摘 要

随着移动通信的利润增长点逐渐从话音业务转向数字业务，各种移动增值业务层出不穷，而移动支付就成为了其中的一个亮点。所谓移动支付是指借助手机、掌上电脑、笔记本电脑等移动通信终端和设备，通过无线方式所进行的银行转账、缴费和购物等商业交易活动。如今移动支付已成为移动电子商务的重要组成部分，是实现移动电子商务的核心。

J2ME 是 Java 2 平台的三个核心版本之一，它为运行在诸如手机、PDA 等资源受限的消费产品上的应用程序提供了健壮而灵活的运行环境，是当前手持设备的首选平台之一。由于其良好的安全性设计，基于 J2ME 的应用有无可比拟的安全性。随着移动通信设备的发展，计算能力的提高，J2ME 的应用前景将更加广阔，基于 J2ME 的胖客户端也将是业界的发展方向。

椭圆曲线密码体制(ECC)与其他公钥加密系统相比能提供更好的加密强度、更快的执行速度和更小的密钥长度，因此 ECC 可用较小的开销（所需的计算量、存储量、带宽、软件和硬件实现的规模等）和时延（加密和签字速度高）实现较高的安全性，特别适用于计算能力和集成电路空间受限（如 IC 卡）、带宽受限（如无线通信）要求高速实现的情况。

文章的研究重点包括以下几个方面：

本文全面总结了移动支付技术的研究情况以及目前国内外移动支付业务的发展情况，并总结了其存在的主要问题。指出 J2ME 比较适合于移动支付的开发，基于 J2ME 的安全平台是移动支付系统的发展趋势。

1. 本文深入分析了 J2ME 系统的安全机制，从底层 JVM 虚拟机安全以及应用程序层安全两个方面对 J2ME 的安全体系结构进行了全面的研究与总结。

2. 本文研究了椭圆曲线密码体制的原理、优缺点及其应用，并在安全性与有效性等方面与其他密钥算法进行了比较，最后针对 J2ME 平台提出了一些优化设计，并进行了实现与测试分析。

3. 根据移动支付的特点和要求，以及移动支付系统的实际运行环境，同时结合 J2ME 平台的安全机制以及优化的椭圆曲线密码算法，提出了一个基于 J2ME 的移动支付安全方案，并给出了移动支付模块的设计与实现，为通信提供用户认证、密钥交换、数据加密和数字签名等服务，从而有效地提高移动通信的安全性。

4. 作为移动电子商务的一个具体应用，本文构建了一个带有 J2ME/MIDP 无线前端和 J2EE 应用服务器后端的移动订票系统，由此阐述移动电子商务系统的设

计与实现。该系统的实现，在理论上和应用上都对企业构建高效率的移动电子商务系统具有一定的指导借鉴意义。

关键字

电子商务，移动支付，J2ME，MIDP，椭圆曲线密码体制

中图分类号

TP393.08 计算机网络安全 TP311.1 程序设计

Abstract

As mobile communications profit growth gradually shift from voice services to digital services, a variety of mobile value-added services emerged endlessly. And mobile payment has become one of the brightest spots. The so-called Mobile Payment is using mobile phones, handheld computers, laptops and other mobile communication terminals and equipment, through wireless means, to do bank transfer, pay bills and shopping and other commercial transactions. Now mobile payment has become an important component of Mobile E-commerce, and the core of the realization of it.

J2ME is one of the three major versions of Java 2 Platforms. It provides robust and flexible runtime environment for applications running on resouce limited devices such as cellphones and PDAs. So J2ME is the platform of choice for today's handheld devices. Because of its good security design, the application based on J2ME has unparalleled security. With the development of mobile communication equipment, the improvement of computing power, J2ME application will become more extensive. And fat client based on J2ME will also be the direction of development.

Compared to other public-key encryption systems, Elliptic Curve Cryptosystem (ECC) provides better encryption strength, the implementation of faster speed and a smaller key length. So it can use smaller overhead (for the amount of computation, storage, bandwidth, software and hardware to achieve scale, etc.) and delay (high-speed encryption and signature) to achieve higher safety, particularly applicable to the calculation of space-constrained capacity and integrated circuits (IC card), the limited bandwidth (such as wireless communications) to achieve high-speed requirements of the situation.

Areas of Research Interests:

We give a comprehensive summary of the mobile payment technology research and the current domestic and international mobile payment business development, and sum up the existence of the main issues. Then we come to the conclution that J2ME is more suitable for mobile payment development, and J2ME-based safety platform is the trend of mobile payment system development.

1. We give an in-depth analysis of the J2ME system's security mechanisms, and present a comprehensive study and summing up for the security architecture of J2ME

from two aspects: JVM virtual machine security of the underlying layer and security of the applications layer.

2. We analyze the principle, advantages and disadvantages and application of the Elliptic Curve Cryptosystem, and compare it with other public-key algorithms in areas of safety and effectiveness. Then we realize and optimize Elliptic Curve Cryptosystem correlation algorithm on J2ME platform.

3. Considering the characteristics and requirements of mobile payment and the actual operating environment of mobile payment system, combined with the security mechanisms of J2ME platform and the optimized ECC algorithm, we present a J2ME-based mobile payment security solutions, and give the design and implementation of the mobile payment module, providing communications for user authentication, key exchange, data encryption and digital signature services, and thereby effectively improving the safety of mobile communications.

4. Finally, as a specific Mobile E-commerce application, we construct a mobile booking system with J2ME/MIDP wireless front-end and J2EE back-end application server, and thereby introduce the design and implementation of Mobile E-commerce System. The realization of the system could be use for reference either in theory or in application for the companies, which want to construct high-efficiency mobile e-commerce systems.

Keyword

E-commerce, mobile payment, J2ME, MIDP, ECC

Chinese Library Classification

TP393.08 Computer Network Security TP311.1 Program Design

本论文的工作得到了英特尔公司（Intel Corporation）基金项目“J2ME Class Libs with Small Footprint, Low Power and High Performance on XScale Processor”的支持，特此感谢

绪 论

近年来，随着互联网技术与通信技术的不断发展，网络化和全球化成为不可抗拒的世界潮流。以因特网为代表的现代信息网络正在以惊人的速度急剧增长，其应用范围也开始从单纯的通信、教育和信息查询向更具效益的商业领域扩张，电子商务便应运而生了。从单纯的网上发布信息、传递信息到在网上建立商务信息中心，从借助于传统贸易手段的不成熟的电子商务交易到能够在网上完成产、供、销全部业务流程的电子商务虚拟市场，从封闭的银行电子金融系统到开放式的网络电子银行，电子商务如火如荼，而其发展也给人们的工作与生活带来了极大的便利。随着电子商务的深入开展，人们在商务活动中的支付方式也发生了革命性的变化。现金、票据和信用卡等传统支付手段逐渐被电子支付方式所取代。

所谓电子支付[1]，指的是电子交易的当事人，包括消费者、厂商和金融机构，使用安全电子支付手段通过网络进行的货币支付或资金流转。与传统的支付方式相比，电子支付基于一个开放的系统平台 （即 Internet)， 使用最先进的通信手段，以数字化的方式进行款项支付，具有方便、快捷、高效、经济的优点。电子商务的主要特征是在线支付。为了保证在线支付的安全，需要采用数据加密和身份认证技术，以便营造一种可信赖的电子交易环境。电子支付正在迎来一个崭新的发展机遇，那就是由移动支付技术带来的在支付手段上的全新变革。移动支付本身方便、快捷的特点，使得移动支付真正可以实现 3A 的移动商务，从而人们可以完全摆脱设备、地点以及时间的限制。移动支付将成为未来几年内的社会热点，并形成巨大的市场空间。

第一章 移动支付技术

1.1 移动支付概述

1.1.1 移动支付的定义

关于移动支付，存在着许多不同的定义。

Seema Nambiar[2] 认为：移动支付是指参加商品交易或者服务的双方利用移动设备进行商品价值的交换过程。 (Mobile payment is the process of two parties exchanging financial value using a mobile device in return for goods or services.)

