摘　要

移动商务是依托快速发展的移动通信网络，通过移动终端，随时随地存储、传输和交流各种商业信息，进行商业活动的创新业务类型。

移动商务虽然发展迅速，但离各方的期望还很远，其中主要的原因是由于移动商务交易过程中涉及用户的敏感信息（如信用卡卡号），人们对移动商务的安全性还心怀疑虑。因而移动商务要想取得普及性的推广和发展，当务之急是解决其安全问题。

目前支撑移动商务的技术有很多，主要有 WAP、SMS、J2ME，但前这两种技术都有其缺陷性，不能解决相关的安全问题。本文主要介绍基于无线 Java 框架的 J2ME/J2EE 移动商务模型，试图以一种崭新的技术和方式来解决移动商务上的安全问题。

J2ME 提供了丰富的 UI 接口和与互联网的网络连接，为开发者开发移动商务程序提供了方便；J2ME 支持本地计算能力，使得可以在手机终端进行数据加密计算；同时 J2ME 在业界得到了广泛的支持，目前主流手机厂商的产品都支持这个功能。正是基于上述特点，使得解决基于 J2ME 的移动商务的安全有着重要的理论意义和实际应用价值。

本文在介绍解决移动商务安全的问题时，先引入了 J2ME 本身支持的点到点安全的移动商务模型，此模型由安全协议 HTTPS 连接支持。在分析了此负载比较重的协议在移动商务上应用的缺陷之后，提出了采用基于动态口令认证的端到端安全的移动商务模型，即对用户客户端（手机终端）采用动态口令的认证方式，以及对客户端和服务器端（移动商务应用服务器）之间的数据采用加密的方式进行传输。本人在实验中，用手机模拟程序来实现提出的安全模型，经过分析测试，实验结果和预期值相符合。

本文的最后对全文做了个总结，并对进一步优化这个模型提出了展望。

关键词

移动商务；无线 Java；J2ME；HTTPS；端到端安全；动态口令认证；数据加密

Abstract

M-commerce, a new service type in business, is rapidly developing based on mobile communication network. By using M-commerce, we can store, transmit and exchange any kind of commercial information through mobile devices whenever we want.

Although M-commerce grows very fast, so far it is a long road to meet people's expectations. The main obstacle is that people cast doubt on the security of M-commerce. Because some clients' private and sensitive information (such as credit card NO) is involved in the process of M-commerce transaction. Therefore in order to make a real advancement and comprehensive development in M-commerce, the key is to solve the security problem.

Presently there are many kinds of technologies supporting M-commerce. Among them are WAP, SMS, J2ME and so on. However the former two have some flaws and fail to solve the security problem. This paper presents a new J2ME/J2EE M-commerce model on the base of wireless Java and tries to solve the security problem of M-commerce with a novel technique and perspective.

J2ME supplies lots of UI interface and network connection with Internet, also, J2ME supports local calculation capability, meanwhile, full support to J2ME is realized in industry. So it's very significant and valuable to solve the security of M-commerce based on J2ME.

In the first part of this paper, a secure point-to-point M-commerce model (which is supported by J2ME itself) is introduced. This model is supported by secure protocol HTTPS connection. Some disadvantage of application of such protocol with a heavy load on M-commerce is analyzed, followed by presenting a new end-to-end secure M-commerce model using dynamic password authentication. In detail, using dynamic password to authenticate client (mobile device) and transmitting encrypted data between client and server (M-commerce application server).

The author also writes a program to emulate the secure model proposed in this paper. By analyzing and testing, the experiment result matched author's expectation very well.

At the end of this paper, the author gives summary to the whole paper and brings forward a prospect of further optimizing this model.

Key words

M-Commerce; Wireless Java; J2ME; HTTPS; End-to-end security;
Dynamic password authentication; Data encryption

第一章 绪言

1.1 移动商务的发展

所谓移动商务（M-Commerce），是指利用手机、PDA 等移动通信设备与因特网有机结合，进行电子商务活动。移动商务包括移动支付、移动股市、手机银行等。

移动支付是指交易双方为了某种货物或者业务，通过移动设备进行商业交易。移动支付使用手机、PDA 等移动终端来完成交易。

移动股市服务通过手机服务使您可以随时随地通过手机查询价格和股市行情，还可以运用进行股票交易。

手机银行服务是无线通信技术与银行业务结合的产物。它将无线通信技术的3A（任何时间、任何地点、任何方式）优势应用到金融业务中，为客户提供在线的、实时的服务。其主要技术模式是以银行服务器作为虚拟的金融服务柜台，客户利用移动支付终端通过移动通信网络与银行建立连接，在银行提供的交互界面上进行操作，完成各种金融交易。

移动商务的主要特点是灵活、简单、方便。它能完全根据消费者的个性化需求和喜好定制，设备的选择以及提供服务与信息的方式完全由用户自己控制。通过移动商务，用户可随时随地获取所需的服务、应用、信息和娱乐。

目前，人们逐渐意识到了融合移动通信技术的电子商务将具有更大的潜力，移动商务的市场前景普遍被业内人士看好。据预测，到 2004 年，全球将出现 10 亿移动电话用户、10 亿因特网用户，其中 5 亿为移动因特网用户；到 2005 年，将有 25%的数据业务通过移动通信设备来传输。特别是在我国，目前已经拥有近 7000 万手机用户和数目众多的 PDA，这些移动终端构成了移动商务巨大的潜在市场，移动商务时代正向我们走来。

1.2 支撑移动商务的技术

因特网、移动通信技术和其它技术的完善组合创造了移动商务。移动商务的主要特点是灵活、简单、方便。它能完全根据消费者的个性化需求和喜好定制，设备的选择以及提供服务与信息的方式完全由用户自己控制。

目前，实现移动商务的技术主要有下面几类：

1、无线应用协议 WAP（Wireless Application Protocol）

WAP 是开展移动商务的核心技术之一。通过 WAP，手机可以随时随地、方便快捷地接入互联网，真正实现不受时间和地域约束的移动商务。WAP 是一种通信协议，它的提出和发展是基于在移动中接入 Internet 的需要。WAP 提供了一套

开放、统一的技术平台，用户使用移动设备很容易访问和获取以统一的内容格式表示的 Internet 或企业内部网信息和各种服务。它定义了一套软硬件的接口，可以使人们像使用 PC 机一样使用移动电话收发电子邮件以及浏览 Internet。目前，许多电信公司已经推出了多种 WAP 产品，包括 WAP 网关、应用开发工具和 WAP 手机，向用户提供网上资讯、机票订购、流动银行、游戏、购物等服务。

2. 短消息 SMS（Short Messaging Service）

SMS 是通过移动网络用手机收发简短文本消息的一种通信机制，它通过存贮转发、实时监测的机制，提供可靠的、低开销的无线数据业务。与普通的寻呼机制不同的是，SMS 是一项有保证的双向服务。发送方可以在将短消息发送出去之后得到一条确认通知，返回传递成功或失败的信息以及不可到达的原因。SMS 系统可以支持短消息与话音、数据、传真等业务的同步传输。目前基于短信的智能 SIM 卡技术开发出了新的短信业务，如手机银行、手机钱包、电话卡充值等，用户可以通过加密了的短消息进行移动支付，实现移动商务应用。

3. Wireless Java（J2ME）

随着越来越多的移动运营商开始把无线 Java 作为利润新增点，设备厂商也投入大量精力来提供相应的技术支持。无线 Java 即 J2ME。J2ME 是 1999 年 6 月由 Sun Microsystems 第一次推向 Java 团体，经过这三年多的发展，逐渐被各种消费电子生产商所接受。使原本致力于单一领域的开发人员能将其技能发挥到跨越不同设备和应用的领域。通过基于 J2ME 的 MIDlet 应用程序，能方便地享受类似于 Internet 上的各种服务，如下载游戏等，也可进行各种在线应用，如证券炒股、信息查询等移动商务活动。

1.3 移动商务是移动增值服务的补充

自从 1995 年首次被推出以来，Java 语言逐渐成为最受开发人员欢迎的语言之一。目前，Java 语言存在三个版本：

J2SE(Java 2，Standard Edition)：Java 2 标准版，为桌面级的应用提供 Java 的运行环境（Java 虚拟机 JVM）和开发包 API，支持 Windows、Solaris 和 Linux 等平台，它也是整个 Java 体系的基础。

J2EE(Java 2，Enterprise Edition)：Java 2 企业版，为多用户的企业级服务器提供综合 Java 平台，它的实现基于 J2SE 的 API，加上服务器端运行的 API。

J2ME(Java 2，Micro Edition)：Java 2 小型版，为智能卡、移动电话等小型设备提供 Java 运行环境（K 字节 Java 虚拟机 KVM）和开发 API，它是 J2SE 的子集。也是无线 Java 技术的基础。

Java 体系结构中，前两个版本分别是针对企业开发和桌面开发而设计的，本文不作介绍。而着重介绍为各类消费电子产品开发而设计的无线 Java 的基础，J2ME。

J2ME 实现移动商务的优势

J2ME 的主要技术优势在于：专门针对移动终端等消费类电子设备设计；可移植性，应用程序能很容易地被移植到其他遵循 J2ME 或 MIDP 并且符合 CLDC 规范的设备上，有良好的跨平台能力,实现了 write once, run anywhere；有着与 J2EE(企业版)后端的无缝结合能力；更低的网络资源消耗与服务器负载，J2ME 客户机应用程序能在断开连接模式下工作并保持数据的同步；丰富的用户界面和网络交互模型，改善了的 UI 用户体验，J2ME API 为呈现功能更强的 GUI 提供了更大的可能性，这些增强的功能包括了诸如事件处理和更丰富的图形等方面；记录管理存储 RMS，MIDlet 中的动态事件处理；事务保护；密码术，J2ME 本身提供了面向 J2ME 的安全性和信任服务 API。

J2ME 是 Java 无线应用技术的重要基础，它主要运行在移动设备上，使移动设备可以和 PC 机一样成为各种网络服务的客户端。

目前的手机是一个封闭的操作系统，除非用厂商自己的软件和工具，否则无法对手机的菜单进行改动，更无法在手机上附加其他应用。而无线 JAVA 技术的应用将摆脱这些传统的束缚，无线 Java 编写的应用程序可以兼容不同的网络协定，简单说就是，只要移动网络运营商提供 Java 服务，用户的 Java 手机就可以拥有无限扩充应用的能力。

移动商务是移动增值服务的补充

移动通信业务的发展历程是由传统的移动语音业务发展到移动基础数据业务，再进一步向更丰富的移动数据增值业务方向发展。

传统的移动增值服务有：SMS，MMS，手机邮件，音乐下载，手机游戏等。无线 Java 服务是一种新的移动数据业务的增值服务，它为用户提供了一个开放的平台，能更好地为用户提供全新图形化、动态化的移动增值服务。

随着现代移动技术和互联网以及其他技术的发展，移动商务在人们的生活中起着十分重要的作用。在这个广阔的领域当中，J2ME 将以其强大的功能，增进对各厂商产品与技术的兼容性，加强服务的交互能力，使移动商务更加个性化和智能化，更大地发挥移动通信的优势。

移动商务存在的问题

移动商务存在的问题有：薄弱的安全性，无线信道资源短缺，面向用户的业务还不完善，终端设计还不够人性化。其中的安全性问题影响移动电子商务发展的关键问题。相对于传统的电子商务模式，移动商务的安全性更加薄弱。如何保护用户的合法信息（账户、密码等）不受侵犯，是一项迫切需要解决的问题。可以采取的方法是吸收传统电子商务的安全防范措施，并根据移动电子商务的特点，开发轻便高效的安全协议。

如何解决好移动商务中的安全问题，这正是本文所要介绍的。

移动商务作为一种新型的电子商务方式，利用了移动无线网络的诸多优点，相对于传统的“有线”电子商务有着明显的优势，是对传统电子商务的有益补充。尽管目前移动商务的开展还存在很多问题，但随着它的发展和飞快的普及，必将成为未来电子商务的主战场。

1.4 论文作者的工作

在基于 J2ME 技术及其框架下

为了解决移动商务中上的安全问题，本人试图从一下几个方面来考虑：

1，J2ME 本身支持安全协议比如 HTTPS 等，使用这些协议来保护移动商务中数据交易的安全性。

2，本人提出采用基于端到端安全的动态口令认证和数据加密的方式来解决其安全问题。这也是本文重点介绍的部分。此方式不使用 J2ME 本身提供的安全协议。

3，由于移动商务涉及实体很多，包括用户、移动运营商、设备提供商、银行、应用服务提供商等实体，相互之间的关系复杂，整个服务框架也非常复杂，因此本文不考虑移动商务的整个框架，只简单的考虑用户使用移动设备和应用服务提供商（如股票交易服务商）之间的交易活动，使在这两者间的数据交易安全地进行。

1.5 论文结构

第一章：介绍了移动商务的发展情况，简要叙述了 J2ME 以及论文作者所要完成的工作。

第二章：介绍无线 Java 移动商务设计模式，说明了 J2ME 在实现移动商务上的优势。

第三章：先介绍了 J2ME 的安全体系结构，重点说明了用 HTTPS 协议来实现安全的移动商务的过程。。

第四章：改进算法，也是本文的重点部分，在分析了第三章所采用的安全方式后，提出了端到端的安全移动商务模型，即采用基于动态口令的身份认证和数据加密的方式来保证交易数据的安全。

第五章：实验部分，用手机模拟程序来实现第四章提出的安全模型，以及一部分性能测试。

第六章：总结和展望，指出本文所完成部分的意义，以及不足之处，对进一步的工作提出看法和意见。

说明：在本文中所谈到的移动设备主要是指手机终端设备。

第二章　无线 Java 在移动商务上的设计模型

引言

在移动通信飞速发展、手机用户增长的同时，移动通讯的业务服务内容也逐渐发生变化。虽然传统的语音业务仍然占据一席之地，但不像前几年那样快速发展，数据通讯已经成为发展新的价值增长点。

数据业务的实现依赖于两个方面：一方面取决于移动运营商；另一方面取决于手机本身。为了最大限度利用移动数字通讯业务，手机的功能自然成为广大开发商、网络服务提供运营商、手机用户问题之一。普通的手机只具备语音、短消息功能，WAP 手机在普通手机基础上增加了 WAP 网页浏览功能。但 WAP 服务需要一直保持上网才可以运行，并且要按使用时间收费，所以基于 WAP 的应用通常不能离线运行。

相比 WAP，支持 J2ME 的手机，简称为 Java 手机，在众多的手机行业中异军突起。Java 手机秉承了 Sun 公司一贯倡导的“Write Once，Run Anywhere”的宗旨，其嵌入式 Java 虚拟机 KVM 平台开放，支持标准的网络通信协议（如：TCP/IP、UDP、HTTP 等），可以充分利用新增的无线数据业务，在手机上实现网页浏览、邮件收发、移动商务等功能。

Java 手机不再是传统的功能固定的手机，而是成为了一个类似于 PC 的小型计算机，而其操作系统就是嵌入式 Java 虚拟机 Java KVM。这个虚拟机是手机厂商在硬件上实现的，它支持动态地自动下载安装、删除应用程序。Java 手机开发人员可以为手机用户定制各种应用程序，从而使 Java 手机更具有个性化，甚至可以实现比较大的企业级应用。因此，Java 手机一出现，就迅速成为众多开发商和运营商关注的焦点。对于于运营商来说，Java 技术为动态部署增值性无线数据服务提供一种安全的平台，无线门户将能够更快地发挥其潜力并迅速繁荣。而无线门户的繁荣，又将为个人和企业移动用户带来更广阔的应用空间，从而为运营商带来更多的利润。

J2ME 的体系结构

J2ME 是 Sun 的 Java 2 平台微型版，采用 3 层结构设计。最低层为配置层（Configuration），包括虚拟机（VM）和类库两部分，这一层与设备层（硬件及操作系统）关系紧密，由 Sun 提供参考规范和源代码，设备厂商根据设备特征进行相应的移植。

图 2.1 所示为 J2ME 的体系结构。

第二章　无线 Java 在移动商务上的设计模型

引言

在移动通信飞速发展、手机用户增长的同时，移动通讯的业务服务内容也逐渐发生变化。虽然传统的语音业务仍然占据一席之地，但不像前几年那样快速发展，数据通讯已经成为发展新的价值增长点。

数据业务的实现依赖于两个方面：一方面取决于移动运营商；另一方面取决于手机本身。为了最大限度利用移动数字通讯业务，手机的功能自然成为广大开发商、网络服务提供运营商、手机用户问题之一。普通的手机只具备语音、短消息功能，WAP 手机在普通手机基础上增加了 WAP 网页浏览功能。但 WAP 服务需要一直保持上网才可以运行，并且要按使用时间收费，所以基于 WAP 的应用通常不能离线运行。

相比 WAP，支持 J2ME 的手机，简称为 Java 手机，在众多的手机行业中异军突起。Java 手机秉承了 Sun 公司一贯倡导的“Write Once，Run Anywhere”的宗旨，其嵌入式 Java 虚拟机 KVM 平台开放，支持标准的网络通信协议（如：TCP/IP、UDP、HTTP 等），可以充分利用新增的无线数据业务，在手机上实现网页浏览、邮件收发、移动商务等功能。

