基于 Jini 的鉴别服务*)

于国良　韩文报
(信息工程大学信息研究系　郑州 450002)

摘　要　现有 Jini 结构中的鉴别机制，通常是服务提供者对服务请求者进行鉴别，而不能实现服务请求者对服务提供者的鉴别。为了保护系统的安全，本文采用带密钥的 Hash 函数给出了一种鉴别方案，以实现服务提供者和服务请求者的双向认证，并通过 java 2 所提供的 JAAS 鉴别和授权服务的开发框架，给出了具体的实现。

关键词　Jini，JVM，鉴别，代理

Authentication Service Based on Jini

YU Guo-Liang　HAN Wen-Bao
(Department of Information Research, Information Engineering University, Zhengzhou 450002)

Abstract　In current Jini systems, users are authenticated by service providers, however, users can not authenticate service providers. To protect system security, this paper suggests a new authentication scheme using hash function with key, so that authentication in both directions between users and service providers can be completed. Also, a new way is provided to implement Jini authentication service using JAAS frame using Java 2.

Keywords　Jini, JVM, Authentication, Proxy

1　引言

Jini 技术作为 Sun 公司 1999 年初推出的分布式计算体系结构，提供了一套机制使得电子设备和应用程序能够“即插即用”，随时加入或退出 Jini 群体，从而形成一个动态的分布式系统。Jini 系统提供了在分布式系统中进行服务建立、查找、通讯和使用的机制。Jini 结构的设计充分利用了 Java 程序代码可以在机器之间移动的能力，使服务端和客户端之间可以临时搭建一个连接通道，来进行服务的提供和应用。其实质是将 java 应用环境由单独的 JVM 扩展到一个 JVM 网络[1]。

Jini 体系结构主要包括三部分：基础结构、编程模型和服务[2]。这三部分彼此独立，又相互关联。其中基础结构是体系结构的核心部分，目标是为设备、服务和用户提供相应机制用于发现、加入网络或与网络分离。Jini 的基本工作机制如图 1 所示：

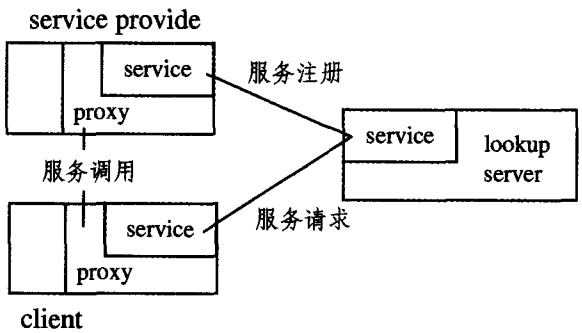

图 1

它包括三个步骤：

1. 服务提供者把服务注册到 Jini 服务群体。
2. 服务请求者请求 Jini 群体中的服务。
3. 服务请求者使用服务提供者所提供的服务。

在任何环境中，安全问题都是非常重要的。对于分布式环境来说，安全问题则更为突出，因为系统的组件无论从物理上还是地理上都是分布的[3]。Jini 体系结构的安全是建立在 java 平台安全性的基础上，利用 JVM 和各种安全模型来实现其安全策略。本文通过 java 2 所提供的 JAAS 鉴别和授权服务(java authentication and authorization service)的开发框架[4]，给出了具体的实现。