而 Lehman Brothers 定义 M-commence 为：使用手持设备通过连接到公用网络上或私有网络上进行交流、获得信息、交易、娱乐等。("Mobile commerce is the use of mobile hand-held devices to communicate, inform, transact and entertain using text and data via connection to public and private networks.") 而移动支付 (mobile payment) 是完成 M-commence 的过程。

根据移动支付论坛[3]（mobile payment forum）的定义，移动支付是指借助手机、掌上电脑、笔记本电脑等移动通信终端和设备，通过无线方式所进行的银行转账、缴费和购物等商业交易活动。通常移动支付所使用的移动终端主要是手机、PDA、移动 PC 等。

尽管移动支付的定义很多，但是有一点是确定的，不管支付的形式如何，支付的双方中至少有一方（或者是双方）要利用移动设备的参与来完成货币价值的交换。

1.1.2 移动支付业务分类

按照金额大小可分为小额支付和大额支付。通常来讲，交易金额小于 10 美元的称为小额支付，大于 10 美元的称为大额支付。

按照支付地点远近可分为近距离支付（又称 F2F 支付，即 Face to Face）和远距离支付。不同地点的支付，技术实现方式也不相同。

按照支付时间前后可分为预付费（存储卡）、后付费（信用卡）和交易当时支付（借记卡）。

第一章 移动支付技术

1.1 移动支付概述

1.1.1 移动支付的定义

关于移动支付，存在着许多不同的定义。

Seema Nambiar[2] 认为：移动支付是指参加商品交易或者服务的双方利用移动设备进行商品价值的交换过程。 (Mobile payment is the process of two parties exchanging financial value using a mobile device in return for goods or services.)

而 Lehman Brothers 定义 M-commence 为：使用手持设备通过连接到公用网络上或私有网络上进行交流、获得信息、交易、娱乐等。("Mobile commerce is the use of mobile hand-held devices to communicate, inform, transact and entertain using text and data via connection to public and private networks.") 而移动支付 (mobile payment) 是完成 M-commence 的过程。

根据移动支付论坛[3]（mobile payment forum）的定义，移动支付是指借助手机、掌上电脑、笔记本电脑等移动通信终端和设备，通过无线方式所进行的银行转账、缴费和购物等商业交易活动。通常移动支付所使用的移动终端主要是手机、PDA、移动 PC 等。

尽管移动支付的定义很多，但是有一点是确定的，不管支付的形式如何，支付的双方中至少有一方（或者是双方）要利用移动设备的参与来完成货币价值的交换。

1.1.2 移动支付业务分类

按照金额大小可分为小额支付和大额支付。通常来讲，交易金额小于 10 美元的称为小额支付，大于 10 美元的称为大额支付。

按照支付地点远近可分为近距离支付（又称 F2F 支付，即 Face to Face）和远距离支付。不同地点的支付，技术实现方式也不相同。

按照支付时间前后可分为预付费（存储卡）、后付费（信用卡）和交易当时支付（借记卡）。

1.1.3 移动支付方式优点

移动支付真正实现了 3A（任何时间、任何地点、任何方式）。它将无线通信技术的 3A 优势应用到金融交易业务当中，其优势是与其他以往的支付方式（传统的支付方式与电子支付方式）的比较中体现出来的。

- 传统的支付方式

传统的支付方式指现金支付。在电子商务出现之前，我们在商业流通领域内的支付主要采用这种支付方式。比如，我们在购物或者享受某种服务时，直接采用的是现金交易，有时也采用支票的形式。即使在今天，采用现金支付的方式购物仍然大量存在。但是，这种支付方式存在一些缺点：现金的交易量一般不是很大，主要是小额支付，同时用户必须随身携带大量现金，这样就存在现金丢失或者被偷的风险。同时交易双方必须是面对面的交易，这样会给异地交易带来很大的不便。当然，这种支付方式也有它的优点：那就是可信度高，采用直接现金交易人们普遍可以接收。

- 电子支付形式

电子支付是 Internet 发展的产物。所谓电子支付是指电子交易的当事人，包括消费者、厂商和金融机构使用安全电子支付手段通过网络进行的货币支付和资金流动。

与传统的支付方式相比，这种支付方式是基于一个开放的系统平台(Internet)，使用最先进的通信手段，以数字化的方式进行支付，具有方便、快捷、高效、经济的优点。

随着计算机技术的发展，电子支付方式越来越多。这些支付方式大致可以分为三类：一类是电子货币类，如电子现金、电子钱包等；一类是电子信用卡类，包括智能卡、借记卡、电子卡等；还有一类是电子支票类，如电子汇款，电子划款等。这些方式各有其特点和运作模式，适用于不同的交易场合。

电子支付克服了传统的支付方式携带现金的缺点，使用也相对比较方便。在电子商务中已经成熟地使用电子支付方式。这种电子支付形式已被广大的人们所接收，尤其是在小额支付电子商务中。

但是电子支付是基于 Internet 的，用户终端是 PC 或其他设备，并且有时候需要在 Internet 上传送敏感信息如银行卡号、密码等。硬件的客观要求在许多交易场合是不具备的，同时安全隐患也阻碍了人们对电子支付的使用。

- 移动支付形式

移动支付是在电子支付的基础上发展起来的。同前面两种支付方式相比，移动支付有着任何时间、任何地点、任何方式的独特优势。它克服了电子支付在固定网络上支付的缺陷。并且随着第三代通信技术的发展以及通信设备的改进，移动支付也已经被人们普遍接收。移动支付最大的优势在于，将银行的信用、商家的营销能力和消费者的利益最大限度地整合了起来。但是需要指出的是，尽管移动支付是下一代支付的主要方式，但它并不能彻底地代替传统的支付方式，而是支付方式的一种补充和功能的增强。在传统支付方式做不到或者做不好的方面才是移动支付方式的商机和价值所在。例如，用户想购买互联网上的商品，使用传统的支付方式十分的不便，而移动支付则在这方面显得尤为方便。

1.2 移动支付业务技术实现方式

分类	技术实现方式	优势	劣势
远距离移动支付	SMS	业务实现简单	安全性差，操作繁琐、交互性差、响应时间不确定
	IVR	稳定性较高，实时性较好，系统实现相对简单，对用户的移动终端无要求，服务提供商可以很方便的对系统进行升级并不断提供新的服务	服务的操作复杂，耗时较长，通信费用相对较高，不适用于大额支付
	WAP	面向连接的浏览器方式、交互性强	响应速度较慢、需要终端支持、终端设置较为复杂、支付成本高、不适合作频繁小额支付
	K-Java/Brew	可移植性强、网络资源消耗与服务器负载较低、界面友好、保密性高	需要 WAP 推动网关、需要终端支持需为不同终端编译不同的版本支持
	USSD	可视操作界面、实时连接、交互速度较快、安全性较高、交易成本低	需要终端支持 移动运营商的支持有地域差异
近距离移动支付	红外	成本较低、终端普及率高、不易被干扰	传输距离有限、信号具有方向性
	NFC	安全性高、速度快、存储量大	成本高、基础设施投入大、需要终端支持

