一个基于逆向搜索的分布式证书路径构建算法

杨杰[1,2]， 丁伟[1,2]

（1. 东南大学计算机科学与工程系，南京 210096）

（2. 中国教育科研网华东北地区网络中心，南京 210096）

摘要：数字证书的验证是 PKI 安全实施的关键，PKIX 规定证书的验证一般分为证书路径的构建和证书路径的验证这两个步骤，然而标准中对证书路径的构建方法未加阐释，本文在分析了现有证书路径构建算法的基础上，提出了一个基于逆向搜索的分布式证书路径构建算法。

关键字：PKI；证书路径构建；分布式算法；

A distributed algorithm to construct a certificate path in the reverse direction

Yang Jie[1,2], Ding Wei[1,2]

(1. Department of Computer Science and Engineering, Southeast University, Nan Jing 210096)

(2. China Education and Research Network Eastern(North) Regional Center, Nan Jing 210096)

Abstract: Certificate verification is a key component in implementation of PKI. And according to the PKIX, it includes two steps to verify a certificate: certificate path construction and certificate path verification. But the PKIX makes silence on certificate path construction and there are very little existing literatures that discuss this issue. So this paper proposes a distributed algorithm in the reverse direction to construct a certificate path.

key words: PKI, certificate path construction, distributed algorithm

1. 前言

公开密钥体系，如 RSA、DSA 和 ElGamal 等，能用于签名和加密等多种目的，因而被广泛适用于各种网络安全协议，如 IPSec、SSL 和 S/MIME 等。

公开密钥体系应用的核心问题是保证用户和其公钥对应关系的真实性[3]，也就是如何保证一个公钥确实为他所声称的主体所拥有。为解决这个问题，引入了数字证书。一个数字证书（以后简称为“证书”）是一个可信第三方，对某个用户和他的公钥的对应关系的一个保证。数字证书由第三方颁发，通过绑定用户标识和公钥，第三方使用自己的私钥对这种绑定关系进行签名来保证此绑定的真实性。任何对第三方信任的用户，都有充分的理由信任这种绑定关系。而所有生成、发布、分发和撤销数字证书的软件、硬件、策略和人员所组成的集合，就是所谓的公钥基础设施 PKI（Public Key Infrastructure）。

证书在使用之前，需要对证书进行验证，根据 PKIX 规范[1]的建议，一个证书的验证过程分为两步：1）证书路径的构建，证书路径是一个证书的序列，该序列满足如下要求：对任一的证书 $x\in(1,n-1)$，x 的主体正好是 $x+1$ 的颁发者；$x=1$ 的证书是一个自签名证书；$x=n$ 的证书是一个终端证书。2）证书路径的验证，主要是确定证书路径中每个证书的有效期、撤销状态、完整性以及是否遵循规定的约束条件等，证书路径验证的算法详见[1]。

PKIX 规定了详细的证书路径验证算法，但是对证书路径的构建算法却只字未提，同时业界对此问题的解决也缺乏统一的标准，学术研究也非常有限。本论文将对现有的一些集中式证书路径构建算法作一些总结分析，然后着重阐释一个分布式的构建证书路径的算法。

2. 证书路径构建的传统集中式算法

根据 PKIX 的标准，证书一般存储在目录里，端系统通过 LDAP 协议进行访问和存取。X509 证书中的主体名称和颁发者名称，一般采取层次性的 X500 命名方式，它们可以直接映射到目录里的一个实体。证书既可以存储在证书颁发者的目录实体下面，也可以存储在证书主体的目录实体下，或者两个地方都存储。端系统通过集中访问这些目录实体，就可以建立从信任锚到目标证书的证书路径，然后对这个路径进行验证。

创建证书路径的方法有两种，一种是正向搜索，也就是从目标证书开始向信任锚步进建立证书路径；一种是逆向搜索，从信任锚开始向目标证书建立路径。

一般根据证书在目录中存储的位置的差异，可以采取不同的证书路径建立方法。如果证书存储在证书主体的目录下，采用正向搜索比较方便。此时，首先检索目标证书主体目录下的所有证书，每个证书都作为一个候选，从候选中依次取出一个证书，搜索此证书