2　相关研究

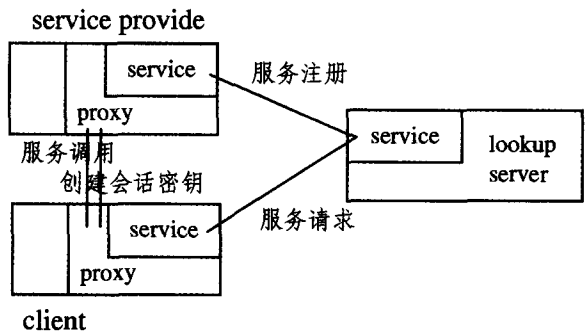

图 2

Jini 技术是一种近年来新出现的体系结构，国内外对 Jini 的研究大部分集中在实用性研究[6,7]、以及利用 Jini 分布式结构来构建应用系统上[7~9]，对于 Jini 结构中的安全问题却很少涉及。文[10]指出了 Jini 面对分布式计算体系结构遇到的安全问题，没有显式的客户和服务器角色的划分、分布式环境中的交互不可预知以及分布式系统容易受到木马程序的攻击等，但对于结合 Jini 本身体系结构的特点，对其安全性的讨论并不多。文[11]中提出了基于 Jini 代理结构的安全模型，在代理之间创建会话密钥，并利用会话密钥建立安全连接，同时

*)基金项目：国家自然科学基金资助项目(No. 19971096, No. 90104035)。于国良　系统分析员，博士研究生，主要研究方向：信息安全、软件工程。韩文报　教授，博士生导师，主要研究方向：信息安全，密码理论，代数编码。

给出了部分实现(如图 2 所示)。

3 基于 Jini 的鉴别方案

在典型的 Jini 分布式体系结构中,服务提供者往往需要对服务请求者进行鉴别以后,才允许合法用户访问。但在 Jini 结构中,作为应用层的服务发现协议,并没有提供对服务请求者进行鉴别的功能。

现有 Jini 结构中的鉴别机制,往往是服务提供者对服务请求者进行鉴别,服务提供者利用预先定义的合法用户列表以及相应用户的密钥来鉴别用户,而且不能实现服务请求者对服务提供者的鉴别。为了保护系统的安全,我们采用带密钥的 Hash 函数给出了一种鉴别方案,以实现服务提供者和服务请求者的双向认证。

设服务提供者 P 和服务请求者 A 的共享密钥为 k,协议中使用的 Hash 函数为 h,使用的对称加密算法为 E,进行鉴别的过程如下:

1. 服务提供 P 把服务标识注册到查询服务器 Q 上。

2. 服务请求者 A 根据服务标识通过查询服务器 Q 查找服务,定位服务提供者 P。

3. 服务请求者 A 根据下载到本地的代理 proxy 与服务提供者 P 建立安全连接。

(1)A 选择一个随机数 n_A,把用户名 ID_A,随机数 n_A,$h(k, ID_A, n_A)$发送给 P。

(2)P 根据 ID_A,随机数 n_A 和共享密钥 k,计算 $h(k, ID_A, n_A)$并与 A 发送过来的值比较实现对用户的鉴别。如果相同,则判断 A 是合法用户,便生成一个随机数 n_P,并把 ID_P, n_P,$h(k, ID_P, n_P)$发送给 A。

(3)A 根据 ID_P,随机数 n_P 和共享密钥 k,计算 $h(k, ID_P, n_P)$并与 P 发送过来的值比较,实现对服务提供者的鉴别。如果相同,则判断 P 是合法提供者,计算 $k_{AP}=h(k, n_A, n_P)$作为会话密钥。并计算 $E[k_{AP}, n_A]$,把其发送给 P。

(4) P 同样计算 $k_{AP}=h(k, n_A, n_P)$作为会话密钥,并计算 $E[k_{AP}, n_A]$,与 A 发送来的值相比较,如果相同则确认 A 已成功计算会话密钥 k_{AP}。P 再计算 $E[k_{AP}, n_P]$将其发送给 A。

(5) A 计算 $E[k_{AP}, n_P]$,并与 P 发送过来的值比较,如果相同则确认 P 已成功计算会话密钥 k_{AP},则双方成功建立了安全连接。

4. 服务请求者 P 和服务提供者 A 利用安全连接使用服务。

4 鉴别的实现

由于 Jini 是基于 java 的一种分布式体系结构,所以它的安全是由 java 平台的安全性决定的。JAAS 就是一种为 java 2 平台提供的基于用户的验证以及访问控制功能的框架,为基于用户、组和角色的访问控制提供了支持。本文的实现就是利用 JAAS 提供一组实现用户鉴别的类,来防止终端用户的破坏。

