ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.2, February 2006, pp.305−314
DOI: 10.1360/jos170305

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 Hilbert 曲线的许可证存储策略及查找算法*

高 迎+, 程涛远, 王 珊

(中国人民大学 信息学院,北京 100872)

Certificates Storage Strategy and Search Algorithm Based on Hilbert Curve

GAO Ying+, CHENG Tao-Yuan, WANG Shan

(Information School, Renmin University of China, Beijing 100872, China)

+ Corresponding author: Phn: +86-10-86758801, E-mail: ygao517@sina.com, http://www.ruc.edu.cn

Gao Y, Cheng TY, Wang S. Certificates storage strategy and search algorithm based on Hilbert curve. *Journal of Software*, 2006,17(2):305−314. http://www.jos.org.cn/1000-9825/17/305.htm

Abstract: It is effective to use trust-management (TM) systems addressing authorization in decentralized environments. However, how to store certificates is an important and unresolved problem in this research area, which determines certificate chain discovery algorithm. In this paper, a new strategy to store certificates is put forward by using two-dimensional certificate information and Hilbert space filling curve. This strategy has not only the characteristic of load balance but also enough flexibility to search for certificates. An optimized certificates search algorithm is put forward, based on the query consisting of partial keyword. In addition to this, an algorithm to discover certificate chain is brought forward for creating the minimum certificates graph reducing the network traffic greatly.

Key words: trust-management; certificate; certificate chain; DHT (distributed hash table)

摘 要: 在分布式环境下,利用信任管理机制来实现存取控制已得到人们的一致认同.但是,许可证的存储策略一直是这个领域中一个尚未完全解决的重要问题,而且它直接影响到许可证链的查找等问题.提出了利用许可证的发布者和主体两维信息,采用分布哈希表和 Hilbert 曲线对许可证进行分布定位的新的许可证存储策略.这种存储策略不仅具有很好的负载平衡的特性,而且为许可证的查找提供了充分的灵活性.同时,利用 Hilbert 曲线生成时的递归特性及其所具有的局部保持性,实现了在分布式环境中基于部分关键字的许可证查找.在此基础上,提出一种许可证链查找算法,实现了在查询过程中构造最小的许可证图,从而大幅度减少网络中的信息传输量.

关键词: 信任管理;许可证;许可证链;分布哈希表
中图法分类号: TP393 文献标识码: A

近年来,为了解决大规模分布环境下的存取控制问题,一些信任管理系统相继提出,并取得了很好的效果[1−6].RT(role-based trust-management language)[2,7]是具有强大功能和高灵活性的信任管理系统的典型代表.它

* Supported by the National Natural Science Foundation of China under Grant Nos.60473069, 604963205 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2005AA4Z3010 (国家高技术研究发展计划(863))

Received 2004-08-13; Accepted 2005-05-08

结合了基于角色的存取控制(role based access control,简称 RBAC)和信任管理系统(trust-management,简称 TM)的优势,克服了传统的存取控制链(access control list,简称 ACL)和基于用户能力系统(capability-based system)方法不支持匿名处理、可扩展性差的缺点.因此,它更能适应大规模的分布式环境的要求.在这种信任管理的机制中,存取控制的决定过程就是在分布式网络中查找从资源所有者到资源请求者的授权许可证链的过程.

在信任管理机制中,大量许可证如何分布存放是实现存取控制的一个主要研究问题[4,5,8,9],RT 信任管理系统同样面临着这个问题[1].由于许可证的分布存储策略直接决定了许可证链的查找方法,因此,分布存储策略和在此基础上的许可证链查找算法都是期待解决的问题.如果许可证集中存放,用户在查找过程中进行许可证的定位会很简单,但是它违背了 RT 系统所具有的分布特性和可扩展性[2].目前提出的分布存放的策略有 3 种[1]:许可证与许可证的发布者一起存放;许可证与许可证的主体一起存放;部分许可证与发布者一起存放,部分许可证与主体一起存放.前两种方式很可能造成许可证过分集中,从而产生瓶颈现象,而且它们只支持单向的许可证链查找,不能灵活地选择查找方向,从而构造较小的许可证图;在最后一种方式中,用户需要利用构造最小许可证图的策略,选择许可证的存放位置,同时还要考虑避免出现瓶颈现象.这种方式对用户的要求很高,在实际情况中并不可行.