表 1-1 移动支付业务技术实现方式优劣势比较[4]

不同地点的支付，技术实现方式不同。远距离移动支付的主要技术实现方式[4]有 SMS、WAP、IVR、Kjava/BREW、USSD（Unstructured Supplementary Service Data，非结构化补充服务数据业务）等；近距离移动支付的主要技术实现方式有红外、NFC（以 FeliCa IC 技术最为典型，是日本索尼公司开发的一种近距离非接触智能芯片）等。各种移动支付业务技术实现方式优劣势比较详见表 1-1。

1.3 移动支付业务发展现状

1.3.1 主要地区移动支付业务发展现状

各国移动支付业务发展水平[4]存在很大的差异。各国根据自己的实际情况选取了不同的技术实现方式。如日本采用由本土公司索尼开发的 FeliCa IC 技术，韩国主要采用红外技术，非洲一些国家主要采用 SMS 技术等。各国的移动支付业务采用的商业模式也不尽相同。日韩主要是运营商主导模式；在欧洲，第三方联合运营模式发展较好。

- 日本

日本是移动支付业务发展最为成功的国家之一。三家移动运营商 NTTDoCoMo、KDDI 和软银（原 VodafoneK.K）分别于 2004 年 7 月、2005 年 7 月和 2005 年 11 月推出了移动支付业务，采用的都是索尼公司开发的 FeliCa 技术。经过几年的发展，移动支付业务在日本已具备相当的规模。截止到 2007 年 4 月，NTTDoCoMo 移动支付业务用户 2150 万户，占其 FOMA 用户的 44%。

2005 年 11 月以信用卡公司 JCB 为主导，由多家运营商和信用卡机构发起成立的"移动支付联盟"旨在建立跨发卡机构、跨运营商的移动支付标准，推进消费者使用手机购物。

2006 年 4 月 NTTDoCoMo 将移动支付业务渗透到消费信贷，推出 DCMX 品牌的移动信用卡，用户可选择小额和大额两种信贷方式，使移动支付业务在日本的发展更进一步。

日本的移动运营商普遍采取注资金融机构的方式主导产业链发展。NTTDoCoMo 采用注资的方式拥有了两家信用卡公司的股份。2005 年 4 月注资 1000 亿日元（9.45 亿美元）获得三井住友信用卡公司 34%的股份；2006 年 3 月又注资 10 亿日元获得瑞穗关联企业 UCCard 18%的股权。日本第二大移动运营商 KDDI 也采取了同样的方法，于 2006 年 4 月宣布将和东京三菱 UFJ 共同出资

不同地点的支付，技术实现方式不同。远距离移动支付的主要技术实现方式[4]有 SMS、WAP、IVR、Kjava/BREW、USSD（Unstructured Supplementary Service Data，非结构化补充服务数据业务）等；近距离移动支付的主要技术实现方式有红外、NFC（以 FeliCa IC 技术最为典型，是日本索尼公司开发的一种近距离非接触智能芯片）等。各种移动支付业务技术实现方式优劣势比较详见表 1-1。

1.3 移动支付业务发展现状

1.3.1 主要地区移动支付业务发展现状

各国移动支付业务发展水平[4]存在很大的差异。各国根据自己的实际情况选取了不同的技术实现方式。如日本采用由本土公司索尼开发的 FeliCa IC 技术，韩国主要采用红外技术，非洲一些国家主要采用 SMS 技术等。各国的移动支付业务采用的商业模式也不尽相同。日韩主要是运营商主导模式；在欧洲，第三方联合运营模式发展较好。

- 日本

日本是移动支付业务发展最为成功的国家之一。三家移动运营商 NTTDoCoMo、KDDI 和软银（原 VodafoneK.K）分别于 2004 年 7 月、2005 年 7 月和 2005 年 11 月推出了移动支付业务，采用的都是索尼公司开发的 FeliCa 技术。经过几年的发展，移动支付业务在日本已具备相当的规模。截止到 2007 年 4 月，NTTDoCoMo 移动支付业务用户 2150 万户，占其 FOMA 用户的 44%。

2005 年 11 月以信用卡公司 JCB 为主导，由多家运营商和信用卡机构发起成立的"移动支付联盟"旨在建立跨发卡机构、跨运营商的移动支付标准，推进消费者使用手机购物。

2006 年 4 月 NTTDoCoMo 将移动支付业务渗透到消费信贷，推出 DCMX 品牌的移动信用卡，用户可选择小额和大额两种信贷方式，使移动支付业务在日本的发展更进一步。

日本的移动运营商普遍采取注资金融机构的方式主导产业链发展。NTTDoCoMo 采用注资的方式拥有了两家信用卡公司的股份。2005 年 4 月注资 1000 亿日元（9.45 亿美元）获得三井住友信用卡公司 34%的股份；2006 年 3 月又注资 10 亿日元获得瑞穗关联企业 UCCard 18%的股权。日本第二大移动运营商 KDDI 也采取了同样的方法，于 2006 年 4 月宣布将和东京三菱 UFJ 共同出资

筹建首家以移动电话作为主渠道的银行。

- 韩国

韩国的移动支付业务实现方式主要采用的是红外技术。韩国三大移动运营商SKT、KTF、LGT 分别于 2004 年 3 月、2004 年 8 月、2003 年 9 月联合金融机构开通采用红外技术的移动支付业务。

韩国移动运营商善于对移动支付业务进行整合，其移动支付业务体系非常清晰，业务规划中针对每个发展阶段制定了非常明确的发展目标。

以韩国 SKT 为例，移动支付总业务品牌为 MONETA。在 MONETA 名下又分 MONETAcard（红外线近距离非接触支付，也称离线交易）、MONETApass（乘车卡）、MONETAbank（银行信息查询、转账等）、MONETA stock trading（股票交易）、MONETA sign（身份认证）、MONETA bill（在线购物）等。

对于移动支付业务的发展 SKT 有着非常清晰的规划。SKT 将自己的移动业务发展分为了 4 个阶段，每个阶段都有不同的发展目标：第一步是初步建立移动支付系统；第二步是成功开展移动支付业务；第三步是向其它金融业务领域扩展；第四步是向移动商务领域扩展。

- 欧洲

欧洲国家的移动支付一如其它产业一样，同时进军欧洲多国，所以欧洲品牌多数采用多国运营商联合运作方式，即银行作为合作者但不参与运营。业务模式往往是通过 WAP（无线应用协议）、SMS（短消息业务）、IVR（交互语音应答）等方式接入来验证身份等，操作较为繁琐，不适于时间性要求很高的支付行为，所以多用于 WAP 业务、电子票务等。

- 美国

虽然美国近两年移动数据业务发展日新月异，但移动上网、图片铃声下载、PTT、MMS/SMS 和流媒体是其业务发展的重点，在移动支付业务领域没有太多的举措。

在美国，支付领域的新贵 PayPal 的应用是人们关注的焦点。PayPal 原来就是一个电子支付业务提供商。PayPal 提供的移动支付业务通过短信的方式不仅能购买数字产品，还能买真实的商品，并且提供货到付款服务。

- 非洲

非洲一些国家，如赞比亚、南非、肯尼亚、尼日利亚、刚果等国都推出了移

动支付业务。

在非洲的移动支付业务有以移动运营商的名义推出的，如肯尼亚的 Safaricom、赞比亚的 CelTel 等；有以银行的名义推出的，如肯尼亚联合银行；有第三方推出的，如南非的 Fundamo。