Java 手机不再是传统的功能固定的手机，而是成为了一个类似于 PC 的小型计算机，而其操作系统就是嵌入式 Java 虚拟机 Java KVM。这个虚拟机是手机厂商在硬件上实现的，它支持动态地自动下载安装、删除应用程序。Java 手机开发人员可以为手机用户定制各种应用程序，从而使 Java 手机更具有个性化，甚至可以实现比较大的企业级应用。因此，Java 手机一出现，就迅速成为众多开发商和运营商关注的焦点。对于于运营商来说，Java 技术为动态部署增值性无线数据服务提供一种安全的平台，无线门户将能够更快地发挥其潜力并迅速繁荣。而无线门户的繁荣，又将为个人和企业移动用户带来更广阔的应用空间，从而为运营商带来更多的利润。

J2ME 的体系结构

J2ME 是 Sun 的 Java 2 平台微型版，采用 3 层结构设计。最低层为配置层（Configuration），包括虚拟机（VM）和类库两部分，这一层与设备层（硬件及操作系统）关系紧密，由 Sun 提供参考规范和源代码，设备厂商根据设备特征进行相应的移植。

图 2.1 所示为 J2ME 的体系结构。

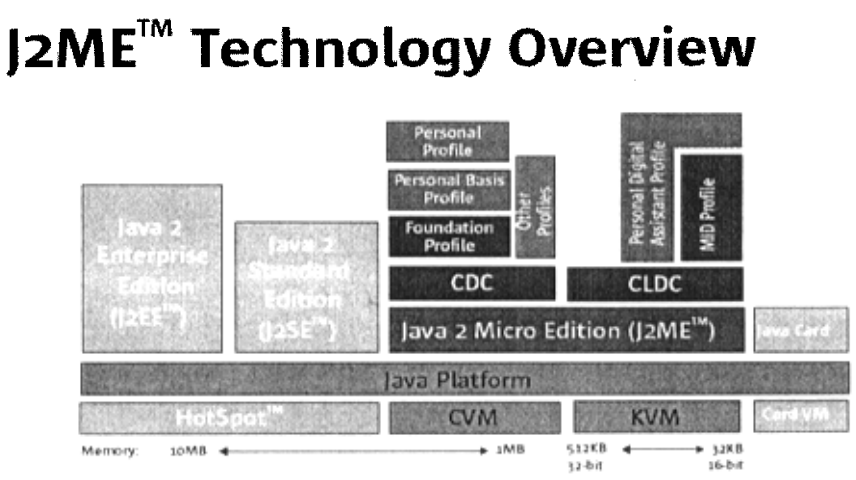


图 2.1　J2ME 的体系结构

当前 J2ME 提供有两个配置：连接设备配置（CDC）和有限连接设备配置（CLDC）。前者主要面向有较大内存和处理能力而只需有限功能的设备，如电视置顶盒、冰箱、汽车导航设备等；后者主要面向对内存和处理能力有较大限制的手持设备，如现在使用的手机、PDA 等，现在及将来大多数这些设备都已经能够接入互联网，其内存范围 160Kb（其中 128Kb 用于虚拟机及类库，至少 32Kb 用于 CLDC 规范所要求的应用程序堆栈空间）到 2M。

设备层之上是描述层（Profile），再之上则是应用层（Application）。描述层扩展了配置层功能为上层应用提供 API，如果说配置层面向设备，描述层则面向应用。可以根据需要在 CDC 或 CLDC 基础之上提供多种描述，一个配置层之上也可以有多个描述。CLDC 之上则主要提供有移动信息设备描述（MIDP），即用于手机、PDA 等移动终端的设备描述，提供 API 以支持无线应用的开发。

CLDC 和 MIDP 提供了以下包用于编程：

java.io、java.lang、java.util，属于 MIDP 的核心包，分别用来提供系统 I/O、语言支持和工具支持。包中的类来自 CLDC 并稍有增加，但都来自 J2SE。

javax.microedition.midlet，定义了 MIDP 应用程序，以及应用程序和它所运行于环境之间的交互。

javax.microedition.lcdui，为 MIDP 应用程序提供用户界面 API。

javax.microedition.rms，用来为 MIDlet 提供持久存储的机制，应用程序可以存储数据，在以后需要的时候获取这些数据。

javax.microedition.io，提供了基于 CLDC 通用连接框架的网络支持。

MIDP 框架的核心是一个 MIDlet 应用程序，这个应用程序继承自 MIDlet 类，而 MIDlet 类是提供了运行时环境和 MIDlet 应用程序之间的接口。

2.1 无线 Java 的体系结构

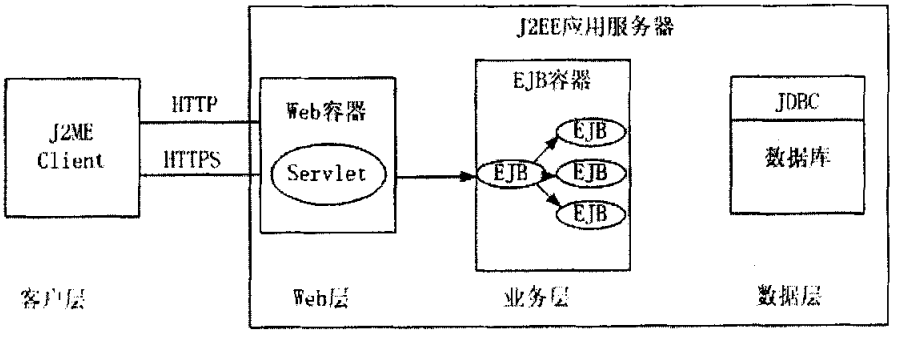


图 2.2 无线 Java 的体系结构

如图 2.2 所示，无线 Java 的体系结构分为四层：

数据层：可能是一个大型数据库或者企业的信息系统；

业务层：又称为应用服务器，一般用 EJB（企业 JavaBean）实现业务逻辑和分布式体系；它是一个运行于运营商无线网络上的后端系统，为无线 Java 用户提供 Java 应用服务，并支持内容供应商提交开发好的应用和服务，同时管理所有与无线 Java 服务相关的商业活动和行为；

Web 层：又称为 Web 服务器，采用 JSP/Servlet/XML/HTML 以及 JavaBean 技术实现无现应用的网站；

客户层：采用 J2ME 技术的移动终端设备，支持作为 J2ME 应用运行平台的 J2ME/MIDP 标准以及 Java 应用管理器，允许用户下载、安装、执行、停止和删除 J2ME 应用。

从 Web 层、业务层和数据层的构成可以看出，这三层属于 J2EE 应用服务器，并不需要专门的无线应用开发的技术，并不一定只有移动设备才可以访问这些应用，也可以用普通的 PC 作为客户端访问。这也意味着可以很方便地把原来的一些基于互联网的应用，很方便地移植到无现应用上来，甚至实现普通互联网应用和无线应用公用一个平台，这将大大地降低开发和维护成本。

MIDP 框架的核心是一个 MIDlet 应用程序，这个应用程序继承自 MIDlet 类，而 MIDlet 类是提供了运行时环境和 MIDlet 应用程序之间的接口。

2.1 无线 Java 的体系结构

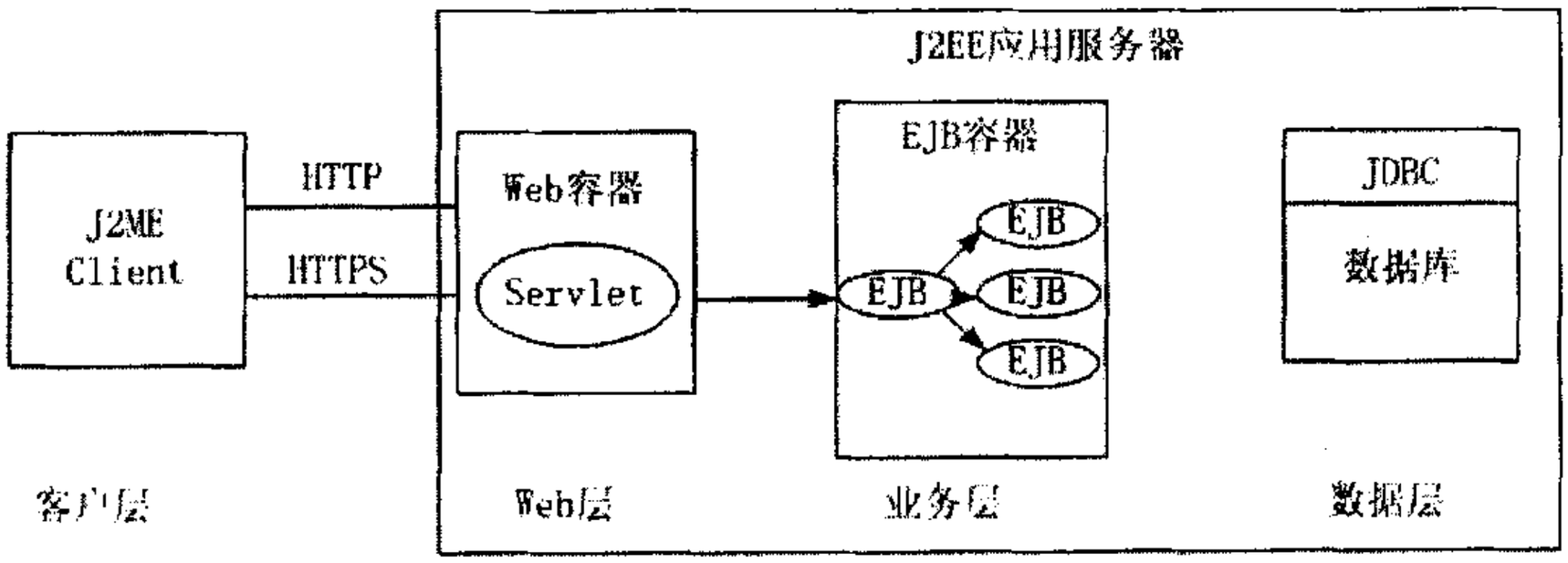


图 2.2　无线 Java 的体系结构

如图 2.2 所示，无线 Java 的体系结构分为四层：

数据层：可能是一个大型数据库或者企业的信息系统；

业务层：又称为应用服务器，一般用 EJB（企业 JavaBean）实现业务逻辑和分布式体系；它是一个运行于运营商无线网络上的后端系统，为无线 Java 用户提供 Java 应用服务，并支持内容供应商提交开发好的应用和服务，同时管理所有与无线 Java 服务相关的商业活动和行为；

Web 层：又称为 Web 服务器，采用 JSP/Servlet/XML/HTML 以及 JavaBean 技术实现无现应用的网站；

客户层：采用 J2ME 技术的移动终端设备，支持作为 J2ME 应用运行平台的 J2ME/MIDP 标准以及 Java 应用管理器，允许用户下载、安装、执行、停止和删除 J2ME 应用。

从 Web 层、业务层和数据层的构成可以看出，这三层属于 J2EE 应用服务器，并不需要专门的无线应用开发的技术，并不一定只有移动设备才可以访问这些应用，也可以用普通的 PC 作为客户端访问。这也意味着可以很方便地把原来的一些基于互联网的应用，很方便地移植到无现应用上来，甚至实现普通互联网应用和无线应用公用一个平台，这将大大地降低开发和维护成本。

2.2 基于无线 Java 体系结构的移动商务模型

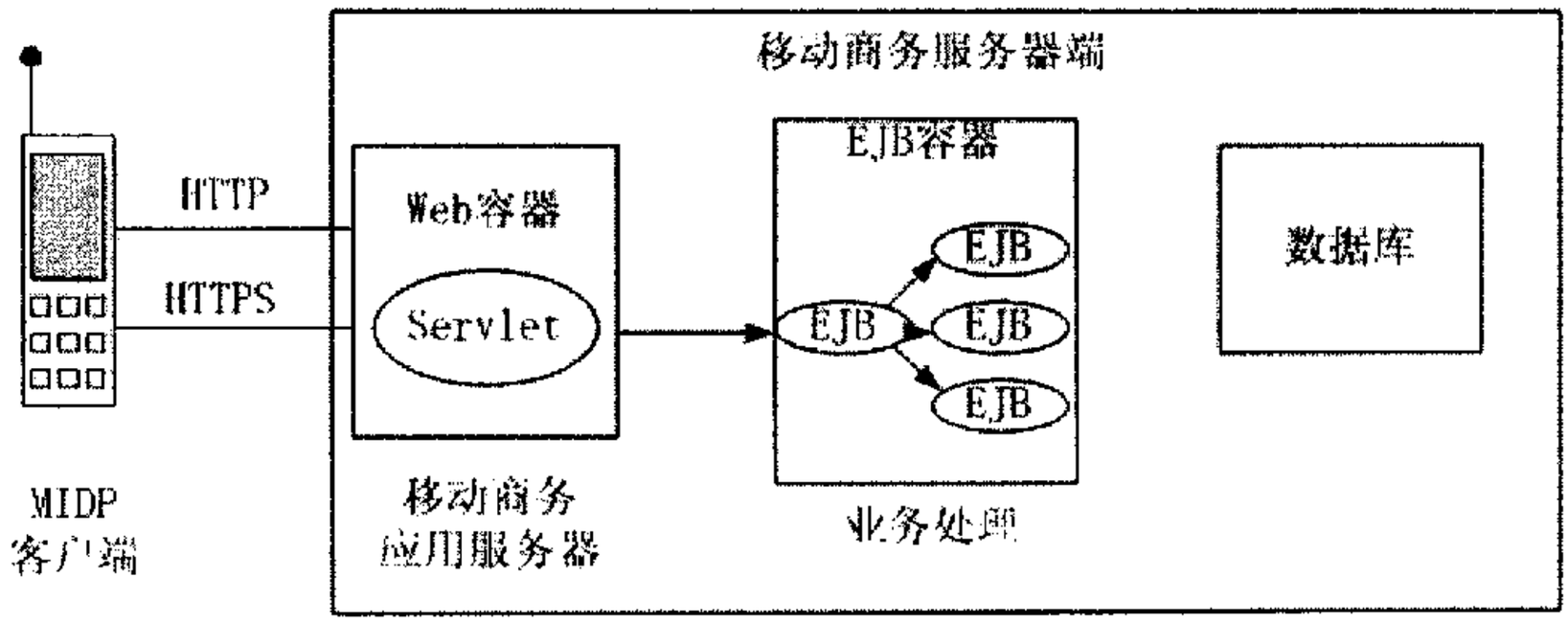


图 2.3　基于无线 Java 体系结构的移动商务模型

基于无线 Java 体系结构的移动商务模型也是分为四层结构，如图 2.3 所示，分为 MIDP 客户端，移动商务应用服务器层，业务层和数据层。用户使用 MIDP 客户端浏览由移动商务服务器提供的信息，并对信息进行操作，客户端与服务器器之间使用 HTTP 或 HTTPS 连接。服务器中的信息通过业务层与数据库进行交换。

移动商务模型包含的实体有用户、移动运营商、设备提供商、银行、应用服务提供商等实体。移动运营商提供移动网络支持，设备提供商提供支持 J2ME 的移动终端设备，应用服务提供商提供各种各样的移动商务应用供用户使用，用户进行移动商务时，通常要进行商务交易，因此银行就来担当资金结算的角色。从这里可以看出，完整的移动商务框架是一个非常复杂的体系，就拿移动支付来说，在国际上目前也还没有形成统一的标准。

因此，在不影响所讨论安全的移动商务这个前提下，我对移动商务模型进行了简化，即只考虑移动终端和移动商务服务器之间安全交易的过程。我相信，如果把这两者之间的安全问题解决了，要解决移动商务服务器和银行等之间的安全问题是非常容易的，毕竟他们同处在有线连接的网络中，有很多成熟的安全方案可供选择和实施。

在第一章谈到了支撑移动商务的技术主要有：WAP 和 J2ME。这里我们再来详细的看看两者之间的比较。

2.3 WAP 与 J2ME 在无线应用上的比较

1　WAP 简介

WAP 是一组向移动网络终端提供互联网应用的开放式协议。它是由一些世界上最主要的通信公司在 1997 年成立的无线应用协议论坛上提出的，其主要目的

是为用户提供一种通过移动终端直接上互联网的手段，由于移动终端的设备资源有限，无线网络的带宽有限等因素，互联网上的 HTTP、UDP、TCP 和 TLS 等协议不再完全适合 WAP，因此，WAP 协议与互联网标准协议之间有一定的差距，在 WAP 应用模型中，WAP 手机，WAP 网关和 WAP 内容服务器三者缺一不可，其核心是 WAP 网关，主要用来连接无线网络与互联网，实现 WAP 协议栈与互联网协议栈之间的相互转化。

2 WAP 与 J2ME 的比较

既然 WAP 与 J2ME 都提供了移动通信设备与互联网的连接，它们之间有什么区别呢？下面对 WAP 以及 J2ME 作一个比较。

首先，WAP 方式不能直接访问互联网，它是通过在移动通信设备中置入一个 WAP 网关访问互联网这样一种方式来实现的，返回时，也是先由互联网返回到 WAP 网关，再由 WAP 网关将内容翻译称移动设备能读懂的 WML，然后返回给移动通信设备进行显示，可以看出，它属于 B/S 结构，计算主要集中在服务器端，系统通信终端只起到一个被动显示的作用，而 J2ME 属于 C/S 结构，它直接支持互联网络协议，可以直接对互联网进行访问。基于 J2ME 的应用不再满足只是被动显示的角色，它支持高效率的分布式计算，这一定当移动通信设备的各种资源，特别时处理能力得到提供以后将更为明显；