为了解决上述问题,本文提出了一种新的许可证存储策略:采用 Chord 协议管理分布式网络,基于许可证的发布者和主体两维信息,利用 Hilbert 空间填充曲线实现许可证从二维空间到一维存储空间的映射[10],使用分布哈希表(distributed hash table,简称 DHT)实现许可证的网络定位[11].这种分布策略能够很好地保持许可证的空间相对位置关系,为许可证及许可证链的查找都提供了充分的灵活性,而且完全避免了许可证查找过程中出现的瓶颈现象,具有很好的负载平衡特性.在此基础上,我们充分利用了 Hilbert 曲线生成时的递归特性及其所具有的局部保持性,实现了基于部分关键字的许可证优化查找算法.这是实现许可证链查找算法的前提.我们提出的基于部分关键字的优化查找算法具有很好的执行效率:参与处理的节点数目与存放相关许可证的节点数目基本接近,发送的查询数目和串行查找的深度大大减小.最后,在此基础上,本文提出了基于最小许可证图的许可证链查找算法.我们从两个方向同时进行许可证链的查找,在进行双向查找的过程中不断试探、比较当前正反两个方向所形成的许可证分支数目的大小,选择小的一端作为查找的前进方向,使得查找到的许可证汇合点与许可证图中分支大的一端尽量接近,从而构造出一个比较小的许可证图.这种策略可以保证查询过程中需要的许可证数量尽量少,从而能够减少网络的信息传输量,提高许可证链的查找效率.

本文第 1 节讨论现有的许可证存放策略存在的问题.第 2 节给出我们提出的一个许可证的存储策略和基于部分关键字的许可证的查找优化算法.第 3 节给出基于最小许可证图的许可证链的查找算法.第 4 节给出实验评价结果.第 5 节介绍相关工作.最后是结论.

1 现有的许可证存放策略存在的问题

Li 在文献[1]中指出,在用于实现存取控制的信任管理系统中,许可证的分布存放决定了许可证链的查找算法.如果全部的许可证都与许可证的发布者一起存放,那么必须采用逆向的许可证链查找算法:从用户申请的角色出发,查找相关许可证,构造许可证图,从而找到许可证链.如果全部的许可证都与许可证的主体一起存放,那么必须采用正向许可证链查找算法,从用户出发才能找到许可证链.由此可见,这两种许可证存放策略严格决定了许可证链查找的执行方向.

同时,Li 在文献[1]中还指出,对于同一个许可证的集合,在判断一个用户是否拥有某一角色的时候,如果从不同的方向进行相关许可证的查找,最后构造出的许可证图的大小是完全不同的.例如,许可证 register.teacher←litong 表明许可证的主体 litong 拥有角色 register.teacher,register 是许可证的发布者.如果有几千个教师拥有这样的角色,而且这些许可证都与发布者一同存放,那么这些许可证将全部存放在 register 所对应的节点上.为了找到满足发布者是 register.teacher 的许可证,必须从 register.teacher 出发使用逆向许可证链查找算法.使用这种查找方法,我们将涉及到几千个相关许可证.但是,如果这几千个许可证分别与各自的主体一同存放,litong 所对应的节点只存放了一个许可证.为了找到主体是 litong 的许可证,我们从 litong 出发进行正向许可证链的查找将

只涉及一个相关的许可证.这个最简单的例子足以说明,对于同一个查询,从两个方向查找得到的许可证图的大小会有很大的差别.

为了尽量在查找中创建一个比较小的许可证图,提高查找效率,避免出现瓶颈现象,Li 在文献[1]中提出由用户来决定将一部分许可证与发布者一同存放,另一部分与主体一同存放,使得许可证的查找可以双向进行.在这种策略中,用户的合理指定将会在查找过程中构造出一个最小的许可证图,同时可以尽量避免瓶颈现象的发生.但在实际操作中,由用户进行这种难度很大的指定是不可行的,很容易因为找不到存在的许可证而作出错误的回答.

Pekka Nikander 和 Lea Viljanen 在文献[12]中提出使用域名服务器来实现 SPKI 许可证的存放,很好地利用了现有的 DNS 服务器机制.但这种机制仍然存在问题:首先,以这种方式进行许可证的存放同样没有解决可能出现的瓶颈现象;其次,为了提高查找许可证链的灵活性,使查找方向能够不受许可证存储的约束而自由选择,他们采用将许可证在发布者和主体所对应的 DNS 服务器中存放两次来实现.这样做必须保证许可证的一致性,因而增加了系统的负担;第三,RT 系统本身具有“平等”的特性,而 DNS 是具有层次关系的服务器,它的这种层次关系对于用户的映射和许可证的查找不仅没有任何帮助,而且会造成查找的负担.

可见,要提高许可证链的查找效率,必须寻找一种适合 RT 特性、适应分布式环境中的开放性和可扩展性的要求,能够为许可证查找提供更多灵活性的存放策略.在此基础上,将构造最小许可证图的工作交给许可证链的查找算法来完成.我们提出利用 Hilbert 空间填充曲线实现许可证从二维空间到一维存储空间的映射[10],并使用分布哈希表实现许可证网络定位[11]的方法,满足了许可证的存放和查找要求.