一些业务提供者除了采用短信方式提供支付业务以外，还提供一些金融服务的短信提醒服务，如短信提醒用户的工资到账、每笔信用卡交易短信提醒等。刚果移动运营商 CelTel 也曾于 2000 年 7 月将 WAP 应用于移动支付业务，但最终由于速度慢、缺乏终端支持等原因没有发展起来。

非洲移动支付业务之所以能够发展起来，甚至比一些发达国家发展得还好，主要是因为在非洲移动支付业务对传统银行业务的补充作用明显。在一些没有传统银行分支机构的地方，消费者可以通过移动支付的方式购买商品。

1.3.2 我国移动支付业务发展现状

对于中国的移动支付业务[5]而言：庞大的移动用户和银行卡用户数量提供了诱人的用户基础，信用卡使用习惯的不足留给移动支付巨大的市场空间，发展前景勿庸置疑。与此同时，移动支付也面临着信用体系、技术实现、产业链成熟度、用户使用习惯等方面的瓶颈。

中国移动支付业务历经 2002～2004 年不温不火的发展过程，2004 年下半年以来，若干主要的第三方移动支付运营商的业务有放量增长的趋势，使得移动支付业务的地域覆盖范围越来越广，产业链其它环节也越来越积极的寻求合作机会。这无疑是一个积极的信号，标志着移动支付的春天。

——用户规模和市场规模

从 2004 年下半年开始，移动支付进入地域快速扩张的阶段。2005 年，移动支付用户数达到 1560 万人，同比增长 134%，占移动通信用户总数的 4%，产业规模达到 3.4 亿元；2007 年，由于产业链的成熟、用户消费习惯的形成和基础设施的完备，移动支付业务进入了产业规模快速增长的拐点；预计到今年年底，移动支付用户数达到 1.39 亿人，占移动通信用户总数的 24%，产业规模达到 32.8 亿元。

——产业生命周期

随着福彩购买、手机话费交纳、公共事业费交纳等业务商业模式的成熟，从 2004 年下半年开始，上述业务呈现迅速发展的态势，在各地扩展的速度很快。然而，由于业务种类有限、用户群规模有限，移动支付的整体市场规模并不大。

移动支付业务的市场规模发展情况如下图 1-1 所示：

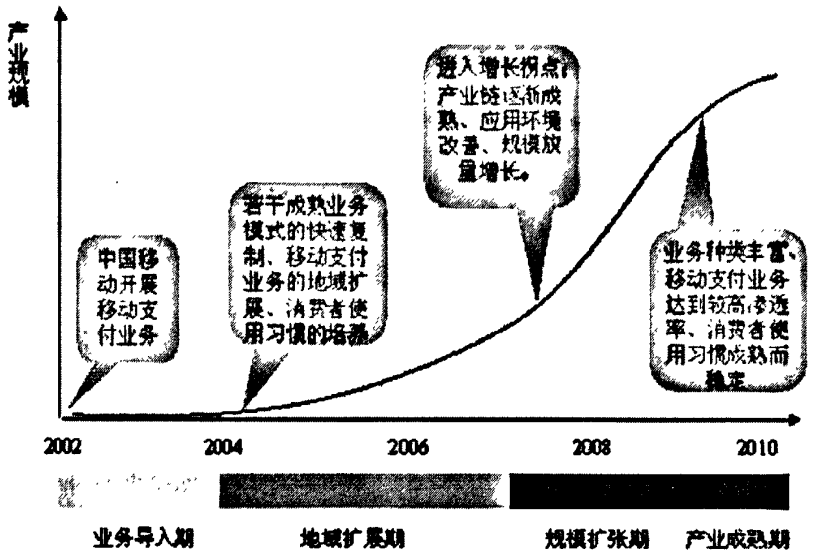

图 1-1　2002～2010 年移动支付生命周期和市场特征图[5]

各阶段特征如下：

*2002～2004 年：移动支付业务一直发展的不温不火，由于技术上的安全问题尚未解决、产业链不成熟、用户使用习惯欠缺，处于缓慢的业务导入期。

*2004 年下半年～2007 年：移动支付领域有若干业务（手机缴费充值、手机购买福彩、公用事业费支付等）的商业模式成熟、用户使用习惯得到培养，其应用的地区快速增多，已经进入地域快速扩展的快车道，但市场规模仍然有限，业务种类单一，处于地域扩展期。

*2007～2009 年：基于较好的用户使用习惯和产业链的逐渐成熟，移动支付的业务种类迅速丰富、消费者使用的比例逐渐升高、进入市场规模快速增长的拐点，处于规模扩张期。

*2009 年以后：移动支付业务渗透率（移动支付用户数/移动通信用户数）达到 24%，提供的业务种类丰富，用户使用习惯成熟稳定，进入产业成熟期。

1.4 移动支付业务存在的问题

移动支付业务虽已推出数年，但发展一直不理想。当前，移动支付业务发展中存在的主要问题[4]有以下三点：运营商和金融机构间缺乏合作；交易的安全问题没有得到很好的解决；缺乏统一的行业标准。

1.4.1 运营商和金融机构间缺乏合作

在提供移动支付业务方面，移动运营商和金融机构之间，一方的优势恰好是另一方的劣势，双方是互补的关系。移动运营商在支付流程管理上缺乏经验，而这恰恰是金融机构的优势所在；金融机构缺乏对移动支付业务传输渠道的控制，而移动运营商不仅控制着移动支付业务的传输渠道——移动通信网，还拥有庞大的移动用户群。移动运营商和金融机构的通力合作是移动支付业务成功开展的必备因素之一。

但从移动运营商和金融机构在实际开展移动支付业务的表现来看，双方的合作不容乐观。移动运营商和金融机构从自身利益考虑，都想成为移动支付产业链的主导者。移动运营商希望借助移动支付提升移动通信业务收入，金融机构希望移动支付成为其支付业务的新发展渠道。另外，移动支付业务带来新的竞争者——移动运营商，这是金融机构不愿意看到的。因此，移动运营商和金融机构的竞争关系大于合作关系。

另外，还有一点不可忽略，绝大多数国家的金融管制政策比较严格，对非金融机构经营金融类业务有着严格的控制，这就使得运营商和金融机构的合作不可能太深入。

1.4.2 安全问题

与所有的支付业务相同，安全问题是影响移动支付业务成功开展的关键因素之一。用户在考虑是否采用移动支付业务时，考虑的首要问题是交易的安全性。实际上，在开放的移动通信网络上传输这些涉及到用户支付信息的敏感数据，不可能保证完全的交易安全性。

由于存在被窃取的威胁，交易的安全认证和数据传输的机密性要求必不可少。

- 客户端的安全认证

客户端的安全认证一种是基于手机终端的设计，主要包括如表 1-2 设计的几种设计方案。

基于手机终端设计的安全认证限制了消费者更换手机的范围。为此，一些移动支付业务提供商采用基于 SMS 和 USSD 的安全认证解决方案。

1.4.1 运营商和金融机构间缺乏合作

在提供移动支付业务方面，移动运营商和金融机构之间，一方的优势恰好是另一方的劣势，双方是互补的关系。移动运营商在支付流程管理上缺乏经验，而这恰恰是金融机构的优势所在；金融机构缺乏对移动支付业务传输渠道的控制，而移动运营商不仅控制着移动支付业务的传输渠道——移动通信网，还拥有庞大的移动用户群。移动运营商和金融机构的通力合作是移动支付业务成功开展的必备因素之一。

