摘 要

无线 Ad-hoc 网络是由一组带有无线收发装置的移动主机节点组成的多跳、没有固定基站和中心节点的临时性自治网络系统。由于 Ad-hoc 网络的临时、动态的拓扑结构，难以使用密钥分配中心分配密钥，从而使得其安全性备受关注。本文围绕无线 Ad-hoc 网络的安全问题，致力于设计与实现一个基于对等通信协议的网络安全平台，并就部分关键问题展开研究。

本文首先对 M.Steiner 等提出的扩展的 Diffie-Hellman 协议（GDH.2）进行分析，指出了它存在的一些安全隐患，并针对其漏洞讨论了一个由西南交通大学移动通信研究所的研究人员提出的改进“协商式”会议密钥分配协议--M-GDH.2 协议。该协议对通信参与者的身份以及传输数据的完整性进行了验证，还对最后产生的会议密钥进行了确认，以增加很小的计算和通信负荷为代价，使得系统安全性能得到了提高。

接着，论文以 Visual C++6.0 为开发工具，设计开发了一个基于对等通信的网络安全平台。该平台以 M-GDH.2 协议为开发重点，实现了网络对等连接、动态显示网络拓扑结构、密钥生成、安全通信以及联机帮助等功能。该平台适用于在没有密钥分配中心的情况下，几个地位平等的用户临时的利用便携式电脑安全的进行信息交流。

由于 M-GDH.2 协议仅针对三个用户，为了使本网络安全平台更为实用，作者以适当降低安全性为代价，针对多用户设计了一个密钥分发方案，参与通信的每个用户都有能力产生一个随机密钥，并分发给其他用户。

最后，论文给出了研究工作总结，并对系统的改进和扩展提出了研究思路。

关键词：Ad-hoc 网络；安全平台；会议密钥

Abstract

A wireless Ad-hoc network is a collection of mobile hosts with wireless transceiver. It is a multi-hop and instant self-organization network without the aid of any established infrastructure or centralized administration. It is difficult to set up a center to distribute key because of its instant、dynamically changing topology, so the security of wireless Ad-hoc networks has become a hot spot . In this thesis, the security issues in wireless Ad-hoc networks are investigated, aiming at designing and implementing a secure communication platform for peer groups. Some related key problems are also studied.

Firstly, the thesis analyses the GDH.2 (Group Diffie-Hellman) protocol, and points out its deficiencies, then discusses an extended contributory conference key agreement protocol--MGDH.2 proposed by the researchers at the Institute of Mobile Communication in Southwest Jiaotong University. Compared with GDH.2, the identity of the participants and the integrality of transmitted data are verified in the scheme, the final conference key is affirmed. Moreover, the security of the new protocol is improved at a slight cost in computation and communication overheads.

Secondly, a secure communication platform based on peer to peer network is designed and implemented, using Microsoft Visual C++6.0 as development tool. The platform puts emphasis on the implementation of M-GDH.2 protocol, and realizes such functions as follows: peer to peer network connection、dynamically displaying network topology 、key generation 、secure communication、help-online and so on. The platform is suitable to the condition that a small group of people would like to exchange the information instantly among their laptop computers without the key distribution center.

Because M-GDH.2 is only designed for three users, in order to make the platform more practical, the author designs a key distribution strategy that is

appropriated for multiuser environment at a slight cost of reducing the security, which allows each of the participants has the ability to generate a random key and distribute it to the others.

Finally, the thesis summarizes the work done during the MSc research, and puts forward the possible extensions and improvements to the existing system.

Key words: Ad-hoc network; secure platform; conference key

第一章 绪 论

无线 Ad-hoc 网络是由一组自主的无线节点或终端相互结合而形成的，独立于固定的基础设施并且采用分布式管理的网络，是一种自创造、自组织和自管理网络[1]。但由于无线链路容易受到链路层的攻击，网络拓扑和成员经常改变使得信任关系经常变化，因而无线 Ad-hoc 网络存在严重的安全性问题。本文围绕无线 Ad-hoc 网络的安全问题，致力于设计与实现一个基于对等通信协议的网络安全平台，并就部分关键问题展开研究。

1.1 无线 Ad-hoc 通信网络概述

无线 Ad-hoc 网络是由移动节点组成的不依赖于任何固定网络设施的临时性自治系统，与传统的蜂窝网络相比，它没有基站，所有节点分布式运行，具有路由器的功能，负责发现和维护到其它节点的路由，向邻节点发送或转发分组。这种网络既可以单独运行，也可以通过网关接入到有线骨干网络（如因特网）[2]。

Ad-hoc 网络的起源可以追溯到 1968 年的 ALOHA 网络和 1973 年美国国防部高级研究计划署 (简称 DARPA) 开始研究的分组无线电网络。IEEE 在开发 IEEE802.11 标准时，将分组无线电网络改称为 Ad-hoc 网络。Ad-hoc 来源于拉丁语，字面上的意思是“为特定目的或场合的”或“仅为这种情况的”。当时分组无线电网络已经用于大规模的军事和救援行动中，采用新的名字，IEEE 希望 Ad-hoc 网络成为为特定目的而临时组建并短期存在的网络。需要指出的是，IEEE802.11 标准定义的 Ad-hoc 网络仅由那些通过无线媒质能够互相进行直接通信的站点组成的网络，即独立的基本服务集（IBSS）。IBSS 没有接入点，为单跳 Ad-hoc 网络[3]，但是目前研究的 Ad-hoc 网络通常是多跳的。1997 年成立了无线 Ad-hoc 网络 MANET（Mobile Ad-hoc NET

work）工作组，专门负责具有数百个节点的无线 Ad-hoc 网络的路由算法的研究和开发，并制定相应的标准。MANET 工作组的工作成绩斐然，已经制定了十几个 Internet 草案标准。

根据节点是否移动，可以将无线 Ad-hoc 网络分为无线 Ad-hoc 网络和传感器网络。在无线 Ad-hoc 网络中，各个无线节点都可以自由移动。事实上，很多文献常常把无线 Ad-hoc 网络等同于移动 Ad-hoc 网络。在传感器网络中，各个无线节点静态的随机分布于某一区域。传感器负责收集区域内的声音、电磁或地震信号等多种信息，将它们发送到网关节点。网关具有更大的处理能力，能够进一步处理信息，或有更大的发送范围，可以将信息送往某个大型网络，使远程用户能够检索到该信息。

1.1.1 无线 Ad-hoc 网络特点

由于无线 Ad-hoc 网络具有其特殊的应用环境，因此它具有区别于其它传统网络的特点[2][4]：

（1） **网络拓扑结构动态变化**

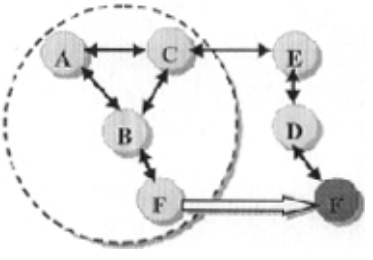


图 1-1：Ad-hoc 网络的拓扑变化

Ad-hoc 网络一个显著的特点是其网络拓扑经常地变化，如图 1-1 所示。由于所有节点的移动性，Ad-hoc 网络的拓扑将发生变化，当节点F移动到了节点B的无线通信范围外，却移动到了节点D的通信范围内时，F将可以和D建立直接的连接，而不能和B建立直接的连接，这时F可以通过D、E、C来和B建立连接。当然除了节点F其他的节点也可以移动，这样整个网络的拓扑就会不停地发生变化。

（2） **采用分布式控制方式**

此特点使得无线 Ad-hoc 网有别于常规移动通信网和无线局域网。因为后两者需要有中心站来进行控制。在无线 Ad-hoc 网中，不设专门的控制中心，把网络的控制功能分散配置到各节点，网络的建立和调整是通过各节点的有机配合实现的。即无线 Ad-hoc 网均衡了网中各节点的特殊性和重要性，从控制能力上看，各节点没有重要和次要之分，从而可防止一旦控制中心被破坏而引起全网瘫痪的危险，提高了网络的抗毁性。同时，网络也根据需要，选定一个或几个节点充当控制中心，并根据情况变化自动更换。

（3） **采用自适应组网技术**

网中各节点能相互协调的遵循一种自组织原则，自动探测网络的拓扑信息，自动选择传输路由，自动进行控制，把网中所有节点组成一个有机整体。即使网络发生动态变化或某些节点严重受损时，仍可迅速调整其拓扑结构以保持必要的通信能力。

（4） **不需要原有通信基础设施的支持**

无线 Ad-hoc 网络不依赖于原有通信基础设施，网内各节点本身就可组成一个完整的通信网络，可满足随时随地组网、使用的要求。

（5） **信道质量较差**

无线 Ad-hoc 网络采用无线传输技术作为物理层通信手段。无线信道由于其本身的物理特性，如衰减大、干扰大、多径效应等，信道质量比有线信道差得多。

（6） **用户终端能源受限**

在无线 Ad-hoc 网络中，用户终端通常为手持电脑、掌上电脑或笔记本电脑，这些装置虽然重量轻、移动性好，但主要靠电池供电，因此在进行网络系统设计时必须考虑到节电问题。

1.1.2 无线 Ad-hoc 网络研究现状

与传统的有线和蜂窝网络相比，Ad-hoc 网络没有基础设施，每个节点都可能随时进入和离开网络，整个网络分布式运行。然而，传统网络中对连接性和业务传输的基本需求，在 Ad-hoc 中也同样需要得到满足。目前关于 Ad-hoc 网络研究中的主要难点问题为路由协议、服务质量、信道访问控制协议、能量消费、节点移动性管理、网络互联、安全性等问题。

路由协议

开发良好的路由协议是建立 Ad-hoc 网络的首要问题，同时也是主要的研究热点和难点。传统的距离向量和链路状态路由协议并不适用于拓扑结构高度动态变化的 Ad-hoc 网络。理想的 Ad-hoc 网络的路由协议应该具有以下性能：分布式运行、无环路、按需运行、考虑安全性、高效地利用电池能量、支持单向链路、维护多条路由[5]。

Ad-hoc 网络的路由协议可以分为表驱动路由和按需路由两大类[6]。在表驱动路由协议(先应路由协议)中，每个节点试图维护到所有已知目的节点的路由表，节点之间周期性和在网络拓扑发生变化时交换路由信息，减少了获得路由的延迟，使源节点能够立即判断目的节点的可达性，但是消费了较多的网络资源，此外它完全浪费了一些资源来建立和重建那些根本没有被使用的路由。在按需路由协议(反应路由协议)中，节点不需要花费资源来维护无用的路由，但路由发现过程费用比较昂贵而且不可预测，路由延迟与先应路由协议中恒定的查表时间相比，更加多变[7,8,9]。

目前研究的最为深入的表驱动路由协议为 DSDV(Destination Sequenced Distance Vector)，按需路由协议为 DSR(Dynamic Source Routing)、AODV(Ad-hoc On-demand Distance Vector Routing)、TORA(Temporally Ordered Routing Algorithm)。其中 DSR 路由协议已经在装有 IEEE802．11WaveLAN 卡的笔记本电脑组成的 Ad-hoc 网络中实现。

此外开发了一些多播路由协议，但是所有路由协议至今还没有一个能完

全符合 Ad-hoc 网络的要求。

■ **服务质量 QoS**

大多数对无线通信网 QoS 问题的讨论采用的模型是单跳、有中心节点（基站）、有有线的基础设施支持。这种情况下主要的设计难点是在移动节点与中心节点之间的 QoS 连接以及移动节点从一个中心移动到另一个中心节点的平衡切换。与上面的模型不同，无线 Ad-hoc 是多跳、分布式控制。所以在无线 Ad-hoc 网络中，不仅要考虑单跳情况下的 QoS 保证，还要保证整个多跳路径上的 QoS,这面临三个问题[10,11]：

1. **信道质量差**：这个问题可以通过使用更强的编码方法、提高信号发射功率、或选择其它路经等方法来解决。然而，信道质量问题的解决通常以损失网络其它性能为代价：更强的编码会导致带宽减少；增大发射功率会增加分组碰撞的概率；选择其它路径会增加其它节点的负担等。这些情况都会造成网络进一步拥塞。
2. **信道访问存在竞争**：因为无线 Ad-hoc 网络的信道是共享的，而且多数信道访问机制是随机访问，还有隐藏终端、暴露终端等问题。若为了支持 QoS 而引入大量控制分组会带来分组碰撞的增加，使节点获得信道访问机会的概率变小，从而导致网络整体性能下降。
3. **拓扑结构动态变化**：这也给 QoS 支持带来了很大困难。要消除或减轻网络拓扑结构变化对 QoS 的影响需要 MAC 层的相应支持以及路由协议能够快速生成新的路径。

■ **信道访问控制协议**

无线 Ad-hoc 网络是无线共享信道，信道访问控制协议的性能对网络的性能有着极大的影响，目前主要有以下几种方式[12]：

1. **ALOHA 方式**：该方式实现算法简单，但传输易发生冲突，网络吞吐量不高。
2. **CSMA 方式**：该方式网络吞吐量较高，但要求传输时延小，电台的发射/接收状态转换时间短，否则将严重影响其性能。

3．CDMA 方式：该方式信道利用率高，网络吞吐量高，但对组播支持不好。

其它各种方式基本上都是以上方式的变种。现在信道访问控制的研究重点是提高信道利用率和网络吞吐量，支持广播、组播及功率控制等[13,14]。

■ **能量消费问题**

能量消费(功率控制)问题涉及到无线网络中的各层。节点能量消费可以分为通信费用和计算费用两部分；前者是指无线网络接口消费的能量。在 Ad-hoc 网络中，移动节点可以位于发射、接收和空闲(旁路)三种模式，其中发射模式的功率消费最大，空闲模式的功率消费最小，缺省模式为空闲模式[15]。后者指协议处理方面消费的能量。通常在这两者之间存在一个折衷，降低通信费用的技术可能增加计算费用，反之亦然。

在目的端能正确接收分组的前提下，减少节点的能量消费可以延长节点和网络的寿命；减少了对邻居节点的干扰，提高网络的吞吐量；减少了数据被窃听的可能性，提高了通信的安全性。

硬件层次的技术，如低功率的 CPU、显示器和能量有效的算法等都得到了应用。在物理层可以通过调整节点的发射功率来减少网络的能量消费；MAC 层的主要措施为减少数据发送的冲突，避免重传，和使进入睡眠状态[16]。在网络层，采用功率控制路由算法，而不是以最短跳数和最小延迟作为路由度量。

■ **网络互连问题**

无线 Ad-hoc 网络通常都是以一个“独立”的通信网络形式存在，即网络不与其它任何网络相连，所有通信都是在网络内部的两个节点之间产生的。但是，实际上，无线 Ad-hoc 网络也存在以下需求[17]：

1. 无线 Ad-hoc 网络中的节点需要访问有线网络中的资源，例如 Internet 上的 WWW 服务、FTP 服务等。
2. 位于不同 Ad-hoc 网络的节点之间的通信。

这就需要无线 Ad-hoc 网络具有与其它网络互连互通的能力。

■ 安全性问题

无线 Ad-hoc 网络存在以下安全性问题：无线链路使 Ad-hoc 网络容易受到链路层的攻击，包括被动窃听和主动假冒、信息重放和信息破坏；节点在敌方环境(如战场)漫游时缺乏物理保护，使网络容易受到已经泄密的内部节点(而不仅仅是外部节点)的攻击，采用分布式的网络体系结构可以提高 Ad-hoc 网络的生存能力；Ad-hoc 网络的拓扑和成员经常改变，节点间的信任关系经常变化。而且由于节点的能源有限，节能问题与实现复杂的加密算法互相矛盾[18,19]。

本文研究的重点就是无线 Ad-hoc 的安全问题，下面详细介绍。

1.2 无线 Ad-hoc 网络的安全问题

在 Ad-hoc 网中没有固定的基站或中心节点，所有节点都是移动的，网络的拓扑结构动态变化，节点之间通过无线信道相连，没有专门的路由器，由节点自身充当路由器。它没有命名服务、目录服务等网络功能。这就导致了在传统网络中的安全机制不再适用于 Ad-hoc 网，概括起来主要有三个原因[20,21]：

第一，传统网络中的加密和认证包括一个产生和分配密钥的密钥管理中心、一个确认密钥的认证机构、以及分发这些经过认证的公钥的目录服务。所有这些服务都是在大家彼此信任的前提下工作。而由于 Ad-hoc 网缺乏足够的物理保护、没有中心节点、节点的计算能力很低等特点，使得传统的加密和认证机制无法在 Ad-hoc 网中实现。

第二，传统网络中的防火墙技术用来保护网络内部与外界通信时的安全。由于所有进出该网络的数据都通过一个节点比如服务器转发，防火墙技术则在该节点上实现，用来控制对网络内部的访问、对外部隐藏网络的内部信息、检查送入文件的病毒等等。防火墙技术假设网络内部物理上是安全的。但是，由于 Ad-hoc 网的拓扑结构是动态变化的，而且没有中心节点，进出

该网络的数据可以通过其中任意节点转发，同时，网络内部的节点因缺乏足够的保护很可能被占领而导致攻击来自于网络内部，网络内部和外部的界限非常模糊。这样一来，防火墙技术显然不能应用于 Ad-hoc 网。

第三，Ad-hoc 网中，不仅拓扑结构、成员、信任关系是动态的，而且网络中产生和传输的数据也具有动态特点。它们包括节点的环境信息、有关网络变化的信息、群组会议交换的信息等等，都有很高的实时性要求。而传统网络服务中的数据库、文件系统和文档服务器等静态数据都不再适用。因此，基于静态配置的传统网络安全方案不能用于 Ad-hoc 网。