2 许可证的分布存放策略

为了避免现有存储策略中存在的问题,我们不再仅仅基于一维的发布者或者主体的信息,而是使用发布者和主体两维信息进行许可证的分布定位.同时,为了能够在许可证链查找过程中灵活地选择查找方向、构造最小的许可证图,需要不断查找具有(*,主体)和(发布者,*)特点的许可证.我们希望这些在二维空间中连续的许可证映射到一维空间中也能够尽量连续,从而方便许可证链的查找.Hilbert 空间填充曲线满足这个要求.它具有局部性保持(locality preserving)的特点,曲线上邻近的点在空间上也是邻近的.因此,我们选择它作为许可证从二维空间到一维空间的映射.这样,满足(*,主体)和(发布者,*)特点的许可证会在 Hilbert 空间填充曲线上分布成几个片断(segment).这种分布既避免了许可证信息存放的过于集中,又不会造成分布的过于分散、查找代价过大.

2.1 许可证信息的映射

为了实现将许可证信息映射到一维空间,并保证均衡分布,我们将许可证信息的发布者和主体分别经过哈希处理,映射到数据空间$[0,2^N]$中.我们规定:在二维空间中,一个许可证信息的横坐标取发布者经哈希函数处理以后的数值,纵坐标取主体经哈希函数处理以后的数值.这样,分布式系统中的许可证信息就成为二维空间的坐标点.接下来,系统使用 Hilbert 空间填充曲线扫描二维空间,得到一个一维的 Hilbert 数字序列 s,即 $S=(S(1),S(2),S(3),\ldots,S(2^{2N}))$.许可证信息经过 Hilbert 曲线映射以后,查找具有(*,主体)或(发布者,*)特点的许可证的问题,也就转换成了对曲线上几个片段的定位问题.

Hilbert 曲线把许可证信息映射成一维的索引后,系统把它存放到分布式系统中.为了在分布式系统中进行高效、正确的查找,系统采用 Chord 协议[13]来组织加入分布式网络的计算机节点.在 Chord 中,参与的计算机节点也采用哈希函数映射到数据空间$[0,2^{2N}]$中.这样,分布式网络中的计算机节点和许可证信息都被映射到一个相同的数据空间之中.

例 1:在图 1 中,library.read←org.member(1)经 Hilbert 曲线映射后,一维索引是 10001100,保存于 D 节点;

org.member←univ.teacher(2)经 Hilbert 曲线映射后,一维索引是 01110100,保存于 C 节点;

univ.teacher←registerb.teacher(3)经 Hilbert 曲线映射后,一维索引是 10101000,保存于 D 节点;

registerb.teacher←li tong(4) 经 Hilbert 曲线映射后,一维索引是 01011100,保存于 B 节点.

2.2 基于部分关键字的许可证的查找

在这种许可证存放策略的基础上,我们提出一种基于部分关键字的许可证的优化查找算法.算法的基本思想是:首先,根据 Hilbert 曲线函数将包含部分关键字的二维查询映射成一组 Hilbert 数字;然后,利用 Chord 协议将这些数字映射到相关的计算机节点;最后,完成满足条件的许可证信息的查找.我们首先介绍两种简单的算法(串行查找和并行查找),然后再介绍我们的优化查找算法 HBLS(Hilbert-Based level search).

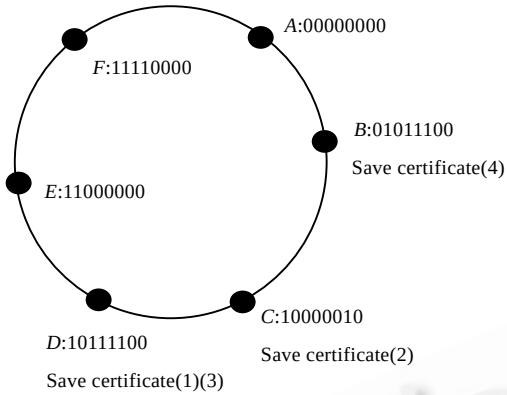


Fig.1 Certificates mapping graph

图 1 许可证映射图

设定 Chord 的数据空间是$[0,2^{2N}]$,发送者和主体各自的一维数据空间是$[0,2^N]$.从节点 N 发出一个查找(发送者,*)或者(*,主体)的查询 Q,经过计算需要查找的 Hilbert 数值有 2^N 个.设定经过检索,系统中有 $k(k<<2^N)$个节点真正保存有满足查询 Q 的许可证信息,有 $m(m<<2^N)$个节点负责 Q 对应的 Hilbert 数值.因为并不是每一个 Hilbert 数值对应的许可证都存在,所以,$m\geq k$.

(1) 并行查找算法:首先计算出查询 Q 对应的 Hilbert 数值(2^N 个).从节点 N 并行发出这些查询.利用 Chord 的查找机制定位到对应的物理节点,获得应答.在并行查找算法中,发送查询 2^N 个.并行查找算法充分利用了网络中的节点并发执行的能力,查找速度最快.但是,这种方式发送的查询数目过多,对网络通信压力很大.