但从移动运营商和金融机构在实际开展移动支付业务的表现来看，双方的合作不容乐观。移动运营商和金融机构从自身利益考虑，都想成为移动支付产业链的主导者。移动运营商希望借助移动支付提升移动通信业务收入，金融机构希望移动支付成为其支付业务的新发展渠道。另外，移动支付业务带来新的竞争者——移动运营商，这是金融机构不愿意看到的。因此，移动运营商和金融机构的竞争关系大于合作关系。

另外，还有一点不可忽略，绝大多数国家的金融管制政策比较严格，对非金融机构经营金融类业务有着严格的控制，这就使得运营商和金融机构的合作不可能太深入。

1.4.2 安全问题

与所有的支付业务相同，安全问题是影响移动支付业务成功开展的关键因素之一。用户在考虑是否采用移动支付业务时，考虑的首要问题是交易的安全性。实际上，在开放的移动通信网络上传输这些涉及到用户支付信息的敏感数据，不可能保证完全的交易安全性。

由于存在被窃取的威胁，交易的安全认证和数据传输的机密性要求必不可少。

- 客户端的安全认证

客户端的安全认证一种是基于手机终端的设计，主要包括如表 1-2 设计的几种设计方案。

基于手机终端设计的安全认证限制了消费者更换手机的范围。为此，一些移动支付业务提供商采用基于 SMS 和 USSD 的安全认证解决方案。

设计方案	详细说明
多重应用芯片	将 SIM 和 WIM（无线认证模块）融合在一个芯片中
双 SIM 手机	手机中有 SIM 和 WIM 两个插槽
外挂 WIM 图卡器	能够连接在手机上的 WIM 图卡器
双插槽手机	手机中有一个内置的智能卡读卡器。手机用户能够将银行发行的借记卡和信用卡插入智能卡插槽
内置于手机中的支付软件	将 WIM 的功能通过软件的方式内置在手机中

表 1-2 基于手机终端设计的移动支付安全认证[4]

- **网络传输层的机密性**

除了在客户端进行交易的安全认证以外，在网络传输层还要保证数据传输的机密性。在现有网络传输层安全草案（如 TLS 和 WTLS）中，提供一个接入到应用层的编解码系统。当前主要采用的编解码系统是 SHA-1 和 3DES。

传统支付业务的安全标准主要有两个：Visa3-D 标准和 MasterCardSPA。在移动支付领域，一些金融机构和社会团体（如 MeT、Mobey 论坛、移动支付论坛等）都在进行交易安全标准的制定工作，但仍然缺乏统一的被广泛认可的交易安全标准。

1.4.3 缺乏统一的行业标准

没有统一标准问题虽然在移动支付业务发展的初期没有明显体现出来，但终将随着移动支付业务的发展日益突出。

韩国的移动支付业务就因为两大运营商 SKT 和 KTF 使用的标准不统一而导致发展受阻。三家运营商在最初提供移动支付业务时都不愿意合作开发这个市场。SKT 的 Moneta 业务和 KTF 的 K-merce 业务需要不同的红外接收器，两种不同的接收器不能互联互通。

GSM 协会于 2007 年 2 月公布制定 NFC 移动支付全球统一标准的新计划——“Pay-BuyMobile”，该计划旨在建立一个在不同设备提供商和金融机构之间实现互操作的全球统一 NFC 移动支付标准。统一的行业标准使参与生产和研发的终端和设备厂商数量更多，有利于降低 NFC 移动支付手机和读卡器的成本，从而使该业务的受众面更大，业务更加普及。

1.5 基于 J2ME 的安全平台

我们已经拥有用来处理移动支付的 WAP、SMS 或 SAT 技术，那为什么还要考虑 J2ME 呢？

与前述技术相比， J2ME 具有以下优点[7]：

- 可移植性。移动支付客户机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。
- 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。
- 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。
- MIDlet 中的动态事件处理。这一功能大大改善可用性和用户体验。
- 网际协议（Internet Protocol（IP））。再没有什么技术比 Java 技术更适合于联网了。
- 尽可能减小 MIDlet 的大小。把 MIDlet 编写得尽可能地小，从而降低用户通过国际漫游下载 MIDlet 所需的费用。
- 记录管理存储（Record Management Store（RMS））。J2ME MIDP 规范提供一个面向记录的数据库系统作为持久存储器，这个系统的名称为记录管理存储（RMS）。该系统提供了两个类、三个接口和五个异常，即使是在重新引导或电池电量低的情况下，它们也能够确保记录完好无损。
- 事务保护。使用 J2ME 密码术，就能对整个移动支付事务进行加密。不仅如此，在 WAP 和 WTLS 的支持下，入口会话就能像在 SSL3.0 中所进行的那样被保护。
- 密码术。J2ME 本身提供了面向 J2ME 的安全性和信任服务 API（Security and Trust Services API for J2ME（JSR 177））。
- 轻松使用 MIDP MIDlet。通过一个 URL，一个单独的步骤就可以安装 MIDlet。通过使用 WAP 2.0 规范，可以将 URL 推至 MIDP 客户机。

1.6 本文的研究内容与组织结构

本文比较全面地介绍了移动支付技术的研究情况以及目前国内外移动支付业务的发展情况，并指出了其存在的主要问题。针对影响移动电子商务发展的重要问题——安全支付问题，本文提出了基于 J2ME 的安全平台，针对 J2ME 平台

实现并优化了椭圆曲线密码机制的相关算法。在此基础上本文设计并实现了移动支付模块，为通信提供用户认证、密钥交换、数据加密和数字签名等服务，从而有效地提高移动通信的安全性。最后，作为移动电子商务的一个具体的应用，本文构建了一个带有 J2ME/MIDP 无线前端和 J2EE 应用服务器后端的移动订票系统，由此阐述移动电子商务系统的设计与实现。

本文的组织结构如下：

第一章：移动支付技术。

介绍了移动支付技术的相关概念，分析了它相对于传统支付方式的优势，另外对国内外移动支付业务的发展现状以及尚存在的问题进行了介绍与分析，最后提出了基于 J2ME 的移动安全平台的解决方案。

第二章：J2ME 平台及其安全技术。

介绍了 J2ME 技术的发展、特点以及体系结构，并深入分析了 J2ME 系统的安全机制。

第三章：椭圆曲线密码体制。

从椭圆曲线相关理论开始，逐步深入地介绍了椭圆曲线密码体制的原理、其优缺点及其应用，并在安全性与有效性等方面与其他密钥算法进行了比较，最后针对 J2ME 平台提出了一些优化设计并进行了测试分析

第四章：移动支付模块的设计与实现。

设计并实现了移动支付模块，为通信提供用户认证、密钥交换、数据加密和数字签名等服务。

第五章：移动电子商务开发实例。

构建了一个带有 J2ME/MIDP 无线前端和 J2EE 应用服务器后端的移动订票系统，并给出了主要部分的设计与实现，由此阐述移动电子商务系统的开发。

第六章：结论和展望。

第二章 J2ME 平台及其安全技术

在今天的生活中，个性化的智能信息用品已经是必须品。这些用品包括移动电话、机顶盒、双向寻呼机、智能卡和掌上电脑等。在几年前开发人员很难想象能够用一种统一的开发方式来开发运行在这些设备上的程序，也很难想象在这些设备上开发兼容的程序，而现在借助 Java 的能力这些都成为可能。

