafCodes* 前 $i$ 比特组成的集合中出现，则说明以 *TempCode* 编码的结点不在 *ShowMidCodes* 结点集的子分支上，并且以该结点分支的所有叶子结点均未被出示，该结点作为离根结点最近且所有子孙结点均未被出示的中间结点被加入到 *ShowMidCodes* 中。

时间复杂度：函数 *GetBinaryCode* 生成 $i$ 比特二进制数需要进行 $i-1$ 次模 2 运算，设进行一次模 2 运算的时间为 $t1$，则 *GetBinaryCode*($j-1$, $i$) 的运行时间为 $(i-1)\cdot t1$。函数 *IsExistPref* 需要进行 *ShowMidCodes*. *Length* 次字符串匹配，*IsExist* 需要进行 *ShowLeafCodes*. *Length* 次字符串匹配，这两个函数在做每一次字符串匹配时最多需要 *TempCode*. *Length*$=i$ 次字符匹配。设判断两个字符是否匹配的时间为 $t2$，则对于第 $i$ 次 For 循环，其总时间开销为 $\sum_{j=1}^{2^i}[(i-1)\cdot t1+i\cdot t2\cdot(ShowMidCodes.Length+ShowLeafCodes.Length)]$。故该算法的总时间开销 $TimeCost=\sum_{i=1}^{\lceil \log_2 N \rceil}\sum_{j=1}^{2^i}[(i-1)\cdot t1+i\cdot t2\cdot(ShowMidCodes.Length+ShowLeafCodes.Length)]$。由于 *ShowMidCodes*. *Length*$\leqslant N-1$（中间结点总数），*ShowLeafCodes*. *Length*$\leqslant N$（叶子结点个数），则：

$$TimeCost<\sum_{i=1}^{\lceil \log_2 N \rceil}\sum_{j=1}^{2^i}[i\cdot t1+i\cdot t2\cdot(N-1+N)]$$
$$=[t1+t2\cdot(2N-1)]\cdot\sum_{i=1}^{\lceil \log_2 N \rceil}2^i\cdot i$$
$$=[t1+(2N-1)\cdot t2]\cdot[(\lceil \log_2 N \rceil-1)\cdot 2^{\lceil \log_2 N \rceil+1}+2]$$

因此，该算法进行的模 2 运算时间复杂度为 $O(N\cdot \log_2 N)$，两个字符匹配的时间复杂度为 $O(N^2\cdot \log_2 N)$。

## 5　方案性能分析

本文提出的细粒度属性证书出示方案的开销主要包括：证书生成阶段属性权威 AA 的计算和通信开销，证书交换阶段证书拥有者的计算和通信开销，证书验证阶段验证者的计算开销。

### 5.1　证书生成阶段开销分析

证书生成阶段 AA 的计算开销主要包括：$N$ 个隐秘特征属性的计算，属性哈希树中 $N-1$ 个中间结点的计算，1 次数字签名运算。其中 $N$ 个隐秘特征属性的计算及 $N-1$ 个中间结点的计算均采用哈希运算，则证书生成阶段的时间开销仅为 $2N-1$ 次哈希运算和 1 次非对称解密运算。证书生成阶段的通信开销为 AA 向用户发送属性证书 $AC$ 和 $N$ 个秘密参数的开销。

### 5.2　证书交换阶段开销分析

证书交换阶段证书拥有者的计算开销主要包括计算 *SerNo* 和计算需要被出示中间结点的开销。其中计算 *SerNo* 的开销可忽略不计，计算被出示中间结点开销则由第 4 节算法分析可知，其需要进行模 2 运算的时间复杂度为 $O(N\cdot \log_2 N)$，两个字符匹配的时间复杂度为 $O(N^2\cdot \log_2 N)$。证书交换阶段的通信开销为移除属性后的证书、$N$ 比特二进制数、$K$ 个秘密参数和不多于 $N-1$ 个中间结点哈希值。

### 5.3　证书验证阶段开销分析

证书验证阶段验证者的开销主要包括 $K$ 个被出示属性的隐秘特征属性计算，利用 $K$ 个隐秘特征属性和接收到的中

（下转第 130 页）

据录入和查询工作，如投标价格、竞争单位信息等。使用OSF后的软件可以在无网络连接的环境下录入和处理招标信息，极大地改善了租户体验，并且增强了软件可用性。查看β态操作和用户信息的界面如图4所示。

**结束语** 本文描述了离线SaaS应用框架(OSF，Offline SaaS Framework)的构件结构，并且详细阐述了其中面向操作的构件框架的设计与实现。所有这些方法和实现都将对离线SaaS应用乃至对环境可感知的智能软件的研究和应用做出一定贡献，框架的实现还有助于提高SaaS应用开发效率，保证其可用性以及改善用户体验。进一步的工作包括在OSF中添加安全性构件、多线程优化构件，以及实现基于NoSQL数据库的面向数据的构件框架。

## 参 考 文 献


[1] Gruszczynski P, Osinski S, Swedrzynski A. Offline business objects: enabling data persistence for distributed desktop applications[C]// Proc. of OTM Conferences. 2005: 960-977

[2] Nguyen P, Sharma D, Tran D. Smart clients and small business model[C]// KES. 2005: 730-736

[3] Michaud P. The challenges of allowing offline usage in a SaaS based system[EB/OL]. http://www.cloudave.com/1714/the-challenges-of-allowing-offline-usage-in-a-saas-based-system, 2011