JAAS 通过提供动态的、可扩展的模型来进行用户验证和控制权限,从而使应用程序有更加健壮的安全机制。同时它还能让用户很轻松地创建自己的登录机制。JAAS 可以同时在客户端和服务器端应用程序上工作。

利用 JAAS 的鉴别应用框架如图 3 所示:

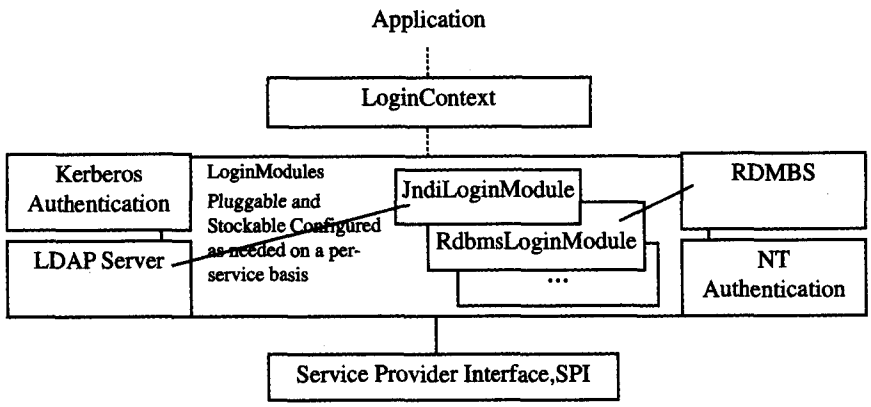

图 3

JAAS 框架通过在应用程序和底层的验证和授权机制之间加入一个抽象层,使抽象层能独立于平台,如上图所示,在底层的验证和授权机制,只是提供了一个接口:服务提供者接口(Service Provider Interface,SPI)来给鉴别模块进行调用。这样,应用程序级的代码只要处理 LoginContext。由 LoginContext 调用下面是一组实现服务提供者接口的动态配置的 LoginModules 模块,由 LoginModule 模块来调用底层的验证和授权机制来进行鉴别即可。

JAAS 提供了一些 LoginModule 的参考实现代码,比如 JndiLoginModule。开发人员也可以自己实现 LoginModule 接口,在这里,我们实现了基于密钥库的鉴别模块来进行鉴别。

当然为了满足可插接性,JAAS 是可堆叠的。一组安全模块可以堆叠在一起,然后被其他的安全机制按照堆叠的顺序被调用。

在我们的实现中使用 JAAS 鉴别涉及到以下几个步骤:

1. 创建一个 LoginContext 的实例。

2. 为了能够获得和处理验证信息,将一个 CallBackHandler 对象作为参数传送给 LoginContext。

3. 通过调用 LoginContext 的 login()方法来进行验证。

4. 通过使用 login()方法返回的 Subject 对象实现一些特殊的功能(假设登录成功)。

程序代码大致如下:

```
LoginContext lc = null;
    try {
        lc = new LoginContext("Sample", new MyCallbackHandler
        ());
    } catch (LoginException le) {
        System.err.println("Cannot create LoginContext. "
            + le.getMessage());
        System.exit(-1);
    } catch (SecurityException se) {
        System.err.println("Cannot create LoginContext. "
            + se.getMessage());
        System.exit(-1);
    }
Subject sub = lc.getSubject();
Subject.doAs(sub, new MyPrivilegedAction());
```

在运行这段代码时，后台进行了以下的工作。

1. 当初始化时，LoginContext 对象首先在 JAAS 配置文件中找到 Sample 文件，然后根据配置文件的内容决定该加载哪个 LoginModule 对象。