作者简介：杨杰（1981/9），男（汉族），工学硕士，研究方向：网络安全、分布式系统；丁 伟（1962），女（汉族），工学博士，教授，博士生导师，研究方向：网络安全、网络行为学。

颁发者目录下的证书，把它们作为新的候选者，以此类推，直到碰见一个信任锚。如果证书只存储在证书主体的目录下，使用逆向搜索将会非常困难。但是如果证书是存储在证书颁发者的目录下，则使用逆向搜索更为方便[3]。

另外，CA 的信任模型对搜索方向的选择也有重要影响，如果一个组织里的 CA 是严格按层次模型划分的，则采取正向搜索会更加方便，因为每次搜索都只有一个候选者。如果 CA 的组织结构为一般的信任模型，如网状结构的模型，根据文献[3]的研究，逆向搜索将会更加简单，因为在搜索的过程中可以使用一些优化技术来迅速排除一些非法的路径，使得证书构建更加快捷，这些技术包括：名字限制和策略执行等。

3. 分布式算法

3．1 集中式算法的问题和分布式算法的可行性

仔细研究证书路径构建的一般算法，不难发现无论是正向搜索或者逆向搜索，都面临一个共同的问题：上层 CA 的组织模型将直接影响证书路径构建的复杂度。例如，一个树状层次模型的 CA 结构中，采用正向搜索建立证书路径，每一步都只有一个候选证书，但是在一个网站模型中，无论是正向还是逆向搜索都可能同时存在多个候选证书，端实体在创建证书路径的过程中，需要遍历所有的这些候选证书，由于传统的证书路径构建是一般在端系统中进行，而端系统又对 CA 的组织结构认知甚少，这就造成证书路径构建的极端复杂性。

同时，通过研究发现，证书路径构建的复杂性还体现在端系统到每个目录检索证书的过程，端系统必须首先学习和认知每个证书的存储位置，才能在构建证书路径的过程中顺利检索，而实际上端系统要认知每个证书的存储位置是非常困难的。相反，每个 CA 对自己发布所有证书的存储位置是却非常清楚，如果能利用 CA 对证书存储位置的知识来创建证书路径，必然会大大降低证书创建的复杂性。在此目标驱动下，下面提出了一个分布式算法，把证书的检索放到各个 CA 去执行，最后把所有 CA 的检索结果综合起来构成证书路径。