(2) 串行查找算法:把查询 Q 对应的 Hilbert 数值进行升序排列,形成一个序列 s:($s1,s2,s3,\ldots,s2^N$).查找算法首先定位数值最小的值 $s1$,返回保存它的节点上存放的满足查询 Q 的所有许可证,以及这个节点的标识符 id.从序列 s 中去掉所有比 id 小的 Hilbert 数值,然后再查询序列 s 中剩余的最小 Hilbert 数值,循环执行,直到序列为空.串行查找算法最多只需要发送 m 个查询,对网络的通信压力最小.但是,在这种算法中,后一步查询依赖于前一步查询的返回结果.所以,串行查找算法执行时间最长,需要用户等待很长的时间.

(3) 优化算法:上面提到的两种算法,或者用户等待时间过长,或者使用了大量的带宽,都不适合实际系统使用.在文献[14]中,Schmidt 提出了一种优化算法,但是,这种算法在数据空间很大的情况下,串行查找的层次过多,仍然存在用户等待时间过长的问题.我们利用 Hilbert 曲线生成时的递归特性,对文献[14]中的算法进行了改进.在查找 Q 时,发起查询的节点并不知道有哪些满足查询的许可证保存在 Chord 网络节点中.我们利用生成 Hilbert 曲线的递归特性来并行地查找许可证信息.在图 2 中,空间 A 在一阶空间中的 Hilbert 数字是 10,那么在二阶空间中,空间 A 的每一个 Hilbert 数字的前两位都是 10.同理,对于属于空间 A 的空间 B,在二阶空间中的 Hilbert 数字是 1 000,那么,在后面的优化过程中,属于这个区间的任何一个 Hilbert 数字的前 4 位都是 1 000.

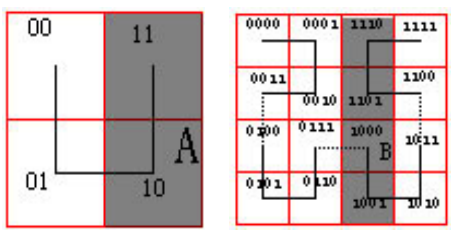


Fig.2 Hilbert space filling curve

图 2 Hilbert 空间填充曲线

性质 **1**(**Hilbert** 曲线递归特性). 在 k 阶空间中,一个区间 A 的二进制 Hilbert 数字是 m,那么,在 k+1 阶空间中,属于空间 A 的所有子空间的二进制 Hilbert 数字的前 $2k$ 位是 m.

利用这个递归特性,我们提出了一种新的优化算法:HBLS(Hilbert-Based level search)查找算法.它的主要目标包括:(1) 使得参与处理的节点数目与存放相关许可证的节点数目基本接近;(2) 尽量减少网络中发送的查询

数目,减轻对网络通信的压力;(3) 减少串行查找的深度,减少用户的等待时间.其主要思想是:在数据空间[0,2^{2N}],从节点 $n1$ 发送查询 Q,按照 Hilbert 曲线生成的迭代过程,首先生成的是 2 位的数字,其次是 4 位,…,直到最后生成的是 $2N$ 位为止.对曲线的第 k 次迭代是对第 k–1 次空间的细化,相应的数字前 $2(k-1)$位保持不变,在第 $2k$ 和 $2k$+1 位根据添加 0 和 1 生成不同的分支.在进行查找时,如果第 k–1 次优化以后的空间内的所有 Hilbert 数字都位于一个节点上,那么,这个空间的优化就结束了.在数字空间很大、Hilbert 数字位数很长的情况下,简单的一次添加两位仍然会串行查找过多,我们采用一次添加 $2k$ 位的方法(k=1,2,3,…)来减少串行查找的层数,提高算法的并行性,从而提高执行效率.

曲线生成过程的优化可以看成是一棵 n 叉树的生成过程.例如,在图 1 中,从节点 A 查找(1001,*),我们选定生成 4 叉树.查找过程如图 3 所示.

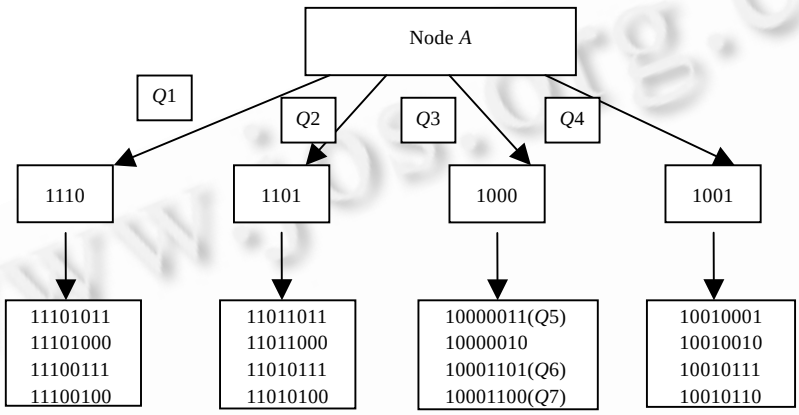


Fig.3 Optimized query tree

图 3 优化查询树

从节点 A 发出查询,首先进行的是 2 阶优化,产生第 1 层 4 个查询.第 1 个查询(查找关键字 1110 0000)和第 2 个查询(1101 0000)都是定位到节点 F.因为节点 F 的前 4 位已经大于这两个查询对应的 Hilbert 数字 1110 和 1101,因此不必再进行优化.第 3 个查询 $Q3$(查找关键字 1000 0000)定位到节点 C,还需要进行优化.在节点 C 上产生更深层次的 Hilbert 数字,然后继续发送查询 $Q5$,$Q6$,$Q7$.查询 $Q4$(查找关键字 1001 0000)定位到节点 D,不再需要进一步的优化.