2. 在登录时，LoginContext 对象调用每个 LoginModule 对象的 login()方法。

3. 每个 login()方法进行验证操作或获得一个 CallbackHandle 对象。

4. CallbackHandle 对象通过使用一个或多个 CallBack 方法同用户进行交互，获得用户输入。

5. 向一个新的 Subject 对象中填入验证信息。

在鉴别实现中，我们在服务提供者 P 和可下载到本地的代理 proxy 上分别部署一个 Login 模块。在可下载到本地的代理 proxy 上，根据 ID_A 信息，以及随机数 n_A 和共享密钥 k，计算 $h(k, ID_A, n_A)$并于服务提供者 P 发送过来的值比较，实现对服务提供者 P 的鉴别。如果相同，则判断服务提供者 P 是合法用户，同时生成一个随机数 n_P，并把 ID_P，n_P，$h(k, ID_P, n_P)$发送给服务提供者 P。代码可以表示为：

if AuthCode= $h(k, ID_A, n_A)$ {……}

在服务提供者 P 的 Login 模块中，根据 ID_p，随机数 n_P 和共享密钥 k，计算 $h(k, ID_P, n_P)$并于代理 proxy 发送过来的值比较，实现对代理 proxy 的鉴别。如果相同，则判断代理 proxy 是合法代理 proxy。代码可以表示为：

if AuthCode=$h(k, ID_P, n_P)$ {……}

5 安全性分析

本文提出了一个 Jini 系统中服务提供者和服务请求者的鉴别协议，并通过双方交互建立安全连接，具有以下特点：

(1) 服务提供者和服务请求者共享一对称密钥，使用带密钥的 Hash 函数实现了双方的双向认证。

(2) 每次认证时，双方都选择不同的随机数，可以防止攻击者的重发攻击。

(下转第 75 页)

(上接第 55 页)

等人还提出一种称为动态集相关的模型，这是一种基于新型的统计独立的并行散列模型。该技术可以降低散列冲突不命中率达到 75%～90%，使用惰性回写技术，吞吐量可以进一步达到 37.5/75Mpps。

4.1 散列算法

散列(Hash)算法的基本思想是以关键字的值为自变量，通过一定的函数关系，计算出对应的函数值来，把这个值解释为节点的存储地址，将节点存入到这个存储单元里去。查找时再根据要查找的关键字用同样的散列函数计算地址，然后到相应的地址单元里去取要找的节点。所以这种方法也被称为关键字-地址转换法。用散列法存储的线性表叫做散列表。在散列表里可以实现对节点进行快速的查找。

散列算法由两个部分构成：散列计算和散列查找，散列计算的计算过程应该做到两个方面：一是计算简单，二是计算应该尽量减小散列冲突。散列计算比较常用的算法是平方法、折叠法、除留余数法等，在本算法中，采用了折叠法加除留余数法。即首先把流的地址字段分段，然后进行顺叠相加，最后把相加的结果和散列表的大小进行模运算，得到散列地址。这样处理计算简单，得到散列地址也有较好的随机性。

散列函数的选取可以减少冲突但是不能避免冲突，因此如何解决冲突就是散列法中一个必须要解决的问题。处理散列冲突的方法基本上有两类，一类叫做拉链法，另一类叫做开地址法。在本算法中，采用了拉链法进行解决散列冲突。

4.2 定时清除

随着时间的推移，冲突次数的增加，存储空间如果不做处理的话将会被耗尽。另一方面，某些流已经退出使用，比如某个 IP 地址已经不在进行网络使用，则在内存中保存的记录也应该被清除，把数据记录保存到外存中，比如数据库中。释放的存储空间可以为后面到达的新流继续使用。在数据包分类算法中，采用了定时清除的方法，每个数据包到达之后用该数据包的时间戳更新所属流的时间标记，以反映该流的活跃时间，然后定时清除已经不再活跃的流。定时清除部分完成定时对散列表中的记录进行扫描，清除已经不再活跃的流，释放存储空间。