1.2.1 无线 Ad-hoc 网络的安全目标

无线 Ad-hoc 网络的安全目标与传统有线网的安全目标是一致的，主要包括以下几方面[20]：

- **可用性**：可用性就是指网络服务对用户而言必须是可用的，也就是确保网络节点在受到各种网络攻击时仍然能够提供相应的服务。这里的网络攻击主要是指拒绝服务攻击。在 Ad-hoc 网络中拒绝服务可以发生在任何一层上：在物理层和媒体接入层，攻击者可以通过无线干扰来扰乱物理信道；在网络层，攻击者可以攻击路由协议；在高层，攻击者可以攻击各种高层服务。针对 Ad-hoc 网络还有一种叫做“剥夺睡眠”的特殊攻击，这种攻击使得移动节点的电池很快耗尽，从而达到使其拒绝服务的目的。
- **机密性**：机密性保证相关信息不泄漏给未授权的用户或实体。由于 Ad-hoc 网络采用的是无线信道，所以更容易受到窃听攻击。所以在网络中传输的敏感信息，比如军事敏感信息，都要保证其机密性。特别是路由信息也要在一定程度上保证其机密性，因为在战场上路由信息的泄漏会使敌方能够判断出移动节点的标识和位置。
- **完整性**：完整性保证信息在传输的过程中没有被破坏或中断。这种破坏或中断包括网络上的恶意攻击和无线信号在传播过程中的衰减以及人为

干扰等。

- **认证**：一个移动节点需要通过认证来确保和它通信的对端就是真正的通信对端，也就是说要确认通信对端的身份。如果没有认证，那么网络攻击者就可以假冒网络中的某个节点来和其它的节点进行通信，那么他就可以获得那些未被授权的资源和敏感信息，并以此威胁整个网络的安全。
- **不可否认性**：不可否认性保证一个节点不能否认其发送出去的信息。这样就能保证一个移动节点不能抵赖它以前的行为。在战场上如果被占领的节点发送了错误的信息，那么收到该信息的移动节点就可以利用不可否认性来通知其它节点该节点已被占领。

为了实现上述目标，就需要对通信数据进行加密，一般的，为了减少Ad-hoc 网络节点的计算负荷和通信时间，通常采用对称密钥对通信数据进行加密，该密钥又称为会议密钥。如何有效、安全的分配会议密钥，已经引起密码学家的高度重视，下面我们就对其研究现状作一介绍。

1.2.2 密钥协商协议与国内外研究现状

一般地，密钥分配协议分为两类：第一类是“集中式”密钥分配协议，即会议密钥由一个可信任的第三方（TTP）单独产生并分配给各个用户，文献[22-25]均介绍了这类协议，该类协议的特点是协议设计简单，但不足之处是过分依赖可信任的第三方，容易受到攻击者的集中攻击，且可能给 TTP 带来计算量和通信量过大的“瓶颈”问题；第二类是“协商式” 密钥分配协议，即会议密钥由通信各方共同协商产生。目前，许多研究均集中在这一方面[27-52]。

Becker 和 Wille[44] 提出了两个具有不同拓扑结构的“协商式” 密钥分配协议，即“超立方体” 密钥分配协议和“章鱼” 密钥分配协议。Burmester 等人提出了一个“树型”密钥分配协议[45]， M.Steiner 等提出了三个“协商式” 密钥分配协议[50]，这些协议推广了 Diffie-Hellman 方案[26]（简称 DH

问题），即把两个通信实体扩展到多个。下面就简单介绍一下这些协议。

- “超立方体”协议

假设系统有 $n=2^d$ 个通信用户。每个用户对应“超立方体”的一个顶点，显然每个用户正好可以用一个 d 位的二进制地址表达。该协议共运行 d 轮，第 j 轮时，地址为 I 的用户将与地址为 $I \oplus 2^{j-1}$ 的用户执行一次 Diffie-Hellman 协议，经过 d 轮运行后，n 个用户间将拥有共同的会议密钥。图 1-2、图 1-3 是四个通信实体间的运行过程。其中 p 是大素数，a 是 Z_p 上的本原元，S_A、S_B、S_C 及 S_D 分别是通信实体 A、B、C、D 的秘密指数。

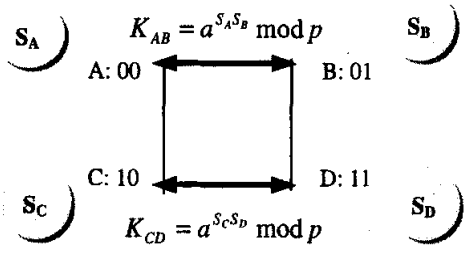


图 1-2“超立方体”协议第一轮执行过程

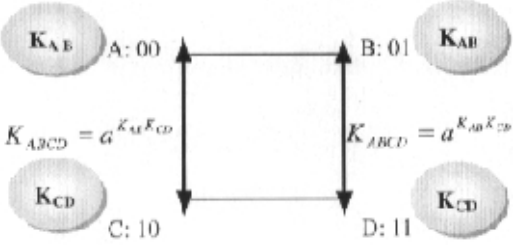


图 1-3“超立方体”协议第二轮执行过程

则经过协议的两轮运行后，四个通信实体间将拥有共同的会议密钥：

$$K_{ABCD}=a^{K_{AB}K_{CD}} \bmod p$$

- “章鱼”协议

“章鱼”协议的拓扑结构见图 1-4。

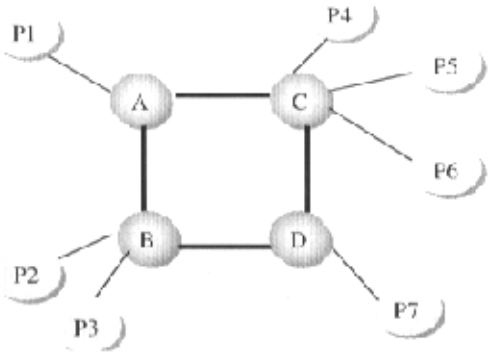


图 1-4 “章鱼”协议的拓扑结构

把系统中 n 个通信用户分为四个子集，每个子集有一个主结点。用 A、B、C、D 表示四个主结点，I_A、I_B、I_C、I_D 表示四个子集，$P_i(i\in[1,n-4])$ 属于某一个子集的附属结点。假设 q 是一个大素数，G 是一个阶为 q 的循环群，α 是 G 上的一个本原元，映射 $\Phi:G\to Z_q$。

第一步：

1、所有主结点 $X\in\{A,B,C,D\}$ 选择秘密指数 N_X，所有 $P_i(i\in[1,n-4])$ 选择秘密指数 N_{Pi}。

2、所有主结点 $X\in\{A,B,C,D\}$ 与附属结点 $P_i(i\in I_X)$ 执行一次 Diffie-Hellman 协议。A、B、C、D 分别拥有秘密：

$$a=K(I_A),b=K(I_B),c=K(I_C),d=K(I_D)$$

其中 $K(J)=\prod_{i\in J}\Phi(k_i),J\subseteq\{1,2,\ldots n-4\}$。

第二步：A、B、C、D 四个主结点间执行两轮“超立方体”协议。

第三步：A 发给附属结点 $P_i(i \in I_A)$ 数据：$\alpha^{K(I_B \cup I_A \setminus \{i\})}, \alpha^{\Phi(\alpha^{K(I_C \cup I_D)})}$

B 发给附属结点 $P_i(i \in I_B)$ 数据：$\alpha^{K(I_A \cup I_B \setminus \{i\})}, \alpha^{\Phi(\alpha^{K(I_C \cup I_D)})}$

C 发给附属结点 $P_i(i \in I_C)$ 数据：$\alpha^{K(I_D \cup I_C \setminus \{i\})}, \alpha^{\Phi(\alpha^{K(I_A \cup I_B)})}$

D 发给附属结点 $P_i(i \in I_D)$ 数据：$\alpha^{K(I_C \cup I_D \setminus \{i\})}, \alpha^{\Phi(\alpha^{K(I_A \cup I_B)})}$

$P_i(i \in [1, n-4])$ 收到数据后，用自己保留的秘密计算会议密钥：

$$K = \alpha^{\Phi(\alpha^{K(I_A \cup I_B)})\Phi(\alpha^{K(I_C \cup I_D)})}$$

■ **“树型”协议**

“树形”协议的拓扑结构见图 1-5。

假设 p，q 是一个大素数且 $q \mid (p-1)$，G 是 Z_p^* 的一个循环子群且阶为 q，α 是 G 的一个本原元，共有 n 个通信实体 M_i（i =1,2,...n）参加。结点 M_a 的孩子结点为 M_{2a} 和 M_{2a+1}，M_1 是根结点。

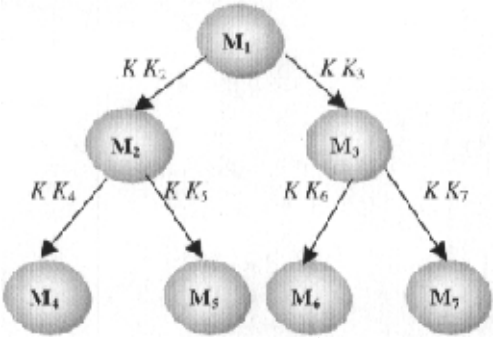


图 1-5 “树型”协议

Step 1:

1、M_a 选择秘密指数 $r_a \in Z_q^*$

2、M_a 计算 $z_a = \alpha^{r_a} \bmod p$

3、$M_a \to M_{\lfloor a/2 \rfloor} : z_a$ if $a > 1$

$M_a \to M_{2a} : z_a$ if $2a \le a$

$M_a \to M_{2a+1} : z_a$ if $2a+1 \le a$

Step 2: M_a 计算：

$K_a = z_{\lfloor a/2 \rfloor}^{r_a} \bmod p$ if $a > 1$

$K_{2a+i} = z_{2a+i}^{r_a} \bmod p$ for $i = 0,1$ if $2a+i \le n$

Step 3:

1、M_1 选择会议密钥 K

2、$M_1 \to M_{2+i} : Y_{2+i} = K \cdot K_{2+i} \bmod p$, for $i = 0,1$

3、$l = 0$

Step 4+l: 如果 M_a 在树中处于第 l 级($\lfloor \log_2 a \rfloor = l$)，则：

1、M_a 解密 Y_a 得到 K

2、$M_a \to M_{2a+i} : Y_{2a+i} = K \cdot K_{2a+i} \bmod p, i = 0,1$ if $2a+i \le n$

3、$l = l+1$

■ GDH 协议

M.Steiner 等提出了三个“协商式” 密钥分配协议，这些协议推广了 Diffie-Hellman 方案（简称 DH 问题），即把两个通信实体扩展到多个。其中，GDH.1 是为了说明设计思想而提出的一个协议，计算负荷和通信负荷均较大，没有实用价值。GDH.2 和 GDH.3 侧重点不一样，前者主要希望能尽可

能地减少协议执行过程中的通信负荷，而后者则希望能尽可能地减少协议执行过程中的计算负荷。

在第二章中将对 GDH.2 协议作详细介绍。

1.3 主要研究内容与思路

本文主要研究内容有：

- 分析研究无线 Ad-hoc 网络的特点及其存在的安全问题；
- 分析研究国内外有关会议密钥协商协议的发展状况；
- 重点分析 M.Steiner 等提出的 GDH.2 协议，指出其存在的安全隐患，然后分析由西南交通大学移动通信所提出的一种改进的多用户密钥协商协议 M-GDH.2，该协议增加了对用户身份的认证及对传输数据的完整性校验，弥补了 GDH.2 协议的缺陷；
- 在 Windows 98/2000 环境下，设计并用 Microsoft Visual C++6.0 实现一个对等通信的安全平台；

第二章 多用户密钥协商协议及其改进

由第一章给出的无线 Ad-hoc 网络特点可以看出，无线 Ad-hoc 网络具有临时性、没有固定的拓扑结构等特点，所以难以使用密钥分配中心分配密钥。那么 Ad-hoc 网络用户间该如何安全通信呢?目前，各国的密码专家都投入了极大的热情来研究该问题，也提出了一些密钥分配协议，如第一章所介绍的“超立方体”协议、“章鱼”协议、“树形”协议等，但由于这些协议拓扑结构过于复杂，实用性不强。M.Steiner 等提出的 GDH.2 协议拓扑结构为线形，结构简单，能够较好的适用于 Ad-hoc 网络，本章将详细分析该协议及其改进方法。

2.1 GDH.2 协议及其漏洞

2.1.1 GDH.2 协议

假设 p，q 是一个大素数且 $q|(p-1)$，G 是 Z_p^* 的一个循环子群且阶为 q，α 是 G 的一个本原元，共有 n 个通信实体 M_i（i =1,2,...n）参加，$N_i \in Z_q$ 是由通信实体 M_i 产生的一个随机数，$\prod(S)$ 表示集合 S 中所有元素的乘积。GDH.2 协议分为两个阶段，第一阶段是每个 M_i（从 M_1 开始）将临时计算值按下标顺序逐次往上传送，最后汇总到 M_n 处，M_n 的作用非常重要，相当于会议主席；第二阶段，M_n 将汇总的数据经过计算处理后，再广播发送到每个通信实体 M_i，M_i 用自己产生的随机数 N_i 对接收的数据进行一次模指数运算，最后得到会议密钥 $K_n = \alpha^{N_1 N_2 ... N_n} \bmod p$。具体的协议见图 2-1。

第一阶段（上传）：第 i 轮， $i \in [1, n-1]$

$$M_i \xrightarrow{\{\alpha^{\frac{N_1 \ldots N_i}{N_k}} \mid k \in [1,i]\}, \alpha^{N_1 N_2 \ldots N_i}} M_{i+1}$$

第二阶段（广播）：第 n 轮，

$$M_i \xleftarrow{\{\alpha^{\frac{N_1 \ldots N_n}{N_i}} \mid i \in [1,n]\}} M_n$$

注：上面的数据均是模 p 以后的计算值

图 2-1：GDH.2 协议

为了便于说明，用有四个通信实体的例子加以说明，如图 2-2 所示。

第一阶段：

Step 1 M_1 选择一个随机数 $N_1 \in Z_q^*$，并把 $\alpha^{N_1} \bmod p$ 发给 M_2。

Step 2 M_2 选择一个随机数 $N_2 \in Z_q^*$，M_2 计算：

$$\alpha^{N_2} \bmod p, \quad \alpha^{N_1 N_2} \bmod p$$

并把下面数据发给 M_3：

$$\alpha^{N_1} \bmod p, \quad \alpha^{N_2} \bmod p, \quad \alpha^{N_1 N_2} \bmod p$$

Step 3 M_3 选择一个随机数 $N_3 \in Z_q^*$，M_3 计算：

$$\alpha^{N_1 N_3} \bmod p, \quad \alpha^{N_2 N_3} \bmod p, \quad \alpha^{N_1 N_2 N_3} \bmod p$$

并把下面数据发给 M_4：

$$\alpha^{N_1 N_2} \bmod p, \ \alpha^{N_1 N_3} \bmod p, \quad \alpha^{N_2 N_3} \bmod p, \quad \alpha^{N_1 N_2 N_3} \bmod p$$

M_4 接收数据后，计算：

$$K_4 = (\alpha^{N_1 N_2 N_3})^{N_4} \bmod p = \alpha^{N_1 N_2 N_3 N_4} \bmod p$$

第二阶段：

M_4 把下列数据广播发送给 M_1，M_2，M_3。

$$\alpha^{N_1N_2N_4} \bmod p,\ \alpha^{N_1N_3N_4} \bmod p,\ \alpha^{N_2N_3N_4} \bmod p$$