算法 **1**. HBLS 查找算法.

输入:

key:查找的一维数据关键字;

currenthilbert:要查找的 Hilbert 数字,初始值为 0;

currentstep:到这一步为止 Hilbert 数字优化的阶数;

step:每次的优化添加的阶数.

输出:这个节点找到的许可证.

```
{
01      if (currentstep=0)
02          跳转 07 步
03      if (当前节点 ID 的前 currentstep 位大于 currenthilbert)
04          返回这个节点找到的许可证
05          结束这个方向的查找
06      else
07          返回在这个节点上查找到的许可证
```

08 *currentstep=currentstep+step*
09 生成 *currentstep* 层的新的 Hilbert 数字
10 判断这些 Hilbert 数字中有哪些不在这个节点上
11 针对每一个新的 Hilbert 数字发出查询
}

2.3 存储均衡的设计

将许可证按照发布者和主体二维信息映射到存储空间来实现许可证的分布已经能够很好地避免瓶颈现象的发生.为了获得更好的负载均衡的效果,我们增加了两种优化处理:(1) 在物理节点加入时进行优化.物理节点在加入 Chord 时,首先获得一个位于$[0,2^m]$内的值 k 作为这个节点的候选标识符,然后系统分别计算出$(k+2^m/5)\%2^m$,$(k+2^m/5*2)\%2^m$,$(k+2^m/5*3)\%2^m$,$(k+2^m/5*4)\%2^m$ 作为候选标识符.系统获得这些标识符后继节点的负载情况,把新的节点加入到负载最重的地方.(2) 后台调整算法.我们在后台定期检测系统中负载最重和负载最轻的节点,然后将负载最轻的节点暂时退出,再使用负载最重节点的负载中间值作为新的标识符,重新加入.因为这种方法的使用代价很大,所以不能频繁地调整节点的负载.

3 基于最小许可证图的许可证链查找算法

采用新的存储策略后,我们在查找过程中可以通过发布包含部分关键字的查询来获得所有的满足要求的许可证.这一点是我们提出的具有更高效率的许可证链发现算法的前提.查找许可证链的过程,就是在当前的许可证集合中使用相关的许可证形成许可证图的过程[2,3].对于同一个许可证的集合,从不同方向进行查找得到的许可证图大小是不同的.许可证图的大小表明了需要从网络中获取的许可证数目的多少,将影响到许可证链的查找效率.

我们的系统采用分布哈希表和 Hilbert 曲线对许可证进行合理的映射和定位,从根本上克服了原来许可证分布的缺点.这种许可证分布存放的映射方法,使得相关许可证的查找不受许可证存放位置的限制,可以灵活地进行许可证链查找方向的选择和调整.我们提出了一种新的许可证链查找算法 Samcg(search algorithm basing on minimal certificate-graph).其主要思想是,根据用户所发出的对某一角色的申请,从两个方向(用户、角色)同时进行许可证链的查找,在进行双向查找的过程中,不断试探、比较当前正反两个方向所形成的许可证分支数目的大小,自动选择小的一端作为查找的前进方向,使得查找到的许可证汇合点与图中分支大的一端尽量接近,从而构造一个比较小的许可证图.

基于最小许可证图查找算法(Samcg)的基本原则和内容:

原则 **1**. 从两个方向分别构造两个独立的许可证图.通过循环执行,两个许可证图不断扩展,直到联通或找不到相关许可证时,算法结束.如果两个图连通,说明存在一个许可证链,否则许可证链不存在.

原则 **2**. 确定当前许可证图的构造方向的判断条件是,根据正向和逆向查找算法维护的当前的许可证的分支数目,选择获得许可证数目比较少的方向来进行下一步的扩展.

原则 **3**. 对于连接角色,形如 $Ar1.r2$,在正向查找过程中,不仅要查找全部满足条件$(*,Ar1.r2)$的许可证,还要查找满足$(*,Ar1)$的许可证,并对找到的许可证进行对发布者和主体同时增加相同属性 $r2$ 的扩展;在逆向查找过程中,也要对找到的满足$(Ar1,*)$的许可证进行扩展.

原则 **4**. 对于交集角色,形如$f_1\cap\ldots\cap f_i\cap\ldots\cap f_k$,在正向和逆向的处理过程中要设置一个交集监控器,用来统计某一角色或用户(D)拥有的交集中角色的次数$(COUNT)$.如果 $COUNT$ 能够达到 k,那么 $f_1\cap\ldots\cap f_i\cap\ldots\cap f_k$ 与 D 之间连通.