分流部分的流程如下：

(1)根据源 IP 地址、目的 IP 地址进行散列计算，计算出关键字 K，散列函数可选取折叠法加除留余数法；

(2)根据散列计算的结果关键字 K 的值，到散列表中进行散列查找。如果找到则得到分流的结果。如果找不到则转(3)；

(3)如果找不到则说明这是一个新流，则为这个流建立一个新的散列表项，插入到关键字 K 对应的同义词子表，作为第一个节点。

5 算法复杂度分析

算法的执行时间由散列计算的时间(记为 T 计算)和散列查找的时间(记为 T 查找)组成，执行时间为 T 计算＋ T 查找 。通常情况下，计算的时间要远小于查找的时间，所以算法的时间约等于 T 查找，而散列查找的时间 TS 要取决于冲突的次数。当查找的关键字是 N，散列表的基本区的大小是 M 的时候，散列查找的平均次数是 $N/2M$，算法的时间复杂度是 $O(N/M)$。同时，算法中使用的流的局部性原理可以加速查找过程，最好的情况下一次查找就能得到结果。

算法的占用存储空间主要是散列表的存储空间 UHash。而散列表主要用于对系统中的流进行存储。假设系统中流的数目是 M，且每个流的记录要占用 N 个字节，则 UHash 要占用的存储空间是 $M \times N$，算法所占用的存储空间大约是 $O(N)$。

结束语 程序的运行结果根据不同的环境，结果有所不同。影响结果的因素有，硬件环境(即仿真计算机的配置)、所确定的散列表的容量、处理的数据包数、处理数据包的相关性、包在散列表中的活跃时间、定时扫描的时间等。算法分析表明算法具有良好的时间复杂度和空间复杂度，算法可以实现快速的分流。

参 考 文 献

1 付歌，杨明福.一个快速的二维数据包分类算法. 计算机工程. 2004,30(6):76～78

2 Telikepalli A. 数据包处理方法和解决方案[J]. 今日电子，2002,7:21～25

3 小高知宏. TCP/IP 数据包分析程序篇[M]. 北京:科学出版社，2003

4 Comer D E,等著.用 TCP/IP 进行网际互联(第二卷:设计、实现与内核)[M]. 张娟,等译. 北京:电子工业出版社，2003

5 徐士良. 计算机常用算法. 北京:清华大学出版社，1995

$S(x^{-1})=S(x^{3^3})=C_3(x)+C_4(x)+C_5(x)$

将 β 带入上式，如果 $p\equiv 7 \bmod 8$，则根据引理 4，5 和 7，$S(\beta)\in\{0,1\}$，因而 $S(\beta^{-1})=1+S(\beta)$，$S(\beta)S(\beta^{-1})=0$；如果 $p\equiv 3 \bmod 8$ 由引理 3 和引理 7，$S(\beta)^2=S(\beta^2)=S(\beta^{-1})=1+S(\beta)\neq S(\beta)$，显然 $S(\beta)\notin\{0,1\}$，因此 $S(\beta)S(\beta^{-1})\notin\{0,1\}$。

3 新的六次剩余序列的线性复杂度

定理 1 令 $p=6f+1\equiv 7 \bmod 8$，则存在一个本原 p th 单位根 β 使得 $S(\beta)=1$，且仅当 j 属于 C_1,C_2,C_3,C_4,C_5 中的三个剩余类时，$S(\beta^j)=0$。

证明：令 γ 一个本原 p 次单位根，则

$$\begin{aligned}\sum_{i=1}^{p-1}S(\gamma^i)&=\sum_{i=1}^{p-1}C_0(\gamma^i)+\sum_{i=1}^{p-1}C_1(\gamma^i)+\sum_{i=1}^{p-1}C_2(\gamma^i)\\&=\sum_{j=1}^{5}C_0(\gamma^{3^j})+\sum_{j=0}^{5}C_1(\gamma^{3^j})+\sum_{j=0}^{5}C_2(\gamma^{3^j})\\&=\sum_{j=1}^{5}C_0(\gamma^{3^j})+\sum_{j=0}^{5}C_1(\gamma^{3^j})+\sum_{j=0}^{5}(\gamma^{3^{j+1}})\\&=\sum_{j=0}^{5}C_0(\gamma^{3^j})=\sum_{k=0}^{p-2}\gamma^{g^k}=1\end{aligned}$$

