单位代码：10293 密　级：

南京邮电大学

硕 士 学 位 论 文

论文题目：无线传感器网络广播认证协议及入侵检测研究

学　　号	1011041031
姓　　名	戴庭
导　　师	黄海平
学 科 专 业	计算机软件与理论
研 究 方 向	基于网络的计算机软件应用技术
申请学位类别	工学硕士
论文提交日期	二零一四年四月

Research on Broadcast Authentication Protocols and Intrusion Detection in Wireless Sensor Networks

Thesis Submitted to Nanjing University of Posts and Telecommunications for the Degree of

Master of Engineering

By

Dai Ting

Supervisor: Assoc. Prof. Huang Haiping

April 2014

南京邮电大学学位论文原创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京邮电大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

本人学位论文及涉及相关资料若有不实，愿意承担一切相关的法律责任。

研究生签名：_____ 日期：_____

南京邮电大学学位论文使用授权声明

本人授权南京邮电大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档；允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索；可以采用影印、缩印或扫描等复制手段保存、汇编本学位论文。本文电子文档的内容和纸质论文的内容相一致。论文的公布（包括刊登）授权南京邮电大学研究生院办理。

涉密学位论文在解密后适用本授权书。

研究生签名：_____ 导师签名：_____ 日期：_____

摘要

广播是无线传感器网络的一项重要操作。它可用于代码的更新、查询以及组员的管理[1]。广播认证是无线传感器网络的一项基本安全服务。常见的基于对称密钥的广播认证协议，特别是 μTESLA 系列，扩展性差（因为 Hash 链长度有限、基站通信范围有限）并且存在安全隐患（因为明文认证）等。

本文首先提出了一个可扩展的广播认证机制，名曰 DH-μTESLA。它是 μTESLA[2]协议和多级 μTESLA 协议[3]的扩展和改进版本。DH-μTESLA 协议包含以下三部分：

1) 提出了基于(t, n)门限和划分树的可再生 Hash 链的构造方案 SRHC-TD，解决了 Hash 链长度受限的问题，并且在相对较低的开销下保证了一定强度的容丢包性/容错包性，最主要的是它能够抵抗选择明文攻击；

2) 提出了基于 d-left 可计数型布隆过滤器的身份认证方案 AdlCBF，在实现了安全认证的同时，确保了存储空间的节约性、系统的可扩展性、查询的高效性；

3) 提出了基于父亲树的分簇路由算法 PTCR，它的主要功能是扩大了网络的范围，使得整个 DH-μTESLA 协议更适用于大范围的稠密的无线传感器网络。

并且，本文利用 Matlab，Visual Studio C++等工具实现了性能的仿真分析，证实了 DH-μTESLA 协议所具备的以上优点。

在无线传感器网络中运行 DH-μTESLA 协议时，网络常常需要采用主动的防御措施。因此，在本文的后半部分，我们将研究传感器网络中的入侵检测问题。

入侵检测是传感器网络的一项重要应用。大量文献表明采用覆盖方法，特别是栅栏覆盖方法，能够增加入侵检测的效率。由于本文中所提出的 PTCR 的簇头轮转机制，新节点的加入以及该算法在实现 800 轮之后的特有的网络拓扑结构的变化，同一局部区域内，节点数目的变化可近似地看做是节点的移动。而且，使用移动的传感器节点能够增强覆盖效果，以避免静态无线传感器网络中存在的覆盖空洞。

然而，传感节点和入侵者的运动轨迹并没有被相关文献详细地研究。同时，移动传感节点之间的相互影响以及传感节点与入侵者之间的影响也没有被详细地研究。为了解决这些问题，本文首先引入虚拟势场的概念。它不但存在于节点之间，而且存在于节点与入侵者之间。考虑到动态相似性，本文使用完全弹性碰撞模型来描述节点的移动；使用点电荷模型来描述入侵者的移动。通过建模，我们能够得到覆盖性能与传感节点、入侵者的移动性之间的关系。最后，我们能够得到虚拟势场下移动传感网的入侵检测的性能要优于静态

无线传感器网络以及一般的移动传感网中的入侵检测的性能。

关键词： 无线传感器网络；*μ*TESLA；可再生 Hash 链；*d*-left 可计数布隆过滤器；(*t*, *n*) 门限；分簇路由算法；入侵检测；栅栏覆盖

Abstract

Broadcast of information in Wireless Sensor Networks (WSN) is an important operation, for example, for code updates, queries and membership information [1]. Broadcast authentication in WSN is a fundamental security primitive. Although symmetric key based μTESLA has been proposed, it suffers from the weakness of low scalability which is resulted from hash chain's finite length, authentication in plaintext and base station's finite communication radius.

This paper presents a scalable broadcast authentication scheme named DH-μTESLA which is the extension and improvement of μTESLA [2] and multilevel μTESLA [3]. It mainly has following three parts:

1) It has proposed the (t, n)-threshold and division tree based self-re-initializable Hash chain scheme (SRHC-TD scheme), which can maintain the infinite lifecycle of Hash chains, low overheads, strong tolerance of message loss or fault, and the ability resist chosen plaintext attack.

2) It has put forward the d-left counting Bloom filter based authentication scheme (AdlCBF scheme), which can make sure the security authentication with memory efficiency, scale expansion and query convenience.

3) It has present the parent-tree based clustering routing algorithm (PTCR algorithm), which can expand the scalability of sensor networks, and make DH-μTESLA be suitable to large scale sensor networks with high density of sensors.

It also presents the experimental results obtained through simulations, which have demonstrated the advantages of the proposed protocol.

In the process of running DH-μTESLA in WSN, the network always needs to execute the proactive defense. Thus, in the latter half of this paper, we also study the intrusion detection in WSNs.

Intrusion detection is a significant application in sensor networks. Considerable literatures indicate that adopting the coverage methods, especially the barrier coverage method can increase the efficiency of detecting intruders. Because of the attributes of the proposed PTCR, (*i.e.* the rotation of cluster heads and the participations of new nodes, and the uneven distribution of nodes after running several rounds, especially 800 rounds), the number and density of nodes in the same local region is changing all the time. This enlightens us to consider the nodes as the mobiles ones and the stationary WSNs as periodly Mobile ones.

It is believed that using mobile sensors can enhance coverage and avoid coverage hole in stationary Wireless Sensor Networks (WSNs). However, the moving trajectories of sensors and intruders have not been investigated properly. Besides, the impacts between mobile sensors and the impacts between a sensor and an intruder have not been discussed thoroughly. In order to address these problems, we first discuss the virtual potential field between sensors and intruders. Using motion similarity, we then formulate the sensor mobility by elastic collision model, and the intruder mobility by point charge model. Afterwards, we obtain the relationship between barrier coverage performance and sensor and intruder mobility. Finally, we show that the coverage performance of Mobile Sensor Networks (MSNs) in virtual potential field can be improved compared to those in stationary WSNs and in general MSNs.

Key words: WSNs; μTESLA; Self-Re-Initializable Hash Chain; *d*-left Counting Bloom Filter, (*t*, *n*)-Threshold, Clustering Routing Algorithm; Intrusion Detection; Barrier Coverage

目录

专用术语注释表

缩略词说明：

WSNs	Wireless Sensor Networks	无线传感器网络
MSNs	Mobile Sensor Networks	移动传感网
TESLA	Timed Efficient Stream Loss-tolerant Authentication protocol	TESLA 广播认证协议
μTESLA	Micro Timed Efficient Stream Loss-tolerant Authentication protocol	μTESLA 广播认证协议
DH-μTESLA	An Improved Broadcast Authentication Protocol for Wireless Sensor Networks	本文多提出的改进的广播认证协议
RHC	Re-initializable Hash Chain	可再生 Hash 链
ERHC	Elegant Re-initializable Hash Chain	精巧构造的可再生 Hash 链
SUHC	Self-Updating Hash Chain	自更新 Hash 链
SRHC	Self-Renewal Hash Chain	理想自更新 Hash 链
NSUHC	New Self-Updating Hash Chain	新的自更新 Hash 链
SUHC-EC	Self-Updating Hash Chain base Erasure Coding	基于纠羽码的自更新 Hash 链
SRHC-FEI	Self-Renewal Hash Chain base Fair Exchange Idea	基于公平交易的自更新 Hash 链
OTS	One-Time Signature	一次性密钥
SRHC-TD	Self-Renewal Hash Chain based on (t, n) Threshold and Division tree	基于(t, n)门限和划分树的可再生 Hash 链的构造方案
BF	Bloom Filter	布隆过滤器
CBF	Counting Bloom Filter	可计数的布隆过滤器
BAS	Bloom filter base Authentication Scheme	基于认证机制的布隆过滤器
BMAP	Bloom filter-based Message Authentication Protocol	基于布隆过滤器的消息认证协议
MAC	Message Authentication Code	消息认证码
dlCBF	d-left Counting Bloom Filter	d-left 计数型布隆过滤器
AdlCBF	Authentication Scheme base d-Left Counting Bloom Filter	基于 d-left 可计数型布隆过滤器的身份认证方案
LEACH	Low-Energy Adaptive Clustering Hierarchy	低能耗的适应性的簇状层级协议
SEP	Stable Election Protocol for clustered heterogeneous wireless sensor networks	层簇同构网络稳定选择协议

DEEC	Distributed Energy-Efficient Clustering algorithm for heterogeneous wireless sensor networks	分布式节能层簇算法
TDMA	Time Division Multiple Access schedule	时分多址调度
PTCR	Parent-Tree based Clustering Routing Algorithm	基于父亲树的分簇路由算法
PKC	Public Key Cryptography	公钥密码体制
DoS	Denial of Service	拒绝服务攻击
ECDSA	Elliptic Curve Digital Signature Algorithm	椭圆曲线数字签名算法
CRT	Chinese Remainder Theory	中国剩余定理

第一章 绪论

1.1 广播认证协议的研究背景及内容

1.1.1 广播认证协议的研究背景及意义

在实用的、安全的网络信息系统中，广播认证是一项重要的服务。在无线传感器网络中，基站经常需要发送命令数据包或者应用更新数据包给其余的传感节点，这些动作或过程都是通过广播来完成的。而广播的安全性会影响整个网络，这就是传感节点需要认证接收数据包的来源、有效性、完整性的原因。

在无线传感器网络中，提出或者设计广播认证协议必须最大程度上满足以下三个条件：1）恶意的接收者很难从发送者那里伪造出任何的数据包；2）必须尽可能地减少通信、计算、存储开销；3）必须在一定程度上能够容忍丢包性和容错性。

基于这三个原则，很多应用在无线传感器网络中的广播认证协议络绎不绝地被提出来。

在 2001 年，一个最典型的广播认证协议，μTESLA[2]被提出来。它是 TESLA 协议的改进方案。而 TESLA 协议是一个工作在一般桌面工作站级别的认证流广播协议[4]。μTESLA 利用延迟发布对称密钥的方法来实现特殊的非对称密钥机制；它能够在降低计算强度的同时提高认证的速度，这是 μTESLA 的主要贡献。同时，μTESLA 也利用了单向 Hash 链保证了密钥的安全性，以及一定程度的容丢包性/容错性。

在 2004 年，一个改进的协议，多级 μTESLA[3]被提出，它从三个方面扩展了 μTESLA 的性能。首先，不同于 μTESLA 中用点对点的单播发送初始参数的方法，它采用预制定并且广播发送初始参数。其次，不同于 μTESLA 中只是用单条 Hash 链来发布广播消息，它采用了多级 Hash 链机制。最后，他采用冗余消息传输以及随机选择策略来发布密钥链头，有效地提高了抗拒绝服务攻击（Denial of Service, DoS）的能力。

在 2005 年，文献[5]提出了一个新的引导机制，它可用于传感器之间的认证。它是多级 μTESLA 的一个局部改进同时也是多级 μTESLA 的应用实例。它改进了发布初始化参数的方法，每个节点都只需要预安装一个组主密钥。然而，考虑到组密钥的可信性以及节点间的非直接通信，这个引导方式并不很实用。

同年，文献[6]提出了一种基于多发送者的广播认证协议。它利用多个发送者来分配用于 μTESLA 实例的参数。而且，它提出了基于撤销树（revocation tree）的机制来立即认证

μTESLA 的参数，该认证方法是由预分配机制实现的，它能够抗 DoS 攻击。然而，随着发送者数量的增加，构建一个撤销树的代价也会相应的增加。这样，存储开销以及通信开销也会增加。

在 2009 年，文献[7]提出了一种基于多用户的广播认证协议。它提出了几种有效的基于公钥的机制，实现了立即广播认证同时增加了安全强度。然而，在目前的无线传感器网络环境中，采用公钥密码体制开销昂贵。

1.1.2 广播认证协议的研究内容及创新点

以上的 μTESLA 系列协议都互不相同，并且，它们对无线传感器网络广播认证的安全性和高效性都有重要的贡献。然而，它们所存在的缺点也同样不容小觑，总结如下：

首先，出现在文献[7]中的传统的非对称机制，例如非轻量级的公钥密码体制（Public Key Cryptography, PKC），由于其过高的计算开销，并不鼓励用在资源受限的无线传感器网络中。

其次，μTESLA 系列中借助 Hash 链来构造特殊的非对称机制，实现了轻量级的认证。但是，Hash 链的长度存在一个矛盾：太短则消耗过快，用尽后，系统必须重新初始化，即需要再生新的 Hash 链，而且 Hash 链的再生一般都还要与系统初次启动一样使用公钥签名技术，这严重有损于系统的效率[13]，造成计算复杂性提高；而太长的话，首先，会造成存储开销增加，其次，会降低 Hash 链的使用效率，最后，发送方在初始化阶段就需要进行链长次的 Hash 运算操作，不仅增加了工作负担而且时间分配也不合理。这一矛盾从来没有在 μTESLA 协议[2]中提起；而在使用了多级密钥链的多级 μTESLA 协议[3]中，该矛盾只是有所减轻，并没有解决。

并且，有限的广播通信半径并没有引起以上协议[2, 3, 5-7]的注意。在文献[2, 3, 5, 7]中，只有一个发送者，即通信范围仍然有限的基站。即使在文献[7]中，采用多个发送者依然不是为了扩大网络的覆盖范围。特别是在布满大量传感器节点的大范围场景下，通信范围的考虑是不容忽视的。

最后，一个常常被 μTESLA 系列[2, 3, 5-7]忽略的安全问题，即所谓的明文认证并不能对一些重要的信息（例如，密钥链首）提供必要的保护。同时，缺少信任的认证交互，会导致入侵者很容易制造虚假的数据包。在极端情况下，基站作为认证中心会受到 DoS 攻击。

为了克服以上广播认证协议的弊端，特别是 μTESLA 和多级 μTESLA 的缺点，本文提出了一种改进的广播认证协议，称为 DH-μTESLA。

有一点需要补充，尽管对于无线传感器网络来说，公钥密码体制的计算开销太大；但有时，一些改良的版本仍然能提供相对简单的方案并且相似的安全性，比如椭圆曲线数字签名算法（Elliptic Curve Digital Signature Algorithm, ECDSA）。ECDSA 是一个轻量级的公钥密码体制，它适用于无线传感器网络中的身份认证。随着越来越多地传感节点加入到网络中，对认证速度、存储空间以及基站的数据安全的需求变得越来越重要甚至必要。由于以上的原因以及为了克服 μTESLA 系列的第一个和第四个弊端，基于 *d*-left counting Bloom filter 的认证机制（AdlCBF）将包含在 DH-μTESLA 协议中。其中，基站是一个超级节点，它负责大部分的加密操作。关于 *d*-left counting Bloom filter 的相关工作将在第二章的第二节介绍。

许多研究者都在致力于解决 Hash 链的矛盾问题（μTESLA 系列的第二个弊端）。这部分的相关工作将会在第二章的第一节介绍。但可惜的是，现存的方案都存在或多或少的安全问题以及效率问题。为了完全克服第二个弊端，本文提出了基于(t, n)门限和划分树的可再生 Hash 链构造方案（SRHC-TD）作为 DH-μTESLA 的安全组件。

最后，为了实现可扩展的通信范围，基于父亲树的分簇路由协议（PTCR）被构造出来用于解决 μTESLA 系列的第三个弊端。同时，它也是 DH-μTESLA 的第三个核心部分。其中，为了延长网络的生命周期并且最优化分簇的结果，本文提出了周期性轮转簇头选择算法，它是 PTCR 的一部分。诚然，在无线传感器网络中，基于能量感知的分簇协议并不是一个很新的主题；但是，它的相关工作将会在第二章的第三节进行分析，并且本文会结合入侵检测介绍 PTCR 的必要性以及可应用的场景。

1.2 入侵检测的研究背景及内容

正如摘要中所介绍，之所以引入移动传感网的入侵检测，是因为：

首先，在无线传感器网络中运行 DH-μTESLA 协议时，网络常常需要采用主动的防御措施。因此，在本文的后半部分，我们将研究传感器网络中的入侵检测问题。

而且，在性能分析那一章我们将会看到：由于 PTCR 中的的簇头轮转机制、新节点的加入、以及该算法在实现 800 轮之后的特有的网络拓扑结构的变化，同一局部区域内，节点数目的会发生变化。这就类似于移动传感网中节点的移动。

1.2.1 入侵检测的研究背景及意义

最近，无线传感器网络被越来越多地应用于安全领域，比如，入侵检测，边境监视以

及破坏监测。栅栏覆盖很适用于这些应用，因为它能够监测到任何跨越栅栏区域的运动[8]。而且，使用移动的传感器节点能够增强覆盖效果，以避免静态无线传感器网络中存在的覆盖空洞。并且，引入传感节点之间的以及传感节点与入侵者之间的虚拟势场，能够清楚地描述运动轨迹以及运动情况。因此，虚拟势场下移动传感网的入侵检测是本文的重点。

首先，如果，跨越带状区域的宽的任意一条路径都至少被 k 个不同的传感节点所覆盖，那么称该带状区域被 k 栅栏覆盖。它首先是在静态无线传感器网络中被提出来的[8]。而在移动传感网络中，栅栏覆盖更加倾向于设计节点重定位算法[9]或者公式化动态因素[10]。然而，没有相关的文献是结合这两点进行研究的。

其次，文献[11]第一次提出移动传感网中虚拟势场的概念。虚拟势场中的每一个节点都被障碍或者其他的节点所推开。这样，网络最终能够蔓延到整个应用环境中。但是，现存的文献中忽略了节点和入侵者之间的虚拟力的作用。

1.2.2 入侵检测的研究内容及创新点

因此，我们现在简单描述下 k 栅栏覆盖，节点和入侵者在移动传感网络中的运动，以及相关的研究问题。假设节点是随机部署在带状区域内的，对于每个传感节点，它的运动情况受到它与其他节点间的斥力的作用。当两个节点靠得足够近的时候（感知范围有重叠），它们之间的斥力开始起作用。于是，它们被彼此推开。这样看起来，节点的运动模式非常类似于物理学中的完全弹性碰撞模型。而对于入侵者而言，只考虑它从带状区域的一条平行边跨越到另一条平行边的运动过程。在此过程中，为了尽量不被传感节点所监测到，它需要尽量远离传感节点的感知范围。这样，它的运动模式就非常类似于物理学中的点电荷模型。因此，以上两个模型能够帮助我们来建立并且量化移动传感节点与入侵者之间的内在的动态关系。

文献[10]中提到，确定一个移动传感网时刻都满足 k 栅栏覆盖是不现实的。只能通过节点和入侵者的任意路劲来考虑可能的覆盖情况，最后得出 k 栅栏覆盖的可能性。同理，本文也只研究 k 栅栏覆盖的概率。

1.3 论文结构安排

文章结构安排安排如下：

第二章讨论广播认证和入侵检测的相关工作。

第三章给出 DH-μTESLA 的具体描述。

第四章描述 SRHC-TD 机制。

第五章描述 AdlCBF 机制。

第六章介绍 PTCR 算法。

第七章描述移动传感网中的 k 栅栏覆盖问题。

第八章性能分析。

第九章总结全文并对未来展望。

第二章 广播认证和入侵检测的相关工作

2.1 广播认证的相关工作

如绪论中所述，DH-μTESLA 协议从三个方面进行了改进。所以，我们将从以下三个方面介绍现有的一些相关工作。

2.1.1 为什么要提出 SRHC-TD

为了解决 Hash 链长度的矛盾（在简介部分已经介绍），许多文献[12-18]都已经提出了新的 Hash 链的构造方案。

2004 年，Goyal V. 提出了可再生 Hash 链 (Re-initializable Hash Chain, RHC) 的构造方案[12]。RHC 的构造主要思想是：当一个 RHC 用尽以后，能够以不可否认的方式安全地再生，从而得到另一个 RHC。

2006 年，在 RHC 的基础上，Zhao Y. C.等人提出了精巧可再生 Hash 链 (Elegant Re-initializable Hash Chain, ERHC) 的构造方案[13]：1) 网络初始化时，发送方随机生成 $L+\lfloor \mathrm{lb}(L) \rfloor+1$ 个随机数（其中 L 是 Hash 函数的参数，即 Hash 函数输出值的长度），将它们的级联记作 S_U。并对这些随机数分别 Hash 后级联，记作 P_U。再以 P_U 为根，计算一条长度为 k 的 Hash 链，链首为 $h^k(P_U)$，以安全方式发送给接收方。2)待 Hash 链耗尽后，发送方生成新的实例 S_U' 和 P_U'，并生成一条链首为 $h^k(P_U')$ 的新链。在发送新链的链首 $h^k(P_U')$ 时，只要发送需公开的部分 S_U，让接收方通过 P_U 和 S_U 的公开部分这两者的结合来验证 $h^k(P_U')$。具体算法详细过程可参见文献[13]。ERHC 在旧链消耗完毕后能够平滑地生成新链，概念上对有限长度 Hash 链进行了自然、合理的扩展，逻辑上构造了无限长的 Hash 链。但是，由于发送方通过透露部分 S_U 信息来认证下一条链的链首，会存在选择明文攻击。

2008 年，Zhang H. J.等人提出了自更新 Hash 链 (Self-Updating Hash Chain, SUHC) 的构造方案[14]。该方案基于 Hard Core Predicate 算法，在分配第一条链的 Hash 密钥值时，顺便分配第二条链链首的 1 bit 的值。这样，当第一条密钥链用完时，第二条链首的所有 bit 都分配完（即得到第二条链首

值）。

同年，Zhang H. J.又对 SUHC 进行了改进，并提出了理想自更新 Hash 链 (Self-Renewal Hash Chain, SRHC) 的构造方案[15]。两方案的主要区别在于随机数选择上不同：SUHC 中是选择满足 $B(SR_i)=\omega'[\ ]$ 条件的随机数 SR_i，并且对 SR_i 做 Hash 计算得到 PR_i，之后是对 PR_i 的操作；而 SRHC 中是选择满足 $B\left(h^{k-i}(S)\|R_i\right)=\omega'[i]$ 条件的随机数 R_i，并直接对 R_i 进行操作。但是，SUCH 和 SRCH 中都存在以下的缺点：将新链链首值的每 1 bit 位，映射到一个随机数，对链首值的安全公布即是对 k（链长）个随机数的安全公布。显然，两方案不可避免地要求所有的随机数都必须完整地被接收，这样才能完整地重构新链的链首，显然剔除了 Hash 链原有的容错性的特点。

2009 年，在 SUHC 的基础上，Zhang M. Q.等人提出了新的自更新 Hash 链 (New Self-Updating Hash Chain, NSUHC) 的构造方案[16]。2010 年，在 NSUHC 的基础上，Zhang W.提出了提出了基于纠羽码 (Erasure Coding) 的自更新 Hash 链 (SUHC-EC) 的构造方案[17]。两方案的基本思想：前者是将新密钥链的种子值（非密钥链链首）从 k 维扩充到 n 维 $(k<n)$，而后者则是将新密钥链的种子值从 1 维扩充到 n 维；接着，两方案都是无重复地选择这 n 个随机数中的一个来发布，经过 k 次之后就能恢复出新种子值。两方案都依赖于一个密钥链种子值服务器，服务器端将新的种子值，通过分割、扩充维数等操作，传输给客户端，客户端再用新的种子值来生成新的密钥链。从某种程度上实现了 Hash 链可再生的概念，但是从 Hash 链值使用者的角度来看，NSUHC 和 SUHC-EC 的构造方案与一般的 CHC[1] (Conventional Hash Chain) 的构造并没有什么不同。

2010 年，Yang X. Y.等人提出了基于公平交易 (Fair Exchange Idea) 的自更新 Hash 链(SRHC-FEI) 的构造方案[18]。该方案在每次发布 Hash 链值的同时，利用 OTS (One-Time Signature) 密钥来对新链链首值的一个 bit 位进行加密传输。分析发现，SRHC-FEI 包含了认证、OTS 等元素，更像是一个应用而不是一个新的关于 SRHC 构造方案。而且，它虽然增强了一定的安全性及公平性，但是大大增加了开销，并且增加了系统的延时。

分析以上经典文献，不难发现，不管是 RHC、ERHC，还是 SUHC、SRHC、SRHC-FEI，它们都是对新链链首值的每一个 bit 位做相应的变换或者映射成一个随机数，对新链链首值的安全发布即是对这些对应的随机数的安全发布。而且，方案中要求所有的随机数都必须完整地被接收才能正确地恢复出新链的链首值。显然，它们都一定程度上减弱了系统的安全性，并且大大地增加了系统的开销。

[1] CHC 是比 RHC 更早出现的概念。它是 Hash 链的最初的形式，同时也是 Hash 链的全称。我们可以简单地将 RHC 看作是多个 CHC 链接而成。

另一方面，NSUHC，SUHC-EC 则是将新链链首值的维数进行适当的扩增，对新链链首的安全发布即是对这扩充后的 n 个随机值中的 k 个值的发布。同理，只有这 k 个随机值都完整地被接收才能正确地恢复出新链的链首值。与 RHC，ERHC 等方案比较，降维帮助了 NSUHC 和 SUHC-EC 减少了开销，但是增加了被中间人攻击的可能。更重要的是，从 Hash 链值的使用者角度来看，只有 RHC，ERHC，SUHC，SRHC 才能真正算是可再生 Hash 链的新颖的构造方案。