算法 **2**. 基于最小许可证图的许可证链查找算法(Samcg).

输入:输入用户名和他所要申请的角色.

输出:许可证链的查找结果.

{

```
01  q1:队列,用来存放正向查找算法中当前层需要处理的全部节点
02  q2:队列,用来存放逆向查找算法中当前层需要处理的全部节点
03  将输入的用户名和角色分别加入到正向和逆向工作队列中
04  while q1 与 q2 队列都不为空时  do {
05        if (qfordnum==0)      /* 表示本次操作选择正向查找算法
06        {
07            清空队列 q1
08            依次获得所有主体为队中节点的许可证
09            if   得到的许可证的发布者就是输入的角色
10               {那么给出肯定回答      return}
11            else {   保存得到的许可证发布者到 q1
12                     统计许可证的数目,并保存到变量 qfordnum 中}}
13        if (qrevernum==0)     /* 表示本次操作选择逆向查找算法
14        {
15           清空队列 q2
16           依次获得所有主体为队中节点的许可证
17           if   得到的许可证的主体就是输入的用户
18              {那么给出肯定回答     return}
19           else{     保存得到的许可证主体到 q2
20                     统计许可证的数目,并保存到变量 qreverdnum 中}}
21        if   当前的两个队列 qford,rever 有公共的元素
22           {正向和逆向查找算法构成的两个独立的许可证图已经连通,给出肯定回答,return}
23        else{比较 qfordnum 与 qrevernum 选择数值小的变量所代表的方向作为下一步的执行方向}
24    }      /* 在 q1,q2 至少一个为空的情况下,两个图仍然没有连通,说明不存在相关的许可证链.
25  给出否定回答     return}
}
```

下面,我们对算法性能进行分析.假设在查找过程中每一个发布者平均发出了β个许可证,同时,每一个主体都接受到了β个许可证,那么,一个搜索深度为 d 的单向搜索算法需要查找的许可证的数目是:

$$\beta+\beta^2+...+\beta^d=\frac{\beta^{d+1}-1}{\beta-1}=o(\beta^d) \tag{1}$$

采用 Samcg 双向搜索算法需要查找的许可证数目是:

$$\beta+\beta^2+...+\beta^k+\beta+\beta^2+...+\beta^n=\frac{\beta^{k+1}-1}{\beta-1}+\frac{\beta^{n+1}-1}{\beta-1}=o(\max(\beta^k,\beta^n) \tag{2}$$

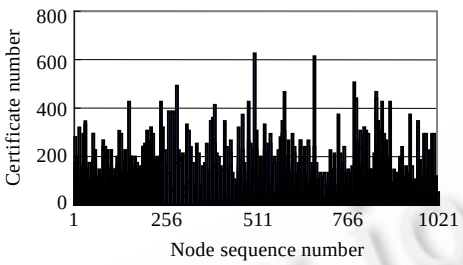
其中 $k+n=d$.所以,Samcg 算法性能最坏情况下是 $o(\beta^d)$,最优是 $o(\beta^{d/2})$,亦即 Samcg 算法可以构造最小的许可证图.

4 实验结果和分析

我们使用由标准 C 语言编写的模拟器在一台 PC 机上模拟实现了一个由 Chord 协议管理的包含若干计算机节点的分布式系统.PC 机的配置为:CPU P4 1.6GHz,内存 1GB,操作系统是 Windows XP.模拟器实现了利用 Hilbert 曲线的许可证从二维空间到一维空间的映射、许可证的均衡分布、基于最小许可证图的许可证链的查找算法.分别通过几组实验详细分析了包括:许可证的负载均衡、基于部分关键字的许可证优化查找算法、不同策略构造的许可证图大小这几个方面的执行情况.

4.1 存储均衡的验证

实验中模拟了 1 024 台物理节点,插入 10 万条许可证信息.图 4 是我们把许可证按照一维信息(发布者)进行定位的分布结果.可以看出,许可证在节点上的分布很不均衡,有些节点过于集中.图 5 是我们把相同的许可证集合按照发布者和主体二维信息映射到 Chord 组织的分布环境中,并经过负载的调整优化以后的结果.在进行负载优化时,设定的存储差异系数(系统中节点的最大负载和最小负载的比值)是 4.两次结果对比表明,我们提出的分布策略完全避免了按照一维信息进行许可证分布存放的策略中会出现的个别节点上的瓶颈现象,并且可以很好地实现许可证的均衡分布.