其次，WAP 过分依赖在服务器和手机终端之间传递数据的网络，如果网络出现故障或暂时瘫痪，则不可避免地对移动互联网产生毁灭性影响，而且 WAP 所不具备的高交互性和安全性也成为其走向企业领域的软肋。对于大多数用户而言，基于 WAP 技术的业务成本也过于高昂，在当前移动网络速度还只有几十 K 的背景下，这种完全依赖网络传输的技术已经被证明不能担当向手机传递数据的重任。

为了应对移动数据的发展，推进移动商务等业务的发展，J2ME 技术作为一种领先的技术进入该领域，它为移动互连引入了一种新的模型，即允许手机从互联网上下载各种应用程序，实现较高效率的在线交易，并可以访问本地存储，利用 Java 平台，开发的程序可以移植到其他系统，手机可以升级、第三方开发的程序也可以很方便的下载到终端上使用。

另外，J2ME 技术为手机赋予本地计算能力，使其具有动态加载应用程序的能力。当手机用户发出请求后，其手机并不是逐一地得到处理中心返回的数据，而是首先一次性地下载 J2ME 编写的计算程序。程序的首次下载也许会花费几十秒，但随后的请求将会在手机内进行处理，然后实时地显示结果。这种本地计算能力的获得就使移动商务解决了延时和相应速度问题，从而更具有人性化，更符合人们的消费习惯。

2.4 移动商务的安全为什么如此重要

有调查显示，人们对于移动商务交易安全性的担忧，就如同早期对互联网电子商务的安全感到不安一样，这已成为普及移动商务服务的一大障碍。

我们首先来看一下移动商务的一个例子。用户使用手机浏览服务商提供的移动商务信息并准备进行交易。一般说来，完成这一步不涉及任何安全操作，因为我们假定这种信息不是敏感信息。

但当用户准备移动支付时，安全需求应运而生。用户需要向服务器提供自己的信用卡号码才能进行结算。由于任何具有信用卡号码的人都可以充当该用户并将帐记在该用户名下，显然我们需要为信用卡信息提供保密。此外，用户还需要确信他将信用卡信息提供给了恰当的服务器而不是偷盗其信用卡的服务器。

移动条件下的商务环境与有线环境间千差万别。移动商务安全面临的主要威胁有：

- 口令的易攻击性。移动设备上的初始存取或口令经常被用户忽视，从而为非法访问应用和数据创造了机会。
- 丢失数据。移动设备的存储能力日益增加。当设备发生故障或丢失时，或者其数据被意外删除且不存在与恢复功能相结合的当前数据备份时，将导致数据的永远丢失。
- 设备的丢失或被盗。移动设备被带到办公室外，其大小决定了它们很容易被放错地方。由于移动设备通常具备非常有限的内置安全特性，所以丢失它们就意味着泄露个人私有数据。

因此，必须针对这些问题提出安全的解决方案。需要提供移动设备和服务器之间数据传递的保密性，而且对交易两端的身份要进行鉴别。

尽管移动商务注重的是更多的本地业务和信息服务，交易值在各地的分布有所差别，但它所涉及的领域将不断扩大。安全性是任何移动商务取得成功最关键的因素。即使相对来说很小的安全问题也有可能失去用户信任。这意味着必须完全了解并解决安全问题才能使移动商务取得全面成功。

第三章　J2ME 支持的点到点安全应用

引言

在过去的几年里，全球无线通信业呈现了爆炸式的增长。与此同时，Internet 在全球也发展迅速，人们越来越依赖于从 Internet 上得到各种信息，同时也希望不但能通过个人计算机，而且能通过移动和无线设备来访问 Internet 上的信息。这样就引发了无线 Internet 革命。

传统移动电话的功能比较少，只有通话和短消息功能，随后移动电话上又增加了一些简单的附加应用，如电话簿和电话铃声编辑功能等。而现在随着 WAP 技术的发展，移动电话增加了访问 Internet 的功能，使用户可以直接在手机上以无线方式(WAP 和 GPRS)浏览网页。

与 WAP 相比，J2ME 在解决移动设备访问 Internet 上更具竞争力。原因在于 Java 语言是跨平台运行的，这一特性使第三方软件开发商可以很容易地介入进来开发应用程序，也可以很方便地将应用程序安装移植到移动电话上，开发周期也大大缩短，而且还能支持应用程序的动态下载和升级。J2ME 除了能够更好地增强完善移动电话上已有的应用外，还进一步增加了字典、图书、游戏、遥控家电和定时提醒等新的应用，并能访问电子邮件、即时消息、股票和电子地图等信息。

除此之外，J2ME 本身还具有如下的安全特性：

• 由于中间没有 WAP 网关，J2ME 应用程序能够提供从后端到移动设备的可伸缩的端到端安全性。当后端发展成消息驱动的 Web 服务框架时，这一点就尤其重要。

• J2ME 应用程序能够在本地存储和处理数据，因此减少了网络流量。这不仅节省了宝贵的无线带宽和减少了延迟时间，而且降低了关键信息被截取或阻断（例如，通过拒绝服务攻击）的可能性。

• J2ME 应用程序有效地利用了设备处理能力。

由于移动设备在人们的生活中起着十分重要的作用，因此有关无线 Java 的应用，一个很重要的方面就是保护移动设备的安全。移动设备的安全主要有两方面：一方面是移动设备运行下载的无线应用程序的安全，由 J2ME 的安全体系结构来保证；另一方面是移动设备无线访问 Internet 传输数据的安全，用数据加密、身份认证和安全的通信协议来保证。

3.1 J2ME 安全体系结构

随着移动设备和移动电子商务在人们的生活中起着越来越重要的作用，安全的话题变得越来越重要，安全性在移动计算和无线环境里显得尤为重要。

J2ME 体系结构的核心概念：MIDP，CLDC，KVM。建立在 CLDC 上的 MIDP，允许程序员开发可从开放网络下载无线应用程序（无线应用程序可能下载到移动终端，可以离线使用）。应用程序下载后，就可以在用户的移动设备上运行。但用户关心的是安全问题，比如运行应用程序是否会损坏移动电话，或者恶意删除设备上的数据以及传递数据到远端服务器。因此安全性在移动计算和无线环境里显得尤为重要。

Java 2 标准版所提供的安全模型提供给开发者一个内建于 Java 平台的强有力的、灵活的安全框架。开发者可以创建精细的安全策略有及为每个应用定义明晰的权限。但由于 Java2 标准版中全部用于安全性的代码超过了 CLDC 和 MIDP 中对可用内存的预算，因此，为 CLDC 和 MIDP 定义的安全模型需要一些适当的简化。

CLDD 和 MIDP 标准化成果的高级目标就是建立一个具有高可移植性、安全的、资源占有少的应用开发平台，使第三方可以为这些资源受限的移动设备进行开发。

CLDC 和 MIDP 的安全模型定义了以下三种不同的级别：

1）底层安全：这层的目的是保证 KVM 执行下载的 MIDP 应用程序，不会毁坏移动信息设备。

2）应用级安全：应用级安全意味着运行一个移动信息设备上的 Java 应用仅仅可以访问设备和 Java 应用环境允许它访问的那些库、系统资源和其它组件。

3）点对点安全：点对点安全性模型是确保任何在移动设备上初试化的事务在设备到为这个事务提供服务的实体（例如 Internet 上的一个服务器）之间的整条路径上是受到保护的。

3.1.1 底层安全

在移动设备中，Java 虚拟机的一个关键要求就是底层虚拟机安全。一个运行在虚拟机之上的应用决不能有害于 Java 虚拟机所在的设备或者损害虚拟机本身。CLDC 规范要求支持 CLDC 标准的 Java 虚拟机必须能够排除无效类文件，这是由类文件检查器来完成的。

由于 Java2 标准类文件检查器的静态和动态内存超出了典型的 CLDC 目标设备的能力。所以 CLDC 规范中定义了新的类文件检查器。在 Sun　KVM 中新检查器的实现需要 12K 的 intel x86 二进制代码，对典型类文件的检查在运行时占用少

3.1 J2ME 安全体系结构

随着移动设备和移动电子商务在人们的生活中起着越来越重要的作用，安全的话题变得越来越重要，安全性在移动计算和无线环境里显得尤为重要。

J2ME 体系结构的核心概念：MIDP，CLDC，KVM。建立在 CLDC 上的 MIDP，允许程序员开发可从开放网络下载无线应用程序（无线应用程序可能下载到移动终端，可以离线使用）。应用程序下载后，就可以在用户的移动设备上运行。但用户关心的是安全问题，比如运行应用程序是否会损坏移动电话，或者恶意删除设备上的数据以及传递数据到远端服务器。因此安全性在移动计算和无线环境里显得尤为重要。

Java 2 标准版所提供的安全模型提供给开发者一个内建于 Java 平台的强有力的、灵活的安全框架。开发者可以创建精细的安全策略有及为每个应用定义明晰的权限。但由于 Java2 标准版中全部用于安全性的代码超过了 CLDC 和 MIDP 中对可用内存的预算，因此，为 CLDC 和 MIDP 定义的安全模型需要一些适当的简化。

CLDD 和 MIDP 标准化成果的高级目标就是建立一个具有高可移植性、安全的、资源占有少的应用开发平台，使第三方可以为这些资源受限的移动设备进行开发。

CLDC 和 MIDP 的安全模型定义了以下三种不同的级别：

1）底层安全：这层的目的是保证 KVM 执行下载的 MIDP 应用程序，不会毁坏移动信息设备。

2）应用级安全：应用级安全意味着运行一个移动信息设备上的 Java 应用仅仅可以访问设备和 Java 应用环境允许它访问的那些库、系统资源和其它组件。

3）点对点安全：点对点安全性模型是确保任何在移动设备上初试化的事务在设备到为这个事务提供服务的实体（例如 Internet 上的一个服务器）之间的整条路径上是受到保护的。

3.1.1 底层安全

在移动设备中，Java 虚拟机的一个关键要求就是底层虚拟机安全。一个运行在虚拟机之上的应用决不能有害于 Java 虚拟机所在的设备或者损害虚拟机本身。CLDC 规范要求支持 CLDC 标准的 Java 虚拟机必须能够排除无效类文件，这是由类文件检查器来完成的。

由于 Java2 标准类文件检查器的静态和动态内存超出了典型的 CLDC 目标设备的能力。所以 CLDC 规范中定义了新的类文件检查器。在 Sun KVM 中新检查器的实现需要 12K 的 intel x86 二进制代码，对典型类文件的检查在运行时占用少

于 100 字节动态 RAM。

类文件检查器的操作有两个阶段，如图 3.1 所示：

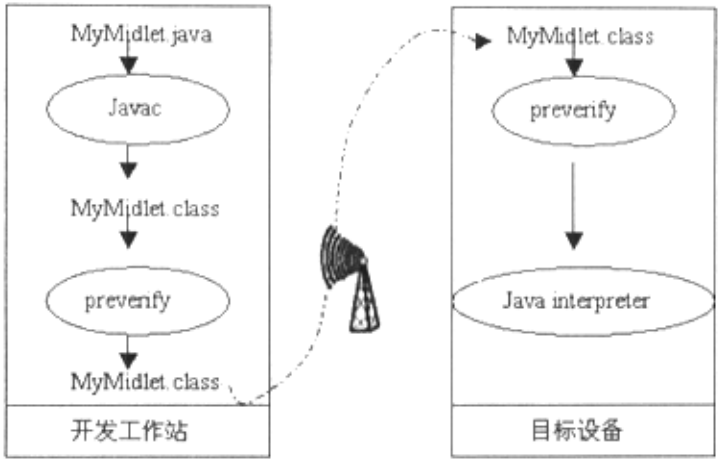


图 3.1　类文件检查的两个阶段

预检查：类文件必须通过一个预检查工具。这个工具向 Java 类文件中插入一些特定的属性。经过预检查后的类文件一般比原始类文件大 5%。

设备里检查：在运行时，虚拟机的运行时检查器组件使用由检查生成的附加属性进行真正的、高效的类文件检查。检查器要求类文件中的方法包含特殊的 StackMap 属性。

3.1.2 应用级安全

虽然类文件检查在保证 Java 平台安全性中扮演了非常重要的角色，但是由类文件检查器提供的安全性对它自身的安全是不够的。类文件检查器只能保证给定的应用是一个有效的 Java 程序。仍然存在几种其它潜在的安全隐患不会被检查器所注意到。比如，访问外部资源如文件系统、打印机、红外设备、本地库或者网络都超出了类文件检查器检查的范围。

CLDC/MIDP 提供了沙箱模型，使得无线 Java 应用程序在一个封闭的环境（沙箱）中能够安全地运行。KVM 提供的沙箱模型与 Java2 标准版中的模型是不同的。它指：

类文件被正确地检查并且保证是有效的类文件。

只有一些预定义的 Java API 可以被程序员使用。

设备上 Java 应用的下载和管理是在虚拟机内部的本地代码上实现的，而且不允许程序员覆盖类加载器或定义他们自己的类加载器。

于 100 字节动态 RAM。

类文件检查器的操作有两个阶段，如图 3.1 所示：

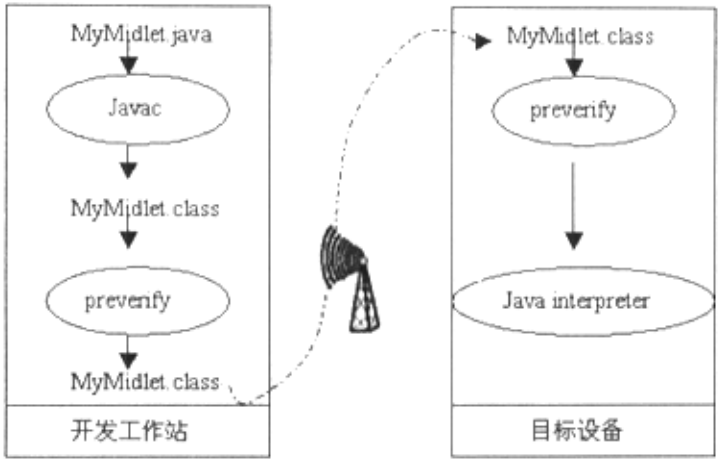


图 3.1　类文件检查的两个阶段

预检查：类文件必须通过一个预检查工具。这个工具向 Java 类文件中插入一些特定的属性。经过预检查后的类文件一般比原始类文件大 5%。

设备里检查：在运行时，虚拟机的运行时检查器组件使用由检查生成的附加属性进行真正的、高效的类文件检查。检查器要求类文件中的方法包含特殊的 StackMap 属性。

3.1.2 应用级安全

虽然类文件检查在保证 Java 平台安全性中扮演了非常重要的角色，但是由类文件检查器提供的安全性对它自身的安全是不够的。类文件检查器只能保证给定的应用是一个有效的 Java 程序。仍然存在几种其它潜在的安全隐患不会被检查器所注意到。比如，访问外部资源如文件系统、打印机、红外设备、本地库或者网络都超出了类文件检查器检查的范围。

CLDC/MIDP 提供了沙箱模型，使得无线 Java 应用程序在一个封闭的环境（沙箱）中能够安全地运行。KVM 提供的沙箱模型与 Java2 标准版中的模型是不同的。它指：

类文件被正确地检查并且保证是有效的类文件。

只有一些预定义的 Java API 可以被程序员使用。

设备上 Java 应用的下载和管理是在虚拟机内部的本地代码上实现的，而且不允许程序员覆盖类加载器或定义他们自己的类加载器。

程序员不能下载任何包含本地函数功能的新库或者访问任何不是由 CLDP 和 MIDP 提供的 Java 库一部分的本地函数。

总之，J2ME 的安全体系结构，能保证移动电话安全地运行下载的无线应用程序。

J2ME 框架除了提供对移动设备本身的保护外，还用协议的方式提供了客户端和服务器端点到点的安全连接。这个协议即是 HTTPS 协议，由于 HTTPS 协议是基于 SSL/TLS 的，因此，这里先简单的介绍一下 SSL/TLS 和 HTTPS 协议。

3.2 SSL/TLS 介绍

在无线网络上传输的信息容易被截获，其中一些可能是敏感信息，比如信用卡号码和其他个人数据。为了使移动设备在移动商务上发挥更大的作用，应用程序必须保护用户信息，通过数据加密，身份鉴别和安全的通信协议。

在移动电子商务中，移动设备用户交易时的一些信息如信用卡账号、股票交易明细等必须得到保护。

电子商务已成功的使用了 HTTPS 协议，它是建立在 SSL/TLS 协议上的 HTTP 协议。SSL 在商业应用中应用广泛，也可用于端到端的安全移动商务应用中。