为了抵制中间人攻击，并且或者高强的安全性以及高效的性能，本文在第四章提出了一种改进的 Hash 链再生方案——一种基于(t, n)门限和划分树的可再生 Hash 链构造方案 (Self-Renewal Hash Chain based on (t, n) Threshold and Division tree, SRHC-TD) 。它是 DH-μTESLA 协议的安全基础，解决了 RHC，NSUHC 等协议中的固有缺点。

2.1.2 为什么要使用 AdlCBF

布隆过滤器 (Bloom Filter, BF) 是一种节约空间的概率数据结构。它支持近似表示和近似成员查询[19]。BF 提供了非常紧密压缩的空间存储：只需要少于 10 bit 的存储空间，就能保证每个元素只含有 1%的假阳性概率；并且，它（前半句）是独立于集合的大小（即集合中元素的个数）。

标准的 BF 只是一个简单的数据结构，它只有插入和查询元素的功能。由于没有删除元素的功能，BF 难以很好地（几乎不能）处理动态的集合。

2000 年，Li F.等人提出的可计数的布隆过滤器[20] (Counting Bloom Filter, CBF) 则兼具插入和删除元素的功能，它能够很好地处理动态的数据而不用重写建立新的过滤器[21]。然而，相比标准的布隆过滤器（BF），CBF 需要使用高达 3 到 4 倍的空间。

BF 和 CBF 在已经被明确而广泛地应用于各种认证机制中[7, 22, 23, 26, 27]。

2007 年，Kui R. 将 BF 和 CBF 用于存储公钥的 Hash 值，并且用于验证网络用户的身份[7]。在基于认证机制的布隆过滤器 (Bloom filter base Authentication Scheme, BAS) 中，BF 被用于处理关于公钥的密钥管理部分。而在混合认证机制 (Hybrid Authentication Scheme, HAS) 中，BF 则与 Merkle Hash Tree 联合，一同来管理公钥。

2010 年，Son J.H. 等人将 BF 作为一个认证者，用以抵抗已认证用户的洪泛攻击[22]。在基于布隆过滤器的消息认证协议 (Bloom filter-based Message Authentication Protocol, BMAP) 中，BF 用于压缩多个消息认证码 (Message Authentication Code, MAC) 。它能够减少消息的尺寸，其代价通常只是获得了一个可忽略不计的假阳性概率。

2012 年，Ayday E. 等人用 BF 来生成并且验证认证信息[23]。与 *RatelessI*[2]相联合，BF 更加具有实用性、可靠性以及身份认证的功能。

借由应用在静态数据集合中的指纹 (Fingerprint) 和完美散列 (Perfect Hashing)[26]的概念，Bonomi F. 引入了 *d*-left 计数型布隆过滤器 (*d*-left Counting Bloom Filter, dlCBF) 的概念[27]。dlCBF 在性能上与 CBF 类似，并且它所需要的空间只有 CBF 的一半。因此，我们结合 dlCBF（而不是 BF 或者 CBF）与 ECDSA 来实现 DH-μTESLA 协议中的身份认证的功能。该功能命名为 AdlCBF (Authentication Scheme base *d*-Left Counting Bloom Filter)，它在认证速度以及存储空间方面具有极好的表现。关于 AdlCBF 的具体实现将会在第五章介绍。

2.1.3 为什么要使用 PTCR

当传感节点被配置完之后，内部的电池是很难进行更换的。因此，关于无线传感网的协议以及应用的设计必须是能量感知的，这样才能延长网络的生命周期。其中，最有效的节能的方法之一，就是组织传感节点成簇状。

2000 年，Heinzelman W.R. 提出了经典的低能耗的适应性的簇状层级 (Low-Energy Adaptive Clustering Hierarchy, LEACH) 协议[28]。顾名思义，它是一个基于分簇的路由协议，它使用了本地簇头的随机轮转机制来实现整个网络中的能量分布的均衡。

2002 年，Heinzelman W.B. 提出了 LEACH-C 协议[29]。它结合了基于簇的节能的路由算法、介质访问算法、特定应用的数据聚集算法，达到了良好的性能表现，包括延长了系统的生命周期，降低了时延，并且提高了应用感知的质量。

2004 年，Smaragdakis G. 等人提出了层簇同构网络稳定选择协议(Stable Election Protocol for clustered heterogeneous wireless sensor networks, SEP) [30]。它是一个面向异构网络的协议，用于延长网络的稳定周期。其中，稳定周期（或者称稳定时间）是指从系统启动到出现第一个死亡节点的这一段时间。总之，SEP 相比较于 LEACH 以及 LEACH-C 协议，具有更长的稳定时间。

2006 年，Li Q. 等人提出了分布式节能层簇算法 (Distributed Energy-Efficient Clustering algorithm for heterogeneous wireless sensor networks, DEEC)[31]。它是面向异构网络的、一个新的、分布式的、节能的分簇协议。在 DEEC 中，先计算每个节点的剩余能量与网络的平均能量的比值，通过这个比值

[2] *RatelessI* 是一个基于无速率码的信息传递机制 (Rateless Information Delivery Mechanism)[24,25]。在文献[23]中，它被用来提供数据的可靠性。

再来计算每个节点被选为簇头的概率。与那些低能量的节点相比，具有较高初始能量以及剩余能量的节点更容易被选为簇头。并且，引入分簇技术能够收集数据并且包含更多有意义的信息。因此，在异构网络中，相比 LEACH，LEACH-C，以及 SEP，DEEC 实现了更长的网络生命周期以及收集了更加有效的数据信息。

同年，Kim D.S. 提出了 LEACH-M 协议[32]。当一个簇内的成员（传感节点）移动时，LEACH-M 可用于宣称它们的身份。而且，它可用于确认在一个时隙内（该时隙是由时分多址调度 (Time Division Multiple Access schedule, TDMA) 分配的），某个传感节点是否能与一个指定的簇头进行通信。

结合 LEACH 中分簇的概念、SEP 中的同构性、DEEC 中的能量均衡、以及其他的一些方面或概念，本文在 DH-μTESLA 协议中提出了基于父亲树的路由算法(Parent-Tree based Clustering Routing Algorithm, PTCR)。它是一个节能的分簇路由算法，扩展了基站的通信范围，并且合理地延长了网络的生命周期。关于 PTCR 算法的具体步骤将在第六章介绍。

2.2 入侵检测的相关工作

在入侵检测成为无线传感网中的重要研究问题之前，它第一次被提出是在基于传感节点的自动系统中[33]。覆盖，作为入侵检测的重要分支，越来越受到研究者的关注。覆盖分为两大类：全覆盖和栅栏覆盖。全覆盖能够保证连通性并且最大化覆盖区域的检测率（本文不做研究）。而栅栏覆盖能够最小化入侵者不被检测的概率。

关于栅栏覆盖的研究内容有很多。包括，可靠密度[34]，性能分析[35]，质量分析[36]，以及节能的相关问题[37-42]。而移动传感网中的栅栏覆盖是另一项重要的分支，它被多次研究[9, 10, 42-45]。文献[10, 42, 43]中都说明了节点的移动性能够提高覆盖的性能。文献[10]罗列出了 k 栅栏覆盖的性能与一系列重要的动态因素之间的关系。文献[42]提出，网络监控中的覆盖问题即是保证目标实体所在的观察区域在一个或多个节点的感知范围内。类似地，文献[43]也研究了移动传感网中检测目标在与不在的问题。同时，文献[9, 44, 45]主要侧重于研究虚拟势场中移动节点的重定位算法。文献[9]提出了节约能耗的节点重定位方法，相比静态随机分布的方法，它只需要较少的节点就达到相同的效果。文献[44]讨论通过移动传感节点的部署来扩大静态区域的覆盖。文献[45]则解决了栅栏覆盖中移动传感节点的理想化运动问题。

目前为止，本文是第一个研究虚拟势场中 k 栅栏覆盖的概率的。通过借用并且适当修改调整物理学中的完全弹性碰撞以及点电荷模型，本文研究了移动入侵者与移动传感节点之间的动态关系，并且

在此基础上评估了 k 栅栏覆盖的性能。

2.3 本章小结

首先，本章主要从三个方面介绍了无线传感网络中广播认证协议的相关工作：可再生 Hash 链，基于布隆过滤器的身份认证，以及分簇路由算法。

其次，考虑到广播认证协议中的安全措施更属于被动防御，而不是主动安全监测，所以，本章后来又介绍了入侵检测的相关工作。

第三章 DH-μTESLA 协议

与 μTESLA 协议类似，但是又有所不同，DH-μTESLA 协议具体可分为 6 个步骤。它们分别是假设、发送者设置、启动新接收者、网络初始化、广播信息、认证信息。

3.1 假设

我们假设基站 (Base Station) 是固定的，并且成千上百的传感节点分布在一个大型的稠密的分布式无线传感器网络中。其中，基站具有无限的存储空间、处理能力以及能量供给。相反，网络中的传感节点具有受限的存储空间、计算能力、带宽以及能量。并且，在安全方面，我们假设基站总是可信的但是传感节点则有可能会受到攻击而妥协。需要注意的是，基站可位于网络中的任何位置，当它处于网络中心位置时，如图3.1所示。

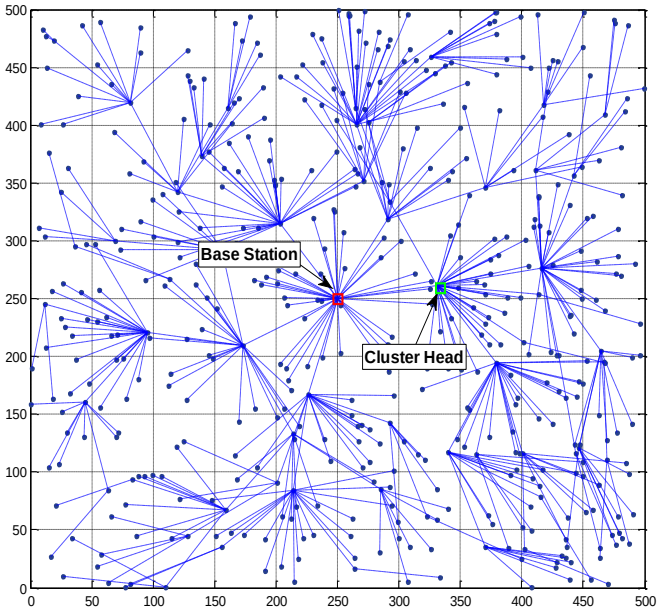

图 3. 1 大规模无线传感器网络拓扑结构的展示图。
网络场景是 500m× 500m，并且在该区域内有 500 个传感节点随机分布。

3.2 发送者设置

首先，发送者（基站或者称为数据处理中心）使用SRHC-TD来生成一系列的私钥。具体的实现过程可以参看4.4.1节的第一部分关于SRHC-TD的初始化部分。然后，发送者构造dlCBF来存储那些合法的传感节点的ID与公钥的指纹 (fingerprint) 信息。其中，节点的ID以及公钥信息，在分布前，就已经预载在每个合法的节点中。具体的实现细节描述参见第四章的构建阶段。

3.3 启动新接收者

新的节点可由AdlCBF机制来认证。具体的实现细节参见第五章。只有合法的节点才能加入网络。为了运行DH-μTESLA，每个合法的节点都必须有一个commitment值（一般是Hash链的链首值）以及其他的关于可再生Hash链的初始参数。这些参数可在第五章的算法5.3或者算法5.4中获得。它们包含有：Hash链的长度n，一个安全的Hash函数$h:\{0,1\}^{*}\to\{0,1\}^{L}$，以及划分对$(m_L,l_L)$。并且，合法的节点同样也获得其他的初始信息。比如有当前时间T_c，发送者与接收者之间的最大时钟误差Δ，时隙i的初始时刻T_i，每个时隙的时长T_{int}，以及延迟发布的尺寸δ。其中，最后三个参数被用在密钥公布机制中。

3.4 网络初始化

利用第六章的介绍的PTCR算法，可建立一个簇型拓扑结构的网络。然后，所有的传感节点被分类为簇头节点或者是普通节点。其中，簇头 (Cluster Head) 节点负责管理它所在的簇内的节点，包括集成、接收、转发消息；而普通节点只需要同它所在簇的簇头进行通信即可。这样，DH-μTESLA协议内基站节点进行广播的过程可以看做是簇内广播以及簇间分层扩散的过程。

3.5 广播信息

每个传感节点的生命周期都可划分为多个固定时长（T_{int}）的时隙。而一条Hash链的生命周期为$n+1$个时隙（n为Hash链的长度）；然而，就像在第四章所介绍的，多链Hash链能够首尾链接成一条可再生的链，并且可再生Hash链的生命周期是无限的，绝对能够完全覆盖每一个传感节点的生命周

期。如图3.2所示，发送者首先使用密钥值来计算当前时隙内每个数据包的MAC值；然后，广播该数据包以及对应的MAC值；之后，经过一个固定的时间间隔 $\delta \cdot T_{\text{int}}$，发送者公布密钥链上的每个密钥值。

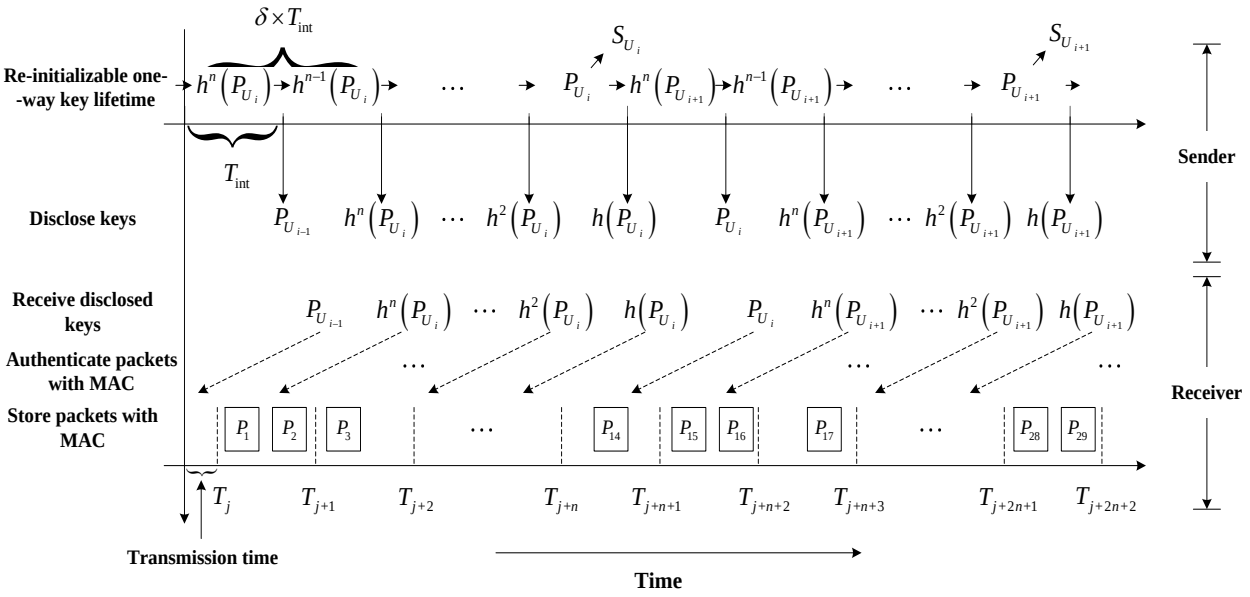

图 3. 2 使用可再生单向 Hash 链的广播过程。每个密钥都在相应的时隙内被使用。

当公布完前一条链的种子值时，例如 S_{U_i}，下一条链的链首值 $h^n(P_{U_{i+1}})$ 就能够立即被计算出来。

3.6 认证信息

当一个传感节点收到密钥公布数据包以及相应的MAC值时，它首先需要检查该数据包的接收时间；然后，它在检查该密钥值以及相应的MAC值。具体的细节将在第三章第四节的第三部分的验证阶段中进行描述。

当一个传感节点接收到普通的广播数据包以及相应的MAC值时，它首先将其存储下来；然后等到相对应的密钥公布之后，再来依次验证它们。

3.7 本章小节

本章主要介绍了 DH-μTESLA 协议的 6 个步骤：假设、发送者设置、启动新接收者、网络初始化、广播信息、认证信息。

第四章 基于(*t*, *n*)门限和划分树的可再生 Hash 链构造方案（SRHC-TD）

为了减少开销并且保证容错性，SRHC-TD 机制考虑到了“重复”以及“序列”的概念。为了保证系统的安全性以及完整性，SRHC-TD 机制考虑到了(*t*, *n*) 门限以及 salt 值的概念。

需要注明的是，划分树是一个新型的数据结构，它常以图的形式直观而清晰地描述重复度的概念。并且它具有一个相对更加精简的形式，称为改进的划分树。

4.1 基本定义

在具体描述构造方案之前，首先给出一些基本的相关定义。

定义 4.1 (分裂) 将长度为 L (bit) 的数分割成 m 个 2^l 进制数的过程称为分裂。简单表示为 $L=(m, l)$，$m = \left\lceil \frac{L}{l} \right\rceil$。如果 L 不是 2^l 的整数倍，则在 L (bit) 个数之前填充小于 2^l 个 0。

定义 4.2 (重复向量、取值向量、重复度) 设 m ($m \geqslant 1$) 个数的可能取值有 n ($n \geqslant 1$) 种，其中有 p_i ($1 \leqslant p_i \leqslant m$，$\sum_{i=1}^{q} p_i = m$) 个数取值为 v_i (v_i 取值各不相同), $i = 1, 2, \cdots, q$，$1 \leqslant q \leqslant \min(m,n)$，则将 $(p_1, p_2, \cdots, p_q)$ 称为重复向量，$(v_1, v_2, \cdots, v_q)$ 称为取值向量，$m-q$ 称为重复度。且 $m-q = \sum_i p_i - q = \sum_i (p_i - 1)$。

定义 4.3 (重复率) 将重复度为 $m-q$ 时 m 个数的取值情况的个数记为 $S_{m,n}^{q}$，则当 q 取遍 $[1, 2, \cdots, \min(m, n)]$ 的所有值时，m 个数的所有取值情况的个数记为 $\sum_{i=1}^{\min(m,n)} S_{m,n}^{i}$，前者与后者的比率称为重复度为 $m-q$ 时的重复率，简记为重复率，用 P_q 表示。$P_q = \frac{S_{m,n}^{q}}{\sum_{i=1}^{\min(m,n)} S_{m,n}^{i}}$，其中，$S_{m,n}^{q} = C_m^{p_1} \cdot C_{m-p_1}^{p_2} \cdots C_{p_q}^{p_q} \cdot \frac{n!}{(n-q)!}$。

定义 4.4 (难度) 设 m ($m \geqslant 1$) 个数，重复度为 $m-q$，则将 q 称为难度。

定义 4.5 (平均难度) 难度的平均加权和，用 D_q 表示。$D_q = \sum_{q=1}^{\min(m,n)} P_q \cdot q$。

4.2 (*m*, *q*)划分

在介绍完以上五个基本定义之后，我们接下来考虑一系列关于(*m*, *q*)划分以及划分树的定义。

定义 4.6 ((*m*, *q*)划分) 将正整数 *m* 无重复地划分成 *q* 个正整数相加的过程称为(*m*, *q*)划分。(*m*, *q*)划分对应一棵 *m* 划分树或者一棵改进的 *m* 划分树。*m* 划分树的树形结构如图 4.1 所示，改进的 *m* 划分树的树形结构如图 4.2 所示。

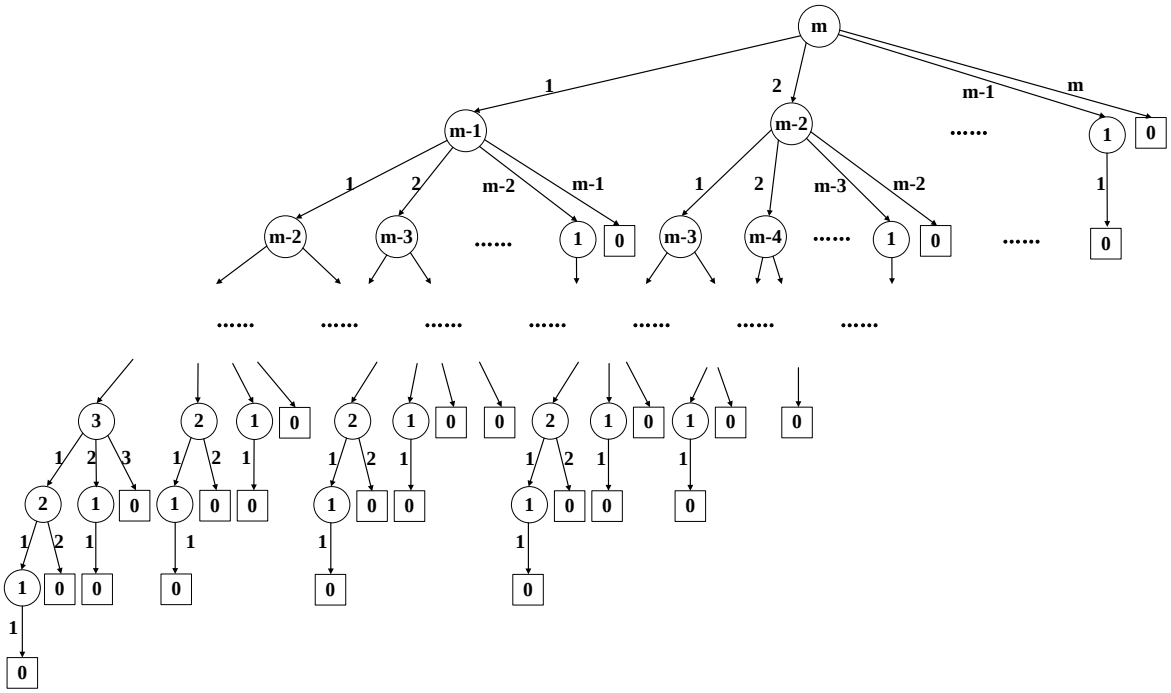

图 4.1　*m* 划分树的结构示意图

定义 4.7 (*m* 划分树) 将具有以下 4 个特点的树称为 *m* 划分树。

1) 根节点值为 *m*。

2) 值为 *l* 的节点有 *l* 个孩子，并且 $l \leqslant m$。

3) *l* 节点的第 *i* 个孩子的值为 $l-i$，*i* 叫做父亲节点到子节点的路径权值，并且 $i \leqslant l$。

4) 叶节点的值为 0。

要找到所有的(*m*, *q*)划分只需要根据 *q* 值直接遍历 *m* 划分树的第 *q* 层(根节点为第 0 层)的叶节点，再从叶节点回溯到根节点，如果回溯路径有重复（即至少存在 2 条回溯路径，它们的所有路径权值组

合相等但排列不同），则取其中一条回溯路径作为一个划分，这样，就找到了所有的非重复的$(m,\ q)$划分。

不妨以$(m, 2)$划分为例。如图 4.1 所示，直接遍历第 2 层的$m-1$个叶节点，即找到了$m-1$条有重复的回溯路径，分别是$(1, m-1),\ (2, m-2),\ \cdots,\ (m-2, 2),\ (m-1, 1)$。再进行比较可知$(i, m-i)$与$(m-i, i)$ (其中，$i=1, 2, \ldots, m-1$)是重复路径，去掉重复路径，得到$\left\lfloor \frac{m}{2} \right\rfloor$条无重复的回溯路径，即找到了$\left\lfloor \frac{m}{2} \right\rfloor$个$(m, 2)$划分。

由于m划分树中存在重复划分的情况，需要回溯来删除重复的划分。不妨直接将路径权值作为考虑因素来进行构造改进的m划分树。

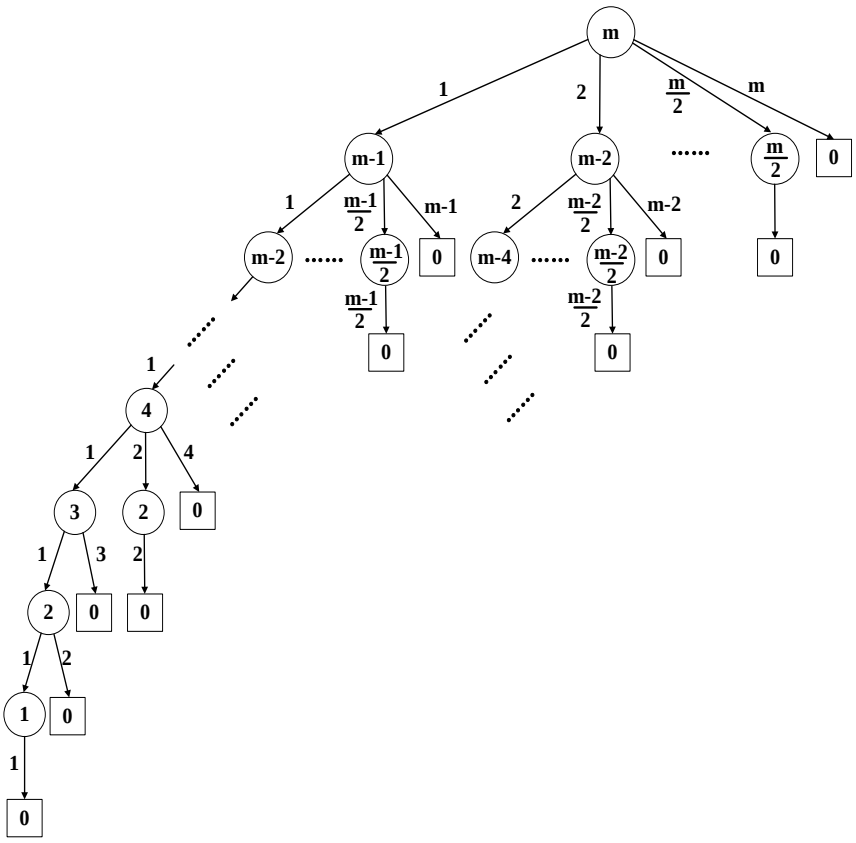

图 4. 2　改进的 m 划分树的结构示意图

定义 4.8 (改进的 m 划分树) 将具有以下 4 个特点的树称为改进的 m 划分树。

1) 根节点值为 m，且有$\left\lfloor \frac{m}{2} \right\rfloor+1$个孩子。

2) 若一个中间节点的值为 l $(0<l<m)$，且其父亲节点到它的路径权值为 r，当$l \geqslant 2r$时，该节

点有$\left\lfloor \frac{l}{2} \right\rfloor - r + 2$个孩子，当$r \leqslant l < 2r$时，该节点有一个孩子(叶节点)。