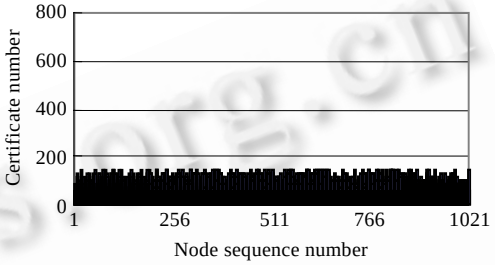


(a) One dimensional distribution

(a) 一维分布

Fig.4 Distribution of certificates with one-dimensional information

图 4 采用一维信息定位的许可证分布图



(b) Two dimensional distribution

(b) 二维分布

Fig.5 Distribution of certificates with two-dimensional information

图 5 采用二维信息定位的许可证分布图

4.2 HBLS算法的验证

在模拟实验中,网络中机器节点数为 1 024,包含 10 万个许可证,HBLS 每次进阶的参数是 step.我们随机选择 500 个关键字进行查询,实验结果取平均值.我们首先通过实验确定合适的 step 值.实验结果如图 6 所示.从图中可以看出,step 越大,发送的查询数目就会越多,同时查询深度就会越小.在 step 取 2 和 3 时可以获得比较满意的实验结果.在 *step*=2 时,发送的查询数目比 *step*=3 时少 18.774,查询深度高 1.13.在查询深度要求不是很苛刻的情况下,为了减少查询的数目,我们选择 *step*=2.

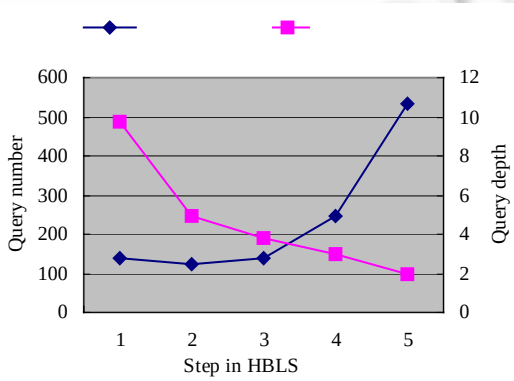


Fig.6 Comparison of different steps

图 6 不同 step 参数的比较

表 1 显示了 HBLS 算法与并行查找算法、串行查找算法和文献[14]中的优化算法的性能比较.

从实验结果中可以看出,并行查找算法需要发送的查询数目是 1 024,串行查找算法的查找深度是 39.333 33.这两个参数决定了这两种算法在分布式网络中不适用.无论是文献[14]中的优化算法,还是 HBLS 算法,都在少量增加参与处理的节点的代价下,极大地降低了发送的查询数目和最大串行查找深度.与文献[14]中

的优化算法相比,HBLS 算法的最大优点是最大串行查找深度减少了 49.4%,明显地降低了用户的等待时间.而且,参与处理的节点数目减少了 10.52%,发送的查询数目减少了 11.46%.因此,综合参与处理的节点、发送的查询数目、最大串行查找深度 3 个参数来考虑,HBLS 算法在 4 种算法中性能最佳.

Table 1 Performance comparison of different algorithms

表 1 不同算法的性能比较

	Nodes saving relevant certificate	Nodes involving in processing	Number of inquiries	Level of search
Parallel algorithm	34.116 67	39.333 33	1 024	1
Serial algorithm	34.116 67	39.333 33	39.333 33	39.333 33
Optimized algorithm in Ref.[14]	34.116 67	58.436 67	136.473 3	9.78
HBLS algorithm	34.116 67	52.29	120.826 7	4.95

4.3 几种查询方法创建的许可证图大小的比较

随机产生 500 个用户,并由 500 个用户随机地作为许可证的发布者和主体,从而模拟产生 10 000 个许可证.对于全部回答为肯定的查询集合,我们统计了下面 3 种情况下创建的许可证图的大小:许可证全部与发布者一同存放,采用逆向许可证链查找算法(backward search algorithm)[1]、许可证全部与主体一同存放,采用正向许可证链查找算法(forward search algorithm)[1]、使用新的许可证存放策略之后,采用我们提出的基于最小许可证图的查找算法(Samcg).结果如图 7 所示,表示随着网络中许可证数目的增加,利用 3 种查找算法分别需要构造的许可证图的大小的对比关系.图中横坐标表示网络中许可证的数目,纵坐标表示需要创建的全部许可证图的大小,即许可证图中包含的许可证的数目.

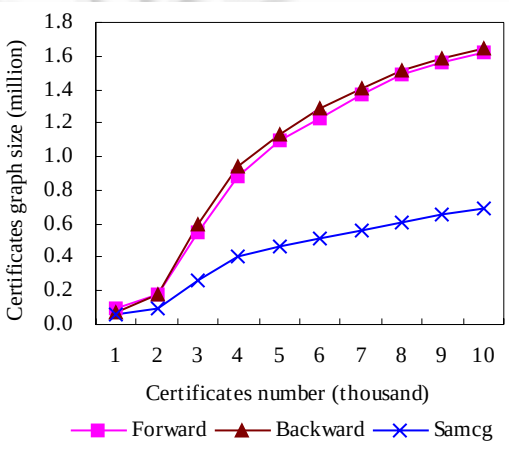


Fig.7 Comparison of different certificate chain search algorithms

图 7 不同的许可证链查找算法的比较

结果图表明,对于随机生成的相同许可证集合和查询集合,采用正向查找算法和逆向查找算法需要构造的许可证图的大小基本相同,但是利用基于最小许可证图算法需要构造的许可证图远远小于上面两种情况,仅仅是它们大小的 40%左右.可见,我们提出的基于最小许可证图的许可证链查找算法极大地减少了在分布环境下需要在网络中传输的信息量,提高了系统的性能.