M_1，M_2，M_3 接收数据后，分别作下列计算。

M_1：$K_4 = (\alpha^{N_2N_3N_4})^{N_1} \bmod p = \alpha^{N_1N_2N_3N_4} \bmod p$

M_2：$K_4 = (\alpha^{N_1N_3N_4})^{N_2} \bmod p = \alpha^{N_1N_2N_3N_4} \bmod p$

M_3：$K_4 = (\alpha^{N_1N_2N_4})^{N_3} \bmod p = \alpha^{N_1N_2N_3N_4} \bmod p$

GDH.2 协议经过一次运行后，通信实体 M_1，M_2，M_3，M_4 将共同拥有会议密钥 K_4。

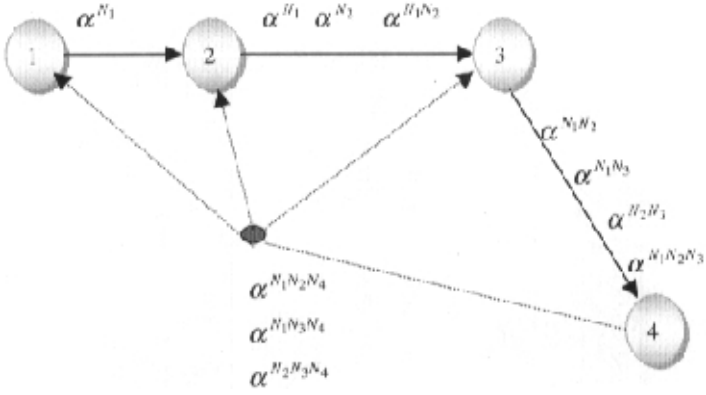
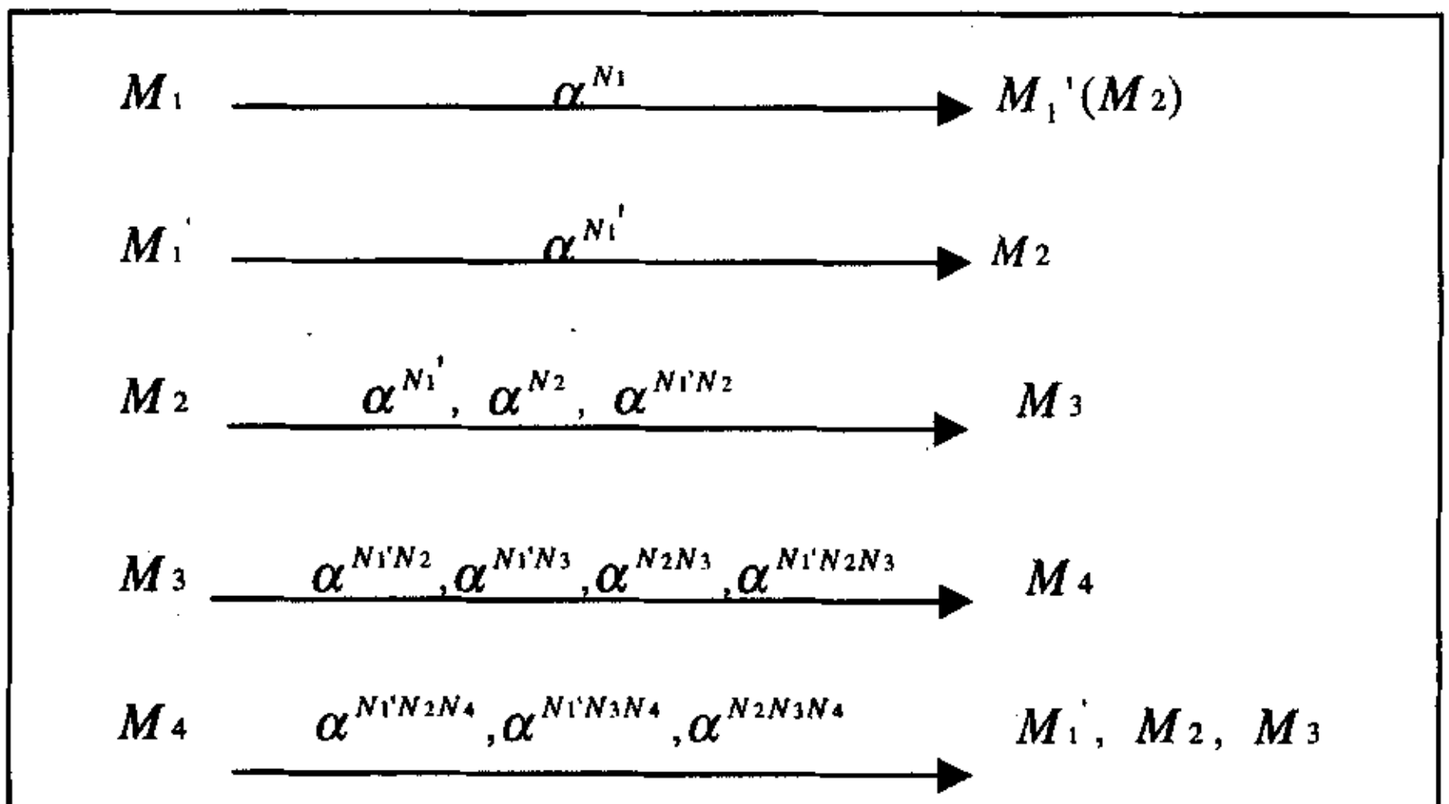


图 2-2 四个通信实体的 GDH.2 协议

2.1.2 对 GDH.2 协议的攻击

假设有一个主动攻击者有能力截获并修改 M_i (i =1,2,...n)之间传递的数据。为了便于说明，假设 GDH.2 协议的运行过程中有四个参与者 M_i（i

=1,2,3,4)，用 M_i' 表示冒充 M_i 的攻击者（当然也可以冒充其他参与者），N_i' 是攻击者 M_i' 产生的一个随机数。具体的攻击步骤见图 2-3。图 2-3 中前两步表示攻击者 M_i' 截获了 M_i 发给 M_2 的数据，并冒充 M_i 将修改后的数据发送给 M_2。



M_i' 对接收到的数据作一次模指数运算,便得到会议密钥:

$$K_n = (\alpha^{N_2N_3N_4})^{N_1'} \bmod p = \alpha^{N_1'N_2N_3N_4} \bmod p$$

这样，M_i' 可以冒充 M_1 与 M_2, M_3, M_4 秘密通信。

图 2-3　攻击 GDH.2 协议的例子

2.2 改进型 GDH.2 协议（M-GDH.2）

从上面的攻击可以看出，GDH.2 协议存在安全隐患的原因是没有对通信参与者 M_i 的身份以及传输数据的完整性进行验证，同时没有对最后产生的会议密钥进行确认。为此，西南交通大学移动通信研究所在文献[52]中提出了一个改进的密钥分配协议 M- GDH.2，下面我们就对其进行详细分析。

2.2.1 改进型 GDH.2 协议（M-GDH.2）

本协议所用符号如表 2-1 所示：

表 2-1 M-GDH.2 协议符号说明表

n	通信实体的个数	M_i	第 i 个通信实体（$i=1,2,\ldots n$）
p, q	都是大素数且 $q \mid (p-1)$	G	是 Z_p^* 的一个子群且阶为 q
α	G 的一个本原元	x_i	是 M_i 的秘密指数（$1 \leqslant x_i \leqslant q-1$）
N_i	M_i 产生的一个随机数且 $Ni \in Z_q$	k_i	$k_i=\alpha^{x_i} \bmod p$，是 M_i 的本地公钥
k_{ij}	$k_{ij} \equiv (\alpha^{x_i x_j} \bmod p) \bmod q$，是 M_i 和 $M_j (i \neq j)$ 的长期共享密钥。	k_{ij}^{-1}	是 $k_{ij} \in Z_q$ 的乘法逆元素
t_i	M_i 发送数据的时间	t_i^*	M_{i+1} 接收数据的时间

该协议分为三个阶段（如图 2-4 所示）：

第一阶段（交换 k_i）：第 i 轮， $i \in [1, n-1]$

$$M_i \xrightarrow{k_i} M_{i+1}$$

第二阶段（上传）：第 i 轮， $i \in [1, n-1]$

$$M_i \xrightarrow[h(\prod\{\alpha^{\frac{N_1 \ldots N_i}{N_k}} \mid k \in [1,i]\} \cdot \alpha^{N_1 N_2 \ldots N_i} \| t_i)]{\{\alpha^{\frac{N_1 \ldots N_i}{N_k} k_{i,i+1}} \mid k \in [1,i]\}, \alpha^{N_1 N_2 \ldots N_i k_{i,i+1}}, t_i} M_{i+1}$$

第三阶段（广播）：第 n 轮，

$$M_i \xleftarrow[h(\alpha^{N_1 \ldots N_n} \| t_n)]{\{\alpha^{\frac{N_1 \ldots N_n}{N_i} k_{in}} \mid i \in [1,n]\}, t_n} M_n$$

图 2-4：M-GDH.2 初始密钥协商协议

第一阶段：交换本地公钥（k_i）

每个 M_i（从 M_1 开始）将本地公钥 k_i 按下标 i 逐次后传，每个 M_{i+1} 收到 k_i 继续后传本地公钥 k_{i+1}；最后终止于发起者 M_1。

第二阶段：传递协商数据

每个 M_i（从 M_1 开始）计算协商数据并按下标 i 逐次后传，每个 M_{i+1} 收到协商数据后使用 k_{ij}^{-1} 进行验证。若确认数据来自 M_i 且没有被修改过，则继续计算协商数据并后传。否则终止协议运行。协商数据最后汇总到 M_n 处，M_n 的作用非常重要，相当于会议主席；

第三阶段：计算共享密钥

M_n 将汇总的数据经过计算处理后，再广播发送到每个通信实体 M_i，收到 M_n 传送的数据后，M_i 计算：$K_n \equiv (\alpha^{\frac{N_1 \cdots N_n}{N_i} k_{in}})^{K_{in}^{-1} \cdot N_i} \bmod p \equiv \alpha^{N_1 \cdots N_n} \bmod p$，然后验证：$h(K_n \| t_n) \stackrel{?}{=} h(\alpha^{N_1 \cdots N_n} \| t_n)$ 是否相等，若相等，则会议密钥 K_n 得到确认；否则，拒绝 K_n，终止协议运行。

仍然用有四个通信实体的例子加以解释。

第一阶段：

M_1 计算 $K_1 = a^{x_1} \bmod p \bmod q$；　M_2 计算 $K_2 = a^{x_2} \bmod p \bmod q$；

M_3 计算 $K_3 = a^{x_3} \bmod p \bmod q$；　M_4 计算 $K_4 = a^{x_4} \bmod p \bmod q$；

然后，M_1 、M_2 交换 K_1、K_2；M_2 、M_3 交换 K_2、K_3；

M_3 、M_4 交换 K_3、K_4；M_4 、M_1 交换 K_4、K_1；

第二阶段：

Step 1　M_1 选择一个随机数 $N_1 \in Z_q^*$，计算：

$$k_{12} = ((\alpha^{x_2})^{x_1} \bmod p) \bmod q = (\alpha^{x_1 x_2} \bmod p) \bmod q$$

并把下面数据发给 M_2：

$$A = \alpha^{N_1 K_{12}} \bmod p\,,\quad B = t_1\,,\quad C = h((\alpha^{N_1} \bmod p) \| t_1)$$

Step 2　M_2 接收数据，若 $t_1^* - t_1 > \Delta t$（Δt 是允许的最大时间间隔），表示超时，终止协议运行；否则，M_2 计算：

$$k_{12} = ((\alpha^{x_1})^{x_2} \bmod p) \bmod q = (\alpha^{x_1 x_2} \bmod p) \bmod q$$

以及 k_{12} 在 Z_q 上的乘法逆元素 k_{12}^{-1}，并验证方程：$h((A^{K_{12}^{-1}} \bmod p) \| B) \stackrel{?}{=} C$ 是否相等。若相等，说明数据确实来自 M_1，且传送的数据没有被修改过，接着进行下一步；否则，终止协议运行。

Step 3　与 Step 1 类似，M_2 把下列数据发送给 M_3：

$$A = \alpha^{N_1 K_{23}} \bmod p\,,\quad B = \alpha^{N_2 K_{23}} \bmod p\,,\quad C = \alpha^{N_1 N_2 K_{23}} \bmod p\,,\quad D = t_2\,,$$

$$E = h((\alpha^{N_1 + N_2 + N_1 N_2} \bmod p) \| t_2)$$

Step 4　与 Step 2 类似，M_3 首先验证接收的数据是否超时，如果没有超时，M_3 计算：

$$k_{23} = ((\alpha^{x_2})^{x_3} \bmod p) \bmod q = (\alpha^{x_2 x_3} \bmod p) \bmod q$$

及 k_{23} 在 Z_q 上的乘法逆元素 k_{23}^{-1}，并验证方程：$h(((A \cdot B \cdot C)^{K_{23}^{-1}} \bmod p) \| t_2) \stackrel{?}{=} E$ 是否相等。若相等，接着进行下一步；否则，终止协议运行。

Step 5　与 Step 1 类似，M_3 把下列数据发送给 M_4：

$$A = \alpha^{N_1 N_2 K_{34}} \bmod p\,,\quad B = \alpha^{N_1 N_3 K_{34}} \bmod p\,,\quad C = \alpha^{N_2 N_3 K_{34}} \bmod p\,,$$

$$D = \alpha^{N_1 N_2 N_3 K_{34}} \bmod p\,,\quad E = t_3\,,\quad F = h((\alpha^{N_1 N_2 + N_1 N_3 + N_2 N_3 + N_1 N_2 N_3} \bmod p) \| t_3)$$

Step 6 与 Step 2 类似，M_4 首先验证接收的数据是否超时，若没有超时，M_4 计算 k_{34} 及 k_{34}^{-1}，并验证方程：$h(((A\cdot B\cdot C\cdot D)^{K_{34}^{-1}} \bmod p)\| t_3) \stackrel{?}{=} F$ 是否相等。若相等，计算会议密钥，并进行下一步；否则，终止协议运行。

$$K_4=(D)^{K_{34}^{-1}N_4} \bmod p=\alpha^{N_1N_2N_3N_4} \bmod p$$

第三阶段：

M_4 把下列数据广播发送给 M_1，M_2，M_3。

$$A=\alpha^{N_2N_3N_4K_{14}} \bmod p,\quad B=\alpha^{N_1N_3N_4K_{24}} \bmod p,\quad C=\alpha^{N_1N_2N_4K_{34}} \bmod p$$

$$D=t_4,\quad E=h((\alpha^{N_1N_2N_3N_4} \bmod p)\| t_4)$$

M_1，M_2，M_3 分别验证接收的数据是否超时，若没有超时，则分别计算并验证：

M_1：$K_4=(A)^{K_{14}^{-1}N_1} \bmod p=\alpha^{N_1N_2N_3N_4} \bmod p$

验证 $h(K_4\| t_4)\stackrel{?}{=}E$ 是否成立。

M_2：$K_4=(B)^{K_{24}^{-1}N_2} \bmod p=\alpha^{N_1N_2N_3N_4} \bmod p$

验证 $h(K_4\| t_4)\stackrel{?}{=}E$ 是否成立。

M_3：$K_4=(C)^{K_{34}^{-1}N_3} \bmod p=\alpha^{N_1N_2N_3N_4} \bmod p$

验证 $h(K_4\| t_4)\stackrel{?}{=}E$ 是否成立。

若上面的式子均成立，则会议密钥 K_4 得以确认；否则，终止协议运行。

2.2.2 M-GDH.2 安全性及性能分析

由于 M-GDH.2 对通信参与者 M_i 的身份以及传输数据的完整性进行验

证，同时对最后产生的会议密钥进行确认，所以本文对 GDH.2 的攻击在 M-GDH.2 将不再存在。除此之外，M-GDH.2 协议还有以下特点：

1、具有“前向安全性”

考虑一种最坏的情况。假设所有通信实体间的长期共享密钥：$K_{ij}(1\le i,j\le n,i\ne j)$ 均泄露，则攻击者可能获得下面数据：$\{\alpha^{\overline{N_1\ldots N_i}}\ |k\in[1,i]\}(1\le i\le n)$，但是要得到会议密钥 $K_n=\alpha^{N_1\ldots N_n}\bmod p$，攻击者必须知道 $N_i(1\le i\le n)$，其难度等价于求解大数的离散对数问题。

2、抵御“Denning-Sacco”攻击

所谓“Denning-Sacco”攻击是指一个主动攻击者即使得到使用过的会议密钥，也不可能冒充一个合法的通信实体参与以后的会话[53]。

假设攻击者知道某次会议密钥：

$$K_n=(\alpha^{\frac{N_1\ldots N_n}{N_i}k_{in}})^{K_{in}^{-1}\cdot N_i}\bmod P=\alpha^{N_1\ldots N_n}\bmod P$$

且知道该次会话的子集 $\{\alpha^{\prod S}\},S\subset\{N_1,\ldots N_n\}$，如果进一步得到 K_{in} 或 K_{in}^{-1}，攻击者就可以象本文对 GDH.2 协议那样对 M-GDH.2 协议进行主动攻击。尽管攻击者知道 $K_n,\alpha^{\frac{N_1\ldots N_n}{N_i}k_{in}},N_i$，但要得到 K_{in} 或 K_{in}^{-1}，其难度等价于求解大数的离散对数问题。

3、抵御重传攻击

如果攻击者截获以前 M_i 传给 M_{i+1} 的数据，对 M_{i+1} 进行重传攻击。由于，M_{i+1} 首先要验证 $t_i^*-t_i>\Delta t$ 是否成立，从而避免了重传攻击。需要说明的是只有通信实体间的时钟同步，该验证才有意义。

4、性能分析

M-GDH.2 协议使用 hash 运算进行数据的完整性验证。由于 hash 函数的运算速度非常快，输出结果非常短，以单向函数 MD5 为例，只有 16 位的输出长度。相对于大数的模指数运算而言，其计算负荷和通信负荷几乎可以忽略不计。与 GDH.2 比较，M-GDH.2 每轮增加了两次模指数运算：一次是求 K_{ij}，另一次是为了验证数据的完整性用 K_{ij}^{-1} 对接收数据的乘积作了一次模指数运算，以及一次求逆元素 K_{ij}^{-1} 的运算，但这是增加协议安全性必须付出的代价。

2.3 M-GDH.2 协议中的用户扩展

我们讨论 M-GDH.2 协议时，并没有考虑通信成员的动态变化。实际上 n 个成员组成的小组进行保密通信时，完全可能有新的成员加入该通信小组，或者该通信小组有成员要退出正在进行的通信。以下对上述两种情况分别加以讨论。

2.3.1 增加成员

增加成员时必须考虑系统的“前向安全性”。也就是说不能让新成员获得该通信小组以前的会议密钥。假设 M_{n+1} 表示新增成员，图 2-5 表示增加成员的过程。

增加成员时，密钥的更新分为两个阶段：

第一阶段：

M_n 选择一个新的随机数 $N_n^* \in Z_q^*$，计算 M_n 和 M_{n+1} 之间的共享秘密：

$$k_{n,n+1} = ((\alpha^{x_{n+1}})^{x_n} \bmod p) \bmod q = (\alpha^{x_n x_{n+1}} \bmod p) \bmod q$$

并把下面数据发给 M_{n+1}：

$$\{\alpha^{\frac{N_1\ldots N_n^*}{N_k}k_{n,n+1}} \mid k\in[1,n]\},\alpha^{N_1N_2\ldots N_n^*k_{n,n+1}},t_n$$

以及

$$h(\prod\{\alpha^{\frac{N_1\ldots N_n^*}{N_k}} \mid k\in[1,n]\}\cdot\alpha^{N_1N_2\ldots N_n^*} \| t_n)$$

与 M-GDH.2 协议执行过程一样，M_{n+1} 验证时间和数据的完整性。并计算 M_n 和 M_{n+1} 之间的共享秘密：

$$k_{n,n+1}=((\alpha^{x_n})^{x_{n+1}} \bmod p) \bmod q=(\alpha^{x_nx_{n+1}} \bmod p) \bmod q$$

M_{n+1} 最后计算得到：

$$\{\alpha^{\frac{N_1\ldots N_n^*}{N_k}} \mid k\in[1,n]\},\alpha^{N_1N_2\ldots N_n^*}$$