3．2 算法描述

算法的工作过程如下：当一个端系统需要验证一个终端证书 cert 时，它将<cert,n>作为参数发送给他的每一个信任锚，委托它们建立证书路径，这里的 n 是用户指定的证书路径的最大长度。信任锚 x 在接受到端系统的请求后，创建一个证书路径创建报文<REQ,<<<x,request ID>,n>,cert>>，REQ 是一个标志字节，x 为信任锚的节点信息，requestID 是一个在每个 CA 中的非重复性计数器，n 是最大路径长度，cert 是要验证的证书。信任锚然后将此请求报文发给自己的证书路径构建 deamon 程序，每个 CA 中都运行着这样的一个 daemon 程序，它们的执行的操作如下面的算法所示。当算法结束，信任锚将结果返回给端系统。daemon 程序执行的算法如下所示。

```
var  Q        ：当前已接受的请求报文队列，队列中每个节点的结构：<传送此请求节点信息，请求报文>；
     C        ：本 CA 颁发的所有 CA 证书的对应的 CA 结点的连接信息；
     T        ：记录本地 CA 到某一目标节点的证书路径。
     REQ：常量，标识一个请求报文；
     ACK：常量，标识一个确认报文；
     ACK_ERR：常量，标识一个报文是查找失败的 ACK；
     ACK_SUC：常量，标识一个报文是查找成功的 ACK；
     CAcert   ：本 CA 的证书；
begin (*接受一个报文 P<t,<h,b>>，P 来自节点 X*)
    if t=REQ
        begin (*将 P 解析为<t,<<<n,ID>,TTL>,cert>>*)
            forall q∈Q do if P<n,ID>=q<n,ID> then exit;
            if cert 中的颁发者与本 CA 匹配  then
                    (*发送报文<ACK,<ACK_SUC,<<n,ID>,CAcert>>>至 X*)
            else begin
                    (*新建一个请求 q1=<REQ,<<<n,ID>,TTL-1>,cert>>)
                    if TTL-1=1 then
```

```
                    (*发送报文<ACK,<ACK_ERR,<n,ID>>>给 X*)
                else
                    begin
                        if C 为空 then
                            (*发送报文<ACK,<ACK_ERR,<n,ID>>>给 X*)
                        else
                            begin
                                (*将<X,P>插入 Q，并获得一个位置号 i*)
                                (*创建一个 Si*)
                                forall c∈C do
                                    (*发送 q1 至 c*)
                                    (*在 Si 中创建一个节点记录 c 的信息*)
                            end
                    end
            end
    end
else if t=ACK
    begin
        (*解析报文为<t,<<acktype,<n,id>>,ackbody>>*)
        if acktype=ACK_ERR then
            begin
                forall <y,q>∈Q do if P<n,ID>=q<n,ID> then 赋值 q 的编号给 i;
                (*从 Si 中找出与 X 结点信息相同的记录并删除*)
                if Si 为空 then
                    begin
                        if n 等于本地节点信息 then exit;
                        else
                            begin
                            (*发送报文<ACK,<ACK_ERR,<n,ID>>>给 y*)
                            (*从 Q 中删除第 i 个节点*)
                            end
                    end
            end
        else if acktype=ACK_SUC then
            begin
                (*解析报文为<t,<<acktype,<n,id>>,<certpath>>>*)
                forall q∈Q do if P<n,ID>=q<n,ID> then 赋值 q 的编号给 i;
                (*从 Si 中找出与 X 结点信息相同的记录并删除*)
                (*将<<n,cert>,<certpath>>插入 T*)
                (*发送报文<ACK,<<ACK_SUC,<n,ID>>,<CAcert,<certpath>>>>给 n*)
                if Si 为空 then
                    if n 等于本节点的信息 then
                        return T;
                    else
                        (*从 Q 中删除第 i 个节点*)
            end
```

end

end

3．3 算法分析

任何时候，系统中只存在着两类报文，一类是证书路径创建请求报文，一类是对请求的 ACK 确认报文。前者的报文定义如下：<REQ,<<<anchor,request ID>,TTL>,cert>>，REQ 是一个标志字节，anchor 为信任锚的节点信息，requestID 是一个在每个 CA 中的非重复性数字，TTL 是最大路径长度，cert 是要验证的证书。ACK 报文又分为两种，一种用于在未找到合适路径时降错误通知上一层 CA，定义语法为：<ACK,<ACK_ERR,<anchor,request ID>>>，ACK 和 ACK_ERR 都是报文标识字节，anchor 和 request ID 与请求报文中的定义相同；另一种用于找到证书路径时返回证书路径的报文，报文格式为：<ACK,<ACK_SUC,<<anchor,request ID>,<cert>>>>，这里的<cert>是一个证书的集合，每个接受到此消息的 CA 都会将自己的证书放入这个集合，并将新的集合返回给上一级 CA。

此算法将终止于 3 种情况：1）到达用户规定的最大路径长度仍然未找到目标证书；2）未达到最大路径长度，但是已经搜索到最底层的 CA 仍然没有找到目标证书；3）搜索到目标证书。前两种情况将返回标识错误的 ACK，第 3 种返回标识成功的 ACK。

此算法中，每个 CA 节点需要的空间复杂度与自己所颁发的 CA 证书个数成线性关系，大大优于传统的基于端系统的集中式证书路径构建算法（此时的空间复杂度依赖于 CA 的组织结构）。算法的报文数，依赖于用户规定的证书最大路径长度和 CA 的信任模型结构。