5 结论和未来工作

利用信任管理技术 SPKI/SDSI,RT 来实现分布环境下的存取控制是一种非常有效的方法.但是,许可证的存储策略一直是这个领域中的一个尚未解决的重要问题,而且直接影响到许可证的查找和回收等问题.本文提出了一种新的存储策略,采用 Chord 协议管理分布式网络,利用分布哈希表和 Hilbert 曲线,基于许可证的发布者和主体两维信息进行能够保持空间相对位置关系的许可证的网络定位.同时,利用 Hilbert 曲线生成时的递归特性及其所具有的局部保持性,实现了在分布式环境中基于部分关键字的许可证查找.并在此基础上提出了基于最小许可证图的许可证链查找算法,使得在查询过程中需要的许可证信息最少,极大地减少了查找过程中的网络信息传输量.如何利用这个存储策略的优势来解决分布环境下许可证的回收问题,是我们下一步的研究方向.

References:

[1] Li NH, Winsborough WH, Mitchell JC. Distributed credential chain discovery in trust management. Journal of Computer Security, 2003,11(1):35−86.

[2] Li NH, Mitchell JC. RT: A role-based trust-management framework. In: Proc. of the 3rd DARPA Information Survivability Conf. and Exposition. Washington, IEEE Computer Society Press, 2003. 201−212. http://theory.stanford.edu/people/jcm/papers/rt_discex03.pdf

[3] Clarke D, Elien JE, Ellison C, Fredette M, Morcos A, Rivest R. Certificate chain discovery in SPKI/SDSI. Journal of Computer Security, 2001,9(4):285−322.

[4] Aura T. Fast access control decisions from delegation certificate databases. In: Proc. of the 3rd Australasian Conf. on Information Security and Privacy (ACISP '98). Brisbane: Springer-Verlag, 1998. 284−295. http://research.microsoft.com/users/tuomaura/Publications/aura-acisp98.pdf

[5] Ellison C, Frantz B, Lampson B, Rivest R, Thomas B, Ylonen T. SPKI certificate theory. Internet RFC 2693, 1999.

[6] Jim T. SD3: A trust management system with certificate evaluation. In: Proc. of the 2001 IEEE Symp. on Security and Privacy. Oakland: IEEE Computer Society Press, 2001. 106−115.

[7] Li NH, Mitchell UC, Winsborough WH. Design of a role-based trust-management framework. In: Proc. of the 2002 IEEE Symp. on Security and Privacy. Los Alamitos: IEEE Computer Society Press, 2002. 114−130. http://www.cs.purdue.edu/homes/ninghui/papers/rt_oakland02.pdf

[8] Ajmani S, Clarke DE, Moh CH, Richman S. ConChord: Cooperative SDSI certificate storage and name resolution. In: Proc. of the 1st Int'l Workshop on Peer-to-Peer Systems (IPTPS 2002). Cambridge: Springer-Verlag, 2002. 141−154. http://pmg.csail.mit.edu/~ajmani/papers/lncs2429.pdf

[9] Ellison C. SPKI Requirements. Internet RFC 2692, 1999.

[10] Hildrum K, Kubatowicz JD, Rao S, Zhao BY. Distributed object location in a dynamic network. In: Proc. of the 14th ACM Symp. on Parallel Algorithms and Architectures. Winnipeg: ACM Press, 2002. 41−52. http://oceanstore.cs.berkeley.edu/publications/papers/pdf/SPAA02.pdf

[11] Asano T, Ranjan D, Roos T, Wiezl E, Widmayer P. Space filling curves and their use in the design of geometric data structures. Theoretical Computer Science, 1996,18(1):3−15.

[12] Nikander P, Viljanen L. Storing and retrieving Internet certificates. In: The 3rd Nordic Workshop on Secure IT Systems. Trondheim, 1998. 1−13. http://www.tml.tkk.fi/~pnr/publications/Nordsec-98.pdf

[13] Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for Internet applications. ACM SIGCOMM Computer Communication Review, 2001,31(4):149−160.

[14] Schmidt C, Parashar M. A peer-to-peer approach to Web service discovery. World Wide Web, 2004,7(2):211−229.

高迎(1973－),女,辽宁鞍山人,博士生,主要研究领域为数据库,信息安全,信息检索.

程涛远(1977－),男,博士生,主要研究领域为数据库,信息安全,信息检索.

王珊(1944－),女,教授,博士生导师,CCF高级会员,主要研究领域为高性能数据库,数据仓库,知识工程.