以及会议密钥：

$$K_{n+1}=\alpha^{N_1N_2\ldots N_n^*N_{n+1}} \bmod p$$

第一阶段（上传）：

$$M_n \xrightarrow{\{\alpha^{\frac{N_1\ldots N_n^*}{N_k}k_{n,n+1}} \mid k\in[1,n]\},\alpha^{N_1N_2\ldots N_n^*k_{n,n+1}},t_n} M_{n+1}$$

$$h(\prod\{\alpha^{\frac{N_1\ldots N_n^*}{N_k}} \mid k\in[1,n]\}\cdot\alpha^{N_1N_2\ldots N_n^*} \| t_n)$$

第二阶段（广播）：

$$M_i \xleftarrow[h(\alpha^{N_1\ldots N_n^*N_{n+1}} \| t_{n+1})]{\{\alpha^{\frac{N_1\ldots N_n^*N_{n+1}}{N_i}k_{i,n+1}} \mid i\in[1,n+1]\},t_{n+1}} M_{n+1}$$

图 2-5 增加成员过程

第二阶段：

M_{n+1}选择一个随机数$N_{n+1} \in Z_q^*$，计算并发送下列数据给$M_i(1 \le i \le n)$：

$$\{\alpha^{\frac{N_1 \ldots N_n^* N_{n+1}}{N_i} k_{i,n+1}} \mid i \in [1, n+1]\}, t_{n+1}$$

以及

$$h(\alpha^{N_1 \ldots N_n^* N_{n+1}} \| t_{n+1})$$

$M_i(1 \le i \le n)$验证时间和数据的完整性，并计算得到会议密钥：

$$K_{n+1} = \alpha^{N_1 N_2 \ldots N_n^* N_{n+1}} \bmod p$$

2.3.2 删除成员

假设只有“会议主席”M_n有权力剔除通信小组中的成员$M_p(p \ne n)$。图 2-6 显示删除通信成员的过程。

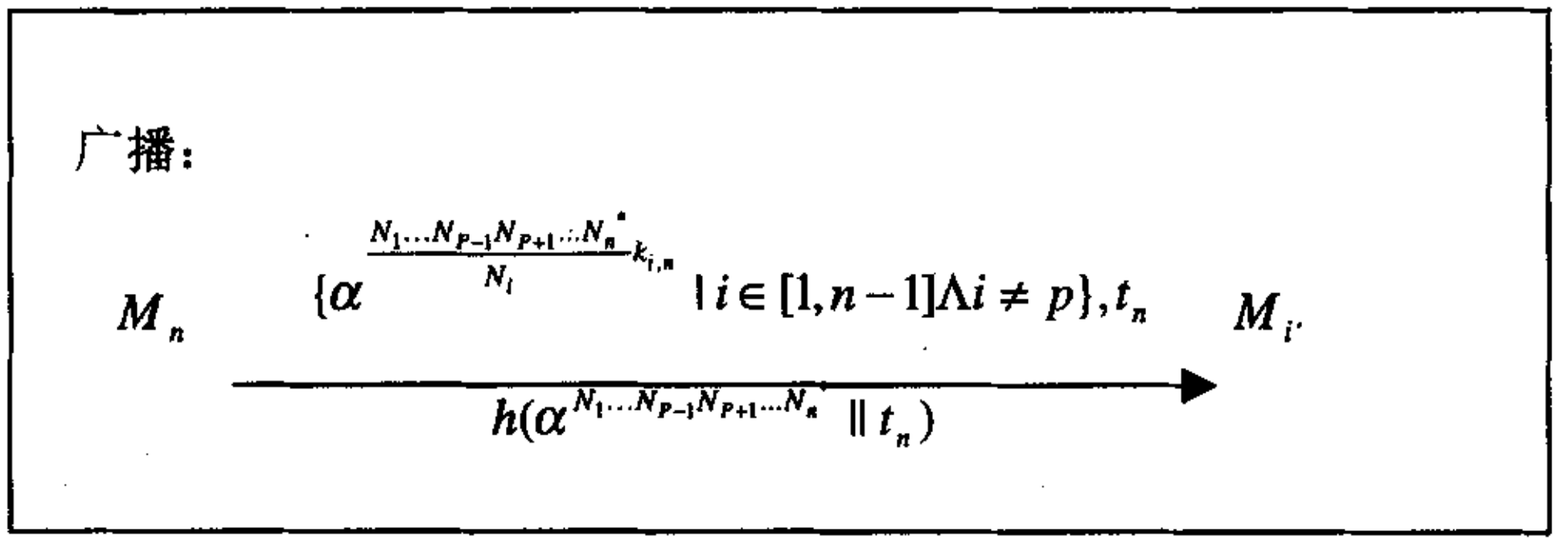


图 2-6 成员的删除过程

M_n选择一个新的随机数$N_n^* \in Z_q^*$，并把下列数据广播发送给$M_i(i \ne p)$：

$$\{\alpha^{\frac{N_1 \ldots N_{P-1} N_{P+1} \ldots N_n^*}{N_i} k_{i,n}} \mid i \in [1, n-1] \Lambda i \ne p\}, t_n$$

以及

$$h(\alpha^{N_1 \ldots N_{P-1} N_{P+1} \ldots N_n^*} \| t_n)$$

$M_i(i \neq p)$ 接收数据后，验证时间和数据的完整性，并计算得到会议密钥：

$$K_{new} = \alpha^{N_1 \ldots N_{P-1} N_{P+1} \ldots N_n^*} \bmod p$$

2.4 本章小结

本章通过对会议密钥分配协议 GDH.2 的分析，指出了其存在的安全漏洞。并在 GDH.2 协议的基础上，讨论了一个改进的 M-GDH.2 协议。相对于 GDH.2 而言，M-GDH.2 只以增加很小的计算和通信负荷为代价，安全性能得到了较大的提高。

第三章 无线 Ad-hoc 网络安全平台设计

一般的，为了防止信息窃听和系统攻击，在无线 Ad-hoc 网络中通常采用密钥对通信数据进行加密。基于密钥的加密算法包括对称算法和公开密钥算法两类。为了减少节点的计算负荷和通信时间，一般使用对称算法，这就要求各节点共同拥有一个对称密钥，该密钥又称为会议密钥。因此如何在参与通信的节点之间产生密钥，是能否实现安全通信的关键所在。目前，许多国内外专家学者公布了自己在这一领域上的研究成果：ZHOU L, HASS Z. J. 在文献[18]中提出“基于信任分散的安全策略”；解放军理工大学的周海刚博士提出“基于信任分散的 Ad hoc 网络安全模型”；还有一些主要针对 Ad-hoc 网络的路由安全问题提出的协议等等。但是大多数关于无线 Ad-hoc 网络安全的研究均侧重于理论分析或理论模型，尚未发现基于仿真平台的研究与实现。于是作者围绕着无线 Ad-hoc 网络中密钥安全问题展开研究，致力于实现一个基于对等通信的网络安全平台。

3.1 系统结构

系统结构如图 3-1 所示：

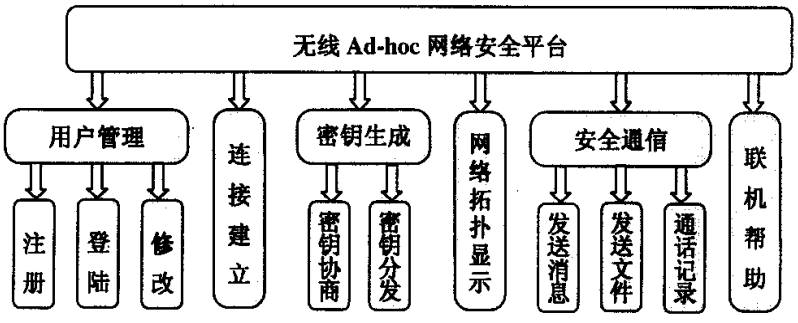


图 3-1：无线 Ad-hoc 网络安全平台结构框图

密钥的生成根据用户数及网络的拓扑结构分为两种情况：

- 环形（线形）拓扑结构

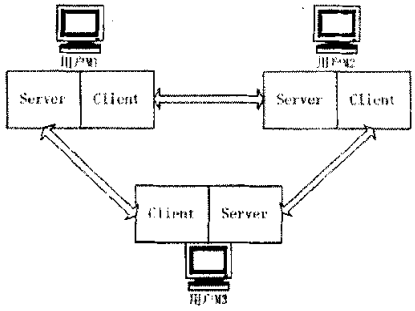


图 3-2：环形对等连接

由于本平台所开发的 M-GDH.2 协议只适合 3 个用户连接成线形或环形的情况，所以当网络的拓扑为图 3-2 所示时，采用 M-GDH.2 协议协商密钥。

- 任意拓扑结构

当网络的拓扑结构不是线形时，M-GDH.2 协议便不再适用。第一章中所介绍的几种密钥协商协议都对网络的拓扑结构有较高的要求，在此都不适用。为了建立一个安全的通信密钥，系统采用了集中式密钥分配协议，考虑到网络中用户变化比较快，而且指定一个用户作为密钥分配中心，容易受到集中攻击，所以系统中的每个用户都可作为密钥分发中心，根据需要产生密钥后，再分发给其他用户。具体算法将在第五章中详细介绍。

3.2 系统功能模块

■用户管理模块

系统一启动就出现如图 3-3 所示的界面，对用户的身份进行验证。

图 3-3：系统登陆界面

用户管理模块包括三个功能：注册、登陆及修改注册信息；

注册：初次使用系统时，需要先进行注册，系统将用户输入的信息按预定的格式写入数

据库，然后返回注册成功的消息，期间有任何异常产生，都会返回注册失败消息。

登陆：用户注册成功后便可登陆系统，系统将用户所输入的用户名及密码与数据库中的内容相比较，验证成功则允许该用户进入系统。

修改注册信息：系统允许用户修改除用户名之外的所有个人注册信息。

■ 连接建立模块

用户登陆成功后，就可向系统中的其他用户发出连接请求，建立连接的界面如图 3-4 所示，有三种方法建立连接：

（1） 单击网上邻居列出的计算机名；

（2） 在复合框的下拉数据中选出对方 IP 地址；

（3） 直接在复合框中输入对方 IP 地址；

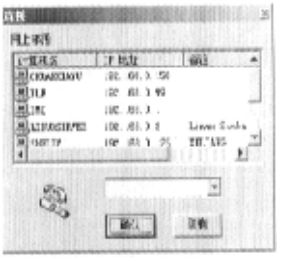


图 3-4：连接建立界面

■ 密钥生成模块

根据用户数及网络的拓扑，密钥的生成分为两种情况：

（1） 当三个用户连接成环状时：可采用 M-GDH.2 协议协商产生会议密钥，

（2） 当多个用户参加通信时：采用集中式密钥分配协议，任一个用户都有能力产生一个随机的密钥，然后再分发给其他用户。

■ 网络拓扑显示模块

形象的显示出本机的连接情况，并可随着网络连接情况的改变而实时的刷新，如图 3-5 所示：

中间一行表示本机；上端表示本机所连接到的主机及其 IP；下端表示本机所接受连接的主机及其 IP。

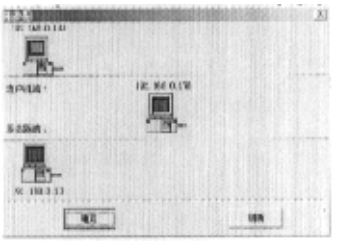


图 3-5：网络拓扑显示

■ 安全通信模块

安全通信模块包括三个功能：聊天、发送文件以及聊天记录。

聊天室如图 3-6 所示，上面形象的显示了进入聊天室的人。

用户可以选择与他连接的某个用户进行聊天，也可以同时向所有与他连接的用户发送聊天信息；发送信息时，可在复选框中的 15 种图标中选择最能够代表自己心情的头像；消息到达时，会有声音提示，同时在文本框中显示消息接收的时间、对方用户名、对方表情以及消息内容等。

图 3-6：聊天室界面

■联机帮助模块

联机帮助模块提供了系统的使用说明，为用户提供在线帮助。

3.3 系统总体设计

3.3.1 开发工具

Microsoft Visual C++ 6.0 提供了 C 和 C++语言的全部功能，具有功能强大、快速、高效等特点。大部分 Visual C++的威力及简单易用来源于 MFC(Microsoft Foundation Classes)[58]，运用 MFC，微软公司已经为用户写好了程序的大部分，使程序设计人员能把精力集中在实现应用程序的功能上。正因为这些优点，因此选择 Microsoft Visual C++ 6.0 为开发工具，工作平台为 Windows 98 或 Windows 2000；

3.3.2 数据库选型

Visual C++ 6.0 提供了多种多样的数据库访问技术[59]，如下所示：

1. ODBC（Open DataBase Connectivity）

2. OLE DB(Object Link and Embedding DataBase)

3. DAO (Data Access Object)

4. MFC ODBC(Microsoft Foundation Classes ODBC)

5. ADO(ActiveX Data Object)

这些技术各有自己的特点，它们提供了简单、灵活、访问速度快、可扩展性好的开发技术。

其中 MFC DAO 类封装了 DAO（数据库访问对象）的大部分功能，从而 Visual C++程序就可以使用 Visual C++提供的 MFC DAO 类方便的访问 Access 数据库，编制简洁的数据库应用程序。本系统中用户的注册信息都存储在 Access 数据库中，于是系统采用 DAO 数据库访问技术对用户的信息进行查询与管理。

3.3.3 对称加密算法

对称加密指的是加密和解密算法都使用相同的密钥。具体如下：

E(p,k)=C ； D(C,k)=p

其中

E = 加密算法； D = 解密算法；

k = 加密密钥； p = 明文（原始数据）； C = 密文；

由于在加密和解密数据时使用了同一个密钥，因此这个密钥必须保密。这样的加密也称为秘密密钥加密。本平台选择了 AES（Advanced Encryption Standard）算法。AES 是美国国家标准和技术协会（NIST）用来代替 DES 的加密标准[60]。2000 年 10 月 NIST 宣布从 15 种候选算法中选出 Rijndael 作为 AES 的标准算法。Rijndael 是一种分组加密算法，分组长度和密钥长度有 128、192 和 256 位等多种选择。因 Rijndael 仅基于简单的位操作运算。因此它具有运算速度快、可对抗差分密码分析和线性密码分析等优点。

3.3.4 系统的安全性能

为了减少用户的计算负荷和通信时间，系统使用对称加密算法-AES 对所传送的数据进行加密，这就要求通信用户共同拥有一个对称密钥，显然，系统中传送的数据是否安全取决于该密钥是否保密。根据网络的拓扑及参加通信的用户数，密钥的生成分为两种情况：当系统只有 3 个用户且呈环形连接时，采用 M-GDH.2 协议确保密钥的安全；当有多个用户参加通信时，系统主要致力于保证用户间对等的传送数据，其安全性略有降低，采用集中式密钥分配方案：任一用户都可作为密钥分配中心，随机产生密钥，然后分发给其他用户。

3.4 本章小结

本章设计了一个安全通信平台，该平台适用于在没有密钥分配中心的情况下，几个地位平等的用户利用手中的便携式电脑安全的进行信息交流。这就类似于一个 Ad-hoc 网络。该平台针对于不同的用户数及网络拓扑结构，设计了不同的密钥生成方案：3 个用户连接成对等环状网时，采用 M-GDH.2 协议协商出密钥；多个用户时，采用密钥分发方案，由任一用户随机产生密钥，再分发给其他用户，其安全性略有降低。最后对平台所包括的用户管理、呼叫建立、密钥生成、安全通信、网络拓扑结构显示以及联机帮助等六大模块的功能一一作了介绍。