SSL 介绍

安全套接层（Secure Socket Layer，SSL）是一种在两台机器之间提供安全通道的协议。它是具有保护传输数据以及识别通信机器的功能。安全通道是透明的，意思是说它对传输的数据不加变更。客户与服务器之间的数据经过加密的，一端写入的数据完全是另一端读取的内容。透明性使得几乎所有的基于 TCP 的协议稍加改动就可以在 SSL 上运行。

SSL 为网络的通信提供良好的私密性。SSL 使应用程序在通信时不用担心被窃听和篡改。SSL 实际上是共同工作的两个协议：“SSL 记录协议”（SSL Record Protocol）和“SSL 握手协议” （SSL Handshake Protocol）。“SSL 记录协议”是两个协议中较低级别的协议，它为较高级别的协议，例如 SSL 握手协议对数据的变长的记录进行加密和解密。SSL 握手协议处理应用程序凭证的交换和验证。

SSL 最初是由网景（Netscape）公司提出的基于 Web 应用的安全协议，经过不断完善，现在的版本是 SSLv3。它包括：服务器认证、 客户认证（可选）、SSL 链路上的数据完整性和 SSL 链路上的数据保密性。对于电子商务应用来说，使用 SSL 可保证信息的真实性、 完整性和保密性。

当一个应用程序（客户机）想和另一个应用程序（服务器）通信时，客户机

程序员不能下载任何包含本地函数功能的新库或者访问任何不是由 CLDP 和 MIDP 提供的 Java 库一部分的本地函数。

总之，J2ME 的安全体系结构，能保证移动电话安全地运行下载的无线应用程序。

J2ME 框架除了提供对移动设备本身的保护外，还用协议的方式提供了客户端和服务器端点到点的安全连接。这个协议即是 HTTPS 协议，由于 HTTPS 协议是基于 SSL/TLS 的，因此，这里先简单的介绍一下 SSL/TLS 和 HTTPS 协议。

3.2 SSL/TLS 介绍

在无线网络上传输的信息容易被截获，其中一些可能是敏感信息，比如信用卡号码和其他个人数据。为了使移动设备在移动商务上发挥更大的作用，应用程序必须保护用户信息，通过数据加密，身份鉴别和安全的通信协议。

在移动电子商务中，移动设备用户交易时的一些信息如信用卡账号、股票交易明细等必须得到保护。

电子商务已成功的使用了 HTTPS 协议，它是建立在 SSL/TLS 协议上的 HTTP 协议。SSL 在商业应用中应用广泛，也可用于端到端的安全移动商务应用中。

SSL 介绍

安全套接层（Secure Socket Layer，SSL）是一种在两台机器之间提供安全通道的协议。它是具有保护传输数据以及识别通信机器的功能。安全通道是透明的，意思是说它对传输的数据不加变更。客户与服务器之间的数据经过加密的，一端写入的数据完全是另一端读取的内容。透明性使得几乎所有的基于 TCP 的协议稍加改动就可以在 SSL 上运行。

SSL 为网络的通信提供良好的私密性。SSL 使应用程序在通信时不用担心被窃听和篡改。SSL 实际上是共同工作的两个协议：“SSL 记录协议”（SSL Record Protocol）和“SSL 握手协议” （SSL Handshake Protocol）。“SSL 记录协议”是两个协议中较低级别的协议，它为较高级别的协议，例如 SSL 握手协议对数据的变长的记录进行加密和解密。SSL 握手协议处理应用程序凭证的交换和验证。

SSL 最初是由网景（Netscape）公司提出的基于 Web 应用的安全协议，经过不断完善，现在的版本是 SSLv3。它包括：服务器认证、 客户认证（可选）、SSL 链路上的数据完整性和 SSL 链路上的数据保密性。对于电子商务应用来说，使用 SSL 可保证信息的真实性、 完整性和保密性。

当一个应用程序（客户机）想和另一个应用程序（服务器）通信时，客户机

打开一个与服务器相连接的套接字连接。然后， 客户机和服务器对安全连接进行协商。作为协商的一部分，服务器向客户机作自我认证。客户机可以选择向服务器作或不作自我认证。 一旦完成了认证并且建立了安全连接，则两个应用程序就可以安全地进行通信。按照惯例，我将把发起该通信的对等机看作客户机，另一个对等机则看作服务器，不管连接之后它们充当什么角色。

SSL 是一种分层协议，它由一个记录层以及记录层上承载的不同消息类型组成。而该记录又会由某种可靠的传输协议如 TCP 来承载。图 3.2 描述了该协议的结构。

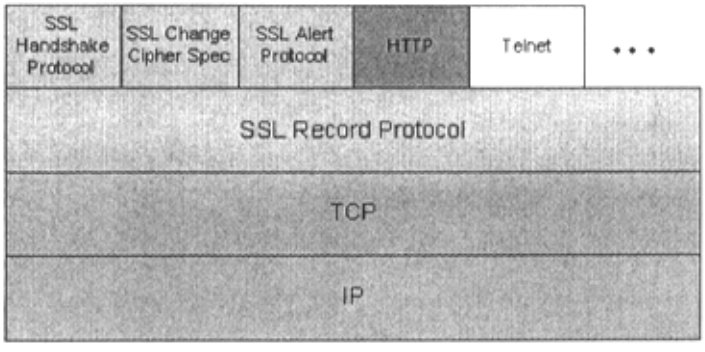


图 3.2 SSL 协议结构

TLS 介绍

由于 SSL 协议是 Netscape 公司提出的协议，为了使此协议得到业界的认可和推广，IETF 于 1996 年 5 月起对一种类似 SSL 的协议进行标准化。TLS 是在 SSLv3 的基础上稍加修改的结果。因此，两个协议之间有着密切的关联，但也有些不同之处，不能互操作。

HTTPS 介绍

HTTP 是第一种使用 SSL 保护其安全的应用层协议。通过 SSL 上的 HTTP 来获取页面的 URL 以 https://开头，以便将其与 HTTP URL 区分开来，这种方案很快就以 HTTPS 成名。

HTTPS 这种方案很简单：客户端连接到服务器，磋商一条 SSL 连接，然后在 SSL 应用数据通道上传输它的 HTTP 数据。

目前的 MIDP 2.0 支持 TLS1.0，SSLv3，支持 HTTPS 在服务器端的身份鉴别，但缺乏对客户端的鉴别机制。

SSL 的连接过程

SSL 的连接分为两个阶段，即握手和数据传输阶段。握手阶段对服务器进行认证并确立用于保护数据传输的加密密钥。必须在传输任何应用数据之前完成握手。

握手阶段：SSL 的握手有三个目的。第一，客户端与服务器需要就一组用于保护数据的算法达成一致。第二，它们需要确立一组由那些算法所使用的加密密钥。第三，握手还可以选择对客户端进行认证，当然一般情况下都不需要对客户端进行认证。

握手的过程如下图所示：

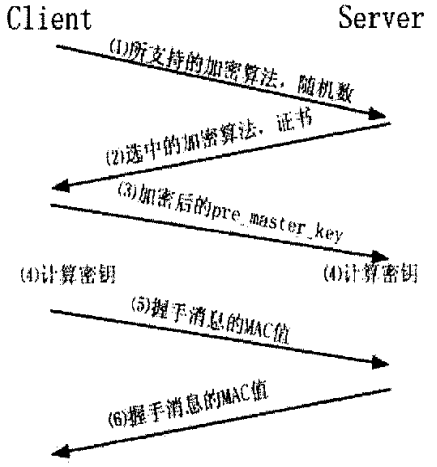


图 3.3　SSL 握手连接示意图

（1）客户端将它所支持的算法列表连同一个密钥产生过程用作输入的随即数发送给服务器。

（2）服务器根据从列表的内容中选择一种加密算法，并将器连同一份包含服务器公用密钥的证书发回给客户端。

（3）客户端对服务器的证书进行验证，并抽取服务器的公用密钥。然后，产生一个称做 pre_master_secret 的随即密码串，并使用服务器的公用密钥对其进行加密。最后，客户端将加密的信息发送给服务器。

（4）客户端与服务器根据 pre_master_secret 以及客户端与服务器的随即数值独立计算出加密和 MAC 密钥。

（5）客户端将所有握手消息的 MAC 值发送给服务器。

（6）服务器将所有握手消息的 MAC 值发送给客户端。

数据传输阶段：握手完成后，客户端和服务器端使用相同的密码算法和会话密钥，客户端和服务器端的数据通过加密后再传输，然后在对方得到解密还原。

数据加密解密是程序自动完成的，这样对用户来说，SSL 就像是一个透明的通道。

服务器端认证的必要性

在进行移动商务或其他在互联网上进行交易的时候，有必要对服务器的身份进行鉴别。通常情况要对客户端的身份进行鉴别。这是因为在进行交易时，对用户来说，用户的数据很多都是敏感信息，用户要确认这些敏感信息的接受端是一个值得信赖的实体，并且确信他们传送给服务器的信息在传输过程中不会被任何其他任何人中途截取或解密。为了标识服务器的身份，引入了数字证书，即 X.509 证书。X.509 是基于公钥密码体制和数字签名的服务。现在的版本是 X.509v3，证书格式如下：

- 版本（version）：区分合法证书的不同版本。
- 序列号（serial number）：一个整数，在 CA 中唯一标识证书。
- 签名算法标识（signature algorithm identifier）：带参数的、用于给证书签名的算法。
- 发行商名字（issue name）：认证中心 CA 的名字。
- 有效期（period of validity）：包含两个日期，即证书的生效日期和终止日期。
- 证书主体名（subject name）：获得证书的用户名，证明拥有相应私钥的主体是公钥的所有者。
- 证书主体的公钥信息（subject' s public-key information）：主体的公钥以及将被使用的密钥的算法标识。
- 发行商唯一标识（issuer unique identifier）：用于标识唯一认证中心。
- 证书主体唯一标识（subject unique identifier）:用于标识唯一证书主体。
- 扩展（extensions）：一个或多个扩展域集。
- 签名（signature）：签名算法标识。

3.3 MIDP 网络接口

MIDP 目标设备作用与一个广泛不同的无线和有线的网络，而且这些网络利用了互不兼容的传输协议。适应当前以及将来的网络技术和互连设备是 MIDP 规范的一个明确目标。

无线数据网络不同于有线数据技术。无线数据网络具有非常窄的带宽，更长的延时以及在这些网络中的连接更容易断掉或者变得不可用。

基于多种考虑，MIDP 专家组决定使用 HTTP 作为 MIDP 库的网络协议，HTTP

数据加密解密是程序自动完成的，这样对用户来说，SSL 就像是一个透明的通道。

服务器端认证的必要性

在进行移动商务或其他在互联网上进行交易的时候，有必要对服务器的身份进行鉴别。通常情况要对客户端的身份进行鉴别。这是因为在进行交易时，对用户来说，用户的数据很多都是敏感信息，用户要确认这些敏感信息的接受端是一个值得信赖的实体，并且确信他们传送给服务器的信息在传输过程中不会被任何其他任何人中途截取或解密。为了标识服务器的身份，引入了数字证书，即 X.509 证书。X.509 是基于公钥密码体制和数字签名的服务。现在的版本是 X.509v3，证书格式如下：

- 版本（version）：区分合法证书的不同版本。
- 序列号（serial number）：一个整数，在 CA 中唯一标识证书。
- 签名算法标识（signature algorithm identifier）：带参数的、用于给证书签名的算法。
- 发行商名字（issue name）：认证中心 CA 的名字。
- 有效期（period of validity）：包含两个日期，即证书的生效日期和终止日期。
- 证书主体名（subject name）：获得证书的用户名，证明拥有相应私钥的主体是公钥的所有者。
- 证书主体的公钥信息（subject' s public-key information）：主体的公钥以及将被使用的密钥的算法标识。
- 发行商唯一标识（issuer unique identifier）：用于标识唯一认证中心。
- 证书主体唯一标识（subject unique identifier）:用于标识唯一证书主体。
- 扩展（extensions）：一个或多个扩展域集。
- 签名（signature）：签名算法标识。

3.3 MIDP 网络接口

MIDP 目标设备作用与一个广泛不同的无线和有线的网络，而且这些网络利用了互不兼容的传输协议。适应当前以及将来的网络技术和互连设备是 MIDP 规范的一个明确目标。

无线数据网络不同于有线数据技术。无线数据网络具有非常窄的带宽，更长的延时以及在这些网络中的连接更容易断掉或者变得不可用。

基于多种考虑，MIDP 专家组决定使用 HTTP 作为 MIDP 库的网络协议，HTTP

是一个丰富而且被广泛使用的协议，可以在不同的无线网络中很简单的实现。HTTP 可以利用大量的服务器端的基础设施，这些设施在有线网络中存在了。另外，HTTP 传输是同步的。

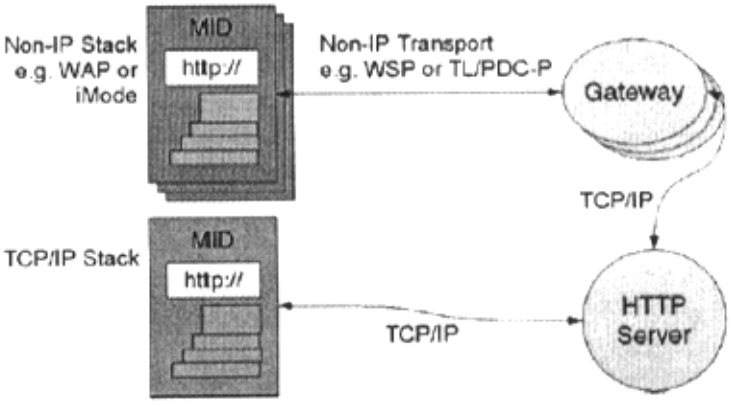


图 3.4 HTTP 网络连接

同时要说明的是支持 HTTP 协议并不意味着移动设备必须支持特定的 IP 协议，MIDP 设备的 HTTP 可以使用 IP 协议像 TCP/IP 或并非 IP 协议像 WAP 或 i-Mode（如图 3.4）来实现。在后一种情况里，使用网关来“桥接”无线协议和 Internet 上的 HTTP 服务器。

MIDP 网络 API 定义在接口 javax.microedition.io.HttpConnection 中。在 MIDP 上的 HTTP 的网络连接状态包括：安装（setup）、连接（connected）和关闭（closed）。

在首先打开到一个服务器的连接建立前，连接处于安装状态。在此状态下，需要连接到服务器的信息，例如请求参数和头。

到一个服务器的连接一旦建立就会进入连接状态。在此状态下，安装状态所设置的信息可能会发送到服务器。

最后一个状态是关闭。当一个连接被关闭并且不再使用的时候进入此状态。

HTTPS 的网络接口和 HTTP 一样，只是 URL 稍加改变，下面就是 HTTPS 网络接口的例子：

```
String url = "https://company.com/file";
HttpConnection c = (HttpConnection) Connector.open(url);
```

在 MIDP 上，HTTPS 是靠 kSSL 提供的支持。kSSL 是专门为无线受限设备提供的在客户端实现的 SSL。目前 kSSL API 在 JCP 中还没标准化。所以还不能直接用 kSSL API。

3.4 HTTPS 在移动商务上的安全应用

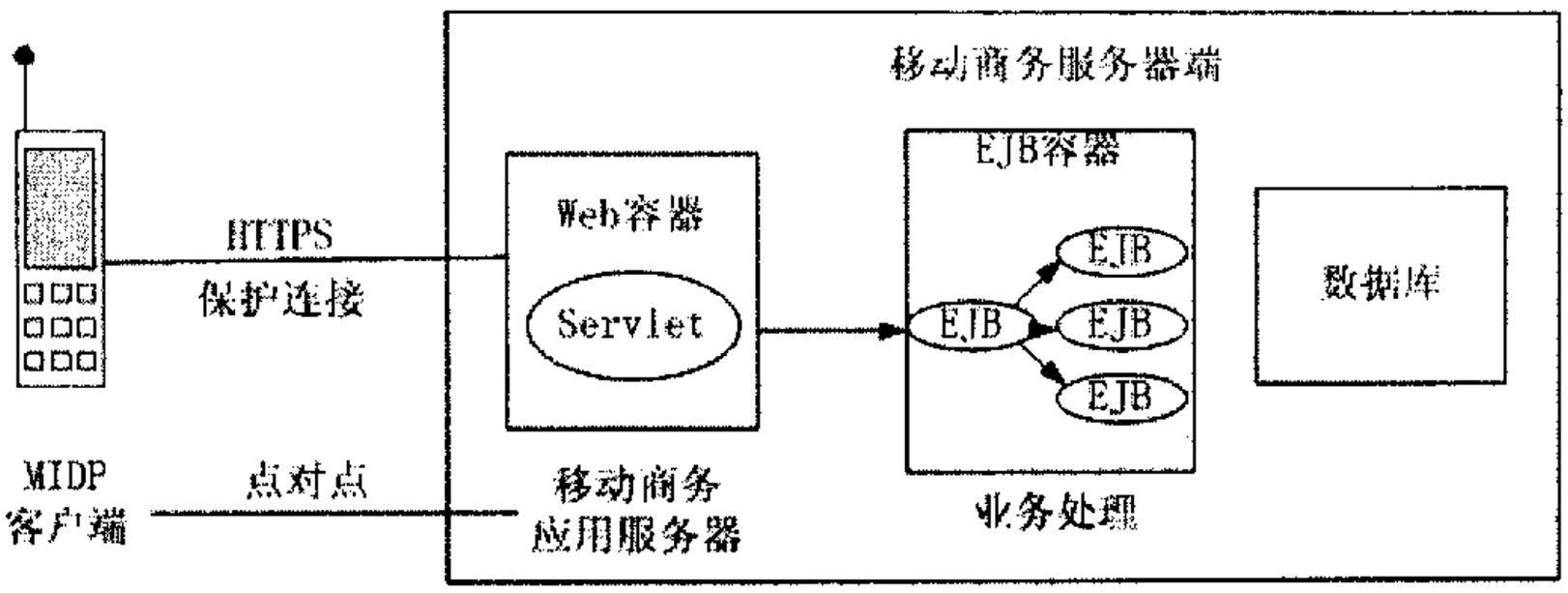


图 3.5　基于 HTTPS 点到点的移动商务安全模型

如图 3.5 所示：此模型的两端，客户端为支持 MIDP 的移动设备，服务器端为移动商务内容提供商提供的 J2EE 应用服务器，以及支持服务器的关系数据库。HTTPS 为移动设备客户端和服务器之间提供了点到点的安全连接，用户在进行移动商务时，客户端与服务器端之间交换的数据都受到安全的保护。