Java 语言的前身是 Oak，当时它的设计目的是用于洗衣机、电视机顶盒等消费电子产品，但是一直没有得到大的发展。因特网的发展，却在无意中成全了 Java，使它很快成为因特网上广为流行的编程语言。

1998 年 1 月，Sun 实验室启动了一个名为 Spotless 项目，来研究将 Java 技术应用于那些资源有限的设备上。在项目的初期，Spotless 仅仅是一个研究项目，但是在摩托罗拉等公司的推动下，此项目不久成为一个商业产品，并由此产生了 Spotless 虚拟机（Virtual Machine）的商业化版本——KVM（K Virtual Machine 或 Kjava Virtual Machine）。

1999 年 6 月，Sun 公司正式向 Java 团体推出 J2ME，并重新划分了 Java 2 的体系结构：Java 2 Enterprise Edition（Java 2 企业版，J2EE）、Java 2 Standard Edition（Java 2 标准版，J2SE）和 Java 2 Micro Edition（Java 2 微型版，J2ME）。J2EE 主要面向企业应用的开发者，J2SE 主要面向桌面应用的开发者，而 J2ME 主要面向的是消费产品和嵌入式设备应用的开发者。

J2ME、J2SE、J2EE，这三者构成了 Java 的完整架构（如图 2-1 所示）。至此，Java 已经由单纯的编程语言发展到一种平台的高度。这三者的结合，可以将从服务器到台式机到移动设备的应用开发集中到一种技术之下。就技术人员而言，这也是一种福音，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与应用的领域。

面对移动通讯时代的到来，在移动通讯的主要工具——手机的功能越来越强大的同时，其平台与操作系统也由于众多厂商的介入而显得愈加复杂纷繁，虽然这对使用者一般没什么影响，但对于程序设计师来说要在那么多不同的平台上开发程序却成为一个令人头疼的问题。在这种情况下，J2ME 的出现就显得尤为重要，利用 Java 的“write once, run anywhere”特性，我们就可以真正做到只要写一次程序，就可以在任何平台下运行。

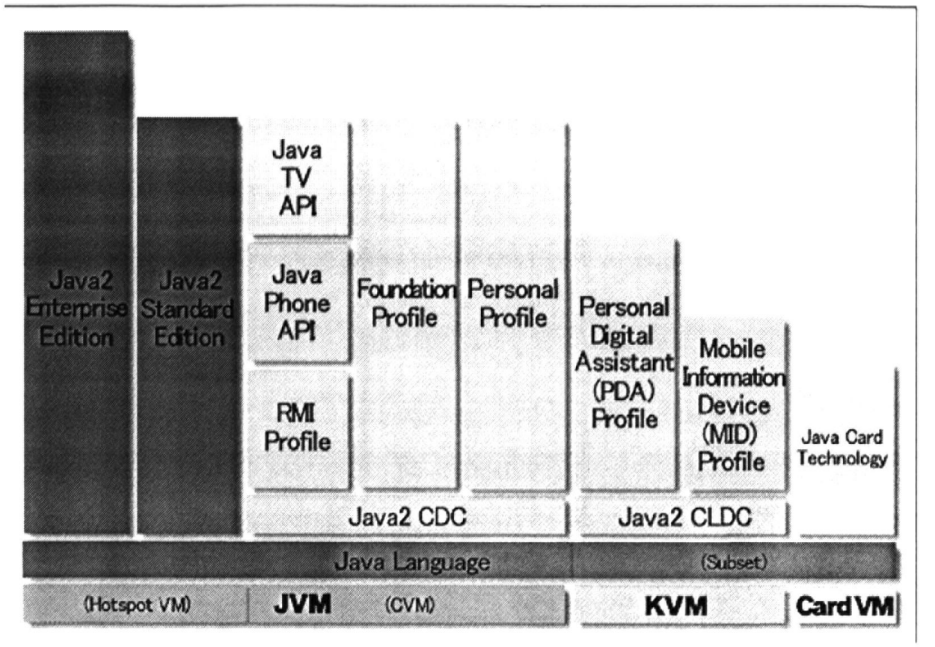

图 2-1　JAVA 版本与其所依据虚拟机所构建的架构[12]

2.1 J2ME 发展概述

J2ME 试图支持 PDA、手机、传呼机、电视机顶盒、远程遥控装置以及其他的嵌入式设备，然而，相同类型的设备可能来自不同的生产厂商，要想为这些形形色色的设备定义一种最优化的技术显然是不可能的。但是，为了保证这些设备之间的互操作性，标准化是必需的。鉴于这些设备的复杂性，Sun 并没有为 J2ME 定义一个技术规范，而是参照这些设备，首先根据处理器性能、存储能力将所有的嵌入式装置分为两类：一种是运算功能有限、电力供应也有限的嵌入式装置（比方说 PDA、手机）；另外一种是运算能力相对较佳、并且在电力供应上相对比较充足的嵌入式装置（比方说冷气机、电冰箱、电视机顶盒）。因为这两种区分，所以 Java 引入了一个叫做 Configuration 的概念，然后把上述运算功能有限、电力有限的嵌入式装置定义在 Connected Limited Device Configuration（CLDC）规格之中；而另外一种装置则规范为 Connected Device Configuration（CDC）规格。这些规格之中定义了两种类型嵌入式装置至少要符合的运算能力、供电能力、内存大小等规范，同时也定义了一组在这些装置上执行的 Java 程序所能使用的类别函式库、这些规范之中所定义的类别函式库为 Java 核心类别的子集合以及与该型态装置特性相符的扩充类别。比方就 CLDC 的规范来说，可以支持的核心类别为 java.lang.*、java.io.*、java.util.*，而可以支持的扩充类别为 java.microedition.io.*。

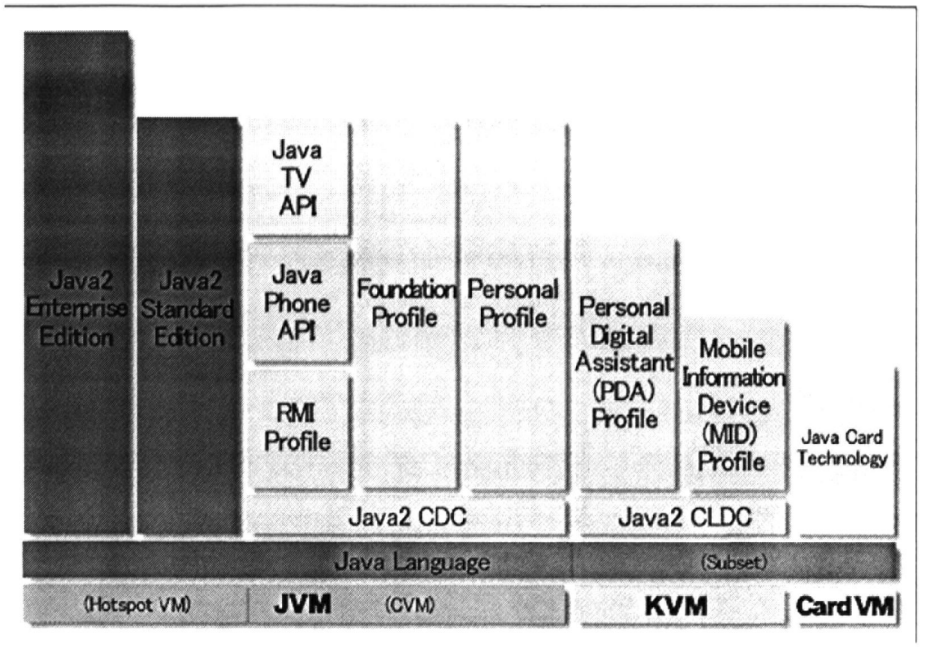

图 2-1　JAVA 版本与其所依据虚拟机所构建的架构[12]