文献[3]中提到在包含双向交叉证书的网站信任结构中，可能会产生路径回路，本算法可以有效的预防这种情况，因为当 CA 接受到一个具有相同<anchor,request ID>的路径构建请求报文时，它将丢弃此报文，并返回错误。

下面对一个实例进行分析。如图 1 所示，EE1 要验证 EE2 的证书，a 为 EE1 的信任锚，虚线为报文传输路径，最后将获得两条有效路径：(a,d,e,ee2)，(a,d,f,e,ee2)，可以根据证书路径长度和其他策略来选择一条最合适的路径作为最终的证书路径。

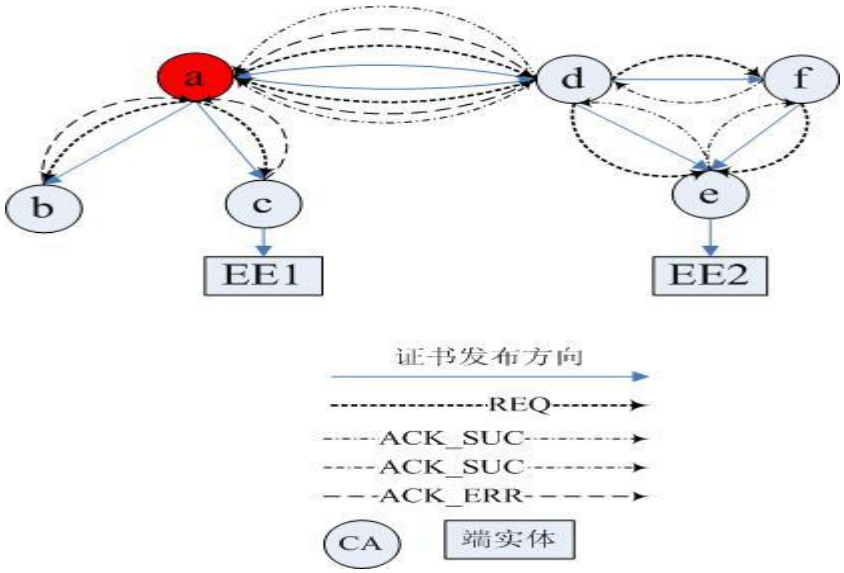

图 1

4． 总结

本论文在深入分析传统的证书路径构建算法的基础上，发现传统集中式的证书路径构建算法中，证书验证的实施者端系统由于对 CA 组织结构的复杂性认识不足，造成了证书路径构建的复杂性，针对此问题，本论文设计了一个分布式的证书路径构建算法，充分利用了 CA 对证书存储位置的知识，来减少证书路径构建的复杂性。通过在华东北地区网络中心自主开发的 CA 服务器 CALock 上实现上面的算法，证明其是可行性。而如何将策略执行等路径搜索优化技术引入此算法，来提高算法的效率，将是下一步的研究重点。

参考文献

[1]RFC 2459, Housley, R., Ford, W., Polk, W., and Solo, D.. “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, January 1999

[2]Ray Hunt. PKI and Digital Certification Infrastructure.

Proceeding of the 9th IEEE International Conference on Networks, 2002

[3]Yassir Elley, Anne Anderson, etc. Building Certification Paths: Forward vs. Reverse, In Network and Distributed System Security Symposium Conference Proceedings:2001.

[4]Understanding Certification Path Construction, PKI Forum white paper, September 2002.

[5]Mahantesh Halappanavar, Ravi Mukkamala: ECPV: Efficient Certificate Path Validation in Public-key Infrastructure. Data and Applications Security XVII: Status and Prospects, IFIP TC-11 WG 11.3 Seventeenth Annual Working Conference on Data and Application Security, August4-6, 2003, Estes Park, Colorado, USA. Kluwer 2004, ISBN 1-4020-8069-7: 215-228.