为了安全地进行移动商务，首先要获得服务器的身份，也就是要对服务器的身份进行鉴别，但同时也要对客户端进行鉴别。

服务器身份鉴别方法：使用基于 SSL 的 HTTPS 协议，在 SSL 握手阶段，MIDP 客户端要求服务器提供 X.509 证书，客户端收到证书后，鉴别证书。如果鉴别失败，表示不认可服务器的身份，握手失败，HTTPS 返回连接建立失败的消息；如果鉴别为真，继续下面的握手阶段。

通过使用 HTTPS 协议，能确认连接时服务器端的身份。

客户端身份鉴别的必要性

这是因为：虽然移动设备基本上是属于某个人的，但是由于某些原因比如丢失等，被他人获得后就可以冒充原设备持有人的身份。因此在发生交易时，通常需要用户输入用户名和密码来证明客户端的身份。

客户端鉴别方法：

```
String user = "user";
String password = "password";
String base = "https://somehost.com/someservlet";
String url = base + "?user=" + user + "&password=" + password;
HttpsConnection hc = (HttpsConnection)Connector.open(url);
```

服务器端通过解析 URL 获得用户名和密码，然后根据用户名查询数据库获得

其对应的密码，如果此密码和解析得到的密码相同，则表明客户端的身份有效，返回登陆成功的消息，否则返回登陆失败的消息。

虽然用 HTTPS 加密保护用户的密码，但是，也可对客户端的身份鉴别方式进一步增强，以防范客户端的重放攻击。可用消息摘要的方式解决这个问题。

这里需要指出的是：由于移动设备内存的受限，不能把 J2SE 中的 JCE 用于 MIDP 平台中，只能使用轻量级的密码算法包。密码算法可以用于数据加密、解密，也可以用于计算消息摘要以验证数据的完整性。

目前有些组织在 J2ME 数据加密方面做出了很大的贡献，他们提供免费的加密 API 包。正是基于这些人的成果，使得我自己不用写加密算法代码来实现加密功能，可以直接简单调用加密 API 来完成我要实现的功能。

在本文中，我用的加密 API 包是由 Bouncy Castle 提供的。

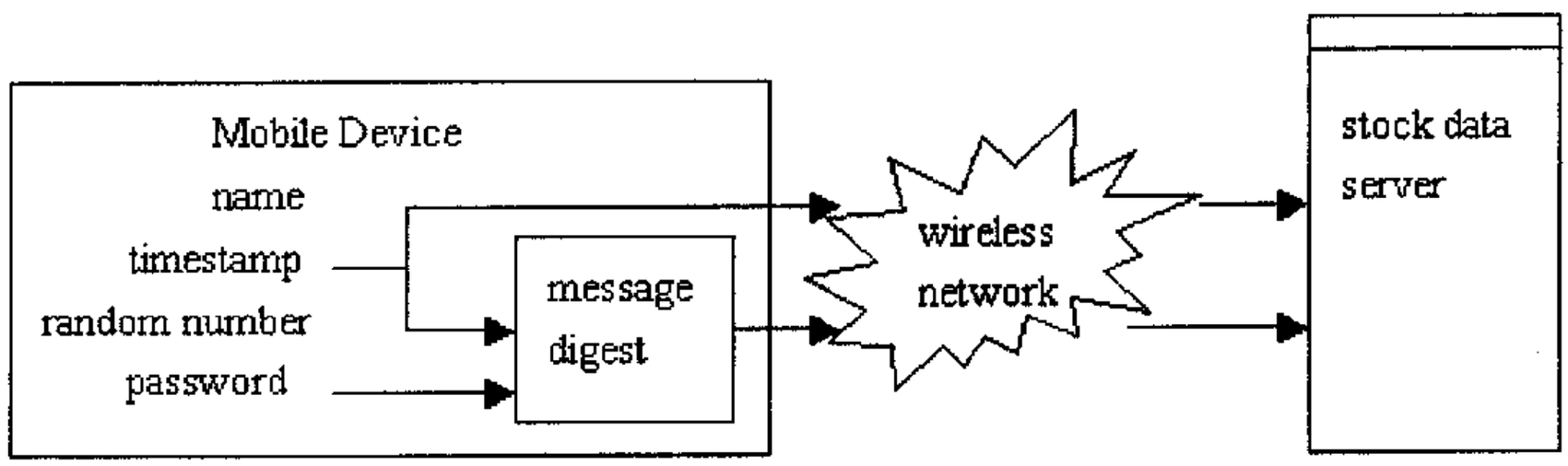


图 3.6　用消息摘要来鉴别客户端身份

如图 3.6 所示：用户端移动电话运行 MIDlet 程序创建一个时间标签和生成一个随机数字，加上用户名和密码，作为消息摘要的输入。然后 MIDlet 把用户名、时间标签、随机数字和消息摘要一起发到服务器(Servlet)。服务器端收到用户密码后，在安全的数据库中搜索与用户名相对应的密码。然后根据收到的用户名、时间标签、随机数字和才从数据库中取出的密码以同样的算法生成消息摘要，如果此摘要与收到的从用户 MIDlet 发来的摘要相同，服务器器就认为用户输入的密码是正确的，允许用户登录。

但是服务器端需要防范重放攻击。一个简单的方法就是服务器保存每个用户上一次尝试登录的时间标签，随后的每次登录尝试，服务器都要查询保存的时间标签。如果当前登录尝试的时间标签比服务器中保存的时间标签要晚，则登录尝试允许。然后用当前登录尝试的时间标签代替保存的时间标签。

下面我们来看看移动商务的一个例子，用手机终端设备来进行在线股票交易。

股票交易的客户端：移动设备上的 MIDlet 小程序。

服务器端：用 servlet 模拟的服务器端。

假设股票交易服务商的地址为：http://www.stock.com/servlet。交易的过程为：用户使用移动设备登陆到股票交易服务商提供的服务器，输入用户名和密码，登陆成功后，选择要交易的股票，输入信用卡账号买入股票。由于涉及到用户的敏感信息，如信用卡账号和购买的股票等，所以必须通过安全的方式来保证交易的安全进行。

用 HTTPS 连接来保证交易的安全性，客户端访问服务器的地址为：https://www.stock.com/servlet。

用户需要鉴别服务器的身份

1．客户端请求服务器提供证书，服务器返回证书。

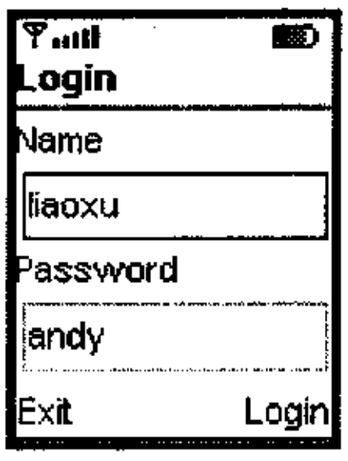
2．客户端鉴别服务器的证书。经过身份鉴别后，继续下面的握手。

3．握手完成后，服务器返回连接成功的消息，进入数据传输阶段。用户可以进行交易了。

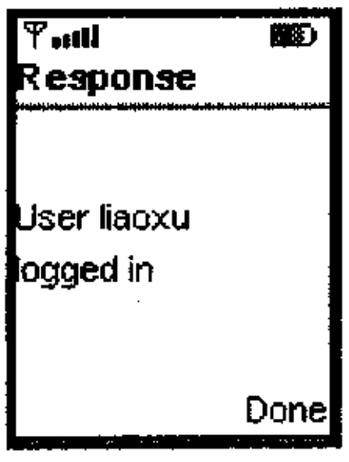
服务器也需要鉴别客户端的身份

1．客户端的身份鉴别。因为在交易时要输入用户的信用卡账号，所以要鉴别客户端的身份，确信信用卡账号是输入用户名所拥有的。使用上面谈到的消息摘要方法来鉴别客户端的身份。

2．客户端身份鉴别成功后，服务器返回登陆成功的消息。



用户端（MIDlet）登录



服务器（Servlet）返回登录信息

图 3.7　用户登录过程

3．选择要买的股票代码和购买数量，输入信用卡账号发送到服务器。服务器解析出相应的数据，完成交易。

如上所述，因为 HTTPS 是安全的协议，使得用户使用移动设备在线进行股票交易时，传输的敏感数据（如信用卡卡号等）得以保护，整个交易过程是安全地进行的。

HTTPS 的安全性几何

SSL/TLS 从诞生以来，还未成为攻击的重点，这和它自身有着良好的安全体系设计是分不开的。攻击 SSL/TLS 的一般方法是对会话密钥采取暴力或搜索密钥的方式来攻击。但是，这两种攻击方法都要截获到相当多的数据才行，对截获的数据进行分析才可能攻击。对移动商务来说，由于不同用户在不同的地方使用移动设备进行交易。在这种不确定情况下，想要截获大量的数据这是非常困难的。因此，使用基于 SSL/TLS 的 HTTPS 的移动商务，安全性是很高的。通常采用 128-bit 的密钥长度就足够了。

第四章　基于端到端安全的移动商务研究和实现

引言

上一章谈到了移动商务的安全性，以及使用 HTTPS 安全协议来保护交易数据的安全。建立在 SSL 技术基础之上的 HTTPS 是超文本传输协议的"安全"版本。SSL 提供一个已加密的 TCP 与网络服务器相连接（通常是 56 比特或 128 比特），这一加密技术在网络服务通讯与 SSL 连接相叠加时提供保护。

由于 SSL 是保护内容而非连接，尽管 HTTPS/SSL 协议非常流行而且功能又很强大，但它们原本是为因特网领域设计的。当开始将 SSL 应用于移动商务程序时，会出现许多严重问题：第一，由于基于 SSL 安全协议的 HTTPS 握手建立的时间比较长，在移动网络上耗费较长时间来建立这样的连接是很大的浪费；第二，在 HTTP 和 HTTPS 之间有着显著的性能差异，加密和解密过程要比简单地发送未经加工的信息花费更多的时间；第三，SSL 只对信息在服务器和客户端之间进行传送时对其进行保护，它并不保护已经位于服务器（比如服务器上的 EJBean）或客户端之中的信息。最后，因为 SSL 只提供点对点的加密技术而不是端对端的方式，所以当信息要传送给多个用户时，SSL 并不能起到作用。

对移动商务来说，重要的是数据交易的安全性，从这点来说，需要关注的是保护内容而非连接。为了解决这些问题，需要一种具有灵活加密方案的端到端安全性方案。本人提出一个新的解决方案：基于时间的动态口令身份认证和数据加密传输。这种方案不使用 HTTPS 协议。

4.1 端到端安全方案模型

在进行移动商务交易时，客户端和服务器端的数据经过加密后再使用 HTTP 协议传输，即采用端到端的安全模型。

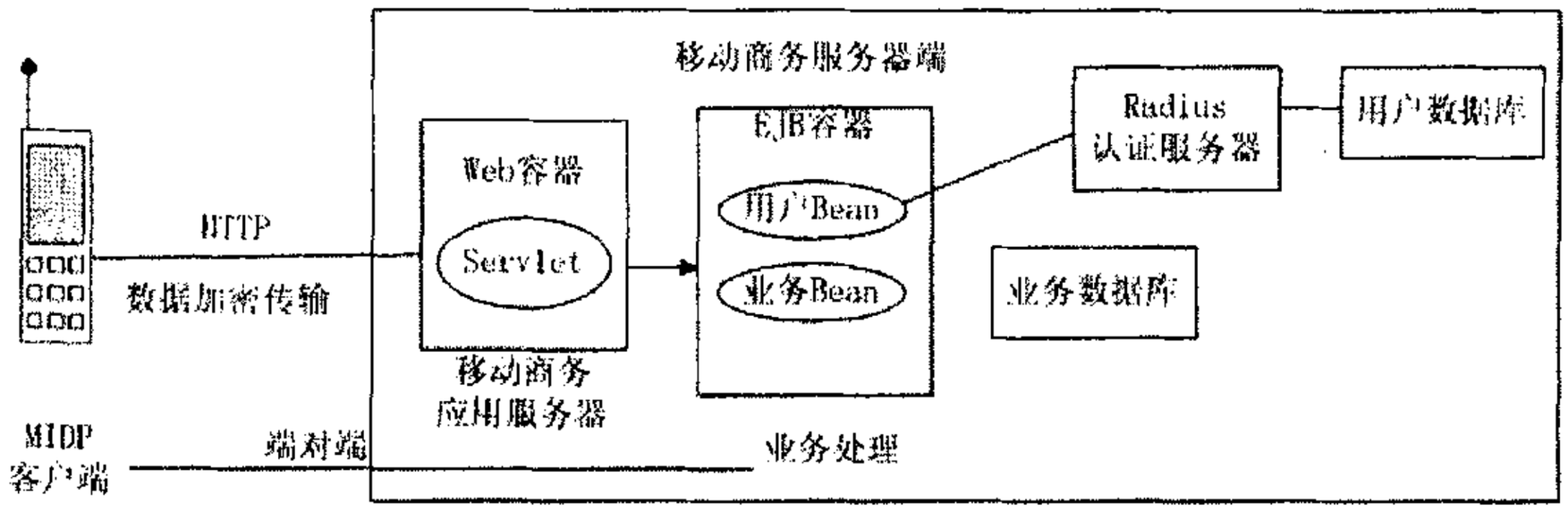


图 4.1　端到端的安全方案模型

图 4.1 所示的端到端安全模型，此模型基于第二章介绍的移动商务模型，客户端支持 J2ME，服务器端支持 J2EE。在这里，对模型做了一些扩展，增加认证

服务器来鉴别客户端的身份。

模型介绍：客户端是 MIDP 移动终端；服务器端由几部分组成：移动商务应用服务器；业务处理，包括用户信息 Bean 和业务（如股票信息等）Bean 等；数据库，包括用户数据库和业务数据库；认证服务器。

端到端模型中重要的是保护传输数据的安全，客户端和服务器端的数据经过加密后再传输。服务器端业务层和表示层之间的数据也是加密传送的。

认证服务器提供严格的身份认证，它用来鉴别客户端的用户身份。认证服务器具有自身数据安全保护功能，所用户数据经加密后存储在数据库中。

该方案可最大限度地提高移动商务的安全性。因为端到端方案保护移动商务交易中的每个薄弱环节，确保数据从用户客户端到最后目的地之间完全的安全性，包括传输过程中的每个阶段。对每一个薄弱环节都采取适当的安全性和私密性措施，可以确保整个交易过程中的安全性。

在前面章节的叙述中也谈到了客户端身份鉴别的重要性，在身份鉴别之前，先介绍一下本文身份认证所采用基于时间的动态口令身份认证。

4.2 基于时间的动态口令身份认证

在无线网络中，用常规静态密码登陆服务器时，有很多安全隐患：密码可能被猜测，截获。采用动态口令能避免使用静态密码的缺陷。

目前在有线连接的网络上使用的动态口令技术有：

时间同步(Time Synchronous)：

认证服务器与客户端之间每隔一段时间同时更换一次密码，以达到动态口令的功效。

事件同步(Event Synchronous)：（比如伪随即数和密钥的配合）

认证服务器与客户端之间以相同数字序列为密码运算因子，由数字序列的变动特性达到动态口令的功效。

非同步(Challenge/Response Asynchronous)：

由认证服务器随机产生一个数字(Challenge)并将此数字传到客户端，使用者将这个数字输入客户端令牌设备后算出该次的密码（Response），由于随机产生的数字每次都不同以达到动态口令。

这些技术都是在有线网络中使用的技术，如何用到移动设备上，这是我考虑的重点。由于 J2ME MIDP 支持本地计算和加密算法，就可以把移动设备当作 Token 设备，这样不需要其他 Token 辅助设备就可以在移动设备客户端计算动态口令。