3) 父亲节点到子节点的路径权值范围是$\begin{cases} \left\{ r, r+1, \cdots, \left\lfloor \frac{l}{2} \right\rfloor \right\} \cup \{l\} & l \geqslant 2r \\ \{l\} & r \leqslant l \leqslant 2r \end{cases}$。

4) 叶节点的值为 0。

这样再根据 q 值直接遍历第 q 层(根节点为第 0 层)的叶节点，由于避免了重复，叶节点的个数即是划分的个数。这样，就找到了所有的非重复的(*m*, *q*)划分。

不妨再以(*m*, 2)划分为例。如图 4.2 所示，直接遍历第 2 层的$\left\lfloor \frac{m}{2} \right\rfloor$个叶节点，找到了$\left\lfloor \frac{m}{2} \right\rfloor$条无重复的回溯路径，分别是$(1, m-1), (2, m-2), \cdots, (\left\lfloor \frac{m}{2} \right\rfloor, \left\lceil \frac{m}{2} \right\rceil)$，即找到了$\left\lfloor \frac{m}{2} \right\rfloor$个(*m*, 2)划分。

4.3 (*t*, *n*)-Mignotte's 门限秘密共享方案

首先，我们给出一些相关定理及定义。

定理 4.1 (中国剩余定理(Chinese Remainder Theory, CRT)) 对于所有的 $1 \leqslant i, j \leqslant k$，当$(m_i, m_j) = 1$，得到中国剩余定理的标准形式：

设$m_1, m_2, \cdots, m_k$是两两互质的 k 个正整数，$k \geqslant 2$，则同余方程组

$$\begin{cases} X \equiv b_1 \bmod m_1 \\ X \equiv b_2 \bmod m_2 \\ \quad \vdots \\ X \equiv b_k \bmod m_k \end{cases} \tag{4.1}$$

有模$M = m_1 \cdot m_2 \cdots m_k$的唯一解$X = \sum_{i=1}^{k} b_i M_i (M_i^{-1} \bmod m_i) \bmod M$，其中，$M_i = \frac{M}{m_i}$，$1 \leqslant i \leqslant k$。

定义 4.9 ((*t*, *n*)-Mignotte's 序列) 设 n 是一个整数，$n \geqslant 2$，且 n，如果给定正整数序列$m_1 < m_2 < \cdots < m_n$，$(m_i, m_j) = 1$，$1 \leqslant i < j \leqslant n$，满足$m_{n-t+2} \cdot m_{n-t+3} \cdots m_n < m_1 \cdot m_2 \cdots m_t$条件，就称该正整数序列是一个(*t*, *n*)-Mignotte's 序列。

如果给定一个(*t*, *n*)-Mignotte's 序列，秘密共享方案运行如下[46]。

1) 随机选择一个整数 S 作为秘密，且要满足：$m_{n-t+2} \cdot m_{n-t+3} \cdots m_n < S < m_1 \cdot m_2 \cdots m_t$条件，如果设

$\alpha = m_1 \cdot m_2 \cdots m_t$， $\beta = m_{n-t+2} \cdot m_{n-t+3} \cdots m_n$，即满足 $\beta < S < \alpha$ ；

2) 根据 $I_i = S \bmod m_i$，可以得到秘密份额 I_i，$1 \leqslant i \leqslant n$ ；

3) 任意选定 *t* 个不同的秘密份额，分别记为：$I_{i_1}, I_{i_2}, \cdots, I_{i_t}$，由中国剩余定理可以恢复出秘密 *S*，且在模 $m_1 \cdot m_2 \cdots m_t$ 下是唯一的。

$$\begin{cases} X \equiv I_{i_1} \bmod m_{i_1} \\ X \equiv I_{i_2} \bmod m_{i_2} \\ \quad \vdots \\ X \equiv I_{i_t} \bmod m_{i_t} \end{cases} \tag{4.2}$$

4.4 可再生 Hash 链 SRHC-TD 的构造方案

设单向散列函数 *h* 的输出的长度为 *L*bit(例如：MD5 算法的输出是 $L = 128$ bit)，并且发送接收双方协定将 *L* 分裂成 (m_L, l_L) 。下面分为五个阶段进行描述。

4.4.1 密钥初始化阶段

在初始化阶段，发送者与接收者协商了 Hash 链的长度 *n*，一个安全的 Hash 函数 $h: \{0,1\}^* \to \{0,1\}^L$，并且 Hash 函数的安全参数为 *L*，表示函数 *h* 的输出为 *L*-bits 长。并且 *L* 能被划分为 (m_L, l_L) 。

1) 初始化种子值。 发送方选择一个合适的 (t, m_L) -Mignotte's 序列 $\{x_1, x_2, \cdots, x_{m_L}\}$，将它们的级联记作 S_U，并且分别对序列中的每个正整数进行 Hash 运算，计算出相应的 m_L 个散列值，将这些散列值的级联记作 P_U 。

2) 初始化一条 Hash 链。发送方以 P_U 为初始种子值，生成一条长度为 *n* 的密钥链：

$$P_U, h(P_U), h^2(P_U), \cdots, h^n(P_U)$$

其中， $n-1$ 为 m_L 的整数倍。

3) 生成下一条链。发送方按照步骤 1) 和 2)，重新选择一个合适的 $(t, m_L)'$ -Mignotte's 序列，再计算一对新的密钥实例 S_U' 和 P_U' ；并且以 P_U' 为初始种子值，生成一条新密钥链：

$$P_U', h(P_U'), h^2(P_U'), \cdots, h^n(P_U')$$

将$h^n(P_U')$分割成m_L个2^{l_L}进制数$\{c_1, c_2, \cdots, c_{m_L}\}$，这$m_L$个$2^{l_L}$进制数的重复度记为$m_L - q_L$，难度记为$q_L$。

4) 设$\alpha = x_1 \cdot x_2 \cdots x_t$，$\beta = x_{m_L-t+2} \cdot x_{m_L-t+3} \cdots x_{m_L}$。如果满足条件$t = q_L$并且$\beta < h^n(P_U') < \alpha$，则以$h^n(P_U')$作为秘密（主密钥）$S$。否则，跳转到步骤 3) 选择另一个合适的秘密。

5) 根据$I_i = S \bmod x_i$，计算m_L个秘密份额（子密钥）$I_1, I_2, \ldots, I_{m_L}$。

6) 计算两组消息认证码，ζ_i和ξ_i；其中，$\xi_0 = h^{n-1}(P_U) \oplus \zeta_1$，$\zeta_1 = \left(x_{c_1+1}, I_{c_1+1}\right) << r_1$，　$r_1 = S_U$。

7) 安全秘密地公布$\left(h^n\left(P_U\right), \xi_0\right)$值给验证者。事实上，这对参数已经在第五章的算法 5.3 和算法 5.4 中以密文形式发送给了各节点。

4.4.2 密钥发送阶段

在密钥发送阶段，发送者需要计算并且发布 Hash 链的链值以及验证信息用于下一阶段的验证。

在第 i ($i = 1, 2, \ldots, n-1$) 轮，发送者需要依次进行以下三个步骤：

1) 计算或者恢复 Hash 链值$h^{n-i-1}\left(P_U\right)$ 和$h^{n-i}\left(P_U\right) = h\left(h^{n-i-1}\left(P_U\right)\right)$。

2) 计算消息认证码ζ_{i+1}和ξ_i。其中$\xi_i = h^{n-i-1}(P_U) \oplus \zeta_{i+1}$，$\zeta_{i+1} = \left(x_{c_k+1}, I_{c_k+1}\right) << r_{i+1}$，$k = i + 1 \left(\bmod m_L\right)$，$r_i$表示$P_U$值的第 i bit、第 $2i$ bit、……组成的数。

3) 计算并公布验证信息帧$\left(h^{n-i}(P_U), \zeta_i, \xi_i\right)$。

Remark.　在第 n 轮，发送者公布种子值P_U。

在$n+1$轮，发送者公布种子值S_U。

4.4.3 密钥认证阶段

如前所述，T_c是指当前的时间，它用来同步整个网络的时钟；Δ表示发送者与接收者之间最大的时钟误差；T_i表示时隙i的初始时刻；T_{int}是指每个时隙的时长，δ表示延迟发布的尺寸。

在第 i ($i = 1, 2, \ldots, n-1$) 轮，接收者需要依次进行以下两个步骤：

如果满足$\left\lfloor \frac{T_c + \Delta - T_i}{T_{\text{int}}} \right\rfloor < i + \delta - 1$，接收者接收来自发送者的发来的验证信息帧$\left(h^{n-i}(P_U), \zeta_i, \xi_i\right)$。

1) 计算并验证 $h\left(h^{n-i}\left(P_U\right)\right)$ 是否与 $h^{n-i+1}\left(P_U\right)$ 相等。其中，$h^{n-i+1}\left(P_U\right)$ 是之前验证后有效的、并存储下来的链值。

2) 计算并验证 $h^{n-i}(P_U)\oplus\zeta_i$ 是否与 ξ_{i-1} 相等。

如果以上两步骤都验证通过，那么接收者就认为发送者发送的数据是安全完整有效的，并且将 ζ_i 存下来。

如果不满足 $\left\lfloor \frac{T_c+\Delta-T_i}{T_{\text{int}}} \right\rfloor < i+\delta-1$，接收者丢弃验证信息帧中的 $h^{n-i}(P_U), \zeta_i$，而保存 ξ_i 值。然后它等待下一个有效的验证信息帧 $\left(h^{n-j}(P_U), \zeta_j, \xi_j\right)$，并且 $j>i$ 。

1) 计算并验证 $h^{j-i+1}\left(h^{n-j}\left(P_U\right)\right)$ 是否与 $h^{n-i+1}\left(P_U\right)$ 相等。其中，$h^{n-i+1}\left(P_U\right)$ 是之前验证后有效的、并存储下来的链值。

2) 计算并验证 $h^{n-j}(P_U)\oplus\zeta_j$ 是否与 ξ_{j-1} 相等。

如果以上两步骤都验证通过，那么接收者就认为发送者发送的数据是安全完整有效的，并且将 ζ_j 存下来。

4.4.4 密钥重组阶段

当所有的密钥链值都公布之后，整条 Hash 链就用尽了。此时，接收者存储了种子值 P_U 以及少于等于 m_L 个 MAC 值（特指 ζ_i）。

1) 计算并验证是否 ζ_{i_1} 和 ζ_{i_2} 内包含相同的序列值 x_{c_k+1} 和子密钥 I_{c_k+1}。其中，满足 $i_1-i_2\equiv 0 \bmod m_L$，$k=i_1+1\left(\bmod m_L\right)$，$i_1,i_2=1,2,\ldots,n-1$。

2) 验证完所有的 MAC 值 ζ_i 后，接收者得到 q_L 个不同的序列值 x_{c_k+1} 以及 q_L 个不同的子密钥值 I_{c_k+1}，以及所有的 m_L 个序列值的排列 (Permutation)。

Remark. 理论上，接收的序列值 x_{c_k+1} 的个数应该为 m_L。它包含重复的情况。即，可能出现 $x_{c_1+1}=x_{c_2+1}$ 的情况。因此，如果不考虑重复的情况的话，那么只有 q_L 个不同的序列值 x_{c_k+1}。简而言之，有序的序列值 $x_{c_1},x_{c_2},\ldots,x_{c_{m_L}}$ 是一个 (m_L,q_L) 划分。

3) 有序验证法：对于所有的 m_L 个序列值 $x_{c_1},x_{c_2},\ldots,x_{c_{m_L}}$，它们的下标值的联合 (Union) 即是新生

的第二条密钥链的链首，暂记为$h^n(P_U')_1$，即$h^n(P_U')_1 = c_1c_2\cdots c_{m_L}$。

4) CRT 验证法：对于$t\,(t = q_L)$对不同$\left(x_{c_k+1}, I_{c_k+1}\right)$，使用 CRT，可得到同余方程组，

$$\begin{cases} x \equiv I_{c_1+1} \bmod x_{c_1+1} \\ x \equiv I_{c_2+1} \bmod x_{c_2+1} \\ \quad\vdots \\ x \equiv I_{c_t+1} \bmod x_{c_t+1} \end{cases} \tag{4.3}$$

模$x_{c_1+1}\cdot x_{c_2+1}\cdots x_{c_t+1}$的唯一解，即新生的第二条密钥链的链首，暂记为$h^n(P_U')_2$。

5) 如果$h^n(P_U')_1 = h^n(P_U')_2$，则表示重组成功。这样我们就能得到新链的链首值$h^n(P_U') = h^n(P_U')_1 = h^n(P_U')_2$。

4.4.5 自更新阶段

重组完之后，下一条链开始工作，另一条新链也生成了，它也有一对实例S_U''和P_U''，以及相应的密钥链链首值$h^n(P_U'')$。再重复迭代以上过程，就能保证 Hash 链连续不断地工作。

4.5 本章小结

本章首先介绍了一些基本的定义，例如，分裂、重复度、重复率、难度等，为之后的(m, q)划分做铺垫。随后就引入了划分、划分树、改进的划分树的概念。

然后，文章又介绍了中国剩余定理、(*t*, *n*)-Mignotte's 序列、(t, n)-Mignotte's 门限秘密共享方案。

在介绍完以上相关知识后，本章从五个部分来详细地描述 SRHC-TD 构造方案的具体步骤：密钥初始化阶段、密钥发送阶段、密钥认证阶段、密钥重组阶段、自更新阶段。

第五章 基于 *d*-Left 可计数型布隆过滤器的身份认证机制

（AdlCBF）

5.1 基本算法

在基于 *d*-left 可计数型布隆过滤器的身份认证机制 (Authentication scheme base *d*-left Counting Bloom Filter, AdlCBF) 中，我们使用 ECDSA 来计算签名 $Sign(A, KS_A)=[r,s]$。签名的生成算法如算法 5.1 所示，而签名的验证算法如算法 5.2 所示。

算法 5.1 (签名生成算法) [23]

INPUT: Authentication information to be signed, A and private key of the source KS_A

OUTPUT: Signature pair $[r,s]$

1: Calculate the hash of the authentication information $h(A)$.

2: Select an integer k randomly from $[1,n-1]$, where n is a big prime number.

3: Calculate $(x_1,y_1)=kG$, where G is a base point of prime order on the elliptic curve;
And set $r=x_1 \bmod n$ ($r\neq 0$, if $r=0\Rightarrow$ repeat 2nd step).

4: Calculate $s=k^{-1}(g+KS_A\cdot r)\bmod n$, where g is the L_n leftmost bits of $h(A)$ ($s\neq 0$, if $s=0\Rightarrow$ repeat 2nd step)

5: Signature pair is $[r,s]$.

算法 5.2 (签名验证算法) [23]

INPUT: $[r,s]$, public key of the source KP_A and authentication information A

OUTPUT: true or false

1: Verify that $[r,s]$ are integers in the interval $[1,n-1]$.

2: Calculate hash for received authentication information and get $h(A)$.

3: Calculate $v=s^{-1}\bmod n$.

4: Calculate $u_1=h(A)\cdot v(\bmod n)$ and $u_2=r\cdot v \bmod n$.

5: Calculate $(x_1,y_1)=u_1\cdot G+u_2\cdot KP_A$.

6: $x_1=r \bmod n=0\Rightarrow$ true, otherwise false.

5.2 构造 *d*-Left 可计数布隆过滤器

在 DH-μTESLA 协议中, 每个传感节点 S_i（$i=1,2,\ldots$）都拥有一个公钥 KP_i 和一个私钥 KS_i。并且它们的公私钥都是由基站 S_B 分配的。并且基站需要构建一个 dlCBF 来存储密钥信息的指纹(fingerprint)，而不是直接分配内存空间来直接存储密钥的信息。

构建一个 dlCBF 总共有三个阶段，如图 5.1 所示。

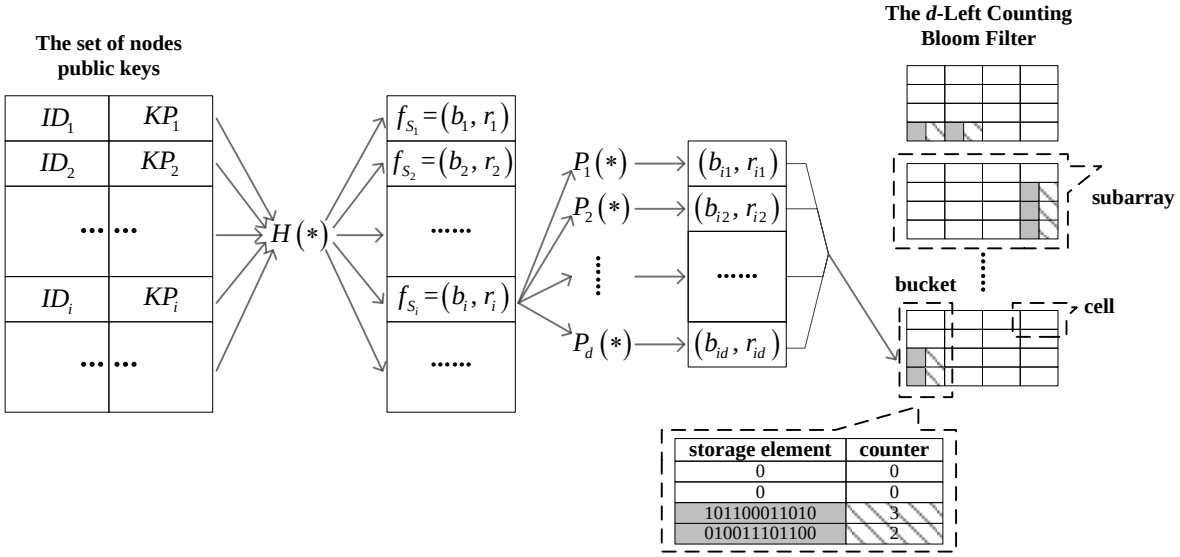

图 5.1　*d*-left 可计数型布隆过滤器的构造过程

首先，基站需要应用一个 Hash 函数 $H(\cdot)$ 来将每个节点的 ID 信息及其公钥对，即 $\langle ID_i, KP_i \rangle$，映射为一真指纹 (True Fingerprint)，表示为 $f_{S_i} = H(ID_i \| KP_i) = (b_i, r_i)$。这个真指纹包含两部分信息。其中，第一部分 b_i 是指桶坐标 (Bucket Index)；而后一部分 r_i (Remainder)，指代的是需要存储的元素，并且这个元素应当存储在桶坐标 b_i 所指代的那个队列 (Array) 的位置。

Remark. 考虑到系统的负载均衡，需要将存储的位置从一个队列扩展到 d 个子队列 (Subarray)。因此，原来的真指纹 f_{S_i} 也相应地需要扩展，变成 d 个指纹。

其次，对于每一个指纹 f_{S_i} 来说，基站应用另外的 d 个伪随机置换函数 $P_1, P_2, \cdots, P_d$ 来计算它的变换值。其中，$P_1(f_{S_i}) = (b_{i1}, r_{i1})$，$P_2(f_{S_i}) = (b_{i2}, r_{i2})$，…，$P_d(f_{S_i}) = (b_{id}, r_{id})$。它们分别对应于 d 个子队列。并且对于每个 $P_j(f_{S_i}) = (b_{ij}, r_{ij})$，$b_{ij}$ 是桶坐标，而 r_{ij} 是需要存储在第 j（$j=1,2,\cdots,d$）个子队列的

元素。

Remark. 有两种碰撞的情况会发生。第一种是两个不同的置换值对应相同的子队列的位置，即 $P_1\left(f_{S_i}\right)=\left(b_{i1},r_{i1}\right)\neq P_1\left(f_{S_j}\right)=\left(b_{j1},r_{j1}\right)$ 但是 $b_{i1}=b_{j1}$。处理这种情况的话，则需要每个子队列的每个桶含有多个单元 (Cell) 来存储不同的元素。另外一种碰撞是两个不同的真指纹具有相同的置换值，即 $f_{S_i}\neq f_{S_j}$ 但是 $P_d\left(f_{S_i}\right)=P_d\left(f_{S_j}\right)$。处理这种情况的话，则需要每个单元不但存储元素值，而且还需要一个计数器来记录存储相同元素的次数。因此，一个桶的存储负载是它里面所有单元的计数器（记录的数）的和。

最后，根据 *d* 个子队列的存储负载，基站选择最左边的、负载最轻的那个作为最后的存储位置。

5.3 新节点的身份认证过程

当一个新的传感节点申请加入到网络时，它首先需要被身份认证。因为只有合法的节点才能获得密钥链的链首等配置信息。考虑到新节点会在整个网络的生命周期内加入到网络中，因此身份认证可能在网络初始化阶段之前也可能在其之后。换句话说，有两种新节点，第一种能够直接与基站进行通信，另一种距离基站较远，即不能与之直接通信。因此，这两种不同的情况都需要分别进行考虑分析。在前一种情况下，基站利用算法 5.3 直接认证新节点的身份。而在后一种情况下，基站使用算法 5.4 帮助新节点以及它最近的簇头节点（也即是改节点需要加入的簇的簇头）进行互相间的身份认证。

算法 5.3 (直接认证新节点)

1: **for** **each** new sensor node S_i $(i=1, 2, \ldots)$, who wants to join the network, **do:**

2: sends authentication message M_a to base station. $M_a=(\ 'a', ID_B, ID_i, KP_i, N_i)$,where $'a'$ is the message type, ID_B is the destination address, ID_i is the source address, N_i is the nonce, and KP_i is the public key of S_i;

3: base station uses the hash function H and the permutations P_1, …, P_d to calculate d bucket indexes and the corresponding fingerprints of $\langle ID_i, KP_i\rangle$; And then it compares the calculated fingerprints and the ones stored in the position the indexes pointing at.

4: **if** there is one pair of fingerprints are the same **do:**

5: base station feedbacks the authentication-response message M_{ar},

$M_{ar}=('ar', ID_i, ID_B, A_1, Sign(A_1, KS_B))$,where $A_1=\left\{T_c\left\|\left(h^n\left(P_U\right)_c, \xi_{0c}, \cdots\right)\right\|T_{sc}\left\|T_{i'}\right\|T_{\text{int}}\left\|\delta\right\|N_B\right\}_{KP_i}$, $'ar'$ is the message type, $ID_{o_1}\left|ID_{o_2}\right|\cdots$ is the destination address, ID_B is the source address, T_c is the current time, $\left(h^n\left(P_U\right)_c, \xi_{0c}, \cdots\right)$ are the commitment and other related parameters of the current hash chain, T_{sc} is the start time of the current time interval , $T_{i'}$ is the start time of interval i', T_{int} is the duration of each time interval, δ is the disclosure delay, N_B is the

nonce, and KS_B is the private key of base station.

6: **else**

7: the node S_i is not the legal one and authentication fails.

8: **end if**

9: **end for**

算法 5.4 (新节点与簇头之间的相互认证)

1: **for** **each** new sensor node S_i $(i=1, 2, \ldots)$, who wants to join the network, **do:**

2: sends authentication message M_a to base station.

$M_a=(\ 'a', ID_B, ID_i, KP_i, N_i)$,where $'a'$ is the message type, ID_B is the destination address, ID_i is the source address, N_i is the nonce, and KP_i is the public key of S_i;

3: when cluster head S_j receives the authentication message of S_i sending to base station, it adds its public key KP_j into the packet, and then forwards the added authentication message M_{aa} to base station.

$M_{aa}=(\ 'aa', ID_B, ID_j, KP_j, N_j, (ID_B, ID_i, KP_i, N_i))$,where $'aa'$ is the message type, ID_B is the destination address, ID_j is the source address, N_j is the nonce, and KP_j is the public key of S_j;

4: base station uses the hash function H and the permutations $P_1, \ldots, P_d$ to calculate d bucket indexes and the corresponding fingerprints of $\langle ID_i, KP_i \rangle$ and $\langle ID_j, KP_j \rangle$; And then it compares the calculated fingerprints and the ones stored in the position the indexes pointing at.

5: **if** there is one pair of fingerprints of $\langle ID_i, KP_i \rangle$ are the same **do:**

6: **if** there is one pair of fingerprints of $\langle ID_j, KP_j \rangle$ are the same **do:**

7: base station feedbacks the authentication-response message M_{ar} and the added-authentication-response message M_{aar},

$M_{ar}=('ar', ID_i, ID_B, A_2, Sign(A_2, KS_B))$, $M_{aar}=('aar', ID_j, ID_B, A_3, Sign(A_3, KS_B))$,

where $A_2=\{'S_j\ is\ legal' \| T_c \| (h^n(P_U)_c, \xi_{0c}, \cdots) \| T_{i'} \| T_{\text{int}} \| \delta \| N_B\}_{KP_i}$, $A_3=\{'S_i\ is\ legal' \| N_B\}_{KP_j}$, $'ar'$ and $'aar'$ are the message type, ID_i and ID_j are the destination addresses, ID_B is the source address, $'S_j\ is\ legal'$ and $'S_i\ is\ legal'$ are the authentication results, and the meanings of other unspecified parameters are the same as those in Algorithm 5.3.

8: **else**

9: base station feedbacks the authentication-response message M_{ar},

$M_{ar}=('ar', ID_i, ID_B, A_4, Sign(A_4, KS_B))$,

where $A_4=\{'S_j\ is\ illegal' \| T_c \| (h^n(P_U)_c, \xi_{0c}, \cdots) \| T_{i'} \| T_{\text{int}} \| \delta \| N_B\}_{KP_i}$.

10: **end if**

11: **else**

12: **if** there is one pair of fingerprints of $\langle ID_j, KP_j \rangle$ are the same, **do:**

13: base station feedbacks the added-authentication-response message M_{aar},

$M_{aar}=('aar', ID_j, ID_B, A_5, Sign(A_5, KS_B))$, where $A_5=\{'S_i\ is\ illegal' \| N_B\}_{KP_j}$.

14: **else**

15: the node S_i and S_j are not the legal ones, and authentication fails.