第四章 多用户密钥协商协议 M-GDH.2 程序实现

实现 M-GDH.2 协议是本系统开发的重点，本章将深入讨论多用户密钥协商协议 M-GDH.2 程序实现。

4.1 密钥协商算法的实现

任何一个用户都可以发起协商。发起者就作为协商算法中的 M1，其后继作为 M2，M2 的后继作为 M3。算法流程见图 4-1。

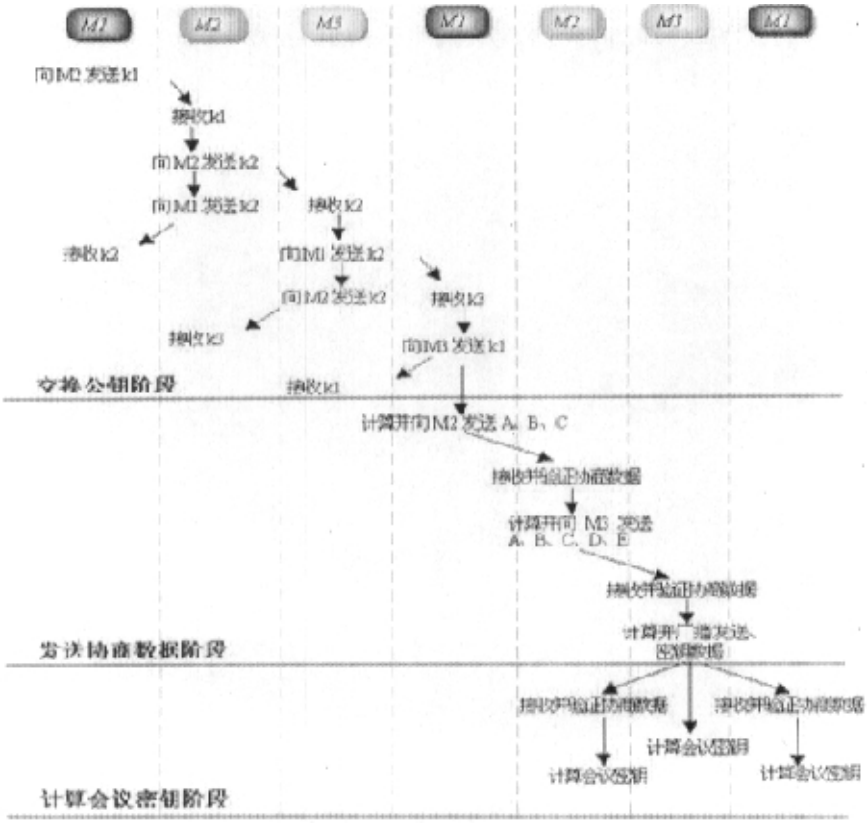


图 4-1：密钥协商算法流程图

算法的实现主要包括以下函数和结构体：

```
typedef struct NegotiationData      // 密钥协商数据的结构体
    {   int order;                  // 发送数据的本地用户身份
        DWORD p;                    // 密钥协商参数
        DWORD q;
        DWORD a;
        DWORD LocalKey;             // 本地用户的私钥
        time_t  t;                  // 开始发送数据的时刻
        CString  MD;                // 用 MD5 算法处理的信息摘要
        DWORD Sdata[MAX_USERS+1];
    } NegotiationData;
*****************************************
PrepareData() ：计算 x,k,N
*****************************************
void CP2p2Dlg::PrepareData()        // 计算 x,k,N
{
    x=(p-2)-lint_rand();            // 函数 lint_rand（）产生一随机数，使得 0<x<p-2
    k=mod_exp(a,x,p);               // k=a^x mod p
    N=lint_rand();                  // N 为随机数
}
*************************************************
M1SendKey() ：向后传递（从客户端到服务器）公钥
*************************************************
void CP2p2Dlg::M1SendKey()              // 向后传递（从客户端到服务器）公钥
{
  MessagePackage msg;
  msg.head=ExchangeKey_CtoS;
  msg.nego.order=1;
  msg.nego.p=p;
  msg.nego.q=q;
  msg.nego.a=a;
  msg.nego.LocalKey=k;
  if(m_pClientSocket[Connect_Order]!=NULL)
  m_pClientSocket[Connect_Order]->Send(&msg,sizeof(MessagePackage));
}
*****************************************************************
server_receive_key(MessagePackage msg)：服务器接收并处理公钥数据
*****************************************************************
BOOL CP2p2Dlg::server_receive_key(MessagePackage msg)//服务器接收处理公钥数据
{
    m_order=msg.nego.order+1;           //获得本地身份
```

```
    if(m_order%MAX_USERS!=1)        //如果交换私钥的过程还没有循环到发起者
    {                               //则继续向前（服务器）传递
        p=msg.nego.p;               //获得协商参数；
        q=msg.nego.q;
        a=msg.nego.a;
        RemoteKey[0]=msg.nego.LocalKey; //RemoteKey[0]存储前驱节点的公钥
        PrepareData();              //计算本地公钥信息
        msg.nego.order++;           //用本地私钥信息更新消息包
        msg.nego.LocalKey=k;
        m_pClientSocket[Connect_Order]
        ->Send(&msg,sizeof(MessagePackage)); //向前驱节点传递本地公钥信息
        msg.head=ExchangeKey_StoC;
        m_ListenSocket.m_pServiceSocket[Connected_Order]
        ->Send(&msg,sizeof(MessagePackage)); //向后继节点传递本地私钥信息
    }
    else                                //若循环到了发起者，则交换结束
    {
        m_order=1;
        RemoteKey[0]=msg.nego.LocalKey;
        msg.nego.order=m_order;
        msg.nego.LocalKey=k;
        msg.head=ExchangeKey_StoC;
        m_ListenSocket.m_pServiceSocket[Connected_Order] //向后继节点传递本地
          ->Send(&msg,sizeof(MessagePackage));          //私钥信息
        msg.head=ExchangeKey_OK;                        //交换结束
    }
    if(msg.head==ExchangeKey_OK)    //如果私钥交换结束，则发起者开始发送
    {                               //A、B、C等数据
        M1SendData();  return(TRUE);
    }
    return(TRUE);
}
void CP2p2Dlg::client_receive_key (MessagePackage msg) //客户端接收并处理公钥数据
void CP2p2Dlg::M1SendData() //M1 向后继节点发送 ABC，从而开始新一轮数据的发送
BOOL BroadcastData(MessagePackage1 msg);//最后一个节点将密钥信息广播发送
void process_key();     //计算得到当前节点与后继节点的共享密钥以及与前驱节点的
                        //共享密钥的乘法模逆元
BOOL receive_final_data(MessagePackage1 msg);       //计算并验证会议密钥
```

限于篇幅，后面几个函数的实现再此就不一一详细介绍了，具体实现可参见源程序代码。

4.2 大素数 p 及其 q 阶本原元 a 的产生

上节所介绍的密钥协商算法中，定义了一个结构体，其中包含了 p、q、a 三个协商参数。下面我们就介绍这三个参数的选择。

4.2.1 产生素数的方法

素数是这样一种数：比 1 大，其因子只有 1 和它本身，没有其它数可以整除它。大素数的产生及测试是密码学领域中的一个重要课题。产生素数的方法可分为以下两类：确定性素数产生方法、概率性素数产生方法[54]。

- **确定性素数**产生方法产生的数必然是素数，然而其产生的素数却带有一定的限制，假若算法设计不佳，便容易构造出带有规律性的素数，使密码分析者能够分析出素数的变化，进而可以猜到该系统中使用的素数。
- **概率性素数**产生方法的缺点在于它只能证明一个数不是合数，而不能证明该数是素数。也就是说，产生的数只是伪素数，为合数的可能性很小，但这种可能性依然存在。优点在于使用概率性素数产生方法，产生伪素数速度较快，构造的伪素数无规律性。这一类算法研究得较多，是当今生成大素数的主要算法，其算法较著名的主要有 Solovay-Strassen 算法、采用 Lucas 函数的算法、MillerRabin 算法。

本文中素数的产生分两步：

（1） 首先采用 MillerRabin 算法产生一个伪素数；

（2） 然后再用 Lucas 定理对其素性进行确认。

下面先来介绍 MillerRabin 算法[55]，其理论依据为定理 1、2；

定义 1 设 n 是正整数，n-1=$2^c m$，其中 c 是非负整数，m 是正奇数。若存在正整数 a，使得 $a^m \equiv 1 (\text{mod n})$；或对某个非负整数 t，$0 \leqslant t < c$，满足 $a^{2t*m} \equiv -1 (\text{mod n})$。则称 n 关于 a 通过 Miller 检测。

定理 1：若 n 是素数，a 是正整数，$(n, a)=1$，则 n 关于 a 通过 Miller 检测。

即若$(n, a)=1$，而 n 不通过关于 a 的 MillerRabin 测试，则 n 不是素数。

定理 2：若 n 是奇合数，则 n 关于 a 通过 Miller 检测的数目最多为$(n-1)/4$，$1\leqslant a<n$。

由定理 1、2，可得到重要结论：若 n 是正整数，选 k 个小于 n 的正整数，以这 k 个数作为基进行 Miller 检测，若 n 是合数，k 次测试全部通过的概率为 $(1/4)^k$。

比如 $k=100$,n 是合数，但测试全部通过的概率为$(1/4)^{100}=6.22*10^{-61}$，这是很小的数，说明这样的事情几乎不可能发生。

产生伪素数 n 的流程如下：

S1：随机产生一个大奇数 n，通过函数 transform(n)，找到 c,m，使得 $n-1=2^c m$；

S2：在{1，2，……，n-1 }中随机均匀的产生一个数 a；$j\leftarrow 0$,计算 $z\leftarrow a^m \bmod n$，若 $z=1$ 或 n-1，则 n 通过测试，结束；

S3：若 $j=c$，则 n 非素数，结束，否则转 **S4**；

S4：$j\leftarrow j+1$, $z\leftarrow z^2 \bmod n$；若 $z=-1$，则 n 通过测试，结束，否则转 **S3**；

****

函数 transform(n)：找到 c,m，使得 $n-1=2^c m$；

****

```
transform(n)
{
   int n0=n-1;
   if(n0%2==0)          n0=n0/2;
   else
        {cout<< "n is not a odd";    return; }
   c=1;
   while (n0%2==0)
      {    n0=n0/2;
           c=c+1;
      }
      m=int((n-1)/int(pow(2,c)));
      cout<<"n-1=pow(2,"<<c<<")*"<<m;
}
```

伪素数 n 产生后，再用 Lucas 定理对其素性进行确认。

[Lucas 定理][55] 设 $n\in N$，若存在一个整数 a，$1<a<n$，且 $a^{(n-1)}\equiv 1(\mathrm{mod}\ n)$ 且对 n-1 的每一个素因子 q，都满足 $a^{(n-1)/q}\bmod n\neq 1$，则 n 为素数。

伪素数 n 的素性确认流程如下：

1) 分解 n-1,使 $n-1=q_1*q_2*\ldots*q_i$，其中 $q_j(i=1,2,\cdots,r)$为不同的素数；

2) $a\leftarrow 1$;

3) $a\leftarrow a+1$;

4) 若 $a>n$，则 n 可能非素数，转 8)；

5) 若 $a^{n-1}\bmod n\equiv 1$，则转 7)；

6) 则转 3)；

7) 若对于任意的 i，$a^{(n-1)/q_i}-1\bmod n\neq 1$，则 n 肯定是素数,否则转 3)；

8) 退出。

4.2.2 求解本原元的方法

如果 p 是一个素数，且 g 小于 p，对于从 0 到 p-1 的每一个 b，都存在某个 a，使得 $g^a\equiv b(\mathrm{mod}\ p)$，那么 g 是模 p 的生成元，也称为本原元。

求解本原元的方法如下：

1. 求解 p-1 的素因子：q_1、q_2、$\ldots q_n$，q 为最大的素因子；
2. g=2;
3. 对所有的 q_1、q_2、$\ldots q_n$，计算 $g^{\frac{p-1}{q_i}}\bmod p$，如果对 q 的任何值，结果都不等于 1，那么 g 是 p 的一个本原元，如果同时 $g^q\bmod p=1$，则 g 就是 p 的 q 阶本原元。如果对 q_1、q_2、$\ldots q_n$ 中的某个值，$g^{\frac{p-1}{q_i}}\bmod p=1$，则转至 4；
4. g=g+1；转至 3，继续判断；

4.3 模指数及模逆元的计算

4.3.1 模指数的计算

如果 $a=b+kn$ 对某些整数 k 成立，那么 $a=b \pmod n$。b 被称为 a 模 n 的余数，a 被称为与 b 模 n 同余。

模运算就象普通的运算一样，它是可交换的、可结合的、可分配的。而且，简化运算每一个中间结果的模 n 运算，其作用与先进行全部运算，然后再简化模 n 运算是一样的。

密码学中用了许多模 n 运算，因为象计算离散对数和平方根这样的问题很困难，而模运算可将所有中间结果和最后结果限制在一个范围内，所以用它计算较容易。对一个 k 位的模数 n，任何加、减、乘的中间结果将不会超过 $2k$ 位长。因此可以用模运算进行指数运算而又不会产生巨大的中间结果。虽然计算某数的乘方并对其取模的运算：$a^x \mod n$ 将导致一系列的乘法和除法运算，但有加速运算的方法：一种方法旨在最小化模乘法运算的数量；另一种旨在优化单个模乘法运算。本文采用一种称为加法链的算法[56]，也称为二进制序列的乘法方法，算法如下：

```
**************************************************
        模取幂运算 计算 x^y mod n    递归算法
**************************************************
DWORD   mod_exp(DWORD x, DWORD y, DWORD n)
{ unsigned long temp;
  if(y==1) return (x%n);
  if((y&1)==0)
  { temp=mod_exp(x,y/2,n);        //递归调用
    return ((temp*temp)%n);
  }
  else
  { temp=mod_exp(x,(y-1)/2,n);    //递归调用
    temp=(temp*temp)%n;
    temp=(temp*x)%n;
    return (temp);
   }
}
```

4.3.2 模逆元的计算

求解模逆元等同于寻找一个 x，使得：$1=(a*x) \bmod n$

也可写作：$a^{-1}=x \pmod n$

一般而论，如果 a 和 n 是互素的，那么 $a^{-1}=x \pmod n$有唯一解；如果 a 和 n 不是互素的，那么 $a^{-1}=x \pmod n$没有解。如果 n 是一个素数，那么从 1 到 n-1 的每一个数与 n 都是互素的，且在这个范围内恰好有一个逆元。

本文采用扩展的欧几里德算法[57]求解模逆元，流程示于图 4-2：

4.4 本章小结

本章介绍了多用户密钥协商协议 M-GDH.2 的程序实现，实现该协议涉及到了许多算法，其中包括随机数产生算法、大素数 p 及其 q 阶本原元 a 的产生算法、模指数以及模逆元的算法等。本章对主要算法的原理以及程序流程做了详细的说明。

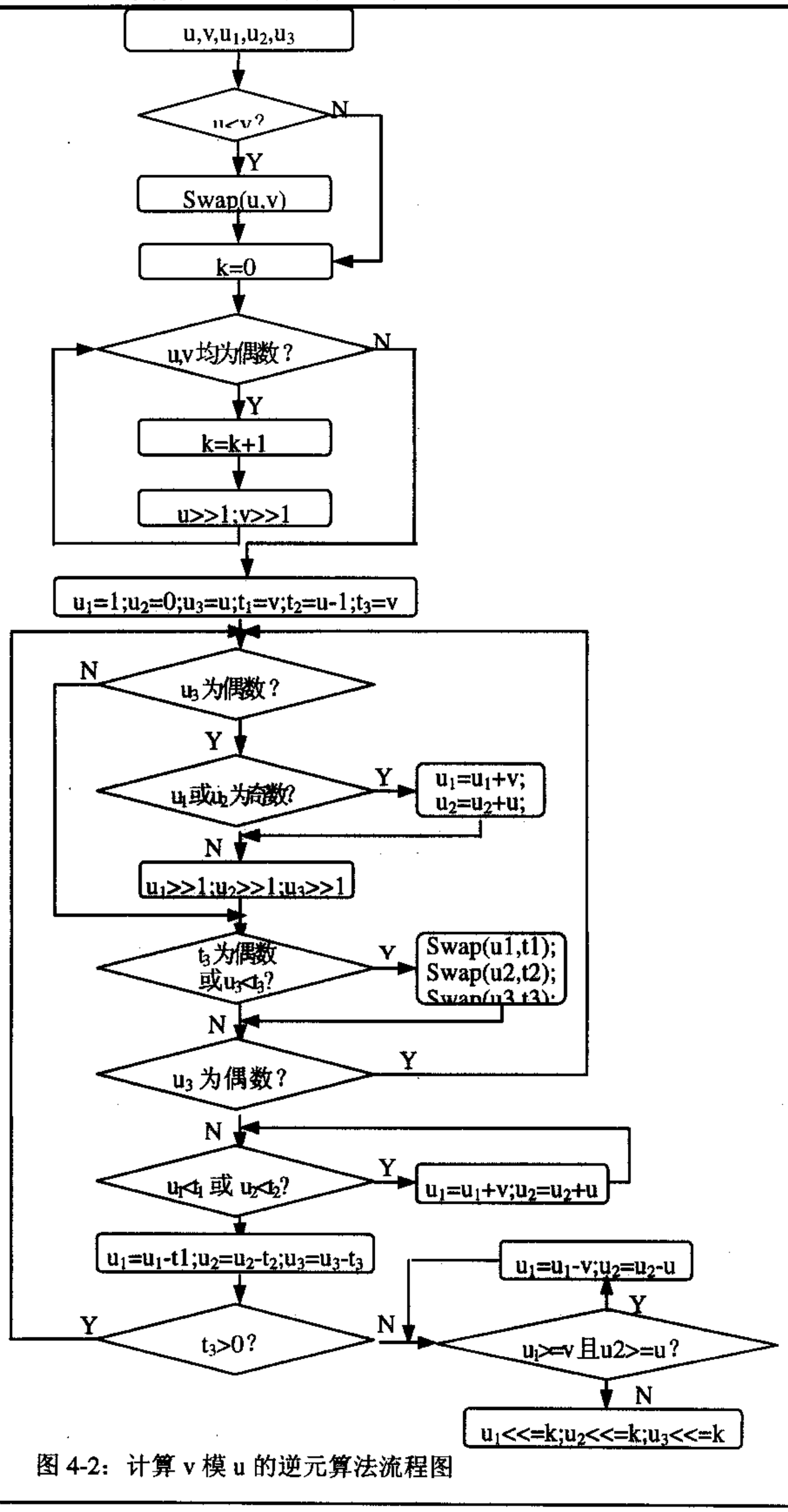


图 4-2：计算 v 模 u 的逆元算法流程图

第五章 无线 Ad-hoc 网络安全平台其它关键模块实现

一个无线 Ad-hoc 网络安全平台，还涉及许多其它功能模块。本章将重点讨论网络对等连接实现、网络拓扑结构显示、密钥分发算法、数据发送模块、数据库管理和文件传送等。

5.1 网络对等连接实现

对等网络（Peer to Peer，简称 P2P）也称为对等连接，是一种新的通信模式，每个参与者具有同等的能力，可以发起一个通信会话。对等互连的实现原理即每个用户既作服务器也作客户端，当一个用户与另一个建立连接时，连接的发起方作为客户端，接收方作为服务器。

Socket 是支持 TCP/IP 协议的网络通信的基本操作单元。可以将 Socket 看作不同主机间的进程进行双向通信的端点。它构成了在单个主机内及整个网际间的编程界面。一般来说，跨机应用进程之间要在网络的环境下进行通信，必须在网络的每一端都建立一个 Socket，两个 Socket 之间是可以建立连接的，也可以是无连接的，并通过对 Socket 的“读”、“写”操作实现网络通信功能。

在 MFC 中 MS 为 Socket 提供了相应的类 CAsyncSocket 和 CSocket，CAsyncSocket 提供基于异步通信的 Socket 封装功能，CSocket 则是由 CAsyncSocket 派生，提供更加高层次的功能，例如可以将 Socket 上发送和接收的数据和一个文件对象（CSocketFile）关联起来，通过读写文件来达到发送和接收数据的目的，此外 CSocket 提供的通信为同步通信，数据未接收到或是未发送完之前调用不会返回。