在本文中，考虑到用户方便性和算法的复杂性，我采用时间同步的动态口令技术。

基于时间的动态口令身份认证，也即采用双因素身份认证和基于时间变化一次性密码，即通过口令计算器（移动设备Token）得到随时变化的、不可预知的、一次性有效的口令，用户在登录时用动态口令代替固定口令提交到中心进行身份认证，通过认证后，该口令即失效，既有效的提高了身份认证的安全性，同时免除了用户记忆密码和经常需要更换密码的麻烦。身份得到认证后，交易数据（比如信用卡卡号）经过加密，然后发送到服务器。

要产生动态口令，用户需要在移动设备上输入个人PIN，口令产生依赖于令牌设备和PIN。这显然要比仅仅一个密码要安全多，因为攻击者需要同时获得PIN和Token设备。另外，由于口令是基于时间动态变化的，对攻击者来说即使获得口令也是无用的；另一方面，由于移动设备的普及和J2ME技术的使用，以及J2ME提供了计算功能，使得一些相对比较复杂的计算可以在移动设备上进行，而不需要发到服务器上计算。这使得移动设备可以当作口令计算器，这也是J2ME的优势所在，使得移动商务的方便性大大提高。

4.2.1 动态口令的产生原理

对于动态口令的产生，有很多种算法。在本文中，我使用的是MD5摘要算法来产生一次性口令，通过把如下数据结合在一起计算MD5值得到口令。

1. 当前时间值，以10秒间隔为计数单位。
2. 用户输入的6-digit PIN码。
3. 设备初始化时从服务器端得到的128位密钥。

当输入 PIN 码时，客户端 MIDlet 小程序会显示计算得到的 MD5 值的前 6 位数字，这个就是一次性口令。这个口令可以被认证服务器识别，因为认证服务器也知道当前时间，用户初始密钥，用户的 PIN 码，通过计算得到用户的动态口令，两者之间做一个比较就可以判断用户的身份是否合法。为了弥补客户端和服务器之间时间的差别，服务器会接受过去 3 分钟到未来 3 分钟之间的密码。认证服务器对每个用户动态口令只识别一次，目的是为了避免重放攻击。为了提高安全性，如果连续 8 次输入口令错误，此用户将被锁定。同时如果一分钟内无按键自动退出程序。

4.2.2 动态口令的优点

● 动态口令的随机性，按时间变化，口令数字出现的概率相同，密码算法抗攻击性强。

● 采用双因素(PIN 及动态口令)身份认证技术和一次性认证原则，保证一次一密；遵循身份认证服务器的标准 RADIUS 协议。

● 用户的关键数据（如 PIN 码和初始密钥）在数据库中以加密形式存储，认证服务器对每个用户的口令连续错误次数都有严格的限制，管理员可为每个用户设置一个口令连续错误次数的最大值，当用户连续输入错误口令的次数大于该最大值，系统自动将该用户的账号锁定。

● 支持基于时间同步机制的身份认证。

● 采用专用认证服务器进行认证，保障用户应用系统的完整性和系统资源。

● 认证服务器与应用系统服务器独立设置，响应速度快。

● 使用方便灵活。

4.3 端到端的安全移动商务应用过程

目前的移动商务如用手机在线交易股票的话，参照目前证券交易所股票交易的方式，或者进行移动支付等，进行必要的实名身份登记还是有必要的。因为在交易过程中涉及到用户的信用卡卡号等，所以必须对用户的身份进行登记。在本文的设计中，要求用户到移动商务服务提供商的柜台去办理，并在管理员的指导下进行注册。注册除了身份登记外，还包括移动设备程序的初始化，移动设备客户端和服务器端的密钥交换。

4.3.1 用户注册以及初始密钥分配

过程如下：

1. 通过 OTA 下载移动商务应用程序，此程序包含用户注册，产品浏览等功能。然后把程序安装在移动设备上。OTA（Over The Air）称为空中下载技术，它使终端用户能够根据个人动态下载移动商务服务商提供的服务。

2. 注册用户信息。信息包括用户名，身份证号，PIN码等。这是用户在移动设备上输入的信息然后发送到服务器注册。管理员还要输入一些补充信息，比如用户姓名，住址等。管理员在处理完信息后，向移动设备发回注册成功的消息。

3. 用户使用移动设备输入用户名和PIN码向服务器请求初始密钥。

4. 服务器生成初始密钥保存到数据库同时把密钥发到移动设备，客户端把密钥保存到RMS管理器中。这样，移动设备和服务器端完成了初始密钥分发。

5. 最后一步，为了提高信用卡的安全性，用户需要输入信用卡账号（通

过设置密码保护）和用户的注册信息绑定。服务器将自动发给用户一个专有号码，用户在移动商务交易时以这个号码取代其信用卡号码，大大减少盗用信用卡的机率。

到此为止：用户注册过程完成。移动设备只保存初始密钥，不保存PIN码以提高设备的安全性。服务器端认证服务器中保存的数据有用户名，初试密钥和PIN码等用户信息。

如果用户要重新初始化初始密钥，可以输入PIN码后发送请求到服务器获得新的初试密钥。

在这里，移动设备就相当于一个令牌（Token），由于用户名和初始密钥是一一对应的，而且客户端的初始密钥就存放到移动设备上，因此用户名就相当于令牌序列号。

说明：用户需记住注册过程中所设定的PIN码，因为此PIN码是鉴别用户身份的重要因素之一。

如本章引言所谈到的，不采用HTTPS协议来鉴别服务器端的身份。但是对移动商务交易来说，为了使客户端相信服务器的身份，必须采用某种机制来确认服务器的身份。对客户端来说，可以采用前面提到的基于时间的动态口令身份认证。本文采用公钥算法来实现服务器端的身份认证，同时也用公钥算法来实现交易数据的加密解密。

4.3.2 服务器身份认证

要对服务器的身份进行鉴别，先要生成公私密钥对。过程如下：

用户在客户端输入PIN码，通过计算生成一个公钥/私钥密钥对，存储到RMS记录管理系统中，同时把公钥发送到服务器端。服务器端收到用户的公钥后，也生成自己的公私密钥对，然后用用户公钥把自己的公钥加密后发送到客户端。客户端接收到服务器的数据后，用自己的私钥解密得到服务器的公钥，并把服务器的公钥存储到RMS记录管理系统中。这样客户端和服务器端就可以使用非对称加密算法，用这个算法既可以完成加密也可以完成身份认证。

这一步也是属于初始化的一个过程，都是在管理员指导下完成的。到此，客户端和服务器端的身份认证方法都有了，用户就可以安全地进行移动商务交易了。

说明：MIDP 本身没有提供加密算法 API，但是，目前有些组织在 J2ME 数据

加密方面做出了很大的贡献，他们提供免费的加密 API 包。正是基于这些人的成果，使得我自己不用写加密算法代码来实现加密功能，可以直接简单调用加密 API 来完成我要实现的功能。

在本文中，我用的加密 API 包是由 Bouncy Castle 提供的。

下面部分代码示意：用Bouncy Castle提供的加密API在客户端生成RSA算法公私密钥对：

```
public void generateRSAKeyPair () throws Exception {
SecureRandom sr = new SecureRandom();
BigInteger pubExp = new BigInteger("10001", 16);
RSAKeyGenerationParameters RSAKeyGenPara =
     new RSAKeyGenerationParameters(pubExp, sr, 1024, 80);
RSAKeyPairGenerator RSAKeyPairGen = new RSAKeyPairGenerator();
RSAKeyPairGen.init(RSAKeyGenPara);
AsymmetricCipherKeyPair keyPair =
      RSAKeyPairGen.generateKeyPair();
RSAprivKey = (RSAPrivateCrtKeyParameters)keyPair.getPrivate();
   //私钥
RSApubKey = (RSAKeyParameters) keyPair.getPublic();
      //公钥
}
```

4.3.3 移动商务的安全交易过程

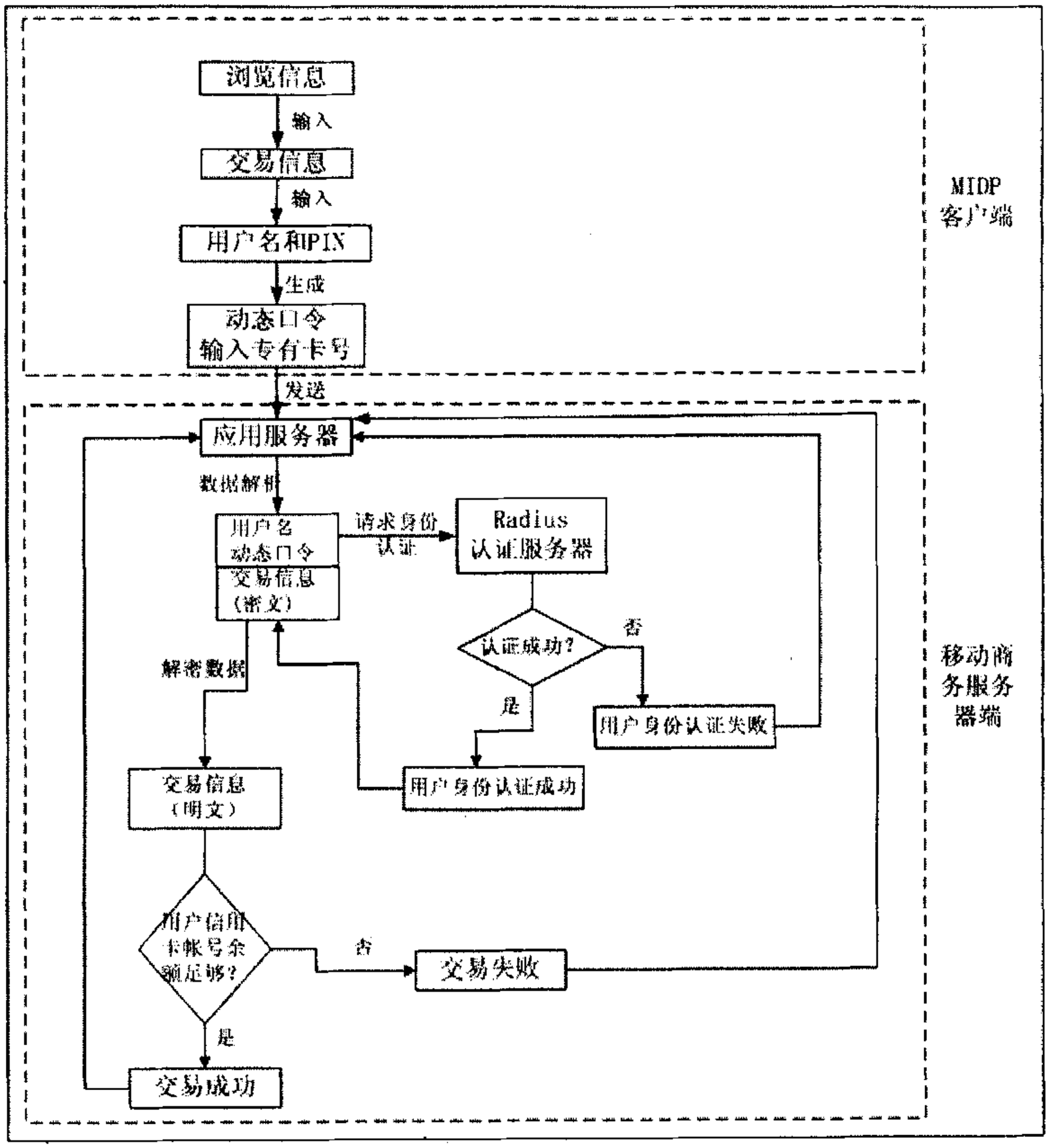


图4.2　移动商务的安全交易过程

1．用户浏览移动商务程序提供的服务信息，可以离线浏览和在线浏览。本文中，仍然举例为股票交易。股票交易服务商为用户提供了丰富的即时的股票资讯。J2ME提供了丰富的用户界面库，使得开发者可以开发出丰富的应用。UI的良好设计也是J2ME的一个非常重要的特点和优势。

2．当用户浏览信息后，对某只股票的走势看好，于是做出买入此股票的决策。在选定买入股票和输入买入股数后，同时把这些购买信息存储到RMS中。准备向服务器出示自己的身份。

3．用户在移动设备上输入用户名和PIN码生成动态口令password。动态口令

的生成采用计算输入混和消息的MD5摘要值。

MD5算法介绍

MD5算法的输入可以是任意长度的消息，对输入按512位的分组为单位进行处理，算法的输出是128位的消息摘要。

MIDlet程序的计算过程如下，下面是用MD5算法生成一次性口令的部分代码：

```
now=new Date();//移动设备当前时间
epoch=""+now.getTime();//换算成时间值
otp=epoch+secret+PIN;// 混和消息（当前时间值 + 初试密钥 + PIN码）
hash=new MD5(otp);//计算混和消息的摘要值
password=hash.asHex().substring(0,6);//选择摘要值的前6个字节作为一次性密码
g.drawString("password: " + password,w/7,h/2,0);
//显示动态口令password
```

计算出动态口令后，输入注册时与信用卡账号相对应的专有号码，取出存储在RMS中的购买信息，然后对专有号码和购买信息等交易数据加密传送，对用户名和动态口令明文传送即可。注意PIN码不在网络上传输，避免在网络中传输被截获的危险。

交易数据和用户名、动态口令组合在一起加密后用HTTP连接发送到服务器端。如HttpConnection c = (HttpConnection) Connector.open(url)，url中包含要发送的所有数据信息。

4. 服务器收到数据后，Servlet解析数据，把用户名和动态口令发送到认证服务器。同时解密客户端一起发过来的加密了的交易数据。认证服务器根据用户名，查找相应的PIN和初始密钥，并调整时间同步，根据同样的MD5摘要算法生成动态口令，计算得到的口令和服务器接收到的口令相比，如果不相同则返回身份认证失败的消息，后续操作全部取消。

用服务器的私钥来解密交易数据，获得购买信息，专用号码等信息。鉴别用户身份有效后，这时即可进行后续操作，比如根据用户信用卡账号上的余额来判断交易是否成功，如果余额足够就向客户端返回交易成功的消息，否则返回交易失败的消息。如果服务器端有一些必要的信息还要传送到客户端，可以用存储着的用户公钥加密然后再发送到客户端，客户端收到数据后，用存储在RMS中的用户私钥解密数据。

5. 交易完成。

如上所述，经过身份鉴别后，用户进行移动交易时，发送的数据通过加密的方式发送到服务器端，提高了安全性。

4.3.4 安全模型中的几个关键问题

1．时间同步

如果认证服务器和客户端移动设备的时钟都是基于标准时间(如北京时间)，理论上两者之间总是时间同步的。但是，某些移动设备的时钟由于一些原因没有设置成标准时间，从而造成了服务器和移动设备时钟的不一致。

在进行移动商务时，采用了双因素（PIN 和动态口令）身份认证，其中很重要的一点，就是动态口令是基于时间的。因此在交易前，用户调整移动设备的时钟以和服务器保持同步是非常重要的。

2．使用 Radius 认证服务器

在上面介绍到的用来鉴别用户身份的 Radius 认证服务器，它是基于 Linux 操作系统，在标准的 Radius 服务器开放源码基础上，添加动态口令模块来完成动态口令认证的，它也提供加密算法把初始密钥、PIN 码等用户的重要信息加密后再保存到数据库中。使用 Radius 的的原因还在于，由于 Radius 广泛用于企业服务器级的身份鉴别，所以对服务器端基本不用做什么改动就可以把 Radius 用于到移动商务用户身份鉴别上，为企业提高了资源的利用效率和节省了开支。

3．HTTP 会话连接

在移动商务交易过程中，用户通过和服务器的连接，在浏览信息和请求交易间不断的切换。如果每次请求都需要进行身份认证，这会给用户带来不便。如果移动商务程序支持会话，很容易解决这个问题。但 MIDP 仅支持标准的 HTTP 协议，它是无状态的。

解决这个问题，有两种方法，一是使用 cookies，一是重写 URLs。但在 MIDP 中，HttpConnection 对象既不支持 cookie 不支持非传统 URL 重写。但考虑到会话追踪在移动商务应用程序中的重要性，就必须实现会话 HTTP 连接。

cookies 证明使用容易并为会话追踪提供了一种标准方法。然而，由于早先一些 cookie 的滥用，越来越认识到 cookies 对于隐密的不安全。对于手持移动设备来说隐私证明是特别关心的事，因为许多人使用手机等管理敏感的个人及金融信息。因此，用 URL 重写是一种更好的方法，多数服务器都支持，包括无线 Java 移动商务模型服务器端的 J2EE servlet 容器。URL 重写技术将会话标识信

息嵌入到自定义格式的 URLs 中。

4．客户端程序的优化

尽管 J2ME 提供了计算能力和支持加密 API，但是由于受本身设备硬件所限制，加上使用了第三方的加密包，使得加密后的数据变得比较庞大。通常情况下，MIDlet 程序套间的大小在 50KB 左右，如果使用第三方的加密包后，程序将增大很多。因此必须使用工具来优化客户端程序，删除未使用的类和接口，来缩减代码大小。