16: **end if**

17: **end if**

18: **end for**

但算法 5.3 和算法 5.4 只能描述单个节点的身份认证过程。当大量的新节点加入到网络中时，簇头可收集这些申请信息，并生成一个认证信息 M_{aa}，然后将其发送给基站。再由基站批处理这些申请信息。因此，以上的算法同样能够适用于多个身份认证同时发生的情况。

5.4 本章小节

本章首先介绍椭圆曲线签名算法 ECDSA 的签名生成算法以及签名验证算法。然后介绍 *d*-left 可计数布隆过滤器的构造过程。接着利用构造好的布隆过滤器描述新节点加入的验证过程。而这一过程包含两种情况，一是基站的直接认证，二是簇头与节点之间的相互认证；它们分别用算法 5.3 和 5.4 来描述。

第六章 基于父亲树的分簇路由算法（PTCR）

基于父亲树的路由算法(Parent-Tree based Clustering Routing algorithm, PTCR)主要描述了这样一个循环迭代的过程。它不断地建立簇，使得节点加入到簇内，选择下一级的簇头，直到所有的或者绝大部分的节点都加入到了某个簇为止。关于 PTCR 算法的基本思想如图 6.1 所示。

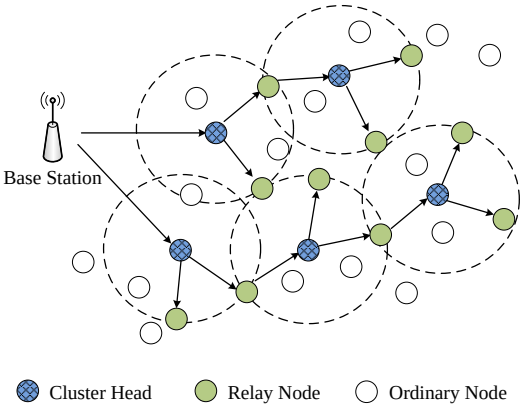

图 6. 1 PTCR 算法建立簇的过程

6.1 基本定义

定义 6.1 (簇头数) 假设节点 i 有 n 个邻居节点，那么它的下一级的簇头数目为 $num(n)$。并且，

$$num(n)=\begin{cases}1 & 0<n\le 5\\ 2 & 5<n\le 10\\ \left\lfloor \frac{n}{10} \right\rfloor+2 & 10<n<90\\ 10 & n\ge 90\end{cases} \tag{6.1}$$

定义 6.2 (节点的综合权值及阈值) 我们定义节点的最大发射功率为 $P_{\max}$，最大能量为 $E_{\max}$，通信半径为 d_o，功率能量系数为 $k_{pe}\left(k_{pe}<1\right)$，距离功率能量系数为 $k_{dpe}\left(k_{dpe}<1\right)$，阈值系数为 $k_T\left(k_T<1\right)$。假设节点 i 的邻居节点为节点 j，且它的发射功率为 P_j，能量为 E_j，与节点 i 的距离为 d_{ij}，距离功率能量的权重为 W_{dep_j}，距离功率能量阈值为 T_{dep_j}。那么，W_{dep_j} 和 T_{dep_j} 可表达为以下形式：

$$W_{dep_j}=k_{dpe}\cdot\left(k_{pe}\cdot E_j+\left(1-k_{pe}\right)\cdot P_j\right)+\left(1-k_{dpe}\right)\cdot d_{ij} \tag{6.2}$$

$$T_{dep_j} = k_T \cdot \left[k_{dpe} \cdot \left(k_{pe} \cdot E_{max} + \left(1 - k_{pe}\right) \cdot P_{max} \right) + \left(1 - k_{dpe}\right) \cdot d_o \right] \tag{6.3}$$

6.2 分簇路由过程

由于 PTCR 算法是个自组织的分布式最优化算法，基站能够自动地发起分簇的操作。分簇路由算法主要有四个步骤：建立邻居、选择簇头、新节点加入、簇头更新。其中，建立邻居和选择簇头是循环迭代地进行的，类似于一个从基站开始的宽度优先算法的过程。而新节点的加入可能在网络运行的任何时刻都会发生，属于触发式的一个过程。至于簇头更新则是在网络分簇结束后，随着网络的运行，网络能量消耗得不均匀，从而采取的一个均衡能量分布的一个手段。

6.2.1 建立邻居

首先，作为一个特殊的簇头，基站按照算法 6.1 开始建立邻居。

算法 6.1 (建立邻居)

1: **for** **each** cluster head node (base station involved) S_i $(i = 1, 2, \ldots)$ **do:**
2: broadcasts building-neighbour message M_{bn},
$M_{bn} = ('bn', 0xff, ID_i)$, where $'bn'$ is the message type, 0xff is the destination address, and ID_i is the source address;
3: **for** **each** receiving node S_j $(j = 1, 2, \ldots)$ **do:**
4: **if** $S_j.Energy > 0$ and $S_j.Type = 'Normal'$ **do:**
5: feedbacks the building-neighbour-response message M_{bnr},
$M_{bnr} = ('bnr', ID_i, ID_j, S_j.Energy)$, where $'bnr'$ is the message type, ID_i is the destination address, ID_j is the source address, and $S_j.Energy$ is the message content about energy information;
6: records ID_i as $S_j.Cluster = ID_i$;
7: **end if**
8: **end for**
9: according to the receiving order, S_i stores the neighbour messages into neighbour-list;
10: **end for**

6.2.2 选择簇头

建立完邻居之后，基站存储了一个邻居列表 (Neighbor List) 用以记录邻居反馈信息的接收时间以及邻居的能量信息。然后，基站根据算法 6.2 和算法 6.3 开始选择簇头。

算法 6.2 (选择簇头 I)

for **each** cluster head node (base station involved) S_i $(i=1, 2, \ldots)$ **do:**
  selects next-level cluster heads according to Algorithm 6.3;
  S_i broadcasts building-cluster-head message M_{bch},
  $M_{bch}=\left('bch', 0xff, ID_i, ID_{c_1}\left|ID_{c_2}\right|\cdots ID_{c_{num(n)}}, CL\right)$, where $'bnr'$ is the message type, ID_i is the source address, $ID_{c_1}\left|ID_{c_2}\right|\cdots ID_{c_{num(n)}}$ is the set of the selected cluster heads, CL is an integer representing the cluster level;
  for **each** selected cluster head S_{c_j} $(j=1, 2, \ldots, num(n))$ **do:**
    alters its type as $S_{c_j}.Type='ClusterHead'$ and set cluster-level as $S_{c_j}.ClusterLevel=CL$;
  end for
end for

算法 6.3 (选择簇头 II)

for **each** node S_i $(i=1, 2, \ldots)$ **do**:
  traverses its neighbour-list, calculates all neighbours' distance-power-energy weight W_{dpe};
  selects $2\times num(n)$ nodes as candidates in descending order of W_{dpe} ;
  sends the candidate-set message M_{cs} to each candidate,
  $M_{cs}=\left('cs', ID_j, ID_i, ID_{c_1}\left|ID_{c_2}\right|\cdots ID_{c_{2\times num(n)}}\right)$, where $'cs'$ is the message type, ID_j is the destination address, ID_i is the source address, $ID_{c_1}\left|ID_{c_2}\right|\cdots ID_{c_{2\times num(n)}}$ is the set of candidates, and $ID_j\in ID_{c_1}\left|ID_{c_2}\right|\cdots ID_{c_{2\times num(n)}}$;
  for **each** candidate **do:**
    builds neighbors according to Algorithm 6.1;
    and then sends the out-of-neighbour-list message M_{ofnl} to S_i,
    $M_{ofnl}=\left('ofnl', ID_i, ID_j, ID_{o_1}\left|ID_{o_2}\right|\cdots\right)$, where $'ofnl'$ is the message type, ID_i is the destination address, ID_j is the source address, and $ID_{o_1}\left|ID_{o_2}\right|\cdots$ is the set of candidate that are nor reachable to S_i.
  end for
  suppose initial value $j=1$;
  while the size of the cluster head set G_{c_i} is less than $num(n)$, **do:**
    if $j\le 2\times num(n)$, **do:**
      S_i picks up the j-th candidate S_{ij};
      if S_{ij}'s distance-power-energy weight $W_{dpe_j}>T_{dpe_j}$, **do:**
        if $j\neq 1$ **do:**
          traverses set G_{c_i};
          if node S_{ij} is totally disconnected with all nodes in set G_{c_i}, **do:**
            S_{ij} joins set G_{c_i}; the size of G_{c_i} adds one;
          end if
        else
          S_{ij} joins set G_{c_i}; the size of G_{c_i} adds one;
        end if
      end if

```
                    j = j + 1;
                else
                    suppose initial value k = 1;
                    node S_i picks up the k-th candidate S_ik;
                    if S_ik doesn't belong to set G_ci, do:
                        S_ik joins set G_ci; the size of G_ci adds one;
                    end if
                    k = k + 1;
                end if
            end while
        end for
```

根据算法 6.1, 6.2, 6.3，基站选择了 $num(n)$ 个第一级簇头（源簇头）。类似地，这些簇头也开始建立邻居，然后选择下一级的簇头。这样以此类推，接下来的操作就是不断地从高级别（数字越小，级别越大）的簇头的邻居中选择低级别的簇头，直到整个网络都已经分完簇为止。此时，网络中的所有的或者说绝大多数的节点都加入到了某个簇中。

6.2.3 新节点的加入

不可避免的是，每时每刻都可能会有新节点加入到网络中。只有通过算法 5.3 和算法 5.4 身份认证过的合法节点才能加入到网络中。在那之后，新的节点使用算法 6.4 来加入到附近的簇。

算法 6.4 (新节点的加入)

```
for each new node S_i, (i = 1, 2, ...) do:
    broadcasts finding-cluster message M_fc,
    M_fc = ('fc', 0xff, ID_i), where 'fc' is the message type, 0xff is the destination address, and ID_i is the source address;
    if it has no reply do:
        S_i would lose effectiveness;
    else if it receives one or more responses from neighbouring cluster heads M_rfc = ('rfc', ID_i, ID_j), where 'rfc' is the message type, ID_i is the destination address, and ID_j is the source address, do:
        joins the closest cluster with cluster head S_j, and sends the joining-cluster message M_jc, M_jc = ('jc', ID_j, ID_i, S_i.Energy), where 'jc' is the message type, ID_j is the destination address, ID_i is the source address, and S_i.Energy is the remainder energy of S_i;
        records ID_j as S_i.Cluster = ID_j;
        cluster head S_j then updates its neighbour-list;
    else if it receives one or more responses from neighbouring normal nodes, do:
        chooses the closest neighbour S_k as its parent-node which is responsible for forwarding messages; and sends the treating-as-parent message M_tap,
```

$M_{tap}=('tap', ID_k, ID_i, S_i.Energy)$, where $'tap'$ is the message type, ID_k is the destination address, ID_i is the source address, and $S_i.Energy$ is the remainder energy of S_i;

records ID_j as $S_i.Cluster = ID_k$;

node S_k then alters its type as $S_k.Type = 'ClusterHead'$ and set cluster-level as $S_k.ClusterLevel = S(S_i.Cluster).ClusterLevel + 1$;

end if

end for

6.2.4 簇头的更新

到现在为止，整个 PTCR 算法就将近结束了。然而，考虑到簇头节点需要频繁地处理、转发数据，它们消耗能量更快。因此，有必要设立一个能量阈值 *E*，当它们的能量低于阈值时，簇头节点使用算法 6.5 发起重新选择簇头的操作。

算法 6.5 (簇头的更新)

for **each** cluster head node S_i $(i=1, 2, \ldots)$ whose energy drops the threshold, **do:**

sends detecting-connectivity message M_{dc} to its upper-level cluster head S_j,

$M_{dc}=('dc', ID_j, ID_i, ID_{n_1}|ID_{n_2}|\cdots)$, where $'dc'$ is the message type, $ID_j = S_i.Cluster$, and $ID_{n_1}|ID_{n_2}|\cdots$ is the set of S_i's neighbors;

cluster head S_j detects the connectivity with S_i's neighbors referring to Algorithm 6.1;

if one or more of S_i's neighbors are reachable to S_j, **do:**

for **each** neighbor node S_k $(k=1, 2, \ldots)$, **do:**

S_k detects the connectivity between itself with S_i's other neighbors;

the one which is reachable to more than half of other neighbors is chosen as the candidate;

end for

if there is one or more candidates, **do:**

S_i chooses the one with most residual energy as the new cluster head; and broadcasts updating-cluster-head message M_{uch},

$M_{uch}=('uch', 0xff, ID_i, ID_{nch})$, where $'uch'$ is the message type, ID_{nch} is the new cluster head's ID;

the new cluster head S_{nch} builds its neighbor-list according to Algorithm 6.1;

the nodes that are unreachable to the new cluster head would join into other clusters referring to Algorithm 6.4;

end if

else

S_i is still considered as the cluster head;

when the power of S_i is exhausted, S_i's neighbors join into the other cluster referring to Algorithm 6.4;

end if

end for

为了平衡所有传感节点的能量分布，当簇头节点的更新操作过于频繁时，基站需要对整个网络实施重新选择簇头的工作。

6.3 本章小节

本章主要以算法伪代码的形式介绍基于父亲树的分簇路由算法 PTCR。首先给出了算法中所需要用到的一些简单定义，然后再介绍 PTCR 的四个主要步骤：建立邻居、选择簇头、新节点加入、簇头更新。这四个步骤每一个都有与之对应的算法描述。

第七章 虚拟势场下移动传感器网络的入侵检测

7.1 准备工作及问题阐述

7.1.1 感知模型

为了方便数学建模处理，本文中我们采用基于圆盘的感知模型。每个节点只能在半径为R的感知范围内感知环境、检测事件。我们使用大写字母S，既表示一个传感节点又表示该节点所处位置的那一点。而对于每个入侵者，把它作为一个点来处理。如果它进去了某个节点的感知范围，那么就表示它被覆盖了或者被检测了。我们使用大写字母I，既表示一个入侵者又表示该入侵者所处位置的那一点。

7.1.2 网络模型

我们考虑这样一个移动传感网络，它由$N(Z)$个移动节点所组成的，这些移动节点在初始化阶段被部署在一个矩形的带状区域Z内，该区域长而窄，如图 7.1 所示。

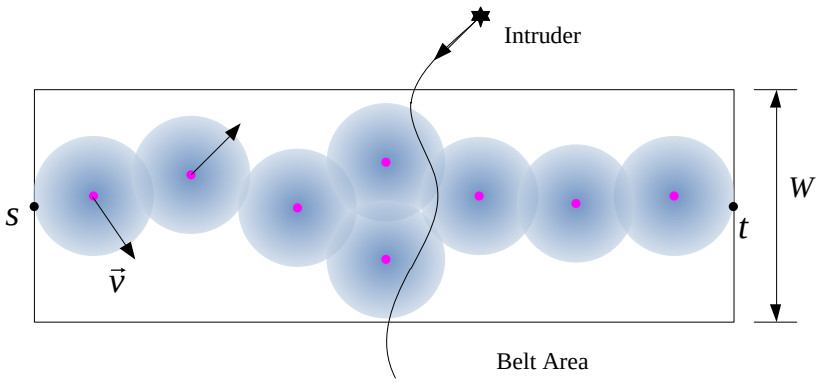

图 7.1 移动传感网中的入侵检测问题

初始化时，我们假设所有的节点的分布都是独立地、随机地、正态地。这样，节点的位置信息就可以用一个静态的二维泊松过程来建模。用$N(Z)$表示区域Z内的所有节点数，我们可以得到：

$$\Pr\left(N\left(Z\right)=k\right)=\frac{e^{-n_Z|Z|}\left(n_Z\left|Z\right|\right)^k}{k!} \tag{7.1}$$

其中，$\Pr\left(Y\right)$表示事件Y发生的概率；n_Z表示传感器的密度；$\left|Z\right|$表示区域Z的面积。

7.1.3 问题阐述

入侵曲线表示入侵者在移动传感网覆盖区域内的移动曲线，即从覆盖区域的一条平行边到另一条之间的这段曲线。在带状区域内，入侵曲线是这样一段穿越路径，它能够完全穿越带状区域的宽。而只有当任意一个入侵者沿着其入侵曲线被至少连续的k个移动节点所检测时，这样一个移动传感网才能被称作k栅栏覆盖。并且用$\Pr\left(\Lambda\geq k\right)$表示移动传感网中$k$栅栏覆盖的概率。文献[38]已证明，

$$\Pr(\Lambda\geq k)=1-\frac{\Gamma\left(k,\Theta_v\cdot\tau\right)}{\Gamma\left(k\right)} \tag{7.2}$$

其中，Λ表示任意一条穿越路径连续被Λ个移动节点所覆盖（或检测）；$\Gamma\left(k,\Theta_v\cdot\tau\right)$和$\Gamma\left(k\right)$分别是不完全欧拉伽马函数以及完全欧拉伽马函数；$\Theta_v$是单位时间内被移动节点覆盖的个数，称作覆盖率；$\tau$是单位时间。

为了求出k栅栏覆盖的概率$\Pr\left(\Lambda\geq k\right)$，我们首先需要利用下面的模型来求出$\Theta_v$。

7.2 移动模型

绪论部分已经提到，节点和入侵者的运动受到虚拟势场的作用。所以，这部分首先会描述虚拟势场的概念。其次，用完全弹性碰撞的模型来描述节点的运动情况。最后，用点电荷的模型来描述入侵者的运动情况。

7.2.1 虚拟势场

通过传统的虚拟势场的概念，我们知道，虚拟力通常存在于相邻的两个传感节点之间。本文中，我们扩展虚拟势场的概念，它不仅存在传感节点之间而且存在传感节点与入侵者之间。

如图 7.2 所示，虚拟斥力$\vec{F}_{ij}$和$\vec{F}_{ji}$是一对作用力与反作用力，它们大小相等，方向相反。而且，它们作用在有重叠感知区域的节点S_i和S_j的质心（本文中即指圆心）位置。

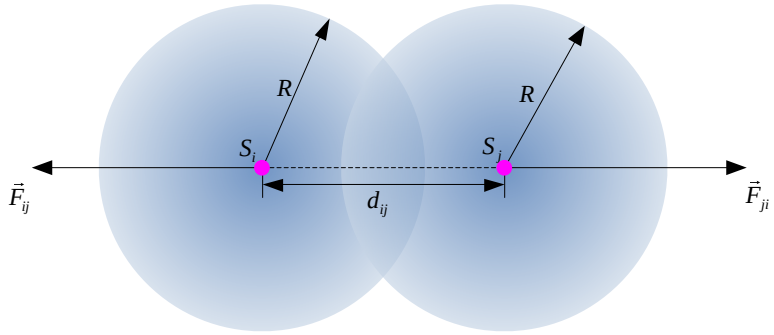

图 7.2 两个传感节点之间的斥力

如图 7.3 所示，当入侵者进入到感知区域时，虚拟斥力开始起作用，并且它们分别作用在传感节点的质心和入侵节点的质心位置。

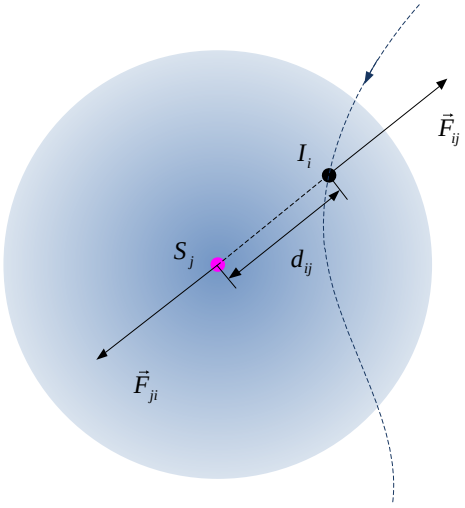

图 7.3 入侵者与传感节点之间的斥力

在物理学中，斥力/引力通常与两物体之间距离的平方成反比。例如，库仑力和万有引力。同理，我们可以得到虚拟斥力的表达式，

$$\left|\vec{F}_{ij}\right| = \left|\vec{F}_{ji}\right| = k_v \frac{1}{d_{ij}^{\,2}} \tag{7.3}$$

其中，d_{ij} 表示两个实体之间的距离；k_v 表示虚拟常量。

7.2.2 完全弹性碰撞模型

为了增加覆盖率，感知区域应该尽量避免出现重叠区域。当两个感知区域有重叠的时候，虚拟斥力将他们相互推开。这个过程很像是物理学中的碰撞模型。不考虑能量的损耗，因此，我们使用完全弹性碰撞模型来描述移动传感节点的运动。

1) 单个完全弹性碰撞模型

碰撞前，分别用v_{i_0}和v_{j_0}来表示传感节点S_i和S_j的速度。为了易于数学处理，我们将S_j作为参照物，得出碰撞前的相对速度v_{ij_0}和v_{jj_0}。其中，$v_{ij_0}=\left|\vec{v}_{i_0}-\vec{v}_{j_0}\right|$，$v_{jj_0}=0$。由于碰撞的瞬间传感节点只受到虚拟斥力作用，没有其他外力。所以，两个传感节点遵守动量守恒定律。碰撞后，分别用v_{ij}和v_{ji}表示传感节点S_i和S_j的相对速度。这样，碰撞后的绝对速度$v_i=\left|\vec{v}_{ij}+\vec{v}_{j_0}\right|$，$v_j=\left|\vec{v}_{ji}+\vec{v}_{j_0}\right|$。

图 7.4 和图 7.5 分别是两个传感节点完全弹性碰撞前后的示意图。图中，碰撞的方向称为法向方向，用$\vec{n}$表示，$\vec{n}$的垂直方向称为切线方向，用$\vec{\tau}$表示。

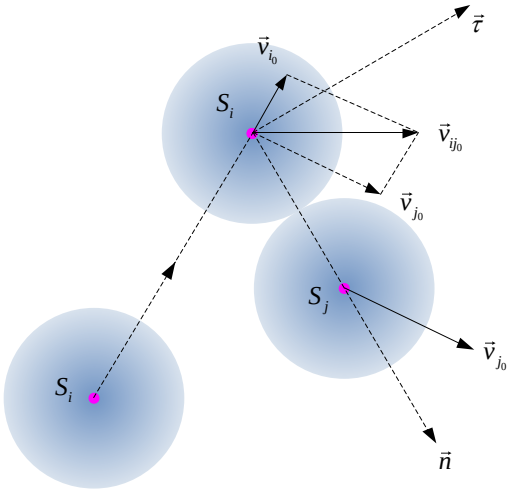

图 7.4 两个传感节点碰撞前的瞬间

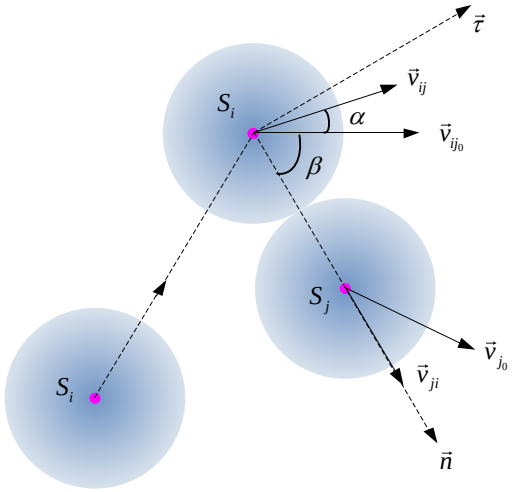

图 7.5 两个传感节点碰撞后的瞬间

由于是完全弹性碰撞，动量守恒并且恢复系数等于 1（即$e=1$）。

$$\begin{cases} m_i v_{ij}\sin(\alpha+\beta)=m_i v_{ij_0}\sin\beta \\ m_i v_{ij}\cos(\alpha+\beta)+m_j v_{ji}=m_i v_{ij_0}\cos\beta \\ m_i v_{ij}\sin\alpha=m_j v_{ji}\sin\beta \\ v_{ij}\cos(\alpha+\beta)-v_{ji}=-v_{ij_0}\cos\beta \end{cases} \tag{7.4}$$

因此，我们能够得到，

$$v_{ij}=v_{ij_0}\sqrt{1-\frac{4m_i m_j}{(m_i+m_j)^2}\cos^2\beta} \tag{7.5}$$

$$v_{ji}=2\frac{m_i}{m_i+m_j}v_{ij_0}\cos\beta \tag{7.6}$$

$$\alpha = \arcsin\left(\frac{m_j}{\sqrt{m_i^{\,2} + m_j^{\,2} - 2m_i m_j \cos 2\beta}} \sin 2\beta\right) \tag{7.7}$$

其中，m_i 和 m_j 表示两个传感节点的质量；v_{ij_0} 表示碰撞前节点 S_i 相对于节点 S_j 的速度；α 是 $\vec{v}_{ij}$ 和 $\vec{v}_{ij_0}$ 之间的角度；β 是 $\vec{v}_{ji}$ 和 $\vec{v}_{ij_0}$ 之间的角度。

由于所有的传感节点都具有相同的质量。显然，我们可以得到，$v_{ij} = v_{ij_0} \cdot \sin\beta$，$v_{ji} = v_{ij_0} \cdot \cos\beta$，$\alpha = \frac{\pi}{2} - \beta$。

这样，节点 S_i 和 S_j 碰撞之后，它们交换法向速度，保持切向速度不变。

2) 多个弹性碰撞模型

对于单个传感节点，它可能会被一个甚至多个邻近节点推开。当多个弹性碰撞同时发生在节点 S_j 和它的邻居之间时，我们通过将不同方向的法向速度进行叠加就能得出节点 S_j 碰撞后的相对速度。多个弹性碰撞如图 7.6 所示。