Socket 编程包括以下几个主要步骤：建立 Socket --->配置 Socket--->通

过 Socket 发送数据--->通过 Socket 接收数据--->关闭 Socket。如图 5-1 所示：

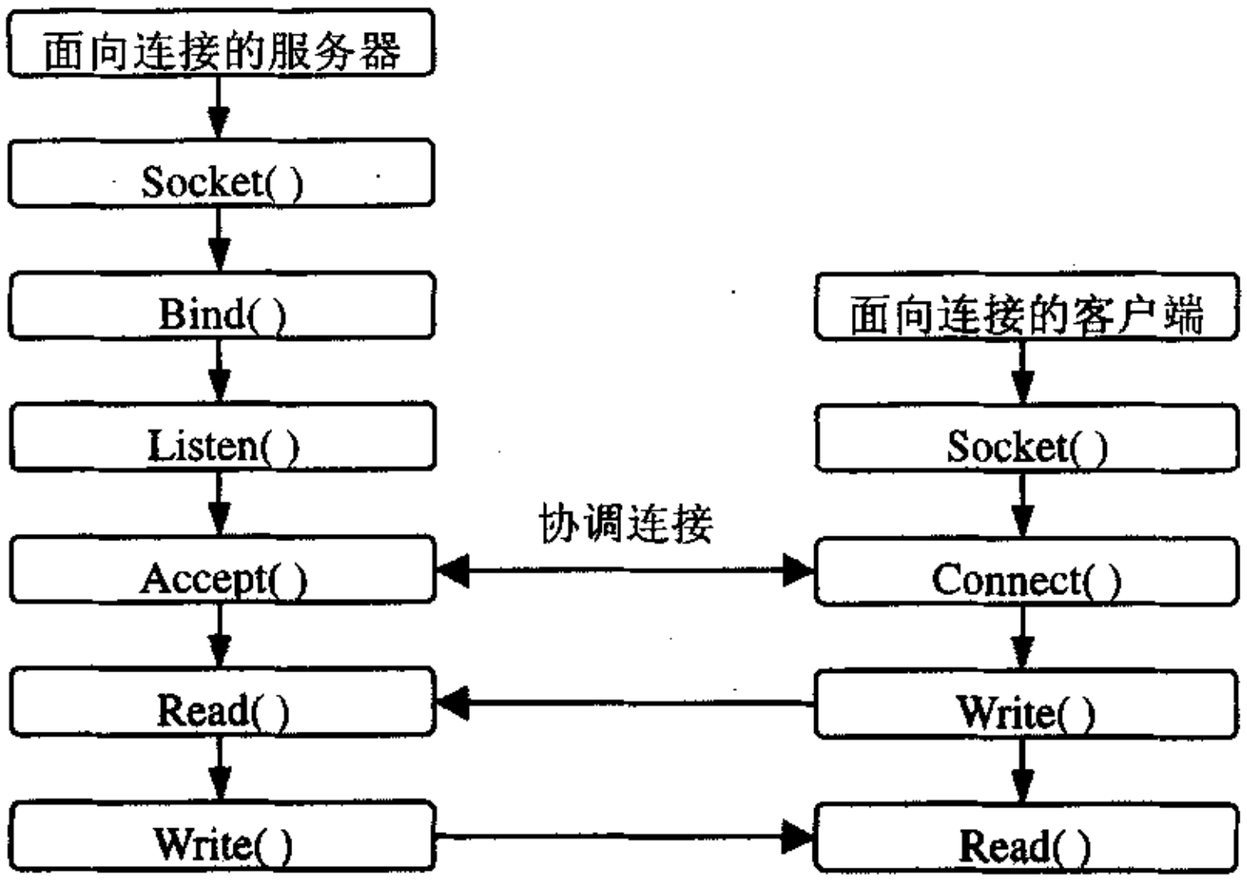


图 5-1：采用面向连接协议时 Socket 的使用

在创建 CAsyncSocket 对象时调用 BOOL CAsyncSocket::Create(UINT nSocketPort = 0, int nSocketType = SOCK_STREAM, long lEvent = FD_READ | FD_WRITE | FD_OOB | FD_ACCEPT | FD_CONNECT | FD_CLOSE, LPCTSTR lpszSocketAddress = NULL)函数，通过指明 lEvent 所包含的标记来确定需要异步处理的事件，对于指明的相关事件的相关函数调用都不需要等待完成后才返回，函数会马上返回，然后在完成任务后发送事件通知，并利用重载以下成员函数来处理各种网络事件：

标记	事件	需要重载的函数
FD_READ	有数据到达时发生	void OnReceive(int nErrorCode);
FD_WRITE	有数据发送时产生	void OnSend(int nErrorCode);
FD_OOB	收到外带数据时发生	void OnOutOfBandData(int nErrorCode);
FD_ACCEPT	服务端等待连接成功时发生	void OnAccept(int nErrorCode);
FD_CONNECT	客户端连接成功时发生	void OnConnect(int nErrorCode);
FD_CLOSE	套接口关闭时发生	void OnClose(int nErrorCode);

重载的函数中都有一个参数 nErrorCode，为零则表示正常完成，非零则表示错误。通过 int CAsyncSocket::GetLastError()可以得到错误值。

CAsyncSocket 类可以提供以下功能，通过这些功能我们可以方便的建立网络连接和发送数据。

- BOOL Create(UINT nSocketPort = 0, int nSocketType = SOCK_STREAM, long lEvent = FD_READ | FD_WRITE | FD_OOB | FD_ACCEPT | FD_CONNECT | FD_CLOSE, LPCTSTR lpszSocketAddress = NULL): 用于创建一个本地套接口，其中 nSocketPort 为使用的端口号，为零则表示由系统自动选择，通常在客户端都使用这个选择。nSocketType 为使用的协议族，SOCK_STREAM 表明使用有连接的服务，SOCK_DGRAM 表明使用无连接的数据报服务。lpszSocketAddress 为本地的 IP 地址，可以使用点分法表示如 10.1.1.3。
- BOOL Bind(UINT nSocketPort, LPCTSTR lpszSocketAddress = NULL): 作为等待连接方时产生一个网络半关联，或者是使用 UDP 协议时产生一个网络半关联。
- BOOL Listen(int nConnectionBacklog = 5): 作为等待连接方时指明同时可以接受的连接数，请注意不是总共可以接受的连接数。
- BOOL Accept(CAsyncSocket& rConnectedSocket, SOCKADDR* lpSockAddr = NULL, int* lpSockAddrLen = NULL): 作为等待连接方将等待连接建立，当连接建立后一个新的套接口将被创建，该套接口将会被用于通信。
- BOOL Connect(LPCTSTR lpszHostAddress, UINT nHostPort): 作为连接方发起与等待连接方的连接，需要指明对方的 IP 地址和端口号。
- void Close( ): 关闭套接口。
- int Send(const void* lpBuf, int nBufLen, int nFlags = 0) ; int Receive(void* lpBuf, int nBufLen, int nFlags = 0): 在建立连接后发送和接收数据，nFlags 为标记位，双方需要指明相同的标记。
- int SendTo(const void* lpBuf, int nBufLen, UINT nHostPort, LPCTSTR lpszHostAddress = NULL, int nFlags = 0); int ReceiveFrom(void* lpBuf, int nBufLen, CString& rSocketAddress, UINT& rSocketPort, int nFlags = 0): 对于无连接通信发送和接收数据，需要指明对方的 IP 地址和端口号，nFlags 为标记位，双

方需要指明相同的标记。

我们可以看到大多数的函数都返回一个布尔值表明是否成功。如果发生错误可以通过 int GetLastError()得到错误值。

由于 CSocket 由 CAsyncSocket 派生，所以拥有 CAsyncSocket 所有功能。

本文通过 Socket 实现对等连接的主要代码如下：

```
**********************************************
    客户端 Socket 类：CClientSocket
**********************************************
class CClientSocket : public CSocket
{public:
    CClientSocket();
    virtual ~CClientSocket();
    virtual void OnReceive(int nErrorCode);
    virtual void OnClose(int nErrorCode);
};
**********************************************
      服务器端监听类：CListenSocket
**********************************************
class CListenSocket : public CSocket
{public:
    CListenSocket();
    virtual ~CListenSocket();
    virtual void OnAccept(int nErrorCode);
  public:
    CServiceSocket *m_pServiceSocket[100];
};
void CListenSocket::OnAccept(int nErrorCode)          //响应连接请求的消息函数
{//接受客户端发出的连接请求，完成连接，实现一个可发收数据的 Socket
    m_pServiceSocket [OrderList]=NULL;
    m_pServiceSocket [OrderList]=new CServerSocket(); //初始化一个新的 Socket
    this->Accept(*(m_pServiceSocket [OrderList])); //生成一个新的连接后的 Socket
    connect_success_update(OrderList);              //接受连接后对服务器端的更新
    OrderList++;
}
**********************************************
    服务器端 Socket 类：CServiceSocket
**********************************************
class CServiceSocket : public CSocket
{public:
    CServiceSocket();
    virtual ~CServiceSocket();
public:
    virtual void OnReceive(int nErrorCode);
    virtual void OnClose(int nErrorCode);
};
```

```
*****************************************
    主对话框类：CP2p2Dlg
*****************************************
class CP2p2Dlg : public CDialog
{ public:
  CListenSocket   m_ListenSocket;
  CClientSocket   *m_pClientSocket[100];
  int             OrderList;                    //接受连接的 Socket 序列号
  int             Connect_Order;
  int             ConnectID;                    //连接的 Socket 序列号
  int             Connected_Order;
  UINT            m_port;                       //连接的端口号
  };
  BOOL CP2p2Dlg::OnInitDialog()                 //初始化
  {……//省略
    Connected_Order=0;
    OrderList=0;
    Connect_Order=0;
    ConnectID=0;
    m_ListenSocket.Create(m_port);              //建立本地服务器的监听套接字
    m_ListenSocket.Listen();                    //开始监听
    int i;
    for(i=0;i<100;i++)
     {  m_pClientSocket[i]=NULL;                //初始化
        m_ListenSocket.m_pServiceSocket[i]=NULL; //初始化
     }
   return TRUE;
  }
  void CP2p2Dlg::OnConnect()
  {
    if( (NULL != (m_pClientSocket[ConnectID] = new CClientSocket))//创建客户端
        && m_pClientSocket[ConnectID]->Create() )          //套接字并连接
    {
        if(m_pClientSocket[ConnectID]->Connect(m_remote_IP,m_port))
        {                                                  //m_remote_IP：服务器地址
         Connect_Order=ConnectID;
         ConnectID=ConnectID+1;
        }
        else    //连接失败
         连接失败提示;
     }
  }
```

5.2 网络拓扑结构显示

影响本机拓扑结构有以下几种情况：

1. 本机发起新的连接请求；

2. 本机接收到新的连接请求；

3. 连接到本机的计算机或本机连接的计算机断开连接或退出系统；

发生上述任何一种情况时都必须相应的调整记录连接情况的参数，才能保证正确的显示本机的连接情况。具体实现如下：

系统中定义了两个结构体，分别记录连接到本机的计算机及本机所连接的计算机的详细情况；

```
struct Client_Info                  //存储客户端的信息
{    BOOL      CorN;                //指明该客户端是连接（1）还是断开（0）
     char      IP[40];              //指明连接方的 IP 地址
};
struct Server_Info                  //存储服务器端的信息
{    BOOL      CorN;                //指明该服务器端是连接（1）还是断开（0）
     char      IP[40];              //指明连接方的 IP 地址
};
int Client_Number;          //连接到本机的客户端数
int Server_Number;          //本机所连接的服务器数
```

当本机发起新的连接请求时,本机执行 OnConnect( )函数，被连接端（服务器端）执行 connect_success_update(int OrderList)函数，具体过程见图 5-1。

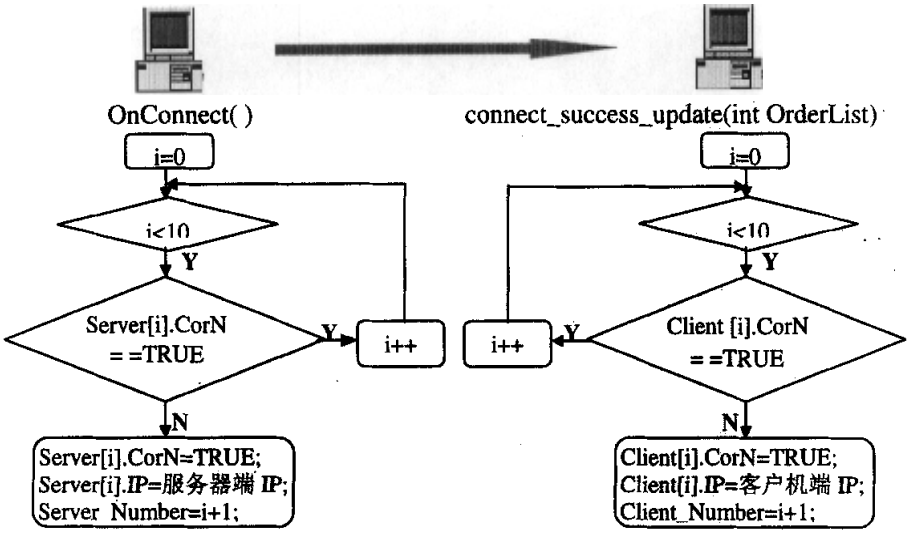


图 5-1：发起新的连接请求时，拓扑结构的变化流程图

当本机断开连接或退出系统时，执行 OnDisconnect( )函数，所有与本机相连的用户执行 disconnect_update(MessagePackage msg)函数，以更新网络拓扑结构。具体流程见图 5-2。

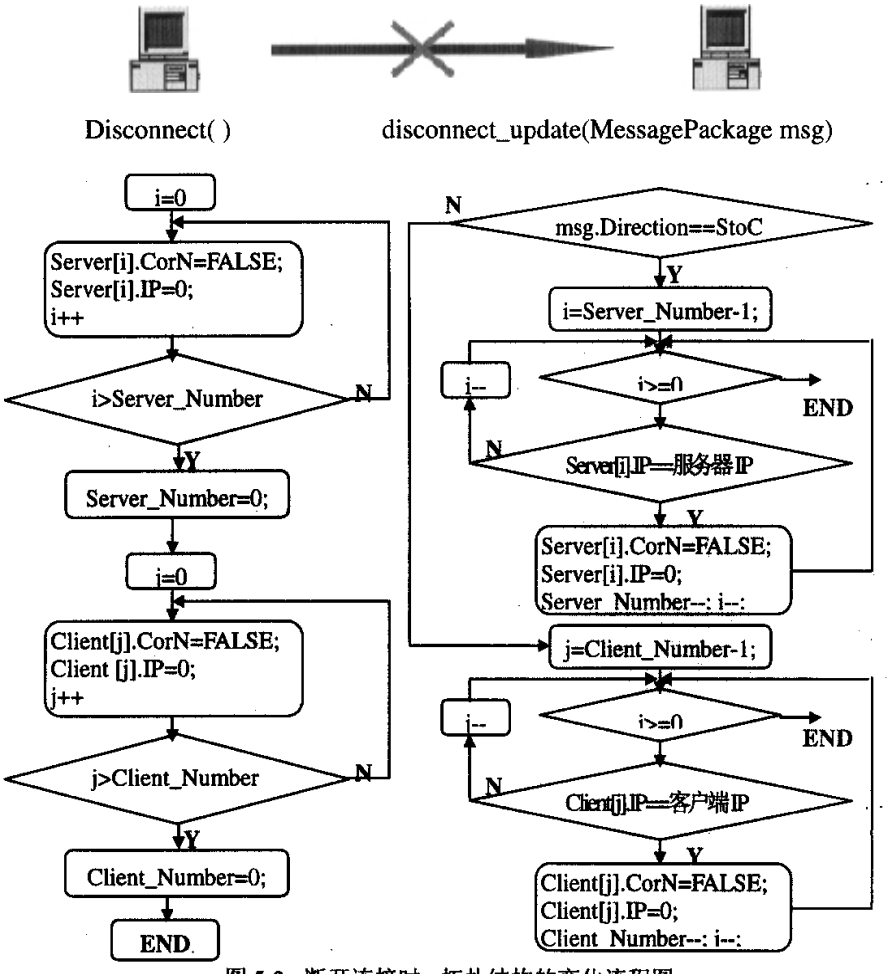


图 5-2：断开连接时，拓扑结构的变化流程图

5.3 密钥分发算法

网络连接完毕后，每一个用户都可作为密钥中心，首先根据需要产生出密钥后，再分发给其他用户。随机密钥产生器如图 5-3 所示：

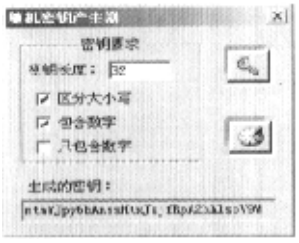


图 5-3：随机密钥产生器

用户可根据需要，设定密钥长度(1~100位)、密钥是否区分大小写、是否包含数字、是否只包含数字等条件后，单击 按钮，就会按照要求随机产生密钥。密钥的产生主要涉及以下几个函数：

```
**************************************************
        函数 Generate：产生密钥
**************************************************
CString Generate(int iLong, BOOL bUpLow, BOOL bNum, BOOL bNumOnly)
{
   int iFlag = 26;
      CString str;
      if(bUpLow == TRUE)        //区分大小写
           iFlag = 52;
      if(bNum == TRUE)          //包含数字
           iFlag = 62;
      if(bNumOnly == TRUE)      //只包含数字
           iFlag = 10;
      for(int i = 0;i<iLong;i++)     //iLong 是密钥的长度
      {
           str += GetChar(iFlag);   // GetChar(iFlag)随机选取一个字符
      }
      if((bNum == TRUE)&&(bUpLow == FALSE))
           str.MakeLower();      //若包含数字但不区分大小写，则将产生的字母变为小写
      return str;
}
**************************************************
        函数 GetChar：随机产生一个字符
**************************************************
CString GetChar(int Flag)
{
   CString s;
   CString m_sArray =
   "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
   CString m_sArray1 = "1234567890";
   switch (Flag)
```

```
{
    case 10: int k = rand();
             k = k % 10;
             s = m_sArray1.Mid(k,1);      //s 为 m_sArray1 字符串中的第 k 个字符
             break;
    case 26:  int k = rand();
             k = k % 26;
             s = m_sArray.Mid(k,1); break;
    case 52:  int k = rand();
             k = k % 52;
             s = m_sArray.Mid(k,1); break;
    case 62:  int k = rand();
             k = k % 62;
             s = m_sArray.Mid(k,1);break;
    default:   int k = rand();
             k = k % 26;
             s = m_sArray.Mid(k,1);break;
    }
        return s;
}
```