4.4 移动商务程序设计

设计移动商务程序时，要考虑两个关键问题，先看看移动商务的连接过程：

进行移动商务时，客户端与后台服务器的连接可以通过两种方法来实现，下面分别进行介绍。

第一种方法是手机上的程序在初始化时向服务器发出连接请求，建立连接后保持连接，供程序随时使用。采用这种方法在实际使用了一段时间后暴露出一些缺点，首先是连接不稳定，因为通过无线网络进行通讯，随时可能因手机信号的中断而造成“虚连接”，即服务器端或手机端都认为该连接有效，继续用该连接来发送数据，但接收方已经无法接收该数据了，并且不能向发送方返回确认信息。最终的结果是因网络超时，该连接会自动中断。在程序中的具体表现为当虚连接产生时，客户端向服务器发出请求后，经过一段时间的等待程序就会异常终止，即使在程序中重复进行数据发送也会因网络超时而失败。此方法的第二个缺点是占用过多的服务器资源，因为服务器对每个连接都会保持一个处理线程，即使客户端长时间没有数据请求，该线程也存在，造成了系统资源的浪费。

所以，很自然地提出了第二种方法：当需要进行数据传输时才建立连接，连接建立后就进行数据的传输，数据传输完成后马上终止该连接。如果连接或传输过程中产生失败，可以提示用户出错原因，由用户选择重试还是退出。采用这种方法的程序在实际使用过程中唯一的缺点就是每次建立连接都会浪费几秒钟的时间，但保证了连接质量的稳定、可靠，并且服务器的性能提升很明显，所以在连接时间上的牺牲还是值得的。

对移动商务来说，为用户提供对移动设备方便的控制方式是非常重要的。因此，在设计移动商务程序时，有两个问题值得注意：一是如何处理连接超时，也就是在网络连接超时时，用户可以随时取消连接；另一个是多线程的处理方式，也就是用户发出连接后，手机还可以响应其他事件。

息嵌入到自定义格式的 URLs 中。

4. 客户端程序的优化

尽管 J2ME 提供了计算能力和支持加密 API，但是由于受本身设备硬件所限制，加上使用了第三方的加密包，使得加密后的数据变得比较庞大。通常情况下，MIDlet 程序套间的大小在 50KB 左右，如果使用第三方的加密包后，程序将增大很多。因此必须使用工具来优化客户端程序，删除未使用的类和接口，来缩减代码大小。

4.4 移动商务程序设计

设计移动商务程序时，要考虑两个关键问题，先看看移动商务的连接过程：

进行移动商务时，客户端与后台服务器的连接可以通过两种方法来实现，下面分别进行介绍。

第一种方法是手机上的程序在初始化时向服务器发出连接请求，建立连接后保持连接，供程序随时使用。采用这种方法在实际使用了一段时间后暴露出一些缺点，首先是连接不稳定，因为通过无线网络进行通讯，随时可能因手机信号的中断而造成“虚连接”，即服务器端或手机端都认为该连接有效，继续用该连接来发送数据，但接收方已经无法接收该数据了，并且不能向发送方返回确认信息。最终的结果是因网络超时，该连接会自动中断。在程序中的具体表现为当虚连接产生时，客户端向服务器发出请求后，经过一段时间的等待程序就会异常终止，即使在程序中重复进行数据发送也会因网络超时而失败。此方法的第二个缺点是占用过多的服务器资源，因为服务器对每个连接都会保持一个处理线程，即使客户端长时间没有数据请求，该线程也存在，造成了系统资源的浪费。

所以，很自然地提出了第二种方法：当需要进行数据传输时才建立连接，连接建立后就进行数据的传输，数据传输完成后马上终止该连接。如果连接或传输过程中产生失败，可以提示用户出错原因，由用户选择重试还是退出。采用这种方法的程序在实际使用过程中唯一的缺点就是每次建立连接都会浪费几秒钟的时间，但保证了连接质量的稳定、可靠，并且服务器的性能提升很明显，所以在连接时间上的牺牲还是值得的。

对移动商务来说，为用户提供对移动设备方便的控制方式是非常重要的。因此，在设计移动商务程序时，有两个问题值得注意：一是如何处理连接超时，也就是在网络连接超时时，用户可以随时取消连接；另一个是多线程的处理方式，也就是用户发出连接后，手机还可以响应其他事件。

1．处理连接超时

除了考虑连接方法，还必须关心网络超时问题，在手机通讯中这个现象尤其严重，主要是因为手机信号不稳定造成的，所以一个连接请求在超出一定的时间后我们必须终止它，避免用户无谓的等待。在 MIDP 中，可以通过 Timer 和 TimerTask 类来实现连接时间的控制。Timer 实例用于计时，TimerTask 实例在超时产生后终止连接并给出提示信息。

2．多线程处理

使用 J2ME 中的单线程技术来建立连接已经满足了无线网络连接的最基本要求，但放在实际的手机程序中就不行了，因为一般的单进程程序都会在连接处等待连接结果，这一过程受网络质量影响或长或短，都会造成程序的暂停。在实际中表现为手机端开始请求建立连接时，手机屏幕变成白屏，无任何显示，也不响应任何事件。为了避免出现这种情况，必须用 J2ME 中的多线程技术来建立连接。在 J2ME 中有两种实现线程的方法：

第一种，java.lang.Runnable 接口定义了一个 run()方法，线程运行时会自动调用这个方法。当这个方法结束时，线程也就结束了。上面用到的 TimerTask 类在 J2ME API 中的定义为 public abstract class TimerTask implements Runnable，它就提供了一个 Runnable 接口，实现了线程的功能。

第二种，java.lang.Thread 类表示一个线程类，它定义了设置和查询线程属性的方法和启动线程运行的 run()方法。这两种方法在实际效果上没有区别，一般情况下都使用 Java.lang.Thread 类来实现线程。Java.lang.Runnable 接口主要用于让某一个类能够在继承父类的同时获得线程的特性。这是由于 Java 语言的单继承特性造成的。

注意：任何时候都不要直接调用 run()方法，调用 start()方法时 run()方法会自动被调用。必须说明一下的是线程的终止。在 Java 中，终止一个线程也有两种方法：第一种是显示的调用 stop()方法来终止一个线程，但因为 stop()方法不能很干净彻底地释放该线程所占用的资源，譬如释放某个变量的逻辑锁，所以这种方法在最新的 Jav API 中已经被抛弃了。第二种方法是目前推荐的做法，即通过结束 run()方法的运行来终止线程。具体实现可以通过定义一个 boolean 型变量，譬如 stopThread 来标识是否需要结束该线程，然后在 run()方法中有规律地检查该值，一旦该值发生变化，就结束 run()方法，线程也就自然终止了。

端到端模型的安全性如何

短到端模型采用动态口令认证和数据加密的方式，由于动态口令依赖于 PIN 码和初始密钥，同时数据加密后再传输，因此此模型是比较安全的。

通常情况下，128-bit 的初始密钥应该是比较安全的。因此特别要注意 PIN 码的长度，在文中用的 PIN 码的长度为 6，对避免攻击者采用暴力攻击的方式应该是足够了。如果想提高安全性，可以增加 PIN 码的长度，但也不是越长越好，因为 PIN 码长度增加给用户记忆带来困难。

从上面的讨论可以看出，移动商务的成功将依赖于是否制订出端到端方案，该方案可最大限度地将风险降低到用户能接受的水平。端到端方案保护移动商务交易中的每个薄弱环节，确保数据从用户客户端到最后目的地之间完全的安全性，包括传输过程中的每个阶段，而不只注意无线传输。对每一个薄弱环节采取适当的安全性和私密性措施，就可以确保整个交易过程中的安全性。

4.5 MIDP 客户端的其他应用

本文中所谈到的用 J2ME 技术实现的移动商务，通常情况下都是客户端和服务器端直接建立连接，用移动设备进行在线的交易活动。除此之外，也可以把移动设备当成一个令牌设备，比如在台式机上通过网页访问服务器时，服务器会返回一个挑战码，客户端需要输入相应的响应码作为登陆密码。这时就可以把支持 J2ME 的移动设备当成令牌设备计算响应码。由于不用其他的令牌设备就可以计算出响应码，这方便了用户的操作。

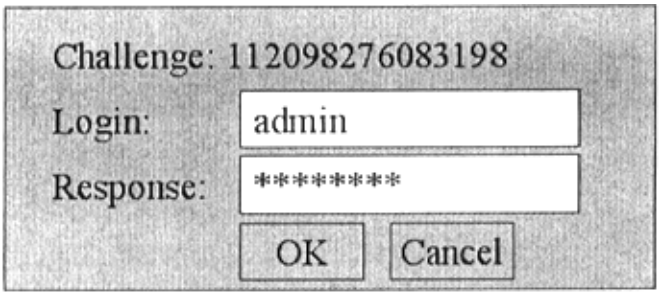


图 4.3　挑战－质询方式的动态口令认证

如上图所示：服务器返回挑战码为 112098276083198 ，用户登陆时要求输入用户名和响应码，响应码由移动设备计算得到。由于服务器每次返回的挑战码不一样，所以这实际也是属于动态口令的认证方式。

端到端模型的安全性如何

短到端模型采用动态口令认证和数据加密的方式，由于动态口令依赖于 PIN 码和初始密钥，同时数据加密后再传输，因此此模型是比较安全的。

通常情况下，128-bit 的初始密钥应该是比较安全的。因此特别要注意 PIN 码的长度，在文中用的 PIN 码的长度为 6，对避免攻击者采用暴力攻击的方式应该是足够了。如果想提高安全性，可以增加 PIN 码的长度，但也不是越长越好，因为 PIN 码长度增加给用户记忆带来困难。

从上面的讨论可以看出，移动商务的成功将依赖于是否制订出端到端方案，该方案可最大限度地将风险降低到用户能接受的水平。端到端方案保护移动商务交易中的每个薄弱环节，确保数据从用户客户端到最后目的地之间完全的安全性，包括传输过程中的每个阶段，而不只注意无线传输。对每一个薄弱环节采取适当的安全性和私密性措施，就可以确保整个交易过程中的安全性。

4.5 MIDP 客户端的其他应用

本文中所谈到的用 J2ME 技术实现的移动商务，通常情况下都是客户端和服务器端直接建立连接，用移动设备进行在线的交易活动。除此之外，也可以把移动设备当成一个令牌设备，比如在台式机上通过网页访问服务器时，服务器会返回一个挑战码，客户端需要输入相应的响应码作为登陆密码。这时就可以把支持 J2ME 的移动设备当成令牌设备计算响应码。由于不用其他的令牌设备就可以计算出响应码，这方便了用户的操作。

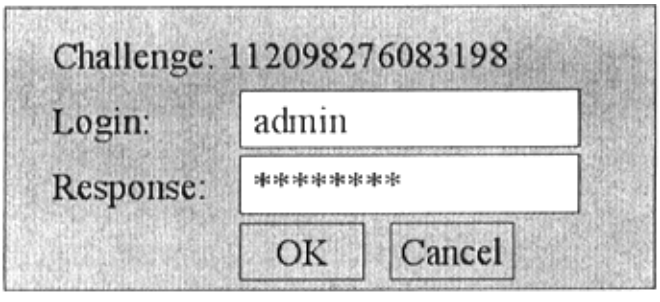


图 4.3　挑战－质询方式的动态口令认证

如上图所示：服务器返回挑战码为 112098276083198 ，用户登陆时要求输入用户名和响应码，响应码由移动设备计算得到。由于服务器每次返回的挑战码不一样，所以这实际也是属于动态口令的认证方式。

4.6 支持移动商务的其他安全方案

本章中所谈到的是基于端到端的安全方案。由于 J2ME 技术的快速发展和业界的广泛支持，很多研究人员都在无线 J2ME 的安全上提出了很多好的方案。这里介绍一个比较新颖的方式：XML 方式。

J2ME 应用程序可以在 HTTP 协议上使用 XML 数据格式与后端服务器和其它 J2ME 应用程序通信。XML 是一种非常健壮的、易于理解的消息格式。这也是为新一代开放、可互操作的 Web 服务所选的通信数据格式。因此，使用 J2ME 的无线设备必须有处理 XML 的能力，以便访问 Web 服务的世界。此时，使用 XML 的优点远胜于其带宽开销。

通过安全 XML 保护内容

XML 是我们为 J2ME 无线应用程序和后端服务之间的数据通信所选的格式。为了提供端到端安全性，我们需要确保 XML 文档的安全。因此，我们需要特殊的 XML 标准以将安全性元信息与单个文档相关联。

已提出了几个 XML 安全性协议以在 XML 应用程序中支持通信数据安全性。其中有下列协议：

安全性断言标记语言（Security Assertion Markup Language (SAML)）是以 XML 消息传输认证和授权信息的协议。它可以用来提供单点登录 Web 服务。

XML 数字签名定义了如何对 XML 文档的部分或全部进行数字签名以保证数据完整性。可以用 XML 密钥管理规范（XML Key Management Specification (XKMS)）格式封装与 XML 数字签名一起分发的公钥。

XML 加密允许应用程序使用对预先约定的对称密钥的引用来加密部分或全部 XML 文档。

第五章　安全模型的模拟实现

引言

本章的目的是实现本人在第四章中所提出的“基于端到端安全”的移动商务的应用。

目前支持 J2ME 的手机很多，比如 Nokia，Motorola 等很多设备商都有某些系列的手机支持 J2ME 功能，一般来说支持 J2ME 的手机也是属于高端手机，因此价格比较贵。由于 J2ME 技术的美好发展前景和广泛应用，同时全球有无数的编程爱好者，为了吸引这些爱好者来开发 J2ME 程序，手机厂商们也竭力为开发者提供方便，独自提供了符合 J2ME 规范的 SDK，同时提供模拟器方便开发者运行和测试程序。要指出的是，虽然厂商提供了支持 J2ME 规范的开发包，但都或多或少的做了些扩展，而这些扩展是基于每个厂商的硬件的。由于每个厂商的硬件不同，所以也许基于独自的开发包开发出的程序就不能移植到其他厂商的手机上。这违背了 J2ME 的宗旨“write once，run anywhere”。因此，我在做模拟程序时，选用 Sun 提供的开发工具，因为 Sun 提出了 J2ME 规范，其他厂商也是遵从这个规范的，因此用 Sun 工具开发出来的 J2ME 程序可以方便的移植到不同厂商的手机上。

5.1 模拟环境

1．需要两台 PC 机器，一台安装 Windows XP 操作系统，这台 PC 机称为 P1；一台安装 Redhat 7.2 操作系统，这台 PC 机称为 P2。

2．在 P1 上安装 J2SDK1.4，J2EE1.3.1，J2ME Wireless Toolkit 2.0。

3．设置环境变量。

JAVA_HOME：JDK 的安装目录

J2EE_HOME：J2EE 的安装目录（支持 EJB）

J2MEWTK_HOME：J2ME Wireless Toolkit 的安装目录

4．下载 Bouncy Castle 加密包（lcrypto-j2me-122.zip）。

5．在 P2 上安装 mysql 数据库（mysql-3.23.57.tar.gz）。

6．在 P2 上安装 Cistron Radius server(radiusd-cistron-1.6.7.tar.gz)，此 Radius 服务器是免费开源的。

第五章　安全模型的模拟实现

引言

本章的目的是实现本人在第四章中所提出的“基于端到端安全”的移动商务的应用。

目前支持 J2ME 的手机很多，比如 Nokia，Motorola 等很多设备商都有某些系列的手机支持 J2ME 功能，一般来说支持 J2ME 的手机也是属于高端手机，因此价格比较贵。由于 J2ME 技术的美好发展前景和广泛应用，同时全球有无数的编程爱好者，为了吸引这些爱好者来开发 J2ME 程序，手机厂商们也竭力为开发者提供方便，独自提供了符合 J2ME 规范的 SDK，同时提供模拟器方便开发者运行和测试程序。要指出的是，虽然厂商提供了支持 J2ME 规范的开发包，但都或多或少的做了些扩展，而这些扩展是基于每个厂商的硬件的。由于每个厂商的硬件不同，所以也许基于独自的开发包开发出的程序就不能移植到其他厂商的手机上。这违背了 J2ME 的宗旨“write once，run anywhere”。因此，我在做模拟程序时，选用 Sun 提供的开发工具，因为 Sun 提出了 J2ME 规范，其他厂商也是遵从这个规范的，因此用 Sun 工具开发出来的 J2ME 程序可以方便的移植到不同厂商的手机上。

5.1 模拟环境

1. 需要两台 PC 机器，一台安装 Windows XP 操作系统，这台 PC 机称为 P1；一台安装 Redhat 7.2 操作系统，这台 PC 机称为 P2。

2. 在 P1 上安装 J2SDK1.4，J2EE1.3.1，J2ME Wireless Toolkit 2.0。

3. 设置环境变量。

   JAVA_HOME：JDK 的安装目录

   J2EE_HOME：J2EE 的安装目录（支持 EJB）

   J2MEWTK_HOME：J2ME Wireless Toolkit 的安装目录

4. 下载 Bouncy Castle 加密包（lcrypto-j2me-122.zip）。

5. 在 P2 上安装 mysql 数据库（mysql-3.23.57.tar.gz）。

6. 在 P2 上安装 Cistron Radius server(radiusd-cistron-1.6.7.tar.gz)，此 Radius 服务器是免费开源的。