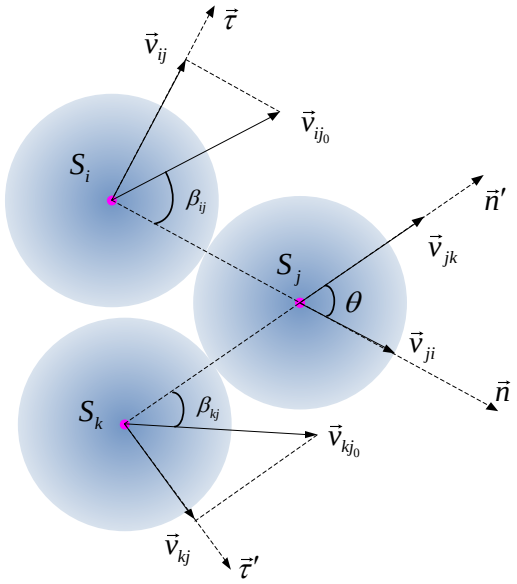

图 7.6 多个传感节点之间的完全弹性碰撞

这样，我们能够得到节点 S_j 碰撞后的绝对速度，

$$\vec{v}_j = \sum_{n \in U} \vec{v}_{jn} + \vec{v}_{j_0} \tag{7.8}$$

其中，v_{j_0} 是节点 S_j 的初始速度；U 表示那些与 S_j 碰撞的邻居节点的集合；v_{jn} 表示 S_j 与 S_n 碰撞后的

相对速度大小。

7.2.3 点电荷模型

为了简化处理，我们假设入侵者的速度大小不变，方向可以不断变化；而传感节点的速度并不会受到它与入侵者之间的虚拟力的作用。这样，当一个入侵者进入到传感节点的感知区域的时候，虚拟斥力会将它推离出去。这很符合入侵者不愿被检测所以尽量避免进入感知范围的实际情况。这一过程与物理学中点电荷模型非常类似。因此，我们使用此模型来描述入侵者的运动情况。入侵者跨越单个传感节点的感知范围的情况如图 7.7 所示。

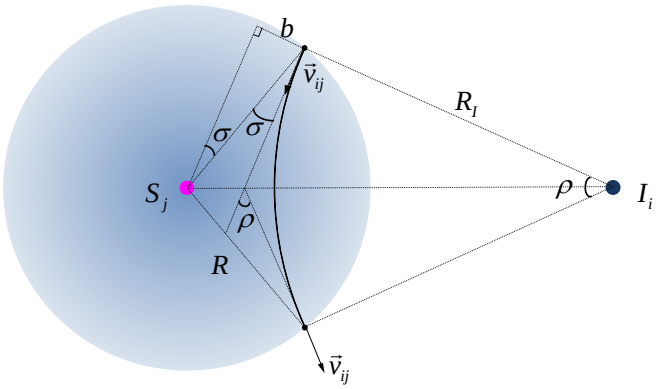

图 7.7 入侵者跨越一个传感节点的感知范围

在此过程中，由于虚拟斥力是内力，入侵者和传感节点整体角动量守恒。由于之前假设入侵者的速度大小是恒定的，所以动能守恒并且入侵轨迹是一个圆弧。

$$\begin{cases} F_{ij}\cdot\vec{e}_r = m_i\cdot\dfrac{d\vec{v}}{dt} \\ m_i\cdot {d_{ij}}^2\cdot\dfrac{d\sigma}{dt} = m_i v_{ij} b \\ \int d\vec{v} = 2v_{ij}\sin\dfrac{\rho}{2} \\ \int \vec{e}_r d\sigma = 2\cos\dfrac{\rho}{2} \end{cases} \tag{7.9}$$

其中，$\sin\sigma = \dfrac{b}{R}$ 并且 $\tan\dfrac{\rho}{2} = \dfrac{R\cos\sigma}{R_I + b}$。

因此，我们可以得到，

$$R_I = R\cos\sigma\cot\frac{\rho}{2} - b \tag{7.10}$$

$$b = \frac{k_v}{m_i v_{ij}^{\ 2}}\cot\frac{\rho}{2} \tag{7.11}$$

$$\rho = 2\,\mathrm{arc\,cot}\left(\frac{Rm_i v_{ij}^{\ 2}}{k_v}\sin\sigma\right) \tag{7.12}$$

其中，$\vec{e}_r$ 表示方向向量；R_I 是入侵者 I_i 在感知区域的轨迹（圆弧）的半径；ρ 是圆弧的中心角；σ 是初速度 $\vec{v}_{ij}$ 与感知半径之间的角度，称为入侵角，并且 $\sigma \in (0, \pi/2)$。

由公式(7.12)得，

$$\sigma = \arcsin\left(\frac{k_v}{Rm_i v_{ij}^{\ 2}}\cot\frac{\rho}{2}\right) \tag{7.13}$$

由于 $\sigma \in \left(0, \frac{\pi}{2}\right)$，所以 $\rho \in \bigcup_{a=0}^{\infty}\left(2\,\mathrm{arccot}\frac{Rm_i v_{ij}^{\ 2}}{k_v} + a\cdot\pi, \pi + a\cdot\pi\right)$。并且，需要满足 $\rho \in (0, \pi)$，所以 $\rho \in \left(2\,\mathrm{arccot}\frac{Rm_i v_{ij}^{\ 2}}{k_v}, \pi\right)$。

7.3 移动传感网中的入侵检测

在这一节我们将公式化移动传感网 MSN 中的入侵检测的问题。根据在第 7.2 节讨论的移动模型，我们需要先求出平均相对速度 $\bar{v}_{ij}$ 以及平均覆盖率 $\bar{\Theta}_v$。

7.3.1 平均邻居数目

由于弹性碰撞时，存在斥力作用，极限情况是两个传感区域是相切的。因此 S_j 的邻居节点位于一个以 S_j 为中心，半径为 $3R$ 的圆形区域内。如图 7.8 所示。如前所述，带状区域的密度为 n_Z，并且用 N_n 表示一个传感节点的邻居节点的数目。因此，我们得到

$$N_n = \left[9\pi R^2 n_Z - 1\right] \tag{7.14}$$

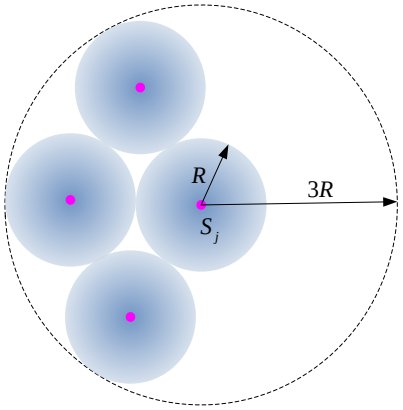

图 7.8 传感节点的邻居区域

7.3.2 平均相对速度

一般情况下，我们将S_j作为参照物，并且把它碰撞前的速度记为$\vec{v}_{j_0}$。并且，对于它的邻居节点而言，它们碰撞前的相对速度记为$\vec{v}_{1j_0},\vec{v}_{2j_0},\cdots,\vec{v}_{N_n j_0}$。因此，根据公式(7.6)，我们可以得到碰撞后$S_j$的相对速度为$\vec{v}_{j1},\vec{v}_{j2},\cdots,\vec{v}_{jN_n}$，其中$v_{jl}=v_{lj_0}\cdot\cos\beta_l$并且$l=1,2,\cdots,N_n$。

文献[38]中证明，速度$\vec{v}_1$和$\vec{v}_2$的平均相对速度为

$$\bar{v}_{12}=\frac{2}{\pi}\left(\bar{v}_1+\bar{v}_2\right)E_{12} \tag{7.15}$$

其中，$E_{12}=\int_0^{\pi/2}\sqrt{1-\frac{4\bar{v}_1\bar{v}_2}{\bar{v}_1{}^2+\bar{v}_2{}^2+2\bar{v}_1\bar{v}_2}\sin^2\varphi}d\varphi$。

因此，我们可以得到$\vec{v}_{j1},\vec{v}_{j2},\cdots,\vec{v}_{jN_n}$的平均相对速度值，为

$$\begin{cases}\bar{v}_{1to2}=\dfrac{2}{\pi}\left(\bar{v}_{j1}+\bar{v}_{j2}\right)E_{1to2}\\ \vdots\\ \bar{v}_{1toN_n}=\dfrac{2}{\pi}\left(\bar{v}_{1toN_n-1}+\bar{v}_{jN_n}\right)E_{1toN_n}\end{cases} \tag{7.16}$$

其中，$E_{1tol}=\int_0^{\pi/2}\sqrt{1-\frac{4\bar{v}_{1tol}\bar{v}_{ji+1}}{\bar{v}_{1tol}{}^2+\bar{v}_{jl+1}{}^2+2\bar{v}_{1tol}\bar{v}_{jl+1}}\sin^2\varphi}d\varphi$并且$l=2,3,\cdots,N_n$。

类似地，入侵者I_i相对节点S_j的平均速度可表示为

$$\bar{v}_{ij}=\frac{2}{\pi}\left(\bar{v}_i+\bar{v}_j\right)E_{ij} \tag{7.17}$$

其中，$\bar{v}_i$ 是入侵者 I_i 的平均速度；$\bar{v}_j=\frac{2}{\pi}\left(\bar{v}_{1toN_n}+\bar{v}_{j_0}\right)E_j$，并且 $\bar{v}_j$ 是碰撞后节点 S_j 的平均速度；$E_{ij}=\int_0^{\pi/2}\sqrt{1-\frac{4\bar{v}_i\bar{v}_j}{\bar{v}_i^{\,2}+\bar{v}_j^{\,2}+2\bar{v}_i\bar{v}_j}\sin^2\varphi}d\varphi$，且 $E_j=\int_0^{\pi/2}\sqrt{1-\frac{4\bar{v}_{1toN_n}\bar{v}_{j_0}}{\bar{v}_{1toN_n}^{\,2}+\bar{v}_{j_0}^{\,2}+2\bar{v}_{1toN_n}\bar{v}_{j_0}}\sin^2\varphi}d\varphi$。

7.3.3 平均覆盖率

经过一段时间 τ 之后，入侵者的入侵轨迹弧度的长度为 $v_{ij}\tau$。等价地，相当于入侵者旋转了 $v_{ij}\tau/R_I$ 的角度。如果入侵者旋转的角度为 ρ 度，这也就是说它跨过了一个面积为 πR^2 的圆（即一个传感节点的感知区域）。区域中节点的密度为 n_Z。因此，覆盖率 Θ_v 可以表示为

$$\Theta_v=\frac{n_Z v_{ij}\pi R^2}{R_I\rho} \tag{7.18}$$

由公式(7.10)，(7.11)和(7.12)，我们可得

$$\Theta_v=\frac{n_Z\pi R^2 m_i v_{ij}^{\,3}}{2\omega\cdot\operatorname{arccot}\omega\cdot\left(\sqrt{R^2m_i^2\bar{v}_{ij}^4-k_v^2\omega^2}-k_v\right)} \tag{7.19}$$

其中，$\omega=\frac{Rm_iv_{ij}^2\sin\sigma}{k_v}$。

因此，平均覆盖率 $\bar{\Theta}_v$ 可表示为

$$\bar{\Theta}_v=\frac{n_Z\pi R^2 m_i \bar{v}_{ij}^{\,3}}{2\bar{\omega}\cdot\operatorname{arccot}\bar{\omega}\cdot\left(\sqrt{R^2m_i^2\bar{v}_{ij}^4-k_v^2\bar{\omega}^2}-k_v\right)} \tag{7.20}$$

其中，$\bar{\omega}=\frac{1}{2\pi}\int_0^{\pi/2}\omega d\sigma=\frac{Rm_i\bar{v}_{ij}^2}{2\pi k_v}$。

7.3.4 移动传感网中的 k 栅栏覆盖

到目前为止，我们得出了平均覆盖率 $\bar{\Theta}_v$，平均相对速度 $\bar{v}_{ij}$。根据公式(7.2)，(7.17)和(7.20)，我们现在能够得出在移动传感网中 k 栅栏覆盖的概率 $\Pr(\Lambda \geq k)$。

7.4 本章小节

本章首先给出了传感节点的感知模型、网络模型，并给出了具体的问题描述。接着围绕着所需要求的问题，提出了虚拟势场下的弹性碰撞模型、点电荷模型。然后根据以上的移动模型再来求出平均相对速度以及平均覆盖率。这样就能求出在移动传感网中 k 栅栏覆盖的概率 $\Pr(\Lambda \geq k)$。

第八章 性能分析

在这一章，我们从四个方面来评估 DH-μTESLA 协议以及入侵检测的性能：

1) SRHC-TD 机制与 RHC[12], ERHC[13], SUHC[14], SRHC[15]在安全性以及复杂度上面的比较。

2) AdlCBF 机制与直接存储方法 (Direct Access Method) 在空间及时间效率上面的比较。以及 AdlCBF 与使用 CBF[20]的认证方案之间的比较；还包括 AdlCBF 的安全强度。

3) PTCR 算法与 LEACH[28], SEP[30], DEEC[31]在能耗、网络生存时间、以及簇头数目之间的比较。

4) 基于虚拟势场的移动传感网 k 栅栏覆盖与静态传感网络的 k 栅栏覆盖、以及一般移动传感网的 k 栅栏覆盖[10]的性能比较。

我们使用 Matlab R2009b 以及 Visual C++ 2010 来做本文的仿真工具。

8.1 SRHC-TD 性能评估

8.1.1 安全性

1) 考虑重复情况的明文空间比较

Hash 链有一个特点：链上的每一个值（除了初始种子值以及已公布的链首值）既是散列计算的输出 (Cybertext) 也是输入 (Plaintext)。所以，在不考虑初始种子值的情况下，Hash 链上的每一个值长度相同。这样带来的一个坏处是会减少明文空间 (Key Space)，从而减少破解的时间。

而 SRHC-TD 在 Hash 链中增加了 salt 值，即 $\zeta_{i+1}=\left(x_{c_k+1},I_{c_k+1}\right)<<r_{i+1}$，从而增大了明文空间，这恰恰是其他的构造方案(RHCs[12-15]) 中所没有提及到的。为了更好地阐述 SRHC-TD 相较于其他普通的 RHCs 在明文空间（正比于破解时间）上的优势，我们将以图表的形式直观地比较它们的性能。

以 MD5 为例，假设输出 L(128bit)划分成(32, 4)，链长 $n=100$。选择 1000 组(t, 32)-Mignotte's 序列($t=10,11,\ldots,16$)，并且假设序列值的平均长度为 4 bit，每组分别计算 MD5 散列级联值 P_U，再将 P_U 进行 100 次 MD5 散列，记录下这 100000 个 Cybertext 中的(32, t)划分情况。并且按照每 10000 个 Cybertext 为一组 test，每组 test 中计算 t 取到不同数值时的概率。将第 i 组 test 中 t 值取 j 时的概率记

为 p_{ij}。得到如图 8.1 的统计概率图。重复以上的步骤 10 次。

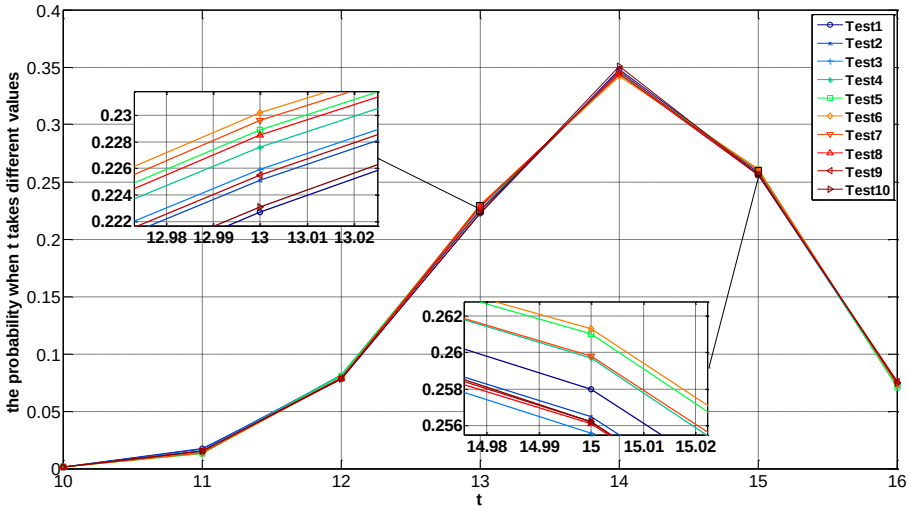

图 8.1 当 t 取不用的值时的概率 p_{ij}

由图 8.1 可知，t 的取值范围大概是{10,11,12,13,14,15,16}，当然不排除会取值到小于等于 10 的自然数的情况存在。图 8.1 中具体的数值如表 8.1 所示。

表 8.1 p_{ij} 的具体数值

Test \ t	10	11	12	13	14	15	16
Test 1	0.0013	0.0176	0.0781	0.2227	0.3480	0.2580	0.0743
Test 2	0.0013	0.0168	0.0806	0.2251	0.3448	0.2565	0.0749
Test 3	0.0014	0.0159	0.0815	0.2259	0.3466	0.2556	0.0731
Test 4	0.0012	0.0142	0.0821	0.2276	0.3445	0.2597	0.0707
Test 5	0.0012	0.0133	0.0807	0.2289	0.3438	0.2610	0.0711
Test 6	0.0012	0.0135	0.0793	0.2302	0.3418	0.2613	0.0727
Test 7	0.0016	0.0137	0.0784	0.2296	0.3435	0.2598	0.0734
Test 8	0.0016	0.0153	0.0785	0.2285	0.3440	0.2561	0.0760
Test 9	0.0013	0.0153	0.0793	0.2255	0.3464	0.2562	0.0760
Test 10	0.0013	0.0155	0.0783	0.2231	0.3506	0.2562	0.0750
$p_{Avg_{t=j}} = 0.1 \times \sum_{i=1}^{10} p_{ij}$	0.0013	0.0151	0.0797	0.2267	0.3454	0.2580	0.0737

由 SRHC-TD 中的 salt 值可知，明文空间平均增加了 128 bit $(128 = 4 \times 32)$，这样，明文空间的大小就变成了 2^{256}。基于改进的 m 划分树的概念，根据表 1 中 t 取不同值时的平均概率可以得出一般 RHCs[12-15] (没有增加 salt 值) 的平均 Key Space 以及 SRHC-TD (加了 salt 值) 的平均 Key Space，如表 8.2 所示。

表 8.2 SRHC-TD 与一般 RHCs 在明文空间上的比较

t	一般 RHSs 的明文空间 表示为 KSu; SRHC-TD 的明文空间表示为 KSa		KSa/KSu
10	KSu_{10}	553,333,286,463,413,155,119,380,296,207,872,000	143,020,679,939,872,390,880,211,676,554,285
	KSa_{10}	79,138,102,863,361,537,004,679,285,453,617,238,668,120,868,942,959,692,426,007,139,840,000	
11	KSu_{11}	5,187,080,514,550,099,279,678,121,388,076,646,400	3,661,891,040,325,389,282,674,949,544,144,523
	KSa_{11}	18,994,523,661,677,418,591,170,431,781,409,944,281,563,099,229,921,542,416,319,529,347,072,000	
12	KSu_{12}	26,806,046,942,907,604,997,886,958,590,845,952,000	75,742,751,459,000,734,383,329,857,043,540,793
	KSa_{12}	2,030,363,751,194,957,174,069,912,361,433,154,199,951,665,857,317,337,853,414,530,889,487,360,000	
13	KSu_{13}	76,637,826,616,149,945,551,514,260,280,729,600,000	1,324,110,606,676,344,898,872,293,485,267,251,454
	KSa_{13}	101,476,959,095,066,836,883,574,848,357,011,830,828,587,992,686,398,439,770,172,405,579,571,200,000	
14	KSu_{14}	117,608,245,430,622,401,776,527,064,367,308,800,000	20,264,471,610,378,457,664,664,760,637,922,372,791
	KSa_{14}	2,383,268,950,675,269,627,341,439,620,767,777,205,252,014,589,252,078,877,364,719,119,523,225,600,000	
15	KSu_{15}	88,465,874,655,133,624,601,621,999,967,436,800,000	279,240,713,721,406,725,647,529,271,034,190,240,345
	KSa_{15}	24,703,273,978,688,019,410,535,964,115,469,307,361,564,474,752,449,796,080,640,681,676,767,395,840,000	
16	KSu_{16}	24,991,467,359,322,430,689,958,992,833,556,480,000	3,545,290,824,160,424,396,558,195,520,471,044,725,362
	KSa_{16}	88,602,019,911,310,565,452,710,747,271,750,192,445,143,064,404,384,392,174,672,129,629,351,424,000,000	
$KSu_{Avg} = \sum_{t=10}^{16} p_{Avg_t} \times KSu_t$		84,877,236,261,416,553,573,765,570,028,577,116,785	161,995,868,252,801,528,012,638,215,965,348,282,400
$KSa_{Avg} = \sum_{t=10}^{16} p_{Avg_t} \times KSa_t$		13,749,761,583,066,344,526,126,464,575,649,604,504,003,162,279,445,035,355,884,359,119,207,993,553,972	

由表 8.2 可知，SRHC-TD 中增加了 salt 值，Key Space 比 RHC, ERHC, SUHC, SRHC 平均增长了 161,995,868,252,801,528,012,638,215,965,348,282,400 倍，这样不管是在暴力破解或者彩虹表破解时都增加了破解时间，从而增加了 Hash 链的安全性，性能更加优于文献[12-15]中没有 salt 值 RHC 构造方案。

2) 双重验证

在 4.4.4 节的密钥重组阶段中，接收方根据已知（已验证）的信息用两种不同的方法来计算得到新生的第二条密钥链的链首值 $h^n(P_U')$。

a) 首先，是有序验证法。它是在 ERHC 方案的基础上改进得到的。与 ERHC 类似，它同样也具有不可否认性的特点。然后，ERHC 不能抵抗选择明文攻击，由下例可以简单描述：

在 ERHC 中，假设散列函数输出值长 L=8 (bit) 时，用 $\lfloor \log_2 L \rfloor + 1 = 4$ (bit) 来表示私钥中‘1’的个数。当第二条密钥链的链首值 $h^n(P_U')$ 中‘1’的个数为 5 时，校验位为‘0101’，中间人可以伪造成‘1’的

个数为 0、1、4，校验位改为‘0000’或‘0001’或‘0100’，不透露或只透露 S_U 的部分信息。而出现这种情况的概率很大，因为只要 $h^n(P_U')$ 中‘1’的个数不为 0，就都会有可能受到此类的选择明文攻击。

而 SRHC-TD 则有效地防止了此类攻击。这是因为，在 4.4.2 节的密钥发送阶段，消息验证码 ζ_i 每次都是不同的。在 ζ_i 中，存在左移操作，并且左移位数 r_i 的选择也是单向性的。

b) 其次，是 CRT 验证法。它是基于(*t*, *n*)-Mignotte's 门限秘密共享方案的。它虽然弱化了“任意 *t* 个子密钥都能计算得出唯一的主密钥”[47]这一属性。但消息认证码 ζ_i 的完整性和可信性能够由它与 Hash 值 $h^{n-i}(P_U)$ 的异或操作来保证；并且延迟发布种子值 S_U 能够保证它的不可否认性。因此，主密钥的唯一性及正确性能够受到保护。

3) 可证明安全

由于 SRHC-TD 是由前后多条链的首尾相接组成的。所以，在研究其可证明安全性时，需要分别考虑两种情况。一是，在一条链上的前后 Hash 值的可证明安全性。二是，前后两条链的首尾相接处的 Hash 值的可证明安全性。

为简便起见，选择 Random Oracle 模型来分析 SRHC-TD 的理论模型。在该模型中，攻击者拥有关于某个安全参数 *k* 的多项式计算能力。方案中所使用的所有算法的安全强度均由安全参数 *k* 决定，并且破解 Hash 算法的概率为关于参数 *k* 的指数函数的倒数，破解 CRT 算法的概率为关于参数 *k* 的幂函数的倒数，破解划分算法的概率为关于参数 *k* 的对数函数的倒数。

a) 在第 *i* 个时间周期结束时刻，接收者（包含攻击者）收到 $\left(h^{n-i}(P_U), \zeta_i, \xi_i\right)$。攻击者需要在第 $i+1$ 个时间周期内成功伪造 $h_{fake}^{n-i-1}, \zeta_{i+1}$，使得 $h\left(h_{fake}^{n-i-1}\right)=h^{n-i}(P_U)$ 以及 $h_{fake}^{n-i-1}\oplus\zeta_{i+1}=\xi_i$ 才能攻击成功。设攻击者的计算能力界限是多项式 $T_{adv}(k)$，在每个时间段内可查询 Oracle 任意次，又假设破解两个 Hash 算法的概率均为 e^{-k}，则攻击成功的概率为：

$$\Pr\left[Adv(k)=1\right]=\Pr\left[h_{fake}^{n-i-1}=h^{n-i-1}(P_U), h_{fake}^{n-i-1}\oplus\zeta_{i+1}=\xi_i\right]=\frac{T_{adv}(k)}{e^k\cdot e^k} \tag{8.1}$$

其中，$T_{adv}(k)$ 具有以下一般形式，$T_{adv}(k)=a_nk^n+a_{n-1}k^{n-1}+\cdots+a_1k+a_0$。于是，我们可以得到 $\Pr\left[Adv(k)=1\right]=\frac{a_nk^n+a_{n-1}k^{n-1}+\cdots+a_1k+a_0}{e^{2k}}$。对 $\Pr\left[Adv(k)=1\right]$ 取极限得到，

$$\lim_{k\to\infty}\Pr\left[Adv(k)=1\right]=\lim_{k\to\infty}\frac{a_n k^n}{e^{2k}}=\lim_{k\to\infty}\frac{a_n\cdot nk^{n-1}}{2\cdot e^{2k}}=\cdots=\lim_{k\to\infty}\frac{a_n\cdot n!}{2^n\cdot e^{2k}}=0 \tag{8.2}$$

b) 在第$n-1$个时间周期结束时刻，第一条链中仅剩P_U值没有公布。接收者（包含攻击者）得到m_L个ζ_i值。攻击者需要在第 n 个时间周期内成功伪造P_U值$P_{U_{fake}}$，使得：$h\left(P_{U_{fake}}\right)=h(P_U)$；且当$i_1-i_2\equiv 0 \bmod m_L$时，$\zeta_{i_1}$和$\zeta_{i_2}$内包含相同的序列值$x_{c_k+1}$和子密钥$I_{c_k+1}$，其中，$k=i_1+1\left(\bmod m_L\right)$，$i_1,i_2=1,2,\ldots,n-1$。且最后用有序验证法得到的解$h^n(P_U')_1$要和 CRT 验证法得到的解$h^n(P_U')_2$相等。攻击者只有伪造了满足以上三个条件的$P_{U_{fake}}$值才能攻击成功。设攻击者的计算能力界限是多项式$T_{adv}(k)$，在每个时间段内可查询 Oracle 任意次，又假设破解 Hash 算法的概率为e^{-k}，破解中国剩余定理算法的概率为k^{-e}，破解划分算法的概率为$\frac{1}{\ln k}$，则攻击成功的概率为：

$$\Pr\left[Adv(k)=1\right]=\Pr\left[\begin{array}{l} h\left(P_{U_{fake}}\right)=h(P_U),\ h^n(P_U')_1=h^n(P_U')_2 \\ x_{c_{i_1+1(\bmod m_L)}+1}=x_{c_{i_2+1(\bmod m_L)}+1},\ I_{c_{i_1+1(\bmod m_L)}+1}=I_{c_{i_2+1(\bmod m_L)}+1}\end{array}\right]=\frac{T_{adv}(k)}{e^k\cdot k^e\cdot\ln k} \tag{8.3}$$