从而，至少存在一个 i 使得 $S(\gamma^i)=1$。则 $\beta=\gamma^i$ 即为所求。

现证明仅当 j 属于 C_1,C_2,C_3,C_4,C_5 中的三个剩余类时，$S(\beta^j)=0$。根据引理 3，只需证明当 i 取 1，2，3，4，5 中的三个数时，$S(\beta^{3^i})=0$。由于

$S(\beta)=C_0(\beta)+C_1(\beta)+C_2(\beta)=1$

因此

$S(\beta^{-1})=S(\beta^{3^3})=1+S(\beta)=0$

由引理 8，有

$S(\beta^3)=C_1(\beta)+C_2(\beta)+C_3(\beta)$，

$S(\beta^{-3})=S(\beta^{3^4})=C_0(\beta)+C_4(\beta)+C_5(\beta)=1+S(\beta^3)$，

$S(\beta^{3^2})=C_2(\beta)+C_3(\beta)+C_4(\beta)$，

$S(\beta^{-3^2})=S(\beta^{3^5})=C_0(\beta)+C_1(\beta)+C_5(\beta)=1+S(\beta^{3^2})$

从而 $S(\beta^3)S(\beta^{3^4})=0$，$S(\beta^{3^2})S(\beta^{3^5})=0$。不失一般性，可假设 $S(\beta^3)=0$，$S(\beta^{3^2})=0$，由证明过程和引理 5 可知，仅当 $j\in C_1\cup C_2\cup C$ 时，有 $S(\beta^j)=0$。

定理 2 周期为 $p=4u^2+27$ 的新六次剩余序列具有如下互反特征多项式：

$$c^*(x)=\begin{cases}(x-1)\prod\limits_{i\in C_0\cup C_4\cup C_5}(x-\beta^i) & p\equiv 7 \bmod 8\\ x^p-1 & p\equiv 3 \bmod 8\end{cases};$$

其中 β 是本原 p 次单位根且满足 $S(\beta)=1$。线性复杂度 L 为

$$L=\begin{cases}1+(p-1)/18 & p\equiv 7 \bmod 8\\ p & p\equiv 3 \bmod 8\end{cases}$$

证明：如果 $p\equiv 7\bmod 8$，由定理 1，当 $a\in C_0\cup C_4\cup C_5$ 时，$S(\beta^a)=1$；当 $b\in C_1\cup C_2\cup C_3$ 时，$S(\beta^b)=0$。且 $S(1)=[(p-1)/2\bmod 2]=1$。则 $c^*(x)$ 为

$$c^*(x)=\frac{x^p-1}{\gcd(x^p-1,S(x))}=(x-1)\prod_{i\in C_0\cup C_4\cup C_5}(x-\beta^i)$$

由于 $2C_0=C_0$，所以 $c^*(x)$ 定义在 GF(2) 上，线性复杂度 $L=1+(p-1)/18$。

如果 $p\equiv 3\bmod 8$，根据引理 8，$S(\beta^j)S((\beta^j)^{-1})\notin\{0,1\}$，$j=1,2,\cdots,p-1$。即 $S(\beta^j)\neq 0$，$j=1,2,\cdots,p-1$。且因为 $p\equiv 3\bmod 8$，$S(1)=(p-1)/2\bmod 2=1$。从而，$\gcd(x^p-1,S(x))=1$，因而 $c^*(x)=x^p-1$，$L=p$。