J2ME Wireless Toolkit（J2MEWTK）开发工具

图 5.1　J2MEWTK 工具

这是 Sun 提供的便捷开发 J2ME 应用程序的工具，用于 Windows 环境。源程序仍需要使用常规的文本编辑器，把编辑好的源文件及资源文件按一定要求放在规定目录下，J2MEWTK 所提供的是菜单或按钮方式的命令。J2MEWTK 中有相应的编译（和预验证一个步骤）、打包、模拟运行的菜单（或按钮），以及其他辅助工具。此工具提供了手机模拟器来运行和测试开发的程序。

模拟环境搭建好后，就涉及到具体的程序功能设计和程序的编写了。

程序功能设计

本实验所完成的例子是：用户用移动设备在线进行安全的股票交易活动。客户端的主要功能是显示从服务器上查询的股票信息，用户可以购买选定的股票并进行交易。客户端还必须支持和服务器连接的初始化，具有动态口令计算和数据加密的功能。服务器端的功能包括解析客户端发过来的数据，鉴别客户端身份，向客户端提供即时的股票信息，保存客户端的交易数据。

客户端模拟的是移动设备，由 J2ME 支持；服务器端模拟的是股票服务提供商所提供的服务器，由 J2EE 支持。

在实验中，用本机（P1）地址作为服务器的地址。服务器的地址为：http://localhost。

代码编写

本人在三年中，通过做用 Java 语言开发的项目以及不断的学习，对 Java 语言非常熟悉，积累了丰富的编程经验。由于 J2ME 的开发本质上和 J2SE，J2EE 的开发没什么区别，因此在完成模型的功能设计后，代码编写就是相对比较容易的工作了。

本实验中的程序编写分为两部分：客户端和服务器端的程序编写。

限于篇幅的关系，这里就不贴出详细的代码。

5.2 客户端程序

客户端程序中很大一部分是界面设计，其次是动态口令计算和数据加密部分。

代码写好后，用 J2MEWTK 工具来编译程序，编译通过后，用手机模拟器运行程序，程序提供了菜单和操作界面。下面几幅图就是从模拟器上捕捉的：

图 5.2　用户注册

图 5.2 所示为用户注册界面，用户输入用户名、身份证号码和 PIN 码发送到服务器端注册。

图 5.3　用户进行股票交易

图 5.3 所示为用户浏览股票信息后，决定买入选定的股票，输入要买入股票

代码编写

本人在三年中，通过做用 Java 语言开发的项目以及不断的学习，对 Java 语言非常熟悉，积累了丰富的编程经验。由于 J2ME 的开发本质上和 J2SE，J2EE 的开发没什么区别，因此在完成模型的功能设计后，代码编写就是相对比较容易的工作了。

本实验中的程序编写分为两部分：客户端和服务器端的程序编写。

限于篇幅的关系，这里就不贴出详细的代码。

5.2 客户端程序

客户端程序中很大一部分是界面设计，其次是动态口令计算和数据加密部分。

代码写好后，用 J2MEWTK 工具来编译程序，编译通过后，用手机模拟器运行程序，程序提供了菜单和操作界面。下面几幅图就是从模拟器上捕捉的：

图 5.2　用户注册

图 5.2 所示为用户注册界面，用户输入用户名、身份证号码和 PIN 码发送到服务器端注册。

图 5.3　用户进行股票交易

图 5.3 所示为用户浏览股票信息后，决定买入选定的股票，输入要买入股票

的数量。

图 5.4 用户登陆

图 5.4 所示为用户输入购买的股票信息后登陆到服务器进行身份鉴别，用户输入 PIN 码，按“确定”后程序马上计算动态口令的值。

动态口令的计算过程：

```
now=new Date();//移动设备当前时间
epoch=""+now.getTime();//换算成时间值
otp=epoch+secret+PIN;// 混和消息（当前时间值 + 初试密钥 + PIN码）
hash=new MD5(otp);//计算混和消息的摘要值
password=hash.asHex().substring(0,6);//选择摘要值的前6个字节作为动态口令。
new TextField("动态口令:", "+ password +" , 6, TextField.UNEDITABLE)
//此动态口令在图5.5中显示
```

图 5.5 用户输入专有卡号确认交易

图 5.5 所示为用户输入专有卡号确认交易。此专有卡号和用户的信用卡账号绑定。

图 5.5 所示，在向服务器发送交易数据之前，先要对交易数据进行加密。对用户名和动态口令就用明文的方式传送。因为动态口令每次都不一样，因此不用加密，另外因为数据发送到服务器端，先要进行客户端的身份认证，因此对用户名和动态口令用明文可以加快认证服务器的鉴别速度，这样服务器只须对交易数据进行解密就可以了。

交易数据包括股票代码，股票买入数量，专有卡号，因此加密数据：

```
String stockCode; //股票代码
Int nums; //股票买入的股数
String cardNo; //专有卡号
byte [] toEncrypt =
        ("stockCode="+stockCode+"nums="+nums+"cardNo="+ cardNo).getBytes();
         //把交易数据组织在一起
AsymmetricBlockCipher eng = new RSAEngine();
eng = new PKCS1Encoding(eng);
eng.init(true, RSApubKey);//用服务器的公钥加密
String encrypted =
          new String(eng.processBlock(toEncrypt, 0, toEncrypt.length));
          //encrypted 即为加密后的交易数据
```

图 5.6 向服务器发送交易数据

图 5.6 所示为向服务器发送交易数据。和服务器连接的网络协议是 HTTP 协议。

```
String base = "https://localhost/servlet";
String url = base
          +"?user="+user+"&password="+password+"&encrypted="+encrypted;
HttpsConnection hc = (HttpsConnection)Connector.open(url);
```

在发送数据时，可能由于网络原因，连接和发送过程要花费一定时间，因此这里采用线程和进度条显示的方式，用户可以随时终止发送。

5.3 服务器端程序

服务器端程序属于后台程序，不与用户直接发生关系，因此要求程序性能比较稳定。

服务器在接收到手机客户端发过来的数据后，对数据进行解析，先解析得到用户名和动态口令（因为这两项数据是明文，不用解密），向认证服务器发出鉴别用户身份的请求。由于认证服务器是在 P2 机器上的，因此在鉴别身份的同时，服务器（P1）同时进行解密客户端发过来的交易数据。

解析数据

解析得到用户名：andy1234，动态口令为：aecf02，加密后的交易数据（一串密文）。

应用服务器向认证服务器提出鉴别用户名为 andy1234 的身份。

用户身份鉴别

Radius 认证服务器收到鉴别请求后，查看系统当前时间。考虑到客户端和服务器端的时钟同步以及网络延迟等原因，因此认证服务器允许客户端与服务器端的时间差异在正负 3 分钟内就行，如果时间差异不在这个范围内，即向服务器返回用户身份无效的消息。如果差异在这范围内，从数据库中取出与用户名对应的 PIN 码，初始密钥，用同样的算法计算 PIN 码，初始密钥以及当前时间的混和值的摘要值，得到动态口令。把这个口令和服务器端收到的动态口令相比较，相同则向服务器返回用户身份有效，服务器即可进行后面的操作；否则向服务器返回用户身份无效，服务器终止后续操作，同时向客户端返回身份认证无效的消息。

解密交易数据

```
AsymmetricBlockCipher eng = new RSAEngine();
eng = new PKCS1Encoding(eng);
eng.init(false, RSAprivKey);//用服务器的私钥解密
String decrypted =
                new String(eng.processBlock(toDecrypt, 0, toDecrypt.length));
                //toDecrypt 为加密的交易数据
//decrypted 是解密后得到的交易数据明文
```

根据交易数据，就可继续进行下面的操作。判断用户信用卡账号上的余额是否大于购买股票的金额，是即向银行请求划掉用户信用卡账号相应金额，同时向

客户端返回交易成功的消息；否则向客户端返回交易失败的消息。

至此，此次交易完成。

5.4 实验结论

在做模拟实验时，首要目标是完整的实现整个移动商务的安全交易过程。基于第四章提出的安全模型，程序包括客户端(MIDlet)程序和服务器端程序。客户端程序包括界面设计，动态口令计算和数据的加密等；服务器端程序包括数据解析，数据解密，以及认证服务器的动态口令小模块。

在完成功能的前提下，基于几个方面的考虑，我主要侧重对客户端的性能进行了分析。安全方面，对交易数据采用 RSA 加密算法，因为 RSA 算法有着密码学界所公认的很高的安全性，同时考虑到移动环境下要截取数据是比较困难的（不同用户在不同地方进行的相同内容的交易），因此没有对模型的安全性做一个详细的分析。对于客户端的身份认证，采用的是基于时间的动态口令和双因素身份认证，由于这种认证方式在有线连接的网络中得到广泛应用，其安全性也是得到业界的一致认可，因此在实验中也没对动态口令的安全性做一个量化的分析。

相比客户端，服务器端在硬件以及软件方面的性能都比较好，而客户端是资源受限的手机移动终端，客户端的性能如何，直接关系到交易的时效性，因此客户端的性能值得关注。

对客户端的性能测试，主要测试了计算动态口令的时间和计算数据加密(128-bit 的密钥长度)用的时间。

在模拟器上测试，得到的测试值如下：

计算动态口令的时间：400ms

计算数据加密的时间：1100ms

虽然客户端支持复杂的计算，但从测试结果可以看出，客户端计算动态口令（摘要）时较快，但是在计算数据加密时比较慢。说明性能瓶颈在于公钥算法的速度较慢。如果客户端是以这样的计算速度，那么这么短的时间基本不会给用户带来不便。另外随着手机硬件性能的提高以及 KVM 的优化，支持 J2ME 的手机的计算性能将会大有提高。

第六章　总结和展望

本文主要讨论了无线 Java 技术在移动商务上的安全应用。先介绍了 MIDP 所支持的 HTTPS 在移动商务上的应用，分析了此负载比较重的协议在移动商务上应用的缺陷，提出了采用基于动态口令认证的端到端安全的移动商务模型。

第一章介绍了移动商务的发展情况，介绍了 Java 及其 J2ME 技术。

第二章主要介绍了移动商务模型，说明了 J2ME 在实现移动商务上的优势。

第三章先介绍了 J2ME 的安全体系结构，重点说明了用 HTTPS 协议来实现安全的移动商务的过程。

第四章是本文的重点部分，在分析了第三章所采用的安全方式后，提出了端到端的安全移动商务模型，即采用基于动态口令的身份认证和数据加密的方式来保证交易数据的安全。

第五章是实验部分，用手机模拟程序来实现我在第四章提出的安全模型，以及一部分性能测试。

上面的总结表明，在本文中提出的端到端安全的移动商务模型，在实验环境中能实现，但经过实验分析发现一些问题，主要集中在手机客户端：

1. 性能问题。虽然客户端支持复杂的计算，在测试中发现，发现计算动态口令（摘要）时非常快，但是在计算数据加密时比较慢。说明性能瓶颈在于公钥算法的速度很慢。计算数据加密的时间比较长会给用户带来不便。做实验时，模拟器是运行在 PC 机上的，如果把程序移植到真正的手机终端上，这个计算时间更长。

2. MIDlet 程序套件由于使用第三方的加密包，虽然经过优化处理，但程序还是比较大，超过了目前手机硬件所能支持的能力。

3. 在本文中，我只简单的考虑了移动商务模型的两端（移动终端和移动商务应用服务提供商），没有考虑和移动运营商、银行等之间的相互关系。

性能问题有待于手机硬件的提高和对 KVM 的优化，KVM 利用可用的特殊硬件和底层的 OS 功能来促进与安全性相关的数学运算。很多机构都在为此做相关的研究工作。

如何在真实的移动商务环境中，实施提出的安全模型，以及如何进一步优化这个模型，这些工作尚有待于我在今后的工作中做更进一步的探讨和研究。

参考文献

[1] 廖旭，凌力，“无线 Java 在移动设备上的安全应用”，计算机工程，Vol.29，No.19

[2] http://java.sun.com/j2me

[3] Roger Riggs, Antero Taivalsaari, Mark VandenBrink, Programming Wireless Devices with the Java 2 Platform, Micro Edition, 4-10, Sun Microsystems, May 2001

[4] http://wireless.java.sun.com/midp/articles/security

[5] Bouncy Castle, the Specification, http://www.bouncycastle.org

[6] Qusay Mahmoud, Secure Java MIDP Programming Using HTTPS with MIDP,http://www.wireless.java.sun, June 2002

[7] Eric Rescorla,崔凯译,SSL 与 TLS Designing and Building Secure Systems，中国电力出版社，2002 年 10 月第一版

[8] William Stallings, Cryptography and Network Security: Principles and Practices, third edition, ISBN 7-5053-9395-2

[9] Designing Wireless Clients for Enterprise Applications with Java Technology, Sun Microsystems, Inc. June 26, 2003(draft)

[10] Build your stock with J2ME, ibm.com/developerWorks

[11] http://www.forum.nokia.com/main/0,6566,034-2,00.html

[12] http://idenphones.motorola.com/iden/developer/developer_known_bugs.jsp

[13] Anders Cervera, Analysis of J2ME for developing Mobile Payment Systems, Master’s Thesis in Information Technology, Electronic Commerce. IT University of Copenhagen, 01.08.2002

[14] Michael Juntao Yuan, Ju Long, Track wireless sessions with J2ME/MIDP, http://www.javaworld.com/javaworld/jw-04-2002/jw-0426-wireless.html

[15] Jonathan Knudsen, MIDP Application Security 3: Authentication in MIDP, http://developers.sun.com/techtopics/mobility/midp/articles/security 3, December, 2002

[16] Jonathan Knudsen, MIDP Application Security 4: Encryption in MIDP, http://developers.sun.com/techtopics/mobility/midp/articles/security4, June, 2003

[17] Eric D.Larson, Developing an End to End Wireless Application Using Java Smart Ticket Demo, February, 2002,

http://developers.sun.com/techtopics/mobility/midp/articles/smartticket

[18] Qusay H.Mahmoud, Learning Wireless Java, O'Reilly Publishing House, ISBN 7-5083-1084-5

[19] Mobile Information Device Profile (2002) What's New in MIDP 2.0 by Jonathan Knudsen November 2002,
http:// wireless.java.sun.com/midp/articles/midp20/

[20] Sun Microsystems, User's Guide, Wireless Toolkit, Version 2.0, March 2003

[21] 谢晓峰，陈璟华，Java 技术在移动增值服务上的应用，中国数据通信，2002.11

[22] 陈粤，李志蜀，罗奕，基于 J2ME 平台的手机通讯程序分析与实现，计算机应用，Vol.23，No.5，May，2003

[23] http://www.radius.cistron.nl
http://www.freeradius.org/faq/cistron.html

攻读硕士期间发表的论文及参与的科研项目

发表的论文

1. “无线 Java 在移动设备上的安全应用”，廖旭，凌力，计算机工程，Vol.29，No.19

参加的项目

1. 开发上海柴油机股份有线公司培训管理系统
2. 开发机房环境动力监控系统
3. 开发 HFC 网管系统
4. 开发天湖山企业信息管理系统

致　谢

光阴似箭日月如梭，三年的研究生生活是短暂而快乐的。在这即将结束我求学生涯的时刻，回首在实验室与老师及同学度过的这一千多个日夜，惜别之情难以言表。

三年中，实验室的老师和同学给了我许多帮助和鼓励。从他们的身上我学到了很多东西。我在研究生期间成长的每一步都与老师的指导和同学们的帮助密不可分，要感谢他们的话语实在是太多太多。

首先感谢我的导师凌力老师。他为我创造了很好的学习环境，在学习科研上给我悉心指导，热心解答我学术上遇到的问题。无论是平时的学习还是毕业论文完成期间，都不断的给我指导，使我受益匪浅；在生活上凌老师给了我无微不至的关心和照顾，使我能够正常完成学业，这一切都给我留下了感人的回忆。我在此深深的感谢凌老师。

感谢钱松荣老师。钱老师为我们实验室提供了很好的硬件环境，创造了宽松但又充满活力的科研环境。

感谢夏勇明老师。夏老师平易近人，关心我的生活。同时也感谢倪卫明博士对我学习上的指导。

感谢三年来在实验室和我一起共事的同学，他们是乔昕、王云、王晓清、徐奎，我们一起完成了项目；还要感谢我的师弟师妹，他们是王银江、姜华阳、杨涛、朱仲亮、董晓健、刘佳娜等，他们给我论文提供了很多很好的建议；也要感谢我的老乡朱敏同学，在学习上给了我很多帮助。

最后，感谢我的女朋友彭焰，在我研究生学习和完成论文期间给我很大的帮助，不断的鼓励着我，和我一起分享成功和喜悦；也要感谢爸爸、妈妈和哥哥，他们一直在默默的支持着我，他们对我的关爱是我不断前进的动力。

复旦大学通信工程系

廖　旭

2004 年 4 月

论文独创性声明

本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．论文中除了特别加以标注和致谢的地方外．不包含其他人或其它机构已经发表或撰写过的研究成果．其他同志对本研究的启发和所做的贡献均已在论文中作了明确的声明并表示了谢意．

作者签名：　　日期：2004.5.3

论文使用授权声明

本人完全了解复旦大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存论文．保密的论文在解密后遵守此规定．

作者签名：　　导师签名：　　日期2004.5.13