密钥产生后，存储在 m_distribute_key 变量中。单击 按钮就开始分发密钥。密钥分发的算法包括两个重载函数：密钥分配中心执行 KeyDistribute() 函数，其功能是将所产生的密钥分发给所有与他相连的用户，流程如图 5-4 所示；

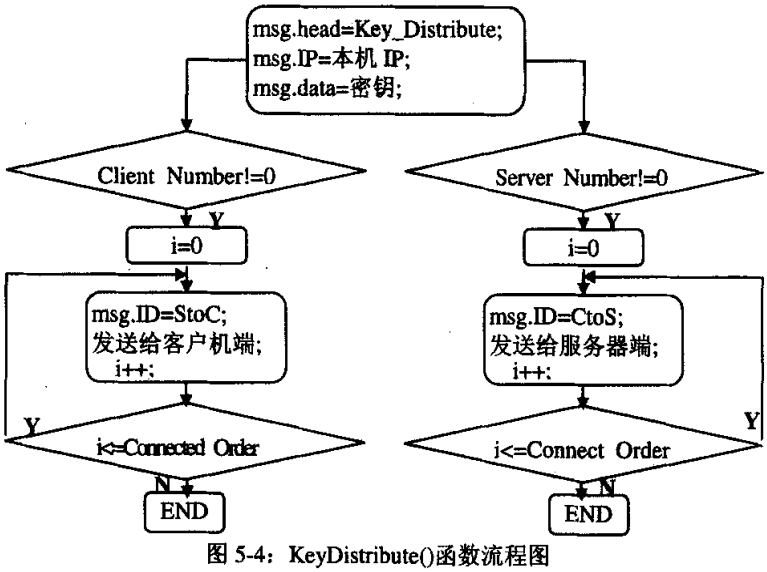


图 5-4：KeyDistribute()函数流程图

客户机端和服务器端收到 head 为 Key_Distribute 的消息后执行 KeyDistribute(msg)函数，其功能是：判断本机除了和分发密钥的用户相连外，是否还和其他用户相连，如果没有，则密钥分发结束；否则继续分发密钥，具体过程如图 5-5。

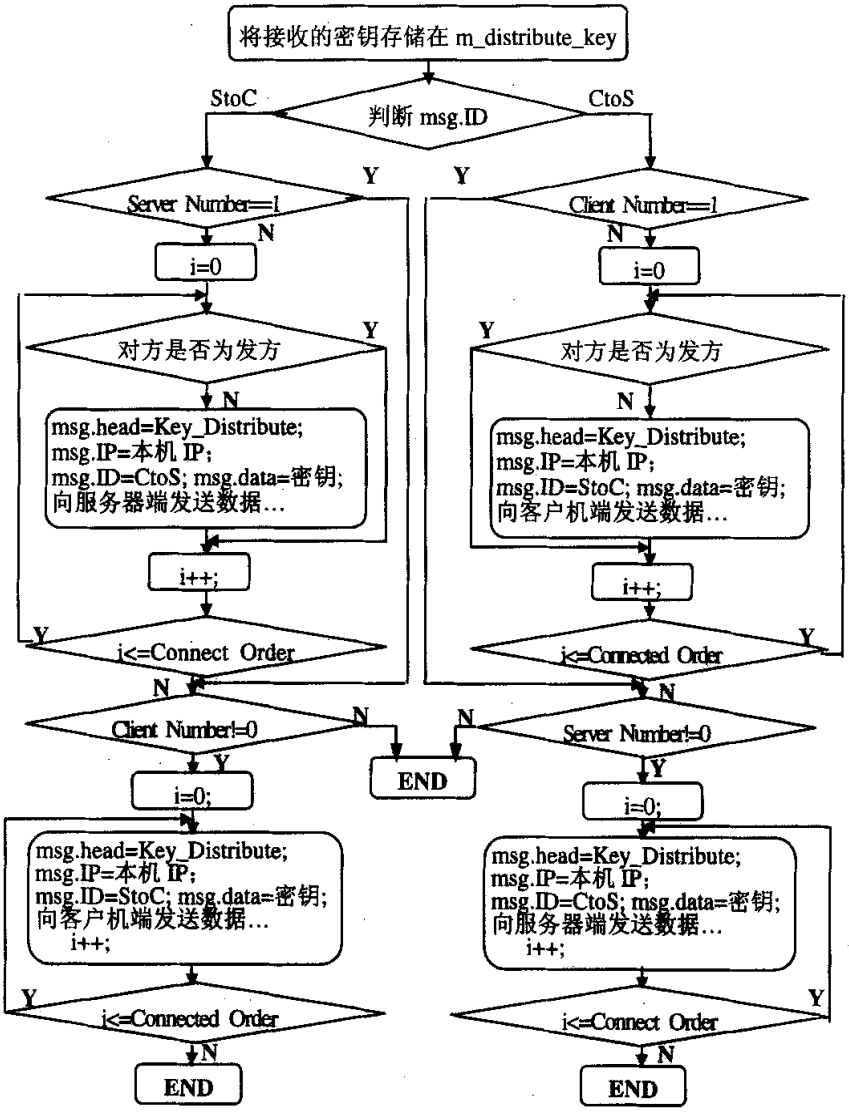


图 5-5：KeyDistribute(msg)函数流程图

5.4 数据发送模块

每个通信用户获得密钥后就可以进行安全聊天和发送文件。进入聊天室时，首先向所有与他相连的（包括他连接的和他接受连接的）用户发出聊天请求，同意聊天的，就一起进入聊天室，不同意的，则断开连接，具体过程见图 5-6。

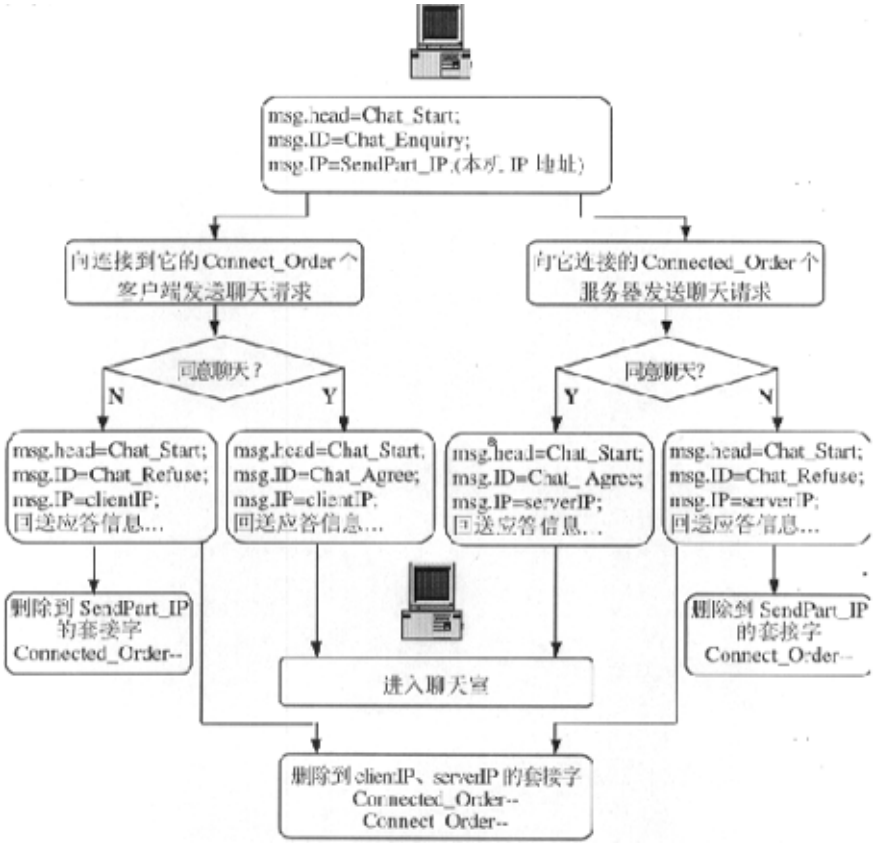


图 5-6：聊天请求流程图

进入聊天室后，发送数据的过程如图 5-7 所示。

```
int   Connect_Order;     //客户端套接字的序号;
int   Connected_Order;   //服务器端套接字的序号;
```

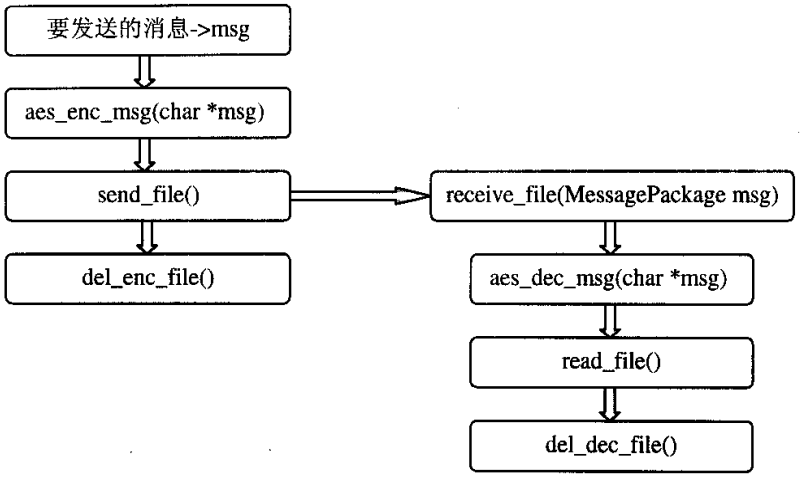


图 5-7：数据发送流程图

```
**********************************************
发送前数据加密函数:aes_enc_msg(char *msg)
**********************************************
void CP2p2Dlg::aes_enc_msg(char *msg)
{
  char *pFileName="plain.txt";
  CFile file(pFileName,CFile::modeCreatelCFile::modeWrite);
  file.Write(msg,strlen(msg));          //将要加密的信息写入文件
  file.Close();
  char o_key[33];
  _ultoa(m_key,o_key,10);           //将 DWORD 类型的密钥转换成字符串
  int i=0;
  int len=strlen(o_key);            //如果字符串长度小于 64 位,则用已有字符循环补充
  for(i=0;(i+len)<32;i++) o_key[len+i]=o_key[i];
  o_key[len+i]='\0';
  char *pCipherFileName="cipher.enc"; //加密后的文件名
  CFile Cfile(pCipherFileName,CFile::modeCreate);
  Cfile.Close();
  switch(aes_file(pFileName,pCipherFileName,1,o_key)) //使用 AES 算法加密文件
  {
   case -1: MessageBox("密钥必须是字符或数字！ ","加密失败",MB_OK);break;
   case -2:
       MessageBox("密钥长度必须是 32 位、48 位或 64 位!","加密失败",MB_OK);break;
```

```
    case -3: MessageBox("明文文件打不开！ ","加密失败",MB_OK);break;
    case -4: MessageBox("密文文件打不开！ ","加密失败",MB_OK);break;
    case -5: MessageBox("加密方式不对！ ","加密失败",MB_OK);break;
    default: m_msgstatus.AddString("加密成功！ \r\n");break;
   }
}
*****************************************
接收后数据解密函数:aes_dec_msg()
*****************************************
void CP2p2Dlg::aes_dec_msg()
{
char o_key[33];
_ultoa(m_key,o_key,10); //将 DWORD(unsigned long)类型的密钥转换成字符串
int i=0;
int len=strlen(o_key);
for(i=0;(i+len)<32;i++) o_key[len+i]=o_key[i];
o_key[len+i]='\0';          //如果字符串长度小于 64 位，则用已有字符循环补充
char *pCipherFileName="Rcipher.enc";
char *pOriginalFile="original.txt";   //解密后的文件名
CFile file(pOriginalFile,CFile::modeCreate);
file.Close();
switch(aes_file(pCipherFileName,pOriginalFile,0,o_key)) //解密
{
  case -1: MessageBox("密钥必须是字符或数字！ ","解密失败",MB_OK);break;
  case -2:
    MessageBox("密钥长度必须是 32 位、48 位或 64 位!","解密失败",MB_OK);break;
    case -3: MessageBox("密文文件打不开！ ","解密失败",MB_OK);break;
    case -4: MessageBox("明文文件打不开！ ","解密失败",MB_OK);break;
    case -5: MessageBox("解密方式不对！ ","解密失败",MB_OK);break;
    default: m_msgstatus.AddString("解密成功！ \r\n");break;
  }
}
```

5.5 数据库管理

5.5.1 DAO 数据库访问技术

MFC DAO 类封装了 DAO(数据库访问对象)的大部分功能，从而 Visual C++程序就可以使用 Visual C++提供的 MFC DAO 类方便的访问 Access 数据

库，编制简洁的数据库应用程序[62]。

MFC 的 DAO 数据库类的名称都有“CDao”前缀。

CDaoWorkspace 对象代表一个 DAO Workspace 对象，在 MFC DAO 体系结构中处于最高处，定义了一个用户同数据库的会话，并包含打开的数据库，负责完成数据库的事务处理。我们可以使用隐含的 Workspace 对象。

CDaoDatabase 对象代表了一个到数据库的连接，每个数据库都拥有自己的 tabledef、querydef、记录集和联系对象的汇集，CDaoDatabase 类提供成员函数对它们进行操作，从而对数据进行操作。

CDaoRecordset 对象，代表一个数据记录的集合，该集合是一个库表或者是一个查询的运行结果中的全部记录。CDaoRecordset 对象有三种类型：表、动态集、快照。

本系统所使用 CDaoRecordset 的类成员主要如下：

成员		意义
数据成员	m_nFields( )	记录集中的字段数
	m_pDAORecordset	基础记录集对象之下的 DAO 接口的指针
	m_pDatabase	链接记录集数据库源的 CdaoDatabase 对象指针
构造	CDaoRecordset	构造一个 CDaoRecordset 对象
	Close	关闭记录集
	Open	创建一个新的记录集对象
属性	IsEOF	判断是否到了末记录
	SetCurrentIndex	对一个标记录集设置一个索引
记录导航操作	AddNew	增加一条新的记录
	Delete	删除当前记录
	Edit	编辑当前记录
	Update	更新当前记录集对象
	MoveFirst	移至首记录
	MoveLast	移至末记录
	MoveNext	移到当前记录的下一条记录
	MovePrev	移到当前记录的上一条记录

5.5.2 用户信息管理

用户必须先注册，等待注册成功；然后根据自己的注册信息进行登陆，登陆成功后才能进入系统，进行用户间通信。

数据库中包括下列字段：

```
CString  m_user;       // 用户名        CString  m_education;   // 学历
CString  m_password;   // 密码          CString  m_position;    // 职务
CString  m_flag;       // 昵称          CString  m_phone;       // 电话号码
CString  m_sex;        // 性别          CString  m_email;       // email
CString  m_age;        // 年龄          CString  m_address;     // 地址
CString  m_married;    // 婚否          short    m_pic;         // 头像 ID
```

用户注册/登陆功能模块的流程如图 5-8 所示：

图 5-8：注册/登陆流程图

注册/登陆功能模块接收到用户端的信息后，首先判断是注册信息还是登陆信息。如果是注册信息，则将该数据按预定的格式写入数据库，然后返回注册成功的消息，期间有任何异常产生，都会返回注册失败消息，提示用户重新注册；如果是登陆信息，则从数据中提取用户名和密码与数据库中的内容进行比较，如果该用户存在，则返回登陆成功消息，反之，返回登陆失败消息。

5.6 文件传送

用户首先选择要发送的文件，并将文件的基本信息--文件名及长度发送给对方，询问对方是否接收该文件，对方同意后，开始调用发送文件线程来发送文件，同时通知对方开始接收，对方收到消息后便调用文件接收线程来

接收文件，其中的参数定义如下：

```
HANDLE      m_send_file_thread_handle;                  //文件传输的线程句柄
DWORD WINAPI send_file_thread(LPVOID param)             //发送文件线程
DWORD WINAPI receive_file_thread(LPVOID param)          //接收文件线程
typedef struct _SOCKET_STREAM_FILE_INFO                 //文件有关信息
{
    TCHAR          szFileTitle[128];                    //文件的标题名
    DWORD          dwFileAttributes;                    //文件的属性
    DWORD          nFileSizeHigh;                       //文件大小的高位双字
    DWORD          nFileSizeLow;                        //文件大小的低位双字
}
**************************
文件传送中消息类型的定义
**************************
#define HEAD_SENDFILE_ENQUIRY          1    //询问对方需要文件吗？
#define HEAD_SENDFILE_CONCENT          2    //允许
#define HEAD_SENDFILE_NO_CONCENT       4    //拒绝
#define HEAD_SENDFILE_CONCENT_RECEIVE  8    //开始接收文件
#define HEAD_SENDFILE_STOP             16   //中止发送或接收
#define HEAD_SENDFILE_SUCCEED          32   //发送成功
#define HEAD_SENDFILE_NO_SUCCEED       64   //发送失败
```