取$T_{adv}(k)=a_n k^n+a_{n-1}k^{n-1}+\cdots+a_1 k+a_0$，对$\Pr\left[Adv(k)=1\right]$取极限得到，

$$\begin{aligned}\lim_{k\to\infty}\Pr\left[Adv(k)=1\right]&=\lim_{k\to\infty}\frac{a_n k^n}{e^k\cdot k^e\cdot\ln k}=\lim_{k\to\infty}\frac{a_n k^{n-e}}{e^k\cdot\ln k}=\lim_{k\to\infty}\frac{a_n\cdot(n-e)k^{n-e-1}}{e^k\cdot\left(\ln k+\frac{1}{k}\right)}=\\ &\cdots=\lim_{k\to\infty}\frac{a_n\cdot(n-e)!}{e^k\cdot\left(\ln k+\frac{n-e}{k}-\cdots+\frac{(n-e-1)!}{k^{n-e}}\right)}=\lim_{k\to\infty}\frac{a_n\cdot(n-e)!}{e^k\cdot\ln k}=0\end{aligned} \tag{8.4}$$

综合以上两种情况，且由极限的定义可知，存在正整数 N，当$k>N$时，对于任意的$\varepsilon>0$，有$\Pr\left[Adv(k)=1\right]<\varepsilon$。所以攻击者攻击成功的概率可忽略不计，即本文所提出的 SRHC-TD 方案是可证明安全的。

8.1.2 容丢包性容错性

Hash 链的长度为$n-1$，它是m_L的整数倍。因此m_L对$\left(x_{c_i+1},I_{c_i+1}\right)$在整条 Hash 链的使用过程中，

能够重复传输，循环往复地发送给接收者。从而避免出现由于某一个子密钥丢失而造成主密钥 $h^n\left(P_U'\right)$ 无法求解的情况。即使在丢包率或者错包率高达 $\left(m_L-q_L\right)/m_L$ 的情况下，主密钥也依然能够被求解。说明 SRHC-TD 符合并且严格遵循(*t*, *n*)门限的 *t*-consistency[47]这一属性。

以上的属性都是 RHC，ERHC，SUHC，SRHC 等中所没有的。

8.1.3 复杂性

Hash 链的保存方法有两种：一是，生成一条链，然后分配专门的空间保存整条链值；二是，只保存链首值，每次用到某个值时再通过链首值来计算。显然，前者降低了计算开销但是增加了存储开销，后者降低了存储开销但是增加了计算开销。

以下分别考虑 Hash 链的两种保存方法，将 SRHC-TD 与 RHC，ERHC，SUHC，SRHC 进行对比。分别从三方面进行比较：计算开销、通信开销、存储开销。表 8.3 说明了各个参数的对应符号。表 8.4 对 SRHC-TD 与一般 RHCs 的详细对比。

表 8.3 各参数名称及其符号表示

参数符号	参数名称
L	Hash 函数的输出长度
n	Hash 链的长度
m	SRHC-TD 中 Mignotte's 序列的长度（序列值的个数）
H	Hash 函数的计算开销
U	联合 (Union) 操作的计算开销
R	RHC 和 ERHC 中生成一个随机数的计算开销
$\mathrm{R_B}$	SUHC 中生成一个随机数的计算开销
$\mathrm{R_{B'}}$	SRHC 中生成一个随机数的计算开销
$\mathrm{R}_{t,m}$	SRHC-TD 中生成一个随机数的计算开销
B	SUHC 中通过 Hard Core Predicate 将一个随机数映射到一个 bit 的计算开销
B′	SRHC 中通过 Hard Core Predicate 将一个随机数映射到一个 bit 的计算开销
I	SRHC-TD 中生成子密钥 I_i 的计算开销
P_r	SRHC-TD 中计算移位数 r_i 的计算开销
M	左移操作的计算开销
X	异或操作的计算开销
C	计算方程组的解的计算开销
len_{H}	k (bit)的通信、存储开销
len_S	Hash 链种子值的通信、存储开销
len_r	生成的随机数的通信、存储开销
$len_{r'}$	Mignotte's 序列值的通信、存储开销
len_{I}	SRHC-TD 中子密钥 I_i 的通信、存储开销

表 8.4 SRHC-TD 与一般 RHC 的计算、通信、存储开销

存储方式			存储整条链	存储链首
RHC	初始化	计算	$(2L+3)\cdot \mathrm{H}+2\mathrm{R}$	$(3L+2)\cdot \mathrm{H}+2\mathrm{R}$
		通信	$2len_{\mathrm{H}}$	$2len_{\mathrm{H}}$
		存储	$2(len_{\mathrm{s}}+len_{r})+(L+4)\cdot len_{\mathrm{H}}$	$2(len_{\mathrm{s}}+len_{r})+4len_{\mathrm{H}}$
	发布 认证 重组	计算	$(2L+3)\mathrm{H}+2\mathrm{R}$	$0.5(L^2+3L-4)\mathrm{H}+2(L-1)\mathrm{R}$
		通信	$2L\cdot len_{r}+(6L-2)\cdot len_{\mathrm{H}}$	$2L\cdot len_{r}+(6L-2)\cdot len_{\mathrm{H}}$
		存储	$2(len_{\mathrm{s}}+len_{r})+(L+4)\cdot len_{\mathrm{H}}$	$len_{r}+(L+2)\cdot len_{\mathrm{H}}+L$
ERHC	初始化	计算	$2(n+L+\lfloor \log_2 L \rfloor+1)\cdot \mathrm{H}$ $+2(L+\lfloor \log_2 L \rfloor+1)\cdot \mathrm{R}+2\mathrm{U}$	$2(n+L+\lfloor \log_2 L \rfloor+1)\cdot \mathrm{H}$ $+2(L+\lfloor \log_2 L \rfloor+1)\cdot \mathrm{R}+2\mathrm{U}$
		通信	0	0
		存储	$(n+2L+1)\cdot len_{\mathrm{H}}+2L\cdot len_{r}$	$(2L+1)\cdot len_{\mathrm{H}}+2L\cdot len_{r}$
	发布 认证 重组	计算	$0.5(2n+L+\lfloor \log_2 L \rfloor+1)\cdot \mathrm{H}$	$0.5(n^2+n+L+\lfloor \log_2 L \rfloor+1)\cdot \mathrm{H}$
		通信	$2(n+L+\lfloor \log_2 L \rfloor+1)\cdot len_{\mathrm{H}}$ $+(L+\lfloor \log_2 L \rfloor+1)\cdot len_{r}$	$2(n+L+\lfloor \log_2 L \rfloor+1)\cdot len_{\mathrm{H}}$ $+(L+\lfloor \log_2 L \rfloor+1)\cdot len_{r}$
		存储	$(n+L+\lfloor \log_2 L \rfloor+1)\cdot len_{\mathrm{H}}+L$	$(n+L+\lfloor \log_2 L \rfloor+1)\cdot len_{\mathrm{H}}+L$
SUHC	初始化	计算	$2(L+1)\cdot \mathrm{H}+\mathrm{R_B}$	$(3L+1)\cdot \mathrm{H}+\mathrm{R_B}$
		通信	$2len_{\mathrm{H}}$	$2len_{\mathrm{H}}$
		存储	$2len_{\mathrm{s}}+len_{r}+(L+3)\cdot len_{\mathrm{H}}$	$2len_{\mathrm{s}}+len_{r}+4len_{\mathrm{H}}$
	发布 认证 重组	计算	$(5L-2)\cdot \mathrm{H}+(L-1)\cdot \mathrm{R_B}+L\mathrm{B}$	$0.5(L^2+6L-4)\cdot \mathrm{H}+(L-1)\cdot \mathrm{R_B}+L\mathrm{B}$
		通信	$(6L-3)\cdot len_{\mathrm{H}}+2L\cdot len_{r}$	$(6L-3)\cdot len_{\mathrm{H}}+2L\cdot len_{r}$
		存储	$(L+2)\cdot len_{\mathrm{H}}+len_{r}+L$	$(L+2)\cdot len_{\mathrm{H}}+len_{r}+L$
SRHC	初始化	计算	$(2L+1)\mathrm{H}+\mathrm{R_{B'}}$	$3L\mathrm{H}+\mathrm{R_{B'}}$
		通信	$2len_{\mathrm{H}}$	$2len_{\mathrm{H}}$
		存储	$2len_{\mathrm{s}}+len_{r}+(L+2)\cdot len_{\mathrm{H}}$	$2len_{\mathrm{s}}+len_{r}+3len_{\mathrm{H}}$
	发布 认证 重组	计算	$(3L-1)\cdot \mathrm{H}+(L-1)\cdot \mathrm{R_{B'}}+L\mathrm{B'}$	$0.5(L^2+5L-2)\cdot \mathrm{H}+(L-1)\cdot \mathrm{R_{B'}}+L\mathrm{B'}$
		通信	$(4L-2)\cdot len_{\mathrm{H}}+2L\cdot len_{r}$	$(4L-2)\cdot len_{\mathrm{H}}+2L\cdot len_{r}$
		存储	$L\cdot len_{\mathrm{H}}+2len_{r}+L$	$L\cdot len_{\mathrm{H}}+2len_{r}+L$
SRHC -TD	初始化	计算	$2(n+m)\cdot \mathrm{H}+2m\cdot \mathrm{R}_{t,m}+2\mathrm{U}+m\mathrm{I}$	$2(n+m)\cdot \mathrm{H}+2m\cdot \mathrm{R}_{t,m}+2\mathrm{U}+m\mathrm{I}$
		通信	0	0
		存储	$(n+2m+1)\cdot len_{\mathrm{H}}+2m\cdot len_{r'}+m\cdot len_{\mathrm{I}}$	$(2m+1)\cdot len_{\mathrm{H}}+2m\cdot len_{r'}+m\cdot len_{\mathrm{I}}$
	发布 认证 重组	计算	$n\mathrm{H}+2n\mathrm{M}+2n\mathrm{X}+2n\mathrm{P}_r+\mathrm{C}$	$0.5(n^2+n)\mathrm{H}+2n\mathrm{M}+2n\mathrm{X}+2n\mathrm{P}_r+\mathrm{C}$
		通信	$4n\cdot len_{\mathrm{H}}+2n\cdot len_{r'}+2n\cdot len_{\mathrm{I}}$	$4n\cdot len_{\mathrm{H}}+2n\cdot len_{r'}+2n\cdot len_{\mathrm{I}}$
		存储	$(n+2m)\cdot len_{\mathrm{H}}+n\cdot len_{r'}+n\cdot len_{\mathrm{I}}+L$	$(n+2m)\cdot len_{\mathrm{H}}+n\cdot len_{r'}+n\cdot len_{\mathrm{I}}+L$

为简便起见，不妨假设 $L\approx n>m$，$\mathrm{H}>\mathrm{B}\approx\mathrm{B'}$，$\mathrm{H}>\mathrm{R}\approx\mathrm{R_B}\approx\mathrm{R_{B'}}\approx\mathrm{R}_{t,m}$，$\mathrm{H}>\mathrm{C}>\mathrm{I}>\mathrm{P}_r\approx\mathrm{M}\approx\mathrm{X}\approx\mathrm{U}$，$len_{\mathrm{H}}>len_{\mathrm{S}}\approx len_{r}\approx len_{r'}\approx len_{\mathrm{I}}$。对表 8.4 的开销数据进行简化，并对以上 5

个方案的开销进行定性的比较。如表 8.5 所示。

表 8.5 SRHC-TD 与一般 RHCs 的计算、通信、存储开销比较

存储方式		存储整条链	存储链首
初始化	计	SRHC < SUHC < RHC < **SRHC-TD** < ERHC	**SRHC-TD** < SRHC < SUHC < RHC < ERHC
	通	ERHC = **SRHC-TD** < RHC = SRHC = SUHC	ERHC = **SRHC-TD** < RHC = SRHC = SUHC
	存	SRHC < SUHC < RHC < **SRHC-TD** < ERHC	SRHC < SUHC < RHC < **SRHC-TD** < ERHC
发布认证重组	计	**SRHC-TD** < ERHC < RHC < SRHC < SUHC	**SRHC-TD** < ERHC < RHC < SRHC < SUHC
	通	**SRHC-TD** < SRHC < ERHC < RHC < SUHC	**SRHC-TD** < SRHC < ERHC < RHC < SUHC
	存	SRHC < SUHC < RHC < **SRHC-TD** < ERHC	SRHC < SUHC = RHC < **SRHC-TD** < ERHC

由表 8.5 可知，SRHC-TD 在复杂度方面有很好的性能表现。其中，在发布-认证-重组阶段，SRHC-TD 的计算和通信开销要远小于其余的 RHCs；而存储开销方面，除了 ERHC，SRHC-TD 只比其余的 RHCs 稍微略高。但是由表 8.4 可知，SRHC-TD 的存储开销与 RHC，SUHC，SRHC 很接近，且大概是 ERHC 的一半。

据我们所知，无线传感器网络中，通信开销和计算开销是其主要的能耗。而且，发送和接收一个数据包的能耗是计算或处理该数据包的好几倍[48]。由于在整个网络的生命周期内，“发布−认证−重组(Distribution-Authentication-Combination, DVC)”阶段的时长远远长于“初始化 (Initialization, Ini)”阶段，因此，图 8.2 用图形化方式给出了在该阶段 SRHC-TD 相较于其他 RHCs 的性能优异性。

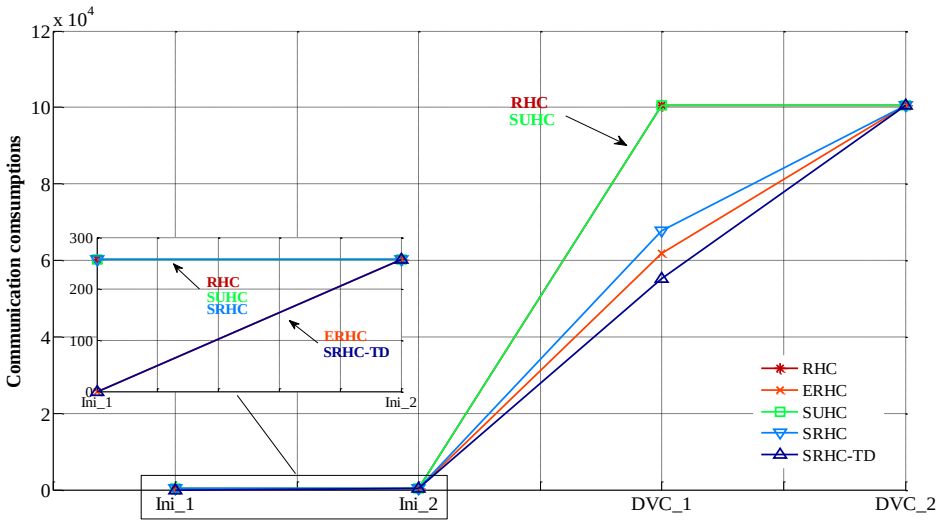

图 8.2 SRHC-TD 与一般 RHCs 的性能比较

8.2 AdlCBF 性能评估

8.2.1 与直接存储方式的比较

构建一个 dlCBF 的基本的仿真配置信息如表 8.6 所示。其中，q 是一个大素数。

表 8.6 仿真的配置信息

参数名称	参数值
元素个数	$m = 500000$
子队列的个数	4
每个子队列中桶的个数	$m/24 = 20833$
每个桶的平均负载	6
每个桶中单元的个数	8
一个计数器的 bit 数	2 bit
存储元素的 bit 数	$r = 14$ bit
Hash 函数	MD5
伪随机置换函数	$P_i\left(H\left(x\right)\right)=aH\left(x\right)\bmod 2^q$
置换函数中的比例参数 a	$\left[2^q\right]$ 中的随机奇数

1) 存储空间的比较

假设 ECDSA 中公钥 KP_A 的长度为 x，那么 $2^x-1\geq n$ ，其中 n 是椭圆曲线的最大值。为了保证所有节点的公钥的唯一性，必须满足 $n\geq m$。因此，我们能够得到 $x\geq \log_2\left(n+1\right)\geq \log_2\left(m+1\right)=18.93$，这样公钥的最小长度为 19 bit。同理可得，ID 的最短长度也为 19 bit。因此，如果 500000 个元素以明文形式存储的话，所需要的存储空间为 $500000\times\left(19+19\right)=19000000\,\text{bit}$。而当 500000 个元素存储在 dlCBF 中时，所需要的存储空间为 $\frac{4m\left(r+2\right)}{3}=10666667\,\text{bit}$。因此，前者（直接存储）是后者（dlCBF 存储）的 1.78125 倍。总而言之，AdlCBF 机制相比于直接存储方式更加节省存储空间。

2) 时间的比较

在构建好的 dlCBF 中增加元素，查询元素，以及删除元素。并且重复以上操作 3000 次。并记下每一项操作的耗时。然后计算每项操作的平均时间。这样一组 test 就结束了。接着，在继续做另外 9 组相同的 test。具体的数值信息如表 8.7 所示。

表 8.7 增加、查询、删除操作的平均耗时

Time (ms) / Test	增加	查询	删除
Test 1	0.23	0.08	0.32
Test 2	0.21	0.07	0.35
Test 3	0.31	0.13	0.40
Test 4	0.25	0.12	0.42
Test 5	0.20	0.10	0.35
Test 6	0.27	0.10	0.39
Test 7	0.30	0.09	0.30
Test 8	0.24	0.08	0.38
Test 9	0.21	0.07	0.35
Test 10	0.23	0.09	0.36
Average Time	0.25	0.10	0.36

如表 8.7 所示，事实上，在 AdlCBF 方案中，dlCBF 的查询时间就是认证时间。在 AdlCBF 中，主要的操作都是由 CPU 所完成的；而在直接存储方式中，主要的操作是由 RAM 或者 Flash 完成的。据我们所知，CPU 的读写速度比 RAM 和 Flash 快 2 到 3 个数量级。根据 AdlCBF 的存储空间以及查询时间，我们可以得到直接查询方法它所需要的时间在 1.78ms 到 17.81ms 范围之间。总之，AdlCBF 具有更高的查询效率。

8.2.2 与使用 CBF 的认证方案的比较

AdlCBF 的假阳性概率的上限为$\left(\frac{1}{2}\right)^{r}\times 4\times 6 = 24\times 2^{-r}$。而使用 CBF 的认证方案的假阳性概率为$\left(2^{-\ln 2}\right)^{c}$。当$c = \frac{r+2}{3}$时，这两个方案都需要相同的存储空间。然而，比较它们的假阳性概率之比，我们可得到 $\left(2^{-\ln 2}\right)^{\frac{r+2}{3}} \Big/ \left(24\times 2^{-r}\right) = 52.65$。因此，我们可以得出，在相同的存储空间的条件下，使用 CBF 的认证方案的假阳性概率远远大于 AdlCBF，是它的的 52.65 倍。

AdlCBF 的存储空间为$\frac{4}{3}m\left(r+2\right)$。而使用 CBF 的认证方案的存储空间为 $4cm$。当$\left(2^{-\ln 2}\right)^{c} = 24\times 2^{-r}$时，两者具有相同的假阳性概率。然而，比较他们的存储空间之比，我们得到$4m\cdot\frac{\ln\left(24\times 2^{-r}\right)}{\ln\left(2^{-\ln 2}\right)} \Big/ \frac{4}{3}m\left(r+2\right) = 2.55$。因此，我们可以得出，在相同的假阳性概率下，使用 CBF 认证方案需要的存储空间大于 AdlCBF，是它的的 2.55 倍。

总之，相比于使用普通 CBF 的认证机制，AdlCBF 具有更好的性能，即更低的假阳性概率或者更少的存储空间。

8.2.3 安全强度

正如第一章绪论中所介绍，利用 ECDSA 算法，传感节点能够被认证，并且基站能够抵抗 DoS 攻击。在 AdlCBF 中，这样的攻击就能够被有效地减轻。这是因为，基站通过 dlCBF 来识别申请者的身份。因此，只需要它的假阳性概率达到了可忽略不计的程度，AdlCBF 就能都很好地工作。

在 8-bit 的 ATmega128L 处理器上，MD5 的能耗只有 5.9 μJ/B，而使用 ECDSA-160 算法的签名验证的能耗为 45.09 mJ[18]。恶意者可以通过网络流量的监听获得一个有效的公钥，然后再利用该公钥伪造虚假数据包，从而洪泛式地攻击基站。然而，恶意者不能篡改信息认证信息 M_a (Authentication Message，见算法 5.3 和算法 5.4) 或者加上额外信息的认证信息 M_{aa} (Added Authentication Message，见算法 5.4) 的 ID 值。即使 ID 值能够被篡改，但是当基站接收到多个相同的认证信息之后，它也能察觉到受到了攻击。那些物理位置上靠近恶意者的传感节点可能会被俘获而滥用。众所周知，这种形式的攻击在基于 PKC 的安全机制中也常出现。

如果合法的传感节点无法接收到认证响应消息 M_{ar} (Authentication Response Message，见算法 5.4)，或者在一轮当中多次接收到错误的响应消息，那么它就会知晓它受到了攻击，并且警告给基站。

如果簇头接收到来自同一个节点的多次认证消息 M_a，或者认证响应消息 M_{ar}，或者加上额外信息的认证响应消息 M_{aar} (Added Authentication Response Message，见算法 5.4)，那么它就会知晓它受到了攻击，并且警告给基站。

基站节点将立即执行现场调查或者启动入侵检测系统来检测恶意者，并且采取相应的补救措施（比如，入侵检测）。这部分内容将在 8.4 节进行介绍。

8.3 PTCR 性能评估

理论上，基站能够位于网络的任何位置。但是，在 LEACH, SEP, DEEC 等算法中，基站常常位于网络的中心位置。因此，为了统一性及考虑到仿真的简易性，我们采取同样的方式，即，将基站置于网络的中心。并且，我们假设传感节点是随机而无重复地分布在网络各处。关于这一节的仿真参数的基本配置如表 8.8 所示。

表 8.8 仿真参数的基本配置

参数名称	参数值
节点个数	500
传感节点的初始能量	0.5 J or 1 J
仿真场景的范围	500 m×500 m
发送者/接收者的电子能量	E_{elec} = 50 nJ / bit
数据集成能耗	E_{DA} = 50 nJ / bit / report
发送放大器能量 (如果 $d_{toCH} \le d_o$)	ε_{fs} = 10 PJ / bit / m^2
发送放大器能量 (如果 $d_{toCH} > d_o$)	ε_{mp} = 0.0013 PJ / bit / m^4
数据包的大小	4000 bit

8.3.1 能量消耗

每一轮次后，使用 PTCR, LEACH, SEP 和 DEEC 的整网剩余能量的比较如图 8.3 所示。显然，在前 800 轮，PTCR 的能耗是最低的，而其他三个算法的能耗大致相等。在 800 轮后，PTCR 的能耗开始比其余的算法要高。

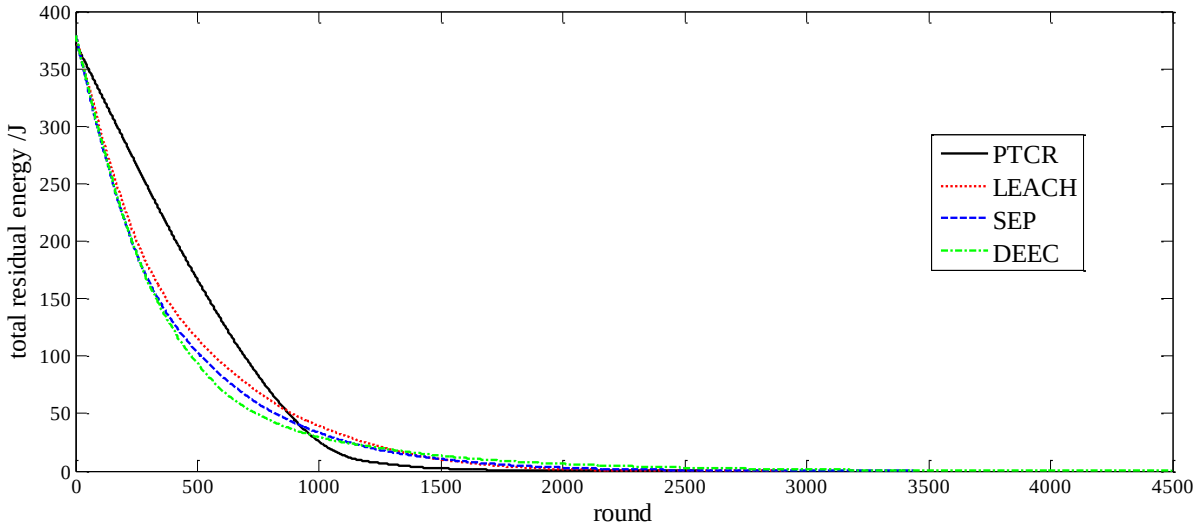

图 8.3 每轮次的剩余能量

每一轮次后，使用 PTCR, LEACH, SEP 和 DEEC 的整网剩余能量的斜率曲线如图 8.4 所示。在前 300 轮，PTCR 的能耗是最慢的，而其他三个算法的能耗波动较大。在 800 轮后，PTCR 的能耗则完全是最快的。