2.1 J2ME 发展概述

J2ME 试图支持 PDA、手机、传呼机、电视机顶盒、远程遥控装置以及其他的嵌入式设备，然而，相同类型的设备可能来自不同的生产厂商，要想为这些形形色色的设备定义一种最优化的技术显然是不可能的。但是，为了保证这些设备之间的互操作性，标准化是必需的。鉴于这些设备的复杂性，Sun 并没有为 J2ME 定义一个技术规范，而是参照这些设备，首先根据处理器性能、存储能力将所有的嵌入式装置分为两类：一种是运算功能有限、电力供应也有限的嵌入式装置（比方说 PDA、手机）；另外一种是运算能力相对较佳、并且在电力供应上相对比较充足的嵌入式装置（比方说冷气机、电冰箱、电视机顶盒）。因为这两种区分，所以 Java 引入了一个叫做 Configuration 的概念，然后把上述运算功能有限、电力有限的嵌入式装置定义在 Connected Limited Device Configuration（CLDC）规格之中；而另外一种装置则规范为 Connected Device Configuration（CDC）规格。这些规格之中定义了两种类型嵌入式装置至少要符合的运算能力、供电能力、内存大小等规范，同时也定义了一组在这些装置上执行的 Java 程序所能使用的类别函式库、这些规范之中所定义的类别函式库为 Java 核心类别的子集合以及与该型态装置特性相符的扩充类别。比方就 CLDC 的规范来说，可以支持的核心类别为 java.lang.*、java.io.*、java.util.*，而可以支持的扩充类别为 java.microedition.io.*。

区分出两种主要的 Configuration 之后，J2ME 接着再定义出 Profile 的概念。Profile 是架构在 Configuration 之上的规格。之所以有 Profile 的概念，是为了要更明确地区分出各种嵌入式装置上 Java 程序该如何开发，具有哪些功能。因此 Profile 之中定义了与特定嵌入式装置非常相关的扩充类别，而 Java 程序在各种嵌入式装置的使用者接口该如何呈现的规定就是定义在 Profile 里头。Profile 之中所定义的扩充类别也是根据底层 Configuration 内所定义的核心类别所建立。

2.2 J2ME 技术的体系结构

2.2.1 CLDC 与 MIDP 概述

目前 J2ME 对于 Configuration 以及 Profile 所组成的架构大致如图 2-2 所示：

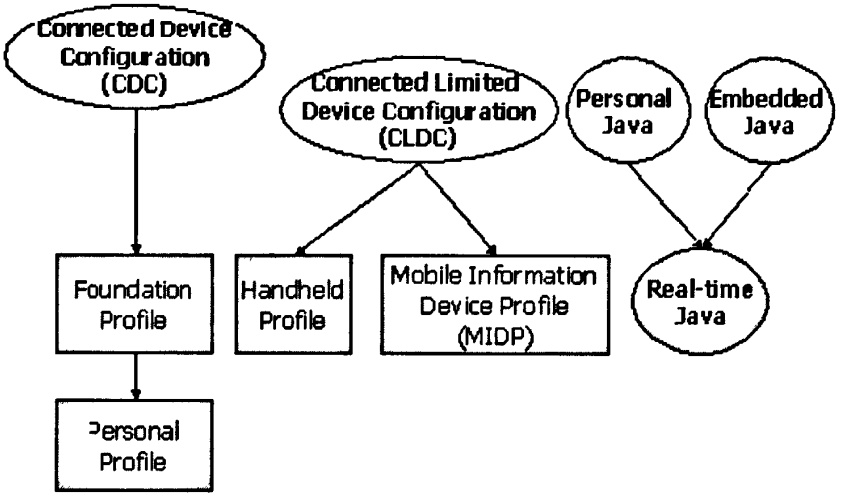

图 2-2　Configuration 与 Profile 架构[12]

从上图可以看出，目前从 CLDC 所衍生出来的 Profile 有两种，一种是 Handheld Profile，大多数用在 PDA 上；而另外一种是 Mobile Information Device Profile（MIDP），这个 Profile 是针对行动装置所定义，比方说呼叫器、移动电话等等，都是属于行动装置。

区分出两种主要的 Configuration 之后，J2ME 接着再定义出 Profile 的概念。Profile 是架构在 Configuration 之上的规格。之所以有 Profile 的概念，是为了要更明确地区分出各种嵌入式装置上 Java 程序该如何开发，具有哪些功能。因此 Profile 之中定义了与特定嵌入式装置非常相关的扩充类别，而 Java 程序在各种嵌入式装置的使用者接口该如何呈现的规定就是定义在 Profile 里头。Profile 之中所定义的扩充类别也是根据底层 Configuration 内所定义的核心类别所建立。

2.2 J2ME 技术的体系结构

2.2.1 CLDC 与 MIDP 概述

目前 J2ME 对于 Configuration 以及 Profile 所组成的架构大致如图 2-2 所示：

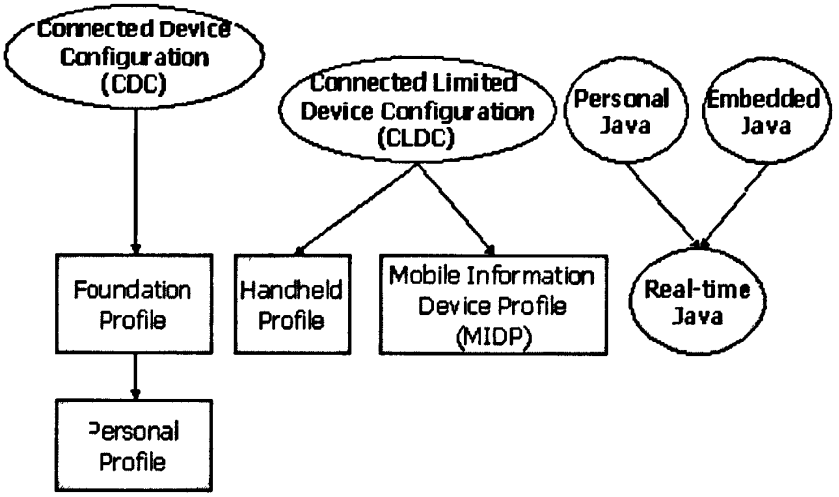

图 2-2　Configuration 与 Profile 架构[12]

从上图可以看出，目前从 CLDC 所衍生出来的 Profile 有两种，一种是 Handheld Profile，大多数用在 PDA 上；而另外一种是 Mobile Information Device Profile（MIDP），这个 Profile 是针对行动装置所定义，比方说呼叫器、移动电话等等，都是属于行动装置。

2.2.2 CLDC 规范

Connected Limited Device Configuration（CLDC）就 Sun 的文件所描述，是定义为「可以放在您的手掌上的装置」，比如 Palm 系列的 PDA 或是手机。而 Connected Device Configuration（CDC）根据 Sun 文件的描述，定义为「可以插在墙壁上的装置」，比如电视机顶盒。不管如何，这两种 Configuration 之中定义的皆为这两种型态的嵌入式装置要执行 Java 程序所需要的最小配备需求。以下（表 2-1）是 CLDC 与 CDC 各自的最小配备需求：