具体的代码限于篇幅在此不再列出，详见源程序。

5.7 本章小结

对等连接是 Ad-hoc 网络的基本特征，也是本论文研究的前提。本章首先介绍了 Visual C++6.0 中 CSocket 类的一些基本函数的定义及功能，然后重点介绍了实现网络对等互连的方法，并给出相关源代码；接下来又详细叙述了网络拓扑显示、密钥分发、数据库管理、消息及文件的发送与接收等无线 Ad-hoc 网络安全平台中其它一些关键模块的实现流程及相关代码。

第六章 结论与展望

本章将对全文研究工作内容进行总结，并对今后工作进行展望，提出相关扩展和改进思路。

6.1 本文工作总结

本文在 Windows 2000 的环境下，用 Microsoft Visual C++设计开发了一个通信的安全平台，如图 6-1 所示：

图 6-1：无线 Ad-hoc 网络安全通信平台

本硕士论文完成的主要工作如下：

- 大量阅读和分析了有关无线 Ad-hoc 网络安全策略的文献资料，对协商式密钥分配协议有了较深入的理解和认识，对国内外的有关最新研究动态有了较全面的了解；
- 通过分析 GDH.2 协议存在的安全隐患，实现了一个改进的多用户密钥协

商协议 M-GDH.2，该协议对通信参与者的身份以及传输数据的完整性进行了验证，同时也对最后产生的会议密钥进行确认，以增加很小的计算和通信负荷为代价，弥补了 GDH.2 协议的缺陷；

- 基于改进的多用户密钥协商协议 M-GDH.2，设计了一个界面友好的安全通信平台，实现了三个和任意多个用户之间安全对等的通信；
- 编程实现了 M-GDH.2 协议所涉及到的各种算法，包括：随机数的产生算法、大素数 p 及其 q 阶本原元 a 的计算、模指数的计算、模逆元的计算等；
- 通过 DAO 数据库访问技术，实现了对用户身份的认证以及对用户注册信息的查询和管理；
- 实现了简单网络拓扑的查找和连接显示，并可随着网络连接情况的变化实时的刷新；
- 可保存用户间详细的通信信息，便于日后查询以及抗抵赖；
- 制作了联机帮助文件，便于用户使用；

6.2 研究工作中的问题、经验与体会

在本平台的开发过程中，作者遇到了一些问题，同时也积累了一些编程经验，现总结如下：

- **密钥长度处理**

M-GDH2 协议协商出的密钥由于受数据类型限制，不可能达到 128 位或更多。如无符号长整型（unsigned long）数据换算成十进制最大长度为 10 位。为达到 AES 对密钥位数的要求，假设协商密钥的各位数字是 a_1、a_2...a_n，AES 所需密钥为 128 位并用字符数组 key[0...128]表示，可用 a_1...a_n 循环填充 key[0]...key[n-1]，key[n]...key[2n-1]，...，直至填满 128 位。

- **用 AES 算法对消息进行加密**

将 AES 源代码编译成静态链接库，并向 MFC 提供加密和解密接口函数。

加密接口函数：int aes_file(pFileName,pCipherFileName,1, key)

pFileName：明文文件名；

pCipherFileName：密文文件名；

1：表示加密； key：密钥；

解密接口函数：int aes_file(pCipherFileName,pOriginalFile,0, key)

pCipherFileName：密文文件名；

pOriginalFile：明文文件名；

0：表示解密； key：密钥；

首先将欲发送的消息存储到文件 pFileName 中：然后调用 AES 提供的文件加密接口函数，将明文文件转化成密文文件 pCipherFileName 传送给接收方。接收方接收成功后再调用文件解密接口函数对密文文件 pCipherFileName 进行解密，恢复成明文文件 pOriginalFile，最后读取出文件中的消息。

- **窗口图标的动态显示**

可以用 TIMER，但是 TIMER 不能有效的定时。因为 TIMER 发送的是窗口消息，当窗口忙于处理键盘、鼠标等消息时就不能及时处理 TIMER，会使间隔时间变得很长。本系统设计中用了一个单独的 TIMER 线程，用 Sleep()定时来解决此问题。具体实现如下：

```
UINT Timer(LPVOID param)
{
    HWND hWnd=(HWND)param;
    while(1)
    {
        Sleep(ms);
        PostMessage(hWnd,CH_PICTURE,NULL,NULL)
     }
}
```

Sleep(ms)后发送自定义消息。消息处理函数就选择某一个 ICON 或 BITMAP 来显示。如 ：

MyBotton.SetBitmap((HBITMAP)Bitmap[i]);

Bitmap 是一个位图数组，存放有 j 个位图。消息处理函数运行一次，i 就累加一次，当 i==j 时，i 就回到 0；这样就产生一个动态图标的效果。

- **数据库表修改后，如何快速更新一个绑定到表的 CRecordSetDAO 记录集**

在程序的编写过程中，往往需要对数据库中的一些字段进行修改，最初每次修改后都需要人工的对 CRecordSetDAO 记录集进行修改，比较麻烦，后来在文献[63]中学到了一个简单的方法，可以快速的更新邦定到该表的 CRecordSetDAO 记录集，方法如下：打开 ClassWizard 中 Member Variables 标签，选中记录集类后，利用 UpdateColoumns 和 Bind All 命令即可实现。

6.3 下一步工作展望

- 由于硬件的限制，本平台只在有线网的环境下调测成功；
- 在消息传送的过程中，只对消息的内容进行了加密，而没有对消息头进行加密。建议针对一次会议，设定一个只有与会者知道的密钥，用这个密钥对初始的一些请求信息进行加密；
- 在密钥分发时，目前是允许任意一个用户随机产生密钥，并分发给其他用户，这样就有可能造成几个用户同时进行密钥的分发，所以建议制定一个原则，比如指定第一个连入网络的用户作为会议主持人，只有他有权作为密钥中心，进行密钥的分发，一旦他离开网络，第二个连入网络的用户接替他作为会议主持人，以此类推。
- 本文的研究重点在于密钥的生成方面，而对于路由选择算法的研究方面有些欠缺。建议在消息的传递过程中，选择一种合适的路由算法，使其能够在最短的时间或者最短的路径内到达目的地。
- 目前只是实现了用户之间数据文件的传输，将来可扩展音频、视频文件的传输和显示功能。

参考文献

[1] Haas Z. J. Wireless Ad hoc Networks, IEEE Journal on Selected Areas in Communications,1999,17(8):1329~1330

[2] C. Perkins. Terminology for Ad-hoc Network. Internet draft-ietf-manet-terms-00.txt, Nov.1997

[3] J. Jubin, J. Tornow. The DARPA Packet Radio Network Protocols, Proceeding of the IEEE, vol.75, Jan. 1987

[4] 英春，史美林. 自组织网体系结构研究，通信学报，Vol.20 No.9, 1999

[5] 赵志峰，郑少仁. Ad hoc 网络体系结构研究，电信科学，2001，1

[6] 张禄林，李承恕. MANET 路由选择协议的比较分析研究，电子学报，Vol.28 No.11, 2000

[7] M. Joa-Ng, I. Lu, A Peer-to-Peer Zone-Based Two-Level Link State Routing for Mobile Ad hoc Networks, IEEE Journal on Selected Areas in Communications, special issue on Wireless Ad hoc Networks, Aug. 1999

[8] David B. Johnson, David A. Maltz etc. The Dynamic Source Routing Protocol for Mobile Ad hoc Networks. Draft-ietf-manet-dsr-05.txt, 2 March 2001

[9] Charles E. Perkins. Elizabeth M. Royer etc. Ad hoc On-Demand Distance Vector(AODV) Routing. Draft-ietf-manet-aodv-08.txt, 2 March 2001

[10] Cansever, Derya H. , Levesqu, Allen H. ,et al. Quality of Service Support in Mobile Ad hoc Networks, in Proceedings of Military Communications Conference, MILCOM99

[11] Corson, M. Scott, Campbell, Andrew T., Towards Supporting Quality of Service in Mobile Ad-Hoc Networks, First Conference in Open Architecture and Network Programming, San Francisco, CA,USA, April 1998

[12] L. Bao, J. Garcia-Luna-Aceves, A New Approach to Channel Access Scheduling for Ad hoc Networks, Eight International Conference on Computer Communications and Networks, 1999

[13] L. E. Moser and P. M. Melliar-Smith. "Ad hoc Mobile Networks", http://beta.ece.ucsb.edu/wireless/

[14] Royer E M,Chai-Keong T. A review of current routing protocols for Ad hoc mobile wireless networks. IEEE Personal Communications, 1999, 6(2): 46~55

[15] Frodigh M, Johansson P, Larsson P. Wireless Ad hoc networking—the art of networking without a network. Ericsson review,2000(4)

[16] IEEE STD 802.11. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications ,1999

[17] Corson S, Macker J. Mobile Ad hoc networking (MANET):routing protocol performance issues and evaluation considerations, request for comments:2501, Internet Engineering Task Force(IETF) ,1999

[18] Lidong Zhou, Zygmunt J. Haas，“Securing Ad Hoc Networks”，http://citeseer.nj.nec.com/ ,1999.

[19] Jean-Pierre Hubaux, Levente Buttyan，“The Quest for Security in Mobile Ad Hoc Networks”，http://citeseer.nj.nec.com/493788.html

[20] Maki, S. Security Fundamentals in Ad hoc Networking. Proceedings of the Helsinki University of Technology, Seminar on Internetworking-Ad hoc Networks, Spring 2000

[21] Karpijoki, V. Signalling and Routing Security in Mobile Ad hoc Networks. Proceedings of the Helsinki University of Technology, Seminar on Internetworking-Ad hoc Networks, Spring 2000.

[22] Kenji Koyama, “Secure conference key distribution schemes for conspiracy”, In advance in Cryptology-Eurocrypt 1992.

[23] M.Burmester and Y.Desmedt, “A secure and efficient conference key distribution system”, In advance in Cryptology-Eurocrypt 1994.

[24] M.Burmester and Y.Desmedt, “ Efficient and secure Conference key distribution ”, In Combridge workshop on security protocols, volume 1189 of lecture Notes in computer science, pp. 119-129, Springer-Verlag, Berlin Germany, Apr. 1996.

[25] I. Ingemarsson, D. T. Tang, and C. K.Wong. A conference key distribution system. IEEE Transactions on Information Theory, 28(5):714–720, Sept. 1982.

[26] W.Diffie, M.Hellman, “New directions in cryptography”, IEEE Transactions on Information Theory, vol.22, no.6, pp.644-654,1976.

[27] Wen-Guey Tzeng. A Practical and Secure Fault-Tolerant Conference-Key

Agreement Protocol. In Proc. Of PKC2000, Lecture Notes in Computer Science. Springer-Verlag, 2000

[28] Y.Kim, A. Perrig, and G. Tsudik. Communication-Efficient Group Key Agreement. In Proc. of International Federation for Information Processing (IFIP SEC 2001), June 2001

[29] O. Pereira and J. J. Quisquater. A Security Analysis of the Cliques Protocols Suites. In 14-th IEEE Computer security Foundations Workshop. IEEE Computer Society Press, June 2001.

[30] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology, 13(3):361-396,2000

[31] D. Pointcheval. Secure Designs for Public-Key Cryptography based on the Discrete Logarithm. To appear in Discrete Applied Mathematics, Elsevier Science, 2001

[32] M. Jakobsson and D. Pointcheval. Mutual Authentication for Low-Power Mobile Devices. In Proc. of Financial Cryptography '2001, 2001

[33] E. Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquater. Provably Group Diffie-Hellman Key Exchange. In Proc. of 8th ACM Conference on Computer and Communications Security, Nov 2001

[34] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In S. Jajodia, editor, 7th ACM Conference on Computer and Communications Security, pages 235–244, Athens, Greece, Nov. 2000. ACM Press.

[35] M.Steiner, G.Tsudik, M.Waidrer, "Diffie-hellman key distribution extended to groups", In ACM conference computer and communication security, pp. 31-37, Mar. 1996.

[36] M.Steiner, G.Tsudik, M.Waidrer, "CLIQUES: A new approach to group key agreement", In IEEE International conference on distributed computing system, May 1998.

[37] N. Asokan and Philip Ginzboorg. Key-agreement in ad-hoc networks. Elsevier Preprint, 2000.

[38] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. Authenticated group key agreement and friends. In Proc. 5th ACM Conference on Computer and Communications Security, pages 17-26, San Francisco, CA USA, November 1998. ACM Press.

[39] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. New multiparty

authentication services and key agreement protocols. IEEE Journal on Selected Areas in Communications, 18(4):628-640, April 2000.

[40] Giuseppe Ateniese and Gene Tsudik. Group signatures a' la carte. In Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 848-849, Baltimore, MD USA, January 1999. ACM Press. USC Technical Report 98-676.

[41] Giuseppe Ateniese and Gene Tsudik. Some open issues and new directions in group signatures. In Proc. 3rd International Conference on Financial Cryptography (FC'99), volume 1648 of LNCS, pages 196-211, Anguilla, British West Indies, February 1999. Springer.

[42] L. Dondeti, S. Mukherjee, and A.Samal. Disec: A distributed framework for scalable secure many-to-many communication. In Proce. of The Fifth IEEE Symposium on Computers and Communications (ISCC 2000), July 2000.

[43] G. Caronni, M. Waldvogel, D. Sun, N. Weiler, and B. Plattner. The Versa Key framework: Versatile group key management. IEEE Journal on Selected Areas in Communications, 17(9), Sept. 1999.

[44] Klaus Becker and Uta Wille. Communication complexity of group key distribution. In Proc. 5th ACM Conference on Computer and Communications Security, pages 1-6, San Francisco, CA USA, November 1998. ACM Press. 11 HUT TML 2000 Tik-110.501 Seminar on Network Security

[45] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In Advances in Cryptology - EUROCRYPT '94, volume 950 of LNCS, pages 275-286, Perugia, Italy, May 1994. Springer.

[46] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architecture. Internet-Draft draft-wallner-keyarch-00.txt, June 1997.

[47] V. Shoup. Using hash functions as a hedge against chosen ciphertext attacks. In B. Preneel, editor, Advances in Cryptology–EUROCRYPT '2000, number 1807 in Lecture Notes in Computer Science, pages 275–288. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2000.

[48] Ingemar Ingemarsson, Donald T. Tang, and C. K. Wong. A conference key distribution system. IEEE Transactions on Information Theory, IT-28(5):714-720, September 1982.

[49] Silja Ma"ki, Maarit Hietalahti, and Tuomas Aura. A survey of ad-hoc

network security. Interim report of project 007- security of mobile agents and ad-hoc societies, Helsinki University of Technology. Laboratory for Theoretical Computer Science, September 2000.

[50] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-hellman key distribution extended to group communication. In 3rd ACM Conference on Computer and Communications Security, pages 31-37, New Delhi, India, March 1996. ACM Press.

[51] Michael Steiner, Gene Tsudik, and Michael Waidner. Key agreement in dynamic peer groups. IEEE Transactions on Parallel and Distributed Systems, 11(08), August 2000.

[52] 袁丁、范平志，“一个安全有效的会议密钥分配方案”,西南交大学报

[53] D.Denning and G.Sacco, “Timestampt in key distribution extended to groups”, In ACM conference computer and communication security, pp.31-37, Mar. 1996

[54] Gordon J. Strong Primes are Easy to Find[J]. Eurocrypt , 1984:21-223

[55] M Rabin. Probabilistic algorithms for testing primality Journal on Number theory[J]. 1980:128-138

[56] Bruce Schneier, 《Applied Cryptography--Protocols,Algorithms, and Source code in C》,机械工业出版社，pp.169-171, 2000

[57] D. Knuth, The Art of Computer programming: Volume 2, Seminumerical Algorithms, 2nd edition , Addison-Wesley,1981

[58] 侯俊杰著，《深入浅出 MFC》，华中科技大学出版社，2001

[59] 黄庆生，汤毅，戴宁著，《精通 Visual C++6.0》，人民邮电出版社，1999

[60] http://csrc.nist.gov/encryption/aes/

[61] 朱三元，网络通信软件设计指南。人民邮电出版社，1997

[62] 蒋东行，林鄂华，Windows Sockets 网络程序设计指南，清华大学出版社，1996

[63] Richard C. Leinecker, Tom Archer 著，张艳，王文学，张谦等译，Visual C++宝典，电子工业出版社，1999

致 谢

在我的硕士研究生学习生活即将结束的时候，我要衷心地感谢我的导师范平志教授。在求学和科研期间，他对我的课程学习、研究方向、硕士论文的选题以及论文的撰写等方面都作了精心的指导，并且给我提供了一个优越的学习环境，使我能顺利的完成学业。范老师严谨的治学态度、渊博的学识以及平易谦和的为人给我留下了深刻的印象，他永远是我最尊敬的老师！

我还要感谢袁丁老师和桑应朋同学，他们在论文的整体设计以及编程方面给了我许多指导和帮助；

感谢戴云博士以及实验室的各位同学，大家朝夕相处，生活上互相关心、遇到问题互相讨论，使我受益匪浅；

最后还要感谢我的父母和爱人，是他们对我的支持和鼓励让我能够全心的投入到研究工作中，并顺利完成学业！

感谢所有帮助过我的老师、朋友，谢谢！

攻读硕士学位期间发表的论文及著作

1. 方旭明、赵旸等合译《无线与移动网络结构》，人民邮电出版社，2002 年 5 月：pp102-129&pp149-171。

2. 赵旸，移动通信系统的安全性研究 ，通信与信息技术，2002 年，Vol.141(No.12)：pp54-56。

3. 赵旸、桑应朋、袁丁，基于对等网络的安全通信系统的设计与实现 ，计算机应用，2003 年第 5 期。

4. 赵旸，即时安全通信会议密钥分配方案 ，已收到《计算机工程》录用通知，拟定于 2004 年 1 月发表。