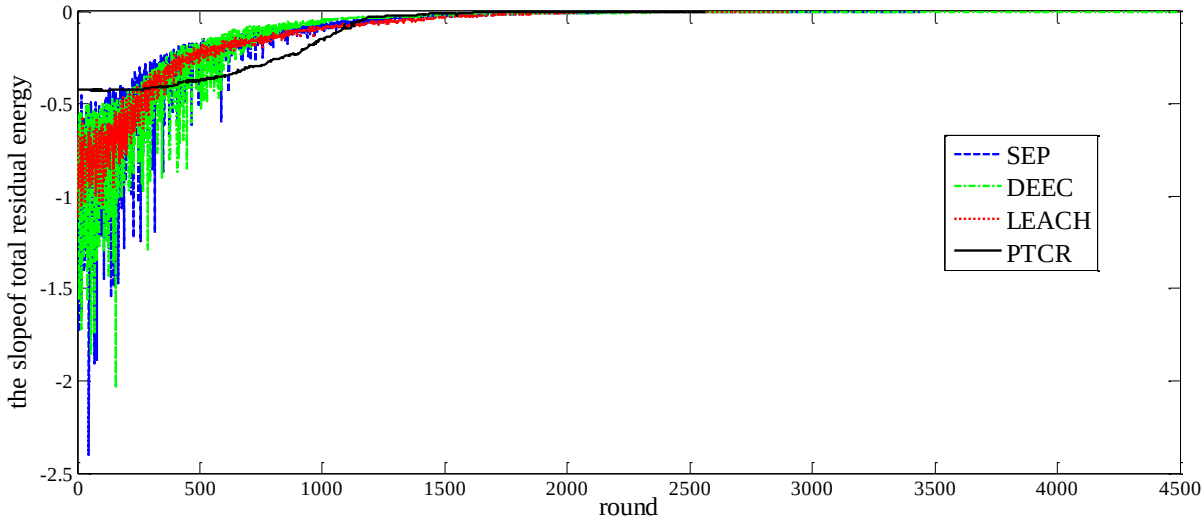

图 8.4 每轮次的剩余能量的斜率

PTCR 的剩余能量在 800 轮前后存在完全不同的极端情况，这一原因将在 8.3.2 节的网络生存时间这节介绍。

另一方面，图 8.5 也显示了 PTCR 与其余的三个算法在能量负载方面的比较。我们计算这四个算法下每个节点的剩余能量的标准差，并以此作为能量负载均衡的测试指标。显然，我们可知，在整个网络的生命周期，PTCR 的标准差总是最小的。换句话说，PTCR 中整网的能量分布是最均衡的。

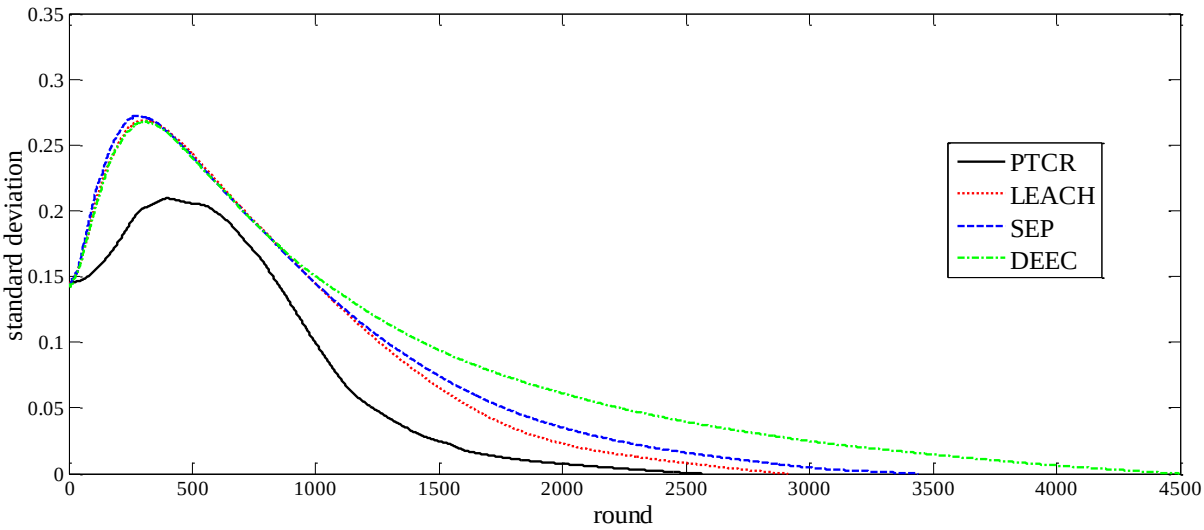

图 8.5 每轮次每个节点剩余能量的标准差

8.3.2 网络生存周期

众所周知，网络的生命周期与死亡节点的数目存在反比的关系。网络中，越少的节点死亡，那么它的生命周期就会越长。在 PTCR, LEACH, SEP 以及 DEEC 这四个算法下，每轮次的死亡节点的数

据统计如图 8.6 所示。

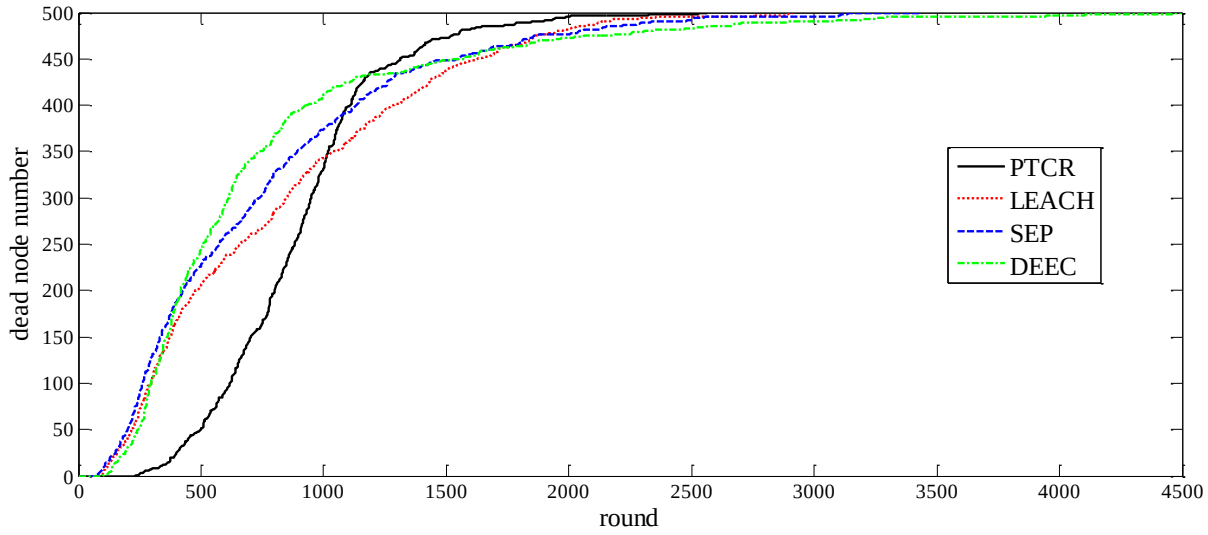

图 8. 6 每轮次死亡节点的数目

由图 8.6 可知，在前 1000 轮中，PTCR 具有最少的死亡节点。并且它的第一个死亡节点出现在第 250 轮。这远远晚于其余的算法（LEACH, SEP 以及 DEEC 这三个算法的死亡节点几乎在一开始就出现了）。然而，在四个算法当中，PTCR 具有最短的生命周期。这是因为，随着死亡节点的增多，原来稠密的网络变得越来越稀疏。经过 1200 轮之后，死亡节点占据整个网络中所有节点的 80%之多，而此时整个网络的剩余能量也是非常小的。换句话说，剩下的活节点（非死亡节点）不足以来组网。因此，如果来研究 1200 轮之后的数据是不合理并且不重要的。以上的分析表明了 PTCR 算法更适用于密度较大的无线传感器网络，而不适用于稀疏的网络。

我们同样观察到，在 800 轮之后，PTCR 的死亡节点的增长速度是最快的。正如之前所描述的，PTCR 的剩余能量在 800 轮前后存在很大的转变。这并不是一个巧合而恰恰是一个比例性问题。死亡节点的数目在某种程度上与网络的剩余能量是成比例的。通过进一步的研究分析，PTCR 的这种矛盾性，主要体现在剩余能量以及网络的生命周期，是与网络的拓扑结构相关的。图 8.7 显示了 PTCR 算法下，网络在运行到第 800 轮时，传感节点（活节点）分布的分散性。并且此种情况在 800 轮之后会愈演愈烈。

在 PTCR 算法中，传感节点的分散性分布只会出现在稀疏的传感器网络中。经过计算，我么可以得出近似的密度门限值。在第 800 轮时，有大约 200 个死亡节点。因此，在本次仿真参数的设置下，节点的密度门限为 $\frac{300}{500\times500}=0.0012/\mathrm{m}^2$ 。这样我们就能够得出，在网络节点的密度大于 $0.0012/\mathrm{m}^2$ 时，研究 PTCR 的性能会更加合适。

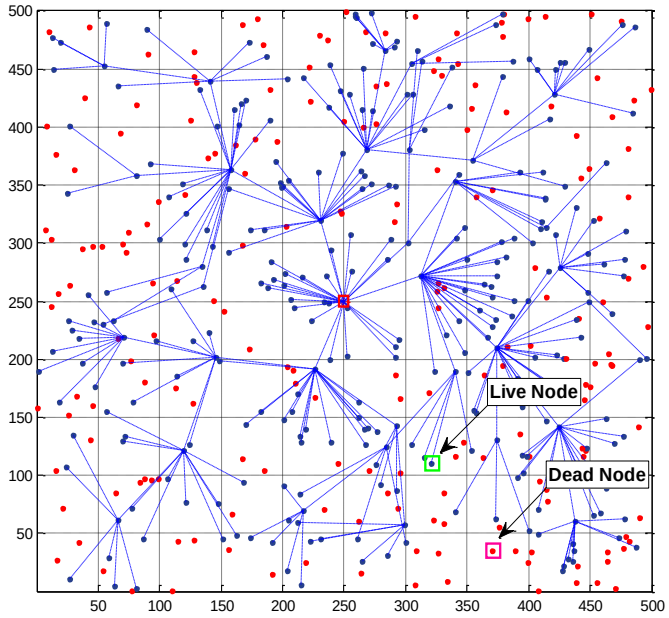

图 8.7 运行 800 轮次时，使用 PTCR 算法的网络的拓扑结构

从另一个角度，我们可以看出在 800 轮次的前后，LEACH，SEP 和 DEEC 算法也具有两极性。并且它们的性能的恰恰与 PTCR 是相反的。如图 8.8 所示，使用了 LEACH 算法的网络在第 800 轮时有 284 个死亡节点。并且它们都距离基站相对较远。此刻，大部分的活节点都位于以基站为中心 350m×350m的范围以内。那么，我们可以求出运行 LEACH 算法 800 轮次后，网络中活节点的密度为 $\frac{500-284}{350\times350}=0.0018/\mathrm{m}^2$。

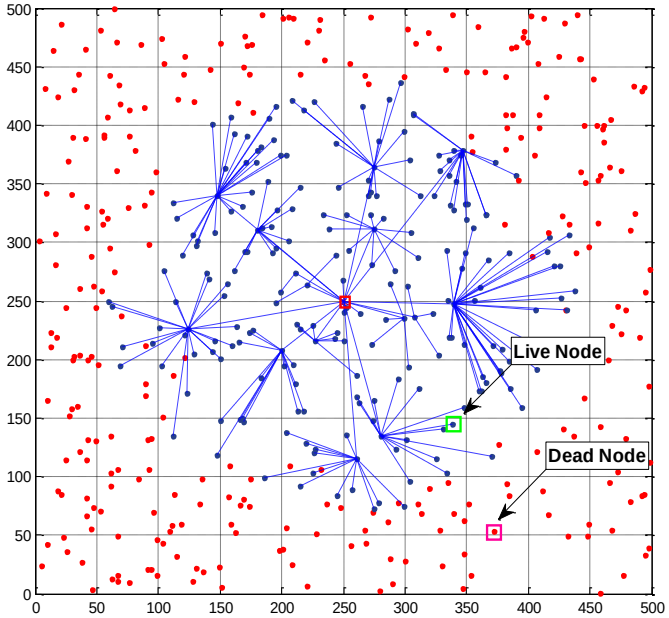

图 8.8 运行 800 轮次时，使用 LEACH 算法的网络的拓扑结构

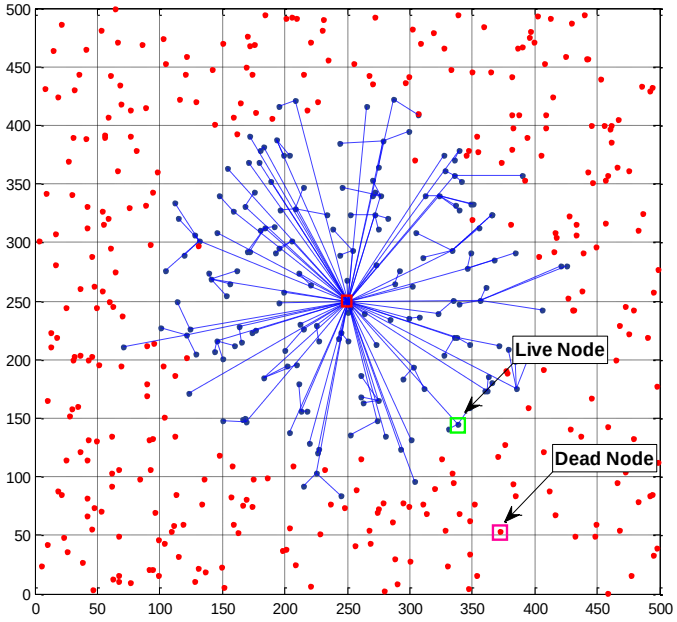

图 8.9 运行 800 轮次时，使用 SEP 算法的网络的拓扑结构

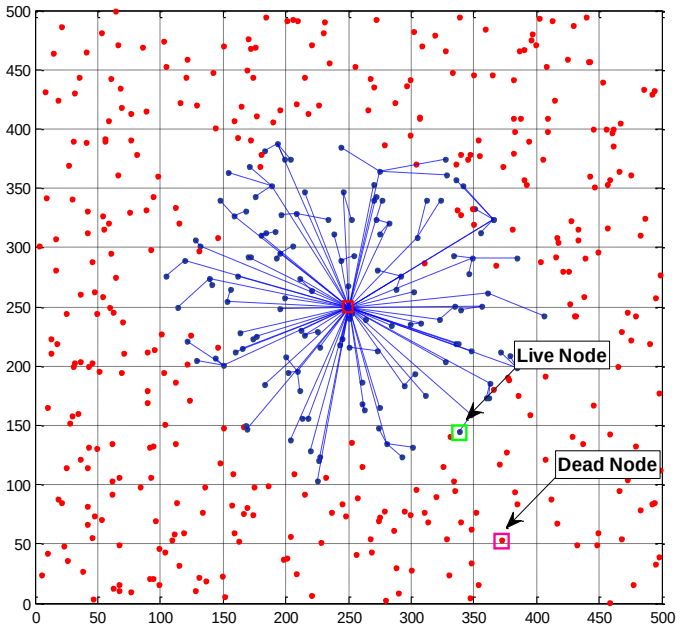

图 8.10 运行 800 轮次时，使用 DEEC 算法的网络的拓扑结构

类似地，图 8.9 显示了使用了 SEP 算法的网络在第 800 轮时有 325 个死亡节点。此刻，大部分的活节点都位于以基站为中心 300m×300m 的范围以内。那么，我们可以求出运行 SEP 算法 800 轮次后，网络中活节点的密度为 $\frac{500-325}{300\times300}=0.0019/\mathrm{m}^2$ 。

图 8.10 显示了使用了 DEEC 算法的网络在第 800 轮时有 362 个死亡节点。此刻，大部分的活节

点都位于以基站为中心 250m×250m 的范围以内。那么，我们可以求出运行 DEEC 算法 800 轮次后，网络中活节点的密度为 $\frac{500-362}{250\times250}=0.0022/\mathrm{m}^2$。

以上三个密度值（$0.0018/\mathrm{m}^2$，$0.0019/\mathrm{m}^2$，$0.0022/\mathrm{m}^2$）都是与初始化时网络中活节点的密度 $\frac{500}{500\times500}=0.0020/\mathrm{m}^2$ 近似相等。

总而言之，PTCR 在节点密度较高的网络中，相较于 LEACH, SEP 和 DEEC 算法更有优势。然而，它不能保证网络中活节点的密度的稳定性。而这一点恰恰是 LEACH, SEP 和 DEEC 所能做到的。因此，将来的工作应当是改进 PTCR 算法，使其在保证网络中活节点的密度的同时获得更好的性能。

8.3.3 簇头数目

簇头节点相较一般节点耗能更快。因此，簇头的数目是评估分簇路由算法好坏的一项重要指标。关于 PRCT, LEACH, SEP 以及 DEEC 这四个算法，在每轮次的簇头数目，由图 8.11 所示。

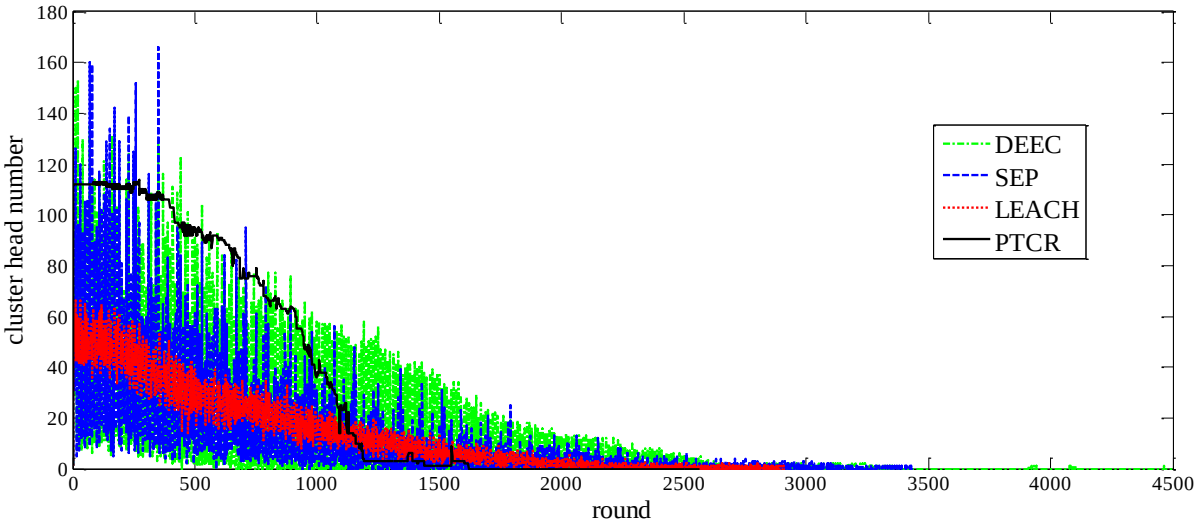

图 8. 11 轮次的簇头节点的数目

图 8.11 中，我们可以看到，SEP 和 DEEC 的曲线波动太大，而 LEACH 相对较为平缓。然而，PTCR 的曲线局部波动缓慢，但是整体上骤减的速度很快。这表明，LEACH 和 PTCR 中簇头选举算法会好于 SEP 和 DEEC。

定理 8.1 PTCR 算法中最优簇头数为 $k_{opt_P}=\sqrt{\frac{M^2\cdot N\cdot\varepsilon_{fs}}{2\pi\left(E_{elec}+E_{DA}\right)}}$。

证明 在 LEACH-C[29]算法中，每轮次的理想簇头数为

$$k_{opt_L} = \sqrt{\frac{N \cdot \varepsilon_{fs} \cdot M^2}{2\pi \cdot \varepsilon_{mp} \cdot d_{toBS}^4}} \tag{8.5}$$

其中，N 表示网络中所有节点的数目；M^2 表示网络场景的范围；d_{toBS} 表示簇头与基站之间的距离。

在 PTCR 算法中，假设第 i 级的簇头数为 k_i，并且普通节点的数目为 N_i，其中 $i = 1, 2, \ldots, n$。特别地，基站看作是第 1 级的簇头。那么我们可以得到 PTCR 算法中簇头数目为

$$N = \sum_1^n (N_i - k_i) = \sum_1^n N_i - 1 - k \tag{8.6}$$

其中 $k_1 = 1$ 并且 $k = \sum_2^n k_i$。

每个簇头都需要接收来自簇内节点的信息，处理信息，再将信息转发给其他簇头。因为簇头之间的距离较短（区别于长距离传输），能耗符合 Friss Free-Space 模型（d^2 能量损失）。因此，第 i 级的簇头在发送一帧时的能耗为

$$E_{CH_i} = l \cdot E_{elec} \cdot (N_i - k_i) + l \cdot E_{DA} \cdot N_i + (l \cdot E_{elec} + l \cdot \varepsilon_{fs} d_{toCH}^2) \cdot k_i \tag{8.7}$$

其中，l 表示每个数据包的长度，单位是 bit；d_{toCH} 表示簇头与普通节点之间的距离。

每个普通节点在一帧内都需要发送数据给簇头。类似地，它的能耗也符合 Friss Free-Space 模型。因此，第 i 级的非簇头节点（普通节点）在发送一帧时的能耗为

$$E_{non\text{-}CH_i} = (l \cdot E_{elec} + l \cdot \varepsilon_{fs} \cdot d_{toCH}^2) \cdot (N_i - k_i) \tag{8.8}$$

因此，每一帧的总能耗为

$$\begin{aligned} E_{total} &= \sum_2^n E_{CH_i} + \sum_1^n E_{non\text{-}CH_i} \\ &= l \cdot E_{elec} \cdot (2N + k + 1 - N_1) + l \cdot E_{DA} \cdot (N + k + 1 - N_1) + l \cdot \varepsilon_{fs} \cdot d_{toCH}^2 \cdot (N + k) \end{aligned} \tag{8.9}$$

LEACH-C 算法能够近似地计算 d_{toCH}^2 的平均值为

$$E[d_{toCH}^2] = \frac{1}{2\pi} \cdot \frac{M^2}{k} \tag{8.10}$$

将公式(8.10)带入到公式(8.9)，可得

$$E_{total} = l \cdot E_{elec} \cdot (2N + k + 1 - N_1) + l \cdot E_{DA} \cdot (N + k + 1 - N_1) + l \cdot \varepsilon_{fs} \cdot \frac{1}{2\pi} \cdot \frac{M^2}{k} \cdot (N + k) \tag{8.11}$$

公式(8.11)两边对 k 求导，我们就能够得出 PTCR 最优的簇头数为

$$k_{opt_P} = \sqrt{\frac{M^2 \cdot N \cdot \varepsilon_{fs}}{2\pi\left(E_{elec} + E_{DA}\right)}} \tag{8.12}$$

根据表 8 的配置信息，我们可以得到最优簇头数应该为 $k_{opt_P} = \sqrt{\frac{500^2 \times 500 \times 10 \times 10^{-12}}{2\pi\left(50+50\right)\times 10^{-9}}} = 45$。然而，在 PTCR 算法中，当新的节点请求加入到网络中时（见算法 6.4），或者当节点需要更新它的簇头时（见算法 6.5），考虑到物理位置、边境节点、能量等因素，这些节点可能并不是加入到一个已知的簇。因为它们可能将最靠近自己的一般节点当做父节点，用于转发信息。而这些父节点充当了簇头的作用，在算法中也认定为簇头。随着网络中活节点的密度越来越稀疏、簇头更新速率越来越频繁，父节点数目也越来越多。这就是为什么在 PTCR 算法中，网络的初始簇头数目（112）大于最优簇头数目（45）的原因。

8.4 基于虚拟势场的移动传感网 *k* 栅栏覆盖性能评估

Remark. 由图 8.7 关于网络使用 PTCR 算法后的网络拓扑结构所知，为了与所提的 DH-μTESLA 协议相结合，我们需要对第七章中虚拟势场下的移动传感网 *k* 栅栏覆盖进行修正。

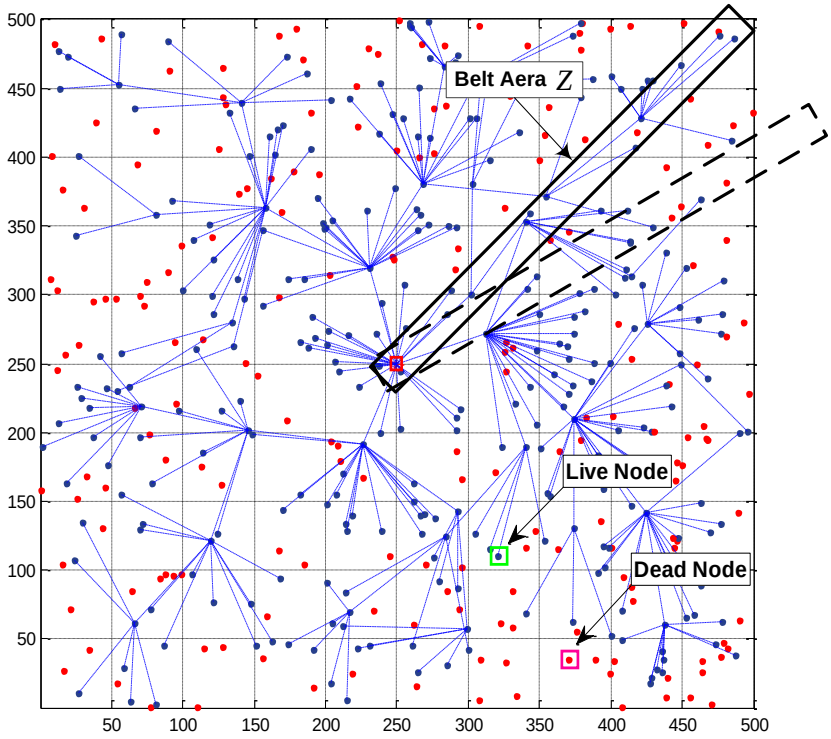

图 8. 12 PTCR 算法下网络拓扑分成多个带状区域

首先，我们以基站为中心，以大小为 Z 的带状区域为轴，将网络拆分成多个带状区域（可能会有重叠），直到网络中每个活节点都至少属于一个带状区域。如图 8.12 所示。