[4] Yang Y. Supporting online web-based teamwork in offline mobile mode too[C]// Proc. of WISE 2000. 2000: 486-490

[5] Ananthanarayanan G, Blagsvedt S O, Toyama K. OWeB: A framework for offline web browsing[C]// Proc. of LA-Web 2006. 2006: 15-24

[6] Ijtihadie R M, Chisaki Y, Usagawa T, et al. Offline web application and quiz synchronization for e-learning activity for mobile browser[C]// Proc. of 2010 IEEE Region 10 Conference. 2010: 2402-2405

[7] Google Gear[OL]. http://code.google.com/apis/gears, 2011-07

[8] Goncalves E E M, Leito A M. Offline execution in workflow-enabled Web applications[C]// Proc. of QUATIC 2007. 2007: 204-207

[9] JPetStroe[OL]. http://java.sun.com/ developer/technicalArticles/J2EE/petstore/ and http://www.jwebhosting.net/ jpetstore, 2011-07

[10] Cheng X, Zhang X D. Accurately modeling workload interactions for deploying prefetching in Web servers[C]// Proc. of International Conference on Parallel Processing. 2003: 427-435


(上接第98页)

间结点恢复属性哈希树根结点的计算，及利用AA的公钥进行签名验证运算。则证书验证阶段需要的时间开销最多为$K+N-1$次哈希运算及1次非对称加密运算，其中$K+N-1$次哈希运算包括计算$K$个隐秘特征属性的$K$次哈希和恢复属性哈希树根结点时不多于$N-1$个中间结点哈希值的计算。

因此，本方案计算开销除了X.509v4证书的正常签名和解签名外，仅额外增加了$(2N-1)+(K+N-1)\leqslant 4N-2$次哈希运算、$O(N\cdot\log_2 N)$次模2运算及$O(N^2\cdot\log_2 N)$次字符匹配运算。从通信复杂度分析，本方案相对于X.509v4证书仅额外增加了$N+K\leqslant 2N$个秘密参数、1个$N$比特二进制数和不多于$N-1$个哈希值。证书的产生、签名及合法性验证比传统的X.509v4标准所付出的额外计算是能够承受的；额外增加的通信量也是很少的。

**结束语** 属性证书作为权威机构认可用户所拥有属性集合的一种有效凭证，被授权管理基础设施(PMI)、信任管理及自动信任协商等领域广泛使用。现有的属性证书标准X.509V4由于直接对证书中所有的属性集合生成签名，在要求出示证书中部分属性的场合，验证方无法根据已出示的部分属性验证签名的有效性。本文提出了一种细粒度的属性证书出示方案，该方案具有与现有标准兼容、灵活性好、安全性高、网络通信负担小等优点，解决了现有研究中存在的一些问题。下一步研究还需要对X.509v4证书中部分属性移除及证书验证时所需的系统进行相应的开发。

## 参 考 文 献


[1] ITU-T. ISO/IEC9594-8 Rec. X.509 The Directory: Authentication Framework[S]. 2000

[2] Nykaen O. Attribute Certificate in X.509 [EB/OL]. http://www.hut.fi/~tpny-kane/netsec/final, 2000

[3] Chadwick D W, Otenko O. The PERMIS X.509 Role Based Privilege Management Infrastructure[C]// Proceedings of SACMAt'02. Monterey, California, USA, 2002

[4] Blaze M, Feigenbaum J, Lacy J. Decentralized Trust Management[C]// Proceedings of the 17th Symposium on Security and Privacy. Oakland: IEEE Computer Society Press, 1996: 164-173

[5] Winsborough W H, Seamons K E, Jones V E. Automated Trust Negotiation[C]// Proceedings of DARPA Information Survivability Conference and Exposition. IEEE Press, 2000: 88-102

[6] Persiano P, Visconti I. User Privacy Issues Regarding Certificates and the TLS Protocol[C]// Proceedings of 7th ACM Conference of Computer and Communications Security Athens. Greece, November 2000

[7] 廖俊国，洪帆，李俊，等. 在信任协商中保密证书的敏感属性[J]. 通信学报，2008，29(6)：20-25

[8] 廖俊国，凌乐真，朱彬. 一种灵活实用的数字证书中敏感属性保密方案[J]. 计算机科学，2010，37(6)：128-131

[9] 龚俭，吴桦，杨望. 计算机网络安全导论[M]. 南京：东南大学出版社，2007

[10] 肖淑婷，吴国新，孙啸寅. 支持属性选择性披露的ATN证书描述方案[J]. 计算机工程，2010，36(9)：142-144

[11] 陈性元，杨艳，任志宇. 网络安全通信协议[M]. 北京：高等教育出版社，2008：195-198

[12] Schneier B. 应用密码学(第二版)[M]. 北京：机械工业出版社，2000

[13] Pedersen T. Non-interactive and Information theoretic secure verifiable secret sharing[C]// Proceedings of CRYPTO'91. volume 576 of Lecture Notes in Computer Science, Springer, 1991: 129-140

[14] Merkle R. Protocols for Public Key Cryptosystems[C]// Proceeding of the IEEE Symposium on Research in Security and Privacy. Oakland, California, April 1980

[15] 严蔚敏，吴伟民. 数据结构(C语言版)[M]. 北京：清华大学出版社，1997：123-124