结论分析 本文构造了一类新的六次剩余序列，同时给出了该序列的线性复杂度和特征多项式。主要定理表明这类序列的线性复杂度在满足条件 $p\equiv 3 \bmod 8$ 时取得最大值 p，作为密钥流序列，它是抗已知明文攻击的；而当 $p\equiv 7 \bmod 8$ 时，线性复杂度为 $(p+1)/2>p/2$。这时，这类序列作为密钥流也是抗已知明文攻击的。

参 考 文 献

1 Blum L, Blum M, Shub M. A simple unpredictable pseudo-random number generator [J]. SIAM J. Comput. 1986, 15: 364～383

2 Golomb S W. Shift Register Sequences [M]. San Francisco, CA: Holden-Day, 1967. Revised edition: Laguna Hills, CA: Aegean Park, 1982

3 Lidl R, Neiderreiter H. Finite fields [M]. In: Encyclop. Math. Its Applic. Reading, MA: Addison-Wesley, 1983, 20

4 Kim J-H Song H-Y. On the Linear Complexity of Hall's Sextic Residue Sequences [J]. IEEE Trans. Inform. Theory, 2001, 47: 2094～2096

5 Hall Jr M. A survey of difference sets [J]. Proc. Amer. Math. Soc., 1956, 7: 975～986

6 Storer T. Cyclotomy and Difference Set [M]. Markham, Chicago, 1967

（上接第 63 页）

（3）协议在双方认证身份后，通过双方选择的随机数和共享密钥计算会话密钥，随机数的使用保证了会话密钥的新鲜性，以便每次建立不同的安全连接。会话密钥的协商时，通过双方交互的返回值，彼此都可以确认对方已成功计算出会话密钥，因此本文的方案是一个认证的密钥协商方案，可以防止中间人攻击。

结束语 通常的鉴别系统中，不只要求服务提供者对服务请求者进行鉴别，还要求实现服务请求者对服务提供者的鉴别，而现有 Jini 结构中的鉴别机制，通常只是服务提供者对服务请求者进行鉴别，并不能实现服务请求者对服务提供者的鉴别。本文通过采用带密钥的 Hash 函数给出了一种鉴别方案，可以实现服务提供者和服务请求者的双向认证，并通过 java 2 所提供的 JAAS 鉴别和授权服务的开发框架，给出了具体的实现。在实际工作中，我们进行了大量的测试，结果表明，我们的方案具有很强的实用性。

参 考 文 献

1 Edwards W K. Core Jini, 2nd edition. Prentice Hall, 2001

2 Sun Microsystems, Inc. Jini Architecture Specification, Version1. 1. http:// www. sun. com/software/Jini/specs/index. html October 2000

3 Sun Microsystems, Inc. Jini Technology Core Platform Specification, Version1. 1. http:// www. sun. com/software/Jini/specs/index. html October 2000

4 Sun Service. Microsystem Java Authentication and Authorization. htp://java. sun. com/products/jaas/

5 Sun Microsystems, Inc. Jini Device Architecture Specification, Version1. 1. http://wwws cup. rpm/cnftware/iini/specs/index. html. October 2000

6 李廷元. Jini 技术实用化研究：[电子科技大学硕士论文]. 2003. 2

7 张晓丽. 基于 JINI 服务的分布式计算体系及其资源发现应用探讨：[中国海洋大学硕士论文]. 2004. 7

8 于铉. 基于 Jini 体系结构的分布式系统研究：[武汉理工大学硕士论文]. 2003. 5

9 高荣. 基于 Jini 的分布式入侵检测系统-体系结构的设计与部分实现：[中国科学技术大学硕士论文]，2004. 5

10 张玉苗、屈鸿. Jini 安全机制研究，福建电脑，2004. 12

11 Andersson F. secure jini in ad hoc networks. Master of Science Thesis Institute of Technology (KTH), 2000

12 KeithEdwards W 著. Jini 核心技术[M]. 王召福，等译. 北京：机械工业出版社，2000. 7

13 Keith Edwards TomRodden W 著. Jini 实例精解[M]. 袁勤勇，等译. 北京：清华大学出版社，2002. 5