并且，我们做下简化和类比，将网络中节点的死亡、新节点的加入等看做是节点的运动。而且死亡和加入的操作的频率与类比后移动节点的速度相关。这样，为了数学上的简易性，我们将仿真器的配置信息设置如表 8.9 所示。

表 8.9 移动传感网络默认参数配置信息

参数名称	参数值
带状区域的面积	$Z = 250\sqrt{2}\times 10\sqrt{2} = 353.6\times 14.1$
传感节点的密度	$n_Z = 0.0012$
传感节点与入侵者速度比值	1:1
感知半径	$R = \sqrt{\varepsilon_{fs}/\varepsilon_{mp}} = 100\sqrt{130}/13 = 87.7$
入侵节点初始速度	10
传感器质量	$m_S = 1$
入侵者质量	$m_I = 1$

第七章中所提出的的移动模型的主要的目的在于增强无线传感网的覆盖，并且从一个新的角度来讨论它的一些动态属性（节点更新频率）。下面，我们将分别对基于虚拟势场的移动传感网 k 栅栏覆盖 (k-barrier coverage of MSNs in virtual potential field, $k$bc in VMSN)，静态传感网络的 k 栅栏覆盖 (k-barrier coverage in stationary WSNs, $k$bc in WSN)，一般移动传感网的 k 栅栏覆盖[10] (k-barrier coverage in general MSNs, $k$bc in gMSN)进行性能分析比较。

在相同场景下，我们执行静态传感网络栅栏覆盖（$k$bc in WSN）以及虚拟势场下移动传感网络栅栏覆盖（$k$bc in VMSN）100 次。分别考虑在网络中传感节点密度分别为 $n_Z = 0.0012/\mathrm{m}^2$，$0.0016/\mathrm{m}^2$ 以及 $0.0020/\mathrm{m}^2$ 时，这两种 k 栅栏覆盖的平均概率密度函数。如图 8.13 所示，概率曲线是 100 次如图 8.12 不同角度的带状区域的平均情况。纵向上，我们可以看出，相同条件下，$k$bc in VMSN 的概率总是大于 $k$bc in WSN 的。横向上，在相同的覆盖率的条件下，$k$bc in VMSN 中的 k 值也总是大于 $k$bc in WSN 中的 k 值。因此，我们可以得出传感节点的移动性能够弥补其数量缺失的不足，并且能够增强栅栏覆盖的效果。

在相同场景下，我们执行两种不同移动模型的 k 栅栏覆盖静态算法（$k$bc in gMSN 和 $k$bc in VMSN）100 次。分别考虑在网络中传感节点密度分别为 $n_Z = 0.0012/\mathrm{m}^2$，$0.0016/\mathrm{m}^2$ 以及 $0.0020/\mathrm{m}^2$ 时，这两种 k 栅栏覆盖的平均概率密度函数。如图 8.14 所示，概率曲线是 100 次仿真下的平均情况。当 k 值不变时，概率值随着传感节点密度的变大而变大。并且图 8.14 显示了，在相同的情况下，$k$bc in VMSN

的性能总是好于 *k*bc in gMSN。

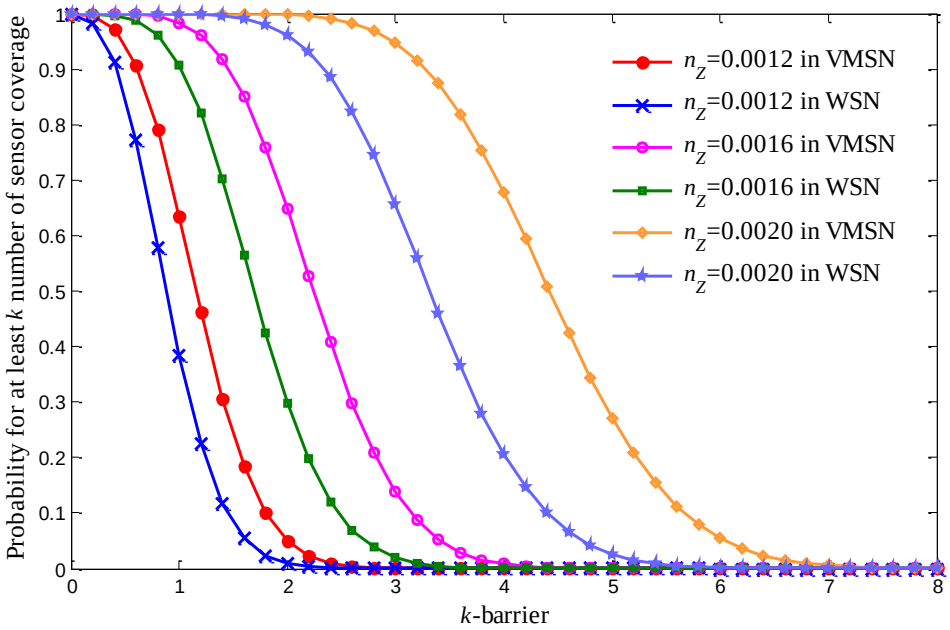

图 8. 13 *k* 栅栏覆盖在静态传感器网络以及虚拟势场移动传感器网络中的概率密度函数

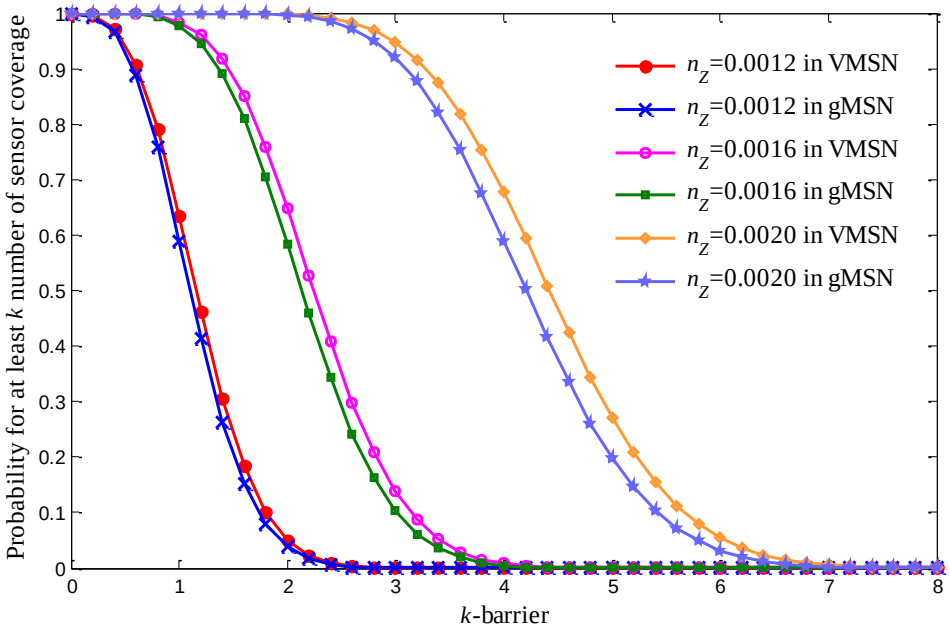

图 8. 14 *k* 栅栏覆盖在两种移动传感器网络中的概率密度函数

当传感节点变得稀疏时，虚拟势场下移动传感网中的弹性碰撞将会减少重叠区域，这样就意味着增加了覆盖率。因此，我们可以得出这样的结论：本章所提出的移动模型能够改进栅栏覆盖的性能。另一方面，点电荷模型也更能准确地描述入侵者的入侵轨迹。

由以上分析可知，基于虚拟势场的移动传感网栅栏覆盖相比其余两种具有较大的检测概率。虽然它是动态的网络，并且不一定能反映网络中节点更新换代的频率，但是，至少，以动态的方法来描述网络拓扑结构的变化，相比较于静态的处理方法更符合 DH-μTESLA 协议的应用场景。并且，我们可以看到，节点的更新速率越快的话，那么网络的入侵检测的能力越强。

8.5 本章小节

本章分别从四个方面来描述整个广播认证协议和入侵检测的性能。分别是：SRTC-TD、AdlCBF、PTCR、以及基于虚拟势场的栅栏覆盖。

其中，在 SRHC-TD 的性能分析中，又分别从安全性、容丢包性/容错性、复杂性三方面来分析；得出 SRHC-TD 安全性高、容错性强、复杂度低的优点。

然后将 AdlCBF 与直接存储方式以及使用 CBF 的认证方案相比较；得出 AdlCBF 的身份认证方案占用空间小、时间高效、假阳性概率低、安全强度高的优点。

接着在 PTCR 的性能分析中又分别考虑了网络的能耗、生存时间、簇头数目。得出，PTCR 算法在前 800 轮性能较强，但是 800 轮后性能突降。说明了 DH-μTESLA 协议更合适大规模的节点密度高的无线传感网络。

最后，由网络分簇拓扑结构的特点更改了栅栏覆盖的配置。将整个网络分成多个带状区域，并求出平均概率密度。得出基于虚拟势场的移动传感网栅栏覆盖性能更优。并且，用节点的运动来近似描述节点的更新换代，从另一个角度分析，得出该替代方法在入侵检测方面效率更高。

第九章 总结与展望

本文虽然名为广播认证协议及入侵检测研究。但是具体从四个方面对现有的广播认证方案及栅栏覆盖进行了改进。得出了如上面八章所描述的一个安全系统。

其中广播认证协议分为三个方面：可再生 Hash 链方案 SRTC-TD，AdlCBF 的身份认证机制，PTCR 分簇路由算法。而入侵检测方面主要是研究基于虚拟势场的移动传感网 *k* 栅栏覆盖方法。

首先，我们首先提出了可再生 Hash 链方案 SRTC-TD，用以克服传统 Hash 链方案的矛盾弊端。SRHC-TD 自更新的属性无缝地集成在 DH-μTESLA 协议来实现广播认证。而且，与现有的 RHCs 方案的比较可以看出，SRHC-TD 具有相对较低的复杂度以及较高的安全强度。

其次，我们使用了 AdlCBF 的身份认证机制。它相较于 μTESLA 和多级 μTESLA 更加安全完整。借用 *d*-left 可计数型布隆过滤器，本文的身份认证方案具备很多优点，比如更低的内存开销、更方便的扩展性、更便易的查询方式，以及更低的能量消耗。

而且，我们也提出了 PTCR 分簇路由算法，它可扩大 DH-μTESLA 协议的网络范围。层簇式的结构使网络的拓扑更加合理。并且，轮转的簇头选举算法能够均衡网络能量的开销。它们使得 DH-μTESLA 协议适用于无线传感器网络中的大型的安全应用。

最后，我们研究了虚拟势场下移动传感网的入侵检测的问题。将节点的更新换代替换成节点的运动。使用动态相似性，我们利用弹性碰撞模型以及点电荷模型分别量化了传感节点以及入侵者的移动性。然后，我们又量化了 *k* 栅栏覆盖概率与传感节点以及入侵者的动态性之间的关系。接着，我们比较三种不同网络（静态无线传感网络，一般移动传感网络，虚拟势场下的移动传感网络）下的覆盖性能。仿真结果表明本文所提出的模型能够从三个方面改进栅栏覆盖的性能：1) 弥补了传感节点数量缺失的不足；2) 更加准确地描述了入侵者的入侵运动轨迹；3) 保证了更大的覆盖范围。

然而，我们仍有一些问题需要在将来的工作中解决：

首先，在 AdlCBF 方案中，采用 ECDSA 算法的计算开销相对较大。这就需要假设基站具备足够强大的计算能力。使用轻量级的加解密系统也许更加适用于实现广播认证。但是这个在目前为止属于愿景的级别，暂时为止，ECDSA 在安全性及能耗上面的妥协平衡性是最优的方案。

其次，PTCR 算法中簇头节点较多，也许将来的工作可以更加合理地处理那些偏远的、不接近簇的节点。而且，由性能分析可知，PTCR 算法是造成网络长期运行后变得越来越稀疏的原因。所以，

也许在将来需要对算法进行改进，使得网络中活节点的分布更加集中。

并且，在处理大型无线传感器网络时，我们可以引入多个基站来处理网络中的节点的广播认证过程。将多个基站分布在网络的各处，各基站之间远程通信，而基站所管辖的区域仍然与单个基站的无线传感器网络的广播认证类似。

而栅栏覆盖方面，如果能够量化节点的更新换代，例如引入随机过程中的泊松分布、排队论中M/G/1 模型等概念描述，就更好了。所以，在今后的工作中，可以试着从这个方面思考，继续研究。

参考文献

[1] Vijay Raman and Indranil Gupta, “Performance tradeoffs among percolation-based broadcast protocols in wireless sensor networks”, International Journal of Parallel, Emergent and Distributed Systems, iFirst article, 2010, 1–22.

[2] Perrig. A., Szewczyk. R., Wen. V., Culler. D., Tygar. D. SPINS: Security protocols for sensor networks. In Proceedings of Seventh Annual International Conference on Mobile Computing and Networks.

[3] Donggang Liu, Peng Ning. Multi-Level μTESLA: Broadcast Authentication for Distributed Sensor Networks [J]. ACM Transaction on Embedded Computing System (TECS), 2004, 3(4): 800-836.

[4] Perrig. A., Canetti. R., Song. D., Tygar. D. Efficient authentication and signing of multicast streams over lossy channels. In Proceedings of the 2000 IEEE Symposium on Security and Privacy.

[5] Wenhuei Chen. A bootstrapping scheme for inter-sensor authentication within sensor networks[J]. Communications Letters, IEEE, Oct. 2005, 9(10):945-947.

[6] Liu, D. Practical broadcast authentication in sensor networks[C]. Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on 17-21 July 2005: 118-129.

[7] Kui Ren. Multi-User Broadcast Authentication in Wireless Sensor Networks[C]. Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on 18-21 June 2007: 223-232.

[8] S. Kumar, T.H. Lai, and A. Arora, “Barrier coverage with wireless sensors,” in Proc. ACM MobiCom, 2005, pp. 284-298.

[9] C. Shen, W. Cheng, X. Liao, and S. Peng, “Barrier coverage with mobile sensors,” in Proc. IEEE Int. Symp. Parallel Archit., Algor., Netw., Jul. 2008, pp. 99-104.

[10] G.Y. Keung, B. Li, and Q. Zhang, “The intrusion detection in mobile sensor network,” IEEE/ACM Trans. on Networking. Vol. 20, No. 4, Aug. 2012, pp. 1152-1161.

[11] A. Howard, M.J. Matarić, and G.S. Sukhatme, “Mobile sensor network deployment using potential field: A distributed scalable solution to the area coverage problem,” in Proc. of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02), Fukuoka, Japan, Jun. 2002, pp. 299-308.

[12] Goyal V. How to re-initialize a hash chain. URL: http://eprint.iacr.org/ 2004/097.pdf

[13] Zhao Yuan-chao, Li Dao-ben. An Elegant Construction of Re-initializable Hash Chains[J]. Journal of Electronics & Information Technology, 2006, 28(9): 1717-1720. (in Chinese)

[14] Haojun Zhang, Yuefei Zhu. Self-Updating Hash Chains and Their Implementations[J]. Lecture Notes in Computer Science, 2006, 4255:387-397.

[15] Haojun Zhang. A Novel Self-Renewal Hash Chain and Its Implementation[C]. Embedded and Ubiquitous Computing, 2008.EUC '08. IEEE/IFIP International Conference on Shanghai, 2008, 2:144-149.

[16] Minqing Zhang, Bin Dong, Xiaoyuan Yang. A New Self-Updating Hash Chain Scheme[C]. Computational Intelligence and Security, 2009.CIS '09. International Conference on Beijing, 2009, 2:315-318.

[17] Wei Zhang.Self-Updating Hash Chains Based on Erasure Coding[C]. Computer, Mechatronics, Control and Electronic Engineering(CMCE), 2010 International Conference onChangchun, 2010, 6:173-175.

[18] Xiaoyuan Yang, Jingjing Wang, Jiayong Chen, Xiaozhong Pan. A Self-Renewal Hash Chain Scheme Based on Fair Exchange Idea(SRHC-FEI)[C].Computer Science and Information Technology(ICCSIT), 2010 3rd IEEE International Conference on Chengdu, 2010, 8:152-156.

[19] Bloom BH. Space/Time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970,13(7):422−426. [doi: 10.1145/362686.362692]

[20] Li Fan, Pei Cao, Jussara Almeida, Andrei Z. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol[J]. IEEE/ACM Transactions on Networking (TON), June 2000, 8(3):281-293.

[21] http://en.wikipedia.org/wiki/Bloom_filter#Counting_filters

[22] Ju-Hyung Son, Haiyun Luo, Seung-Woo Seo. Denial of service attack-resistant flooding authentication in wireless sensor networks[J]. Computer Communications, August, 2010, 33(13): 1531-1542.

[23] E. Ayday, F. Fekri. A secure broadcasting scheme to provide availability, reliability and authentication for wireless sensor networks[J]. Ad Hoc Networks, September, 2012, 10(7): 1278-1290.

[24] A. Shokrollahi, Raptor codes, IEEE Transactions on Information Theory 52 (6), 2006, pp. 2551–2567.

[25] M. Luby, LT codes, in: Proceedings of IEEE Symposium on Foundations of Computer Science, 2002, pp. 271–280.

[26] Andrei Broder, Michael Mitzenmacher. Network Applications of Bloom Filters: A Survey[J]. Internet Mathematics, 2003, 1(4):485-509.
[27] Flavio Bonomi, Michael Mitzenmacher, etc. An Improved Construction for Counting Bloom Filters[C]. ESA'06 Proceedings of the 14th conference on Annual European Symposium. London, UK, 2006, 14:684-695.
[28] Heinzelman, W.R. Energy-Efficient Communication Protocol for Wireless Microsensor Networks[C]. System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on 4-7 Jan. 2000.
[29] Heinzelman, W.B. An Application-Specific Protocol Architecture for Wireless Microsensor Networks[J]. Wireless Communications, IEEE Transactions on Oct 2002, 1(4): 660-670.
[30] Georgios Smaragdakis, Ibrahim Matta, Azer Bestavros. SEP: A Stable Election Protocol for clustered heterogeneous wireless sensor networks. In Proc. of the Int'l Workshop on SANPA (2004).
[31] Li Qing, Qingxin Zhu, Mingwen Wang. Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks[J]. Computer Communications. August, 2006, 29(12): 2230-2237.
[32] Do-Seong Kim. Self-Organization Routing Protocol Supporting Mobile Nodes for Wireless Sensor Network[C]. Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on 20-24 June 2006, 2:622-626.
[33] D. W. Gage, "Command control for many-robot systems," in Proc. 19th Annu. AUVS Tech. Symp., June 1992, pp. 28-34.
[34] P. Balister, B. Bollobas, A. Sarkar, and S. Kumar, "Reliable density estimates for coverage and connectivity in thin strips of finite length," in ACM MobiCom, 2007.
[35] B. Liu, O. Dousse, J. Wang, and A. Saipulla, "Strong barrier coverage of wireless sensor networks," in ACM MobiHoc, 2008.
[36] A. Chen, T. Lai, and D. Xuan, "Measuring and guaranteeing quality of barrier-coverage in wireless sensor networks," in ACM MobiHoc, 2008.
[37] J.K. Li, J.M. Chen, and T.H. Lai, "Energy-efficient intrusion detection with a barrier of probabilistic sensors," in Proc. IEEE INFOCOM, 2012, pp. 118-126.
[38] S. Kumar, T. Lai, M. Posner, and P. Sinha, "Maximizing the lifetime of a barrier of wireless sensors," IEEE Trans. on Mobile Computing, 9(8): 1161-1172, a392010.
[39] A. Chen, S. Kumar, and T.H. Lai, "Local barrier coverage in wireless sensor networks," IEEE Trans. on Mobile Computing, 9(4), 2010.
[40] A. Chen, S. Kumar, and T. Lai, "Designing localized algorithms for barrier coverage," in ACM MobiCom, 2007.
[41] Y. Dong, W.K. Hon, and D.K.Y. Yau, "On area of interest coverage in surveillance mobile sensor networks," in Proc. IEEE IWQoS, Jun. 2007, pp. 87-90.
[42] T.L. Chin, P. Ramanathan, and K.K. Saluja, "Analytic modeling of detection latency in mobile sensor networks," in Proc. ACM IPSN, Apr. 2006, pp. 194-201.
[43] G. Wang, G. Cao, and T. Porta, "Movement-assisted sensor deployment," in Proc. IEEE INFOCOM, Mar. 2004, vol. 4, pp. 2469-2479.
[44] B. Bhattacharya, B. Burmester, Y. Hu, E. Kranakis, Q. Shi and A.Wiese, "Optimal movement of mobile sensors for barrier coverage of a planar region," in Proc. Int. Conf. Combin. Optimiz. Appl., Aug. 2008, pp. 103-115
[45] Ulutas M, Nabiyev V.V, Ulutas G. A new secret image sharing technique based on Asmuth Bloom's scheme[C]. Application of Information and Communication Technologies, AICT 2009 International Conference on 14-16 Oct 2009.
[46] LeinHarn, Changlu Lin. Strong (n, t, n) verifiable secret sharing scheme[J]. Information Sciences 180 (2010) 3059-3064.
[47] Li Huan, Cao Jie-rui, Xiong Jun-wu. Constructing Optimal Clustering Architecture for Maximizing Lifetime in Large Scale Wireless Sensor Networks[C]. IEEE International Conference on Parallel and Distributed Systems (ICPADS), 2009: 182-189.

附录 1 攻读硕士学位期间撰写的论文

[1] **Ting Dai**, Haiping Huang, *et al*., "Research on Migration Strategy of Mobile Agent in Wireless Sensor Networks", International Journal of Distributed Sensor Networks, Vol. 2013, Article ID 642986, 13 pages, 2013.

[2] 黄海平, **戴庭**, 等, 一种基于(*t*, *n*)门限和划分树的可再生 Hash 链构造方案, 通信学报, 第 34 卷, 第 4 期, 2013 年 4 月, 70-81 页.
Haiping Huang, **Ting Dai**, *et al*., "Novel Self-renewal Hash Chain based on (*t*, *n*) Threshold and Division Tree", Journal of Communications, Vol. 34, No. 4, pp. 70-81, Apr. 2013. (in Chinese)

[3] **Ting Dai**, Haiping Huang, *et al*., "Novel Self-Renewal Hash Chain based on Ito-Saito-Nishizeki Secret Sharing Scheme", The Journal of China Universities of Posts and Telecommunications, Vol. 19(Suppl. 2), pp. 122-127, Oct. 2012.

[4] Li Liu, **Ting Dai**, *et al*., "Simulation of the Park Behaviors in a Shopping Mall of Dalian", Advanced Material Research, Vol. 317-319, pp. 2133-2137, Aug. 2011.

附录 2 攻读硕士学位期间授权的专利

[1] 黄海平，**戴庭** 等. 一种基于标签传感网的集装箱物流跟踪与定位方法. 专利号 ZL201110157187.2, 申请日 2011 年 6 月 13 日, 授权日 2013 年 11 月 27 日.
Haiping, Huang. **Ting**, **Dai**. et al. 2013. A Method of Tracking and Locating Container Logistics based on Radio Frequency Sensor Networks. Chinese Patent Application ZL201110157187.2, filed June 13, 2011, and issued November 27, 2013.

[2] 黄海平，**戴庭** 等. 基于自动机和生命游戏的无线传感器网络广播认证协议. 专利号 ZL201110126518.6, 申请日 2011 年 5 月 1 日, 授权日 2013 年 9 月 25 日.
Haiping, Huang. **Ting**, **Dai**. et al. 2013. A Broadcast Authentication Method based on Deterministic Finite Automation and Game of Life in Wireless Sensor Networks. Chinese Patent Application ZL201110126518.6, filed May 13, 2011, and issued September 25, 2013.

附录 3 攻读硕士学位期间参加的科研项目

[1] 2013 年国家自然科学基金，基于隐私保护的无线传感器网络数据安全关键技术研究(61373138)；

[2] 2010 年国家自然科学青年基金，基于密钥技术的多类型混合无线传感器网络安全机制研究，(61003039)；

[3] 2010 年度省科技支撑计划，面向 QoS 的传感网安全可信组网及软件应用技术，(BE2010198)。

附录 4 图表清单

致谢

很幸运，有机会读研。很幸运，把青春的 7 年时光献给了南邮。都快要毕业了，可讽刺的是，我开始怀疑当初自己保研的目的。是喜欢读书，还是逃避进入社会，或者是想证明自己马马虎虎算个学霸，也或许只是随大溜吧。目的不纯，但是我真心感谢这近三年的成长。终于让我悟到了一些“愚见”和“真知”；让我不再惧怕，有勇气重拾起年轻时的梦想。

我的梦想很简单也很明确。虽然我曾一度怀疑自己是否有能力去实现。但是，感谢这三年的沉淀，我终于还是“厚积而薄发”了一回。这次，我就真的朝着出国的梦冲了。

硕士研究生三年，要感谢的人和事太多。

首先，我要感谢爸妈的优生优育，让我有头脑继续读书。

然后，我要感谢导师的辛劳栽培，让我有资格继续读书。

其次，我要感谢基友的相互陪伴，让我有欢乐继续读书。

还有，我要感谢闺蜜的泼辣毒舌，让我有自知继续读书。

接着，我要感谢在我二十五年的岁月里从没出现过的男票和女票们，让我可以专心继续读书。

以及，那些年不解我风情不给我机会的萌叔软妹们，感谢你们让我狠下心来继续读书。

当然，还有那些偷偷爱慕我而从不向我表白的人们，感谢你们让我继续死读书。

至于，那些我曾经因为脑子犯抽而只会读书不会谈情从而伤害过的大叔御姐正太萝莉们，感谢你们不纠缠我让我继续读书。

对了，那些曾经看低我的人，对不起，没有你们我还是会照样有志气继续读书。

最后，我要感谢祖国的房价之高、雾霾之重，让我更加坚定决心去国外继续读书。

差点忘了，还要感谢自己，感谢自己这么心甘情愿死心塌地地继续读书!!!

戴庭

2013 年 2 月 17 日凌晨 3 点 19 分

于南邮学二-410