配备\Configuration	CLDC	CDC
RAM	RAM 与 ROM 再加上闪存（Flash Memory）要为 128K~512K	大于等于 256K
ROM	RAM 与 ROM 加上闪存要为 128K~512K	大于等于 512K
电源	通常是使用电池，所以电源有限	不设限

表 2-1　CLDC 与 CDC 的最小配备需求[9]

2.2.3 MIDP 规范

在 Javasoft 的官方文件之中所架构出来的 J2ME 如图 2-3 所示：

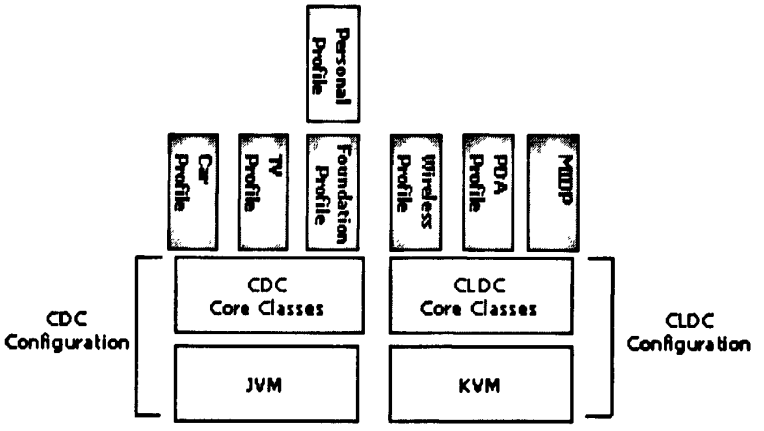

图 2-3　J2ME 架构[12]

根据上图，我们来介绍一下建构在 Configuration 之上的 Profile。在 Profile 之中也定义了特定种类嵌入式装置的最小配备需求。既然 Profile 建构在 Configuration 之上，其意义就是说 Profile 之中所规范的配备需求不可能比 Configuration 还要低。同时，Profile 之中对于显示功能、网络功能、以及耗电能力等相关需求将会比 Configuration 之中所规定的还要高。

以下（表 2-2）是一些 Profile 的配备需求：

配备\Profile	Foundation Profile	Personal Profile	MIDP
RAM	至少 512K	至少 1MB	RAM 与 ROM 至少要为 512K
ROM	至少 1024K	至少 2.5MB	RAM 与 ROM 至少要为 512K
电源	不设限	不设限	通常是使用电池，所以电源有限
网络连接能力	部分功能	部分功能	具有低频宽的无线通讯能力
其它	要有额外的 RAM 或 ROM 供应用程序执行	要有额外的 RAM 或 ROM 供应用程序执行	要有额外的 RAM 或 ROM 供应用程序执行并储存资料

表 2-2　Profile 的配备需求[9]

2.2.4 CLDC / MIDP 架构

J2ME/CLDC/MIDP 架构是专为移动通讯设备而设计的。下图（图 2-4）说明 CLDC 与 MIDP 之间的关系。

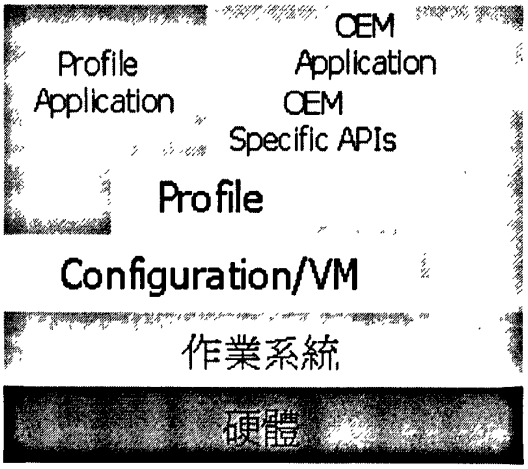

图 2-4　CLDC 与 MIDP 之间的关系[12]

从上图可以看出：

- MIDP 建立在 CLDC 功能的基础之上，根据某个 Profile 规范所撰写的 Java 程序除了可以直接呼叫 Profile 中定义的扩充类别，也可以直接调用 Configuration 里所定义的核心类别子集与扩充类别；
- J2ME 规范允许设备制造商提供与自己设备相对应的 API 供程序调用，一般来说，设备厂商都会提供一些自己的 API 供开发人员使用，这些功能都是有针对性的，因此在不同的设备之间可能无法移植。一般来说，这部分 API 能够提供额外的功能或者在运行性能上有所提高。

此外，从上图还可以看出在 CLDC 之上，有两种 API：

- MIDP APIs：MIDP 规范所要定义的 API；
- OEM Specific API：MIDP 规范所涉及的无线通讯设备多种多样，因此它不可能涉及所有设备的需求。因此这一类的 API 是由 OEM 厂商提供的，以便访问特定设备的特定功能。但基于这些 API 的应用可能不在其他的 MIDP 设备上运行。

需要注意的是，CLDC 和 MIDP 是 J2ME 中两个不同的规范，在 CLDC v1.0 和 MIDP 1.0 规范下。这两个部分是没有交叉的。但是当 MIDP 2.0 发布时，除了在 MIDP 所提供的 API 中进行了改动，还在 CLDC 提供的 API 中增加了部分功能。例如 javax.microedition.io 包是由 CLDC 规范所定义的，但在 MIDP 2.0 发布时 javax.microedition.io 包中增加了对套接字、串口等的支持。

2.3 J2ME 安全体系结构

J2ME 安全体系主要包括两个层次：即 JVM 虚拟机层和应用程序层。具体结构如图 2-5 所示：

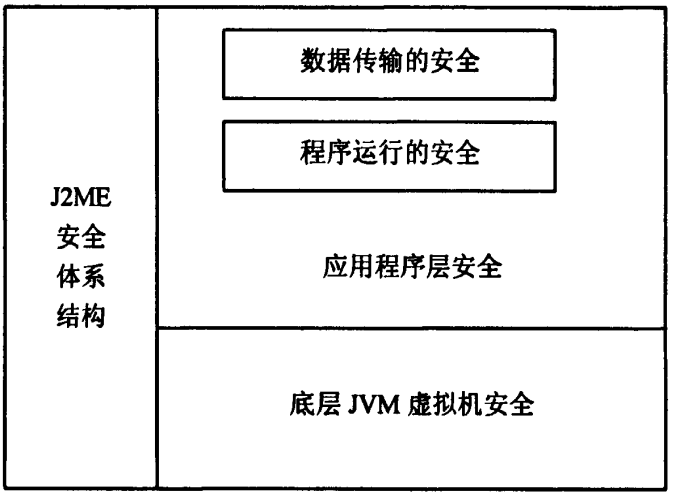

图 2-5　J2ME 安全体系结构简图

2.3.1 底层 JVM 虚拟机安全

虚拟机[21]安全性，完全由 CLDC 提供，MIDP 不定义任何额外的底层安全特性，它确保虚拟机和设备不会由于病态或者恶意编码的类文件导致系统崩溃。

J2ME 通过多种策略来保证底层系统的安全，如类文件检查，内存越界检测和 Java 平台本身独有的沙盒模型（即 CLDC/MIDP-Sandbox）机制，通过这些手段以确保应用程序能在一个较为封闭的环境中安全运行。

1. 字节验证模式[19]

J2SE 提供了包括许可权限、安全策略、保护域和接入控制等安全模式。同时字节码验证模式可以确保存储在 Java 类文件的字节码不会对 JVM 产生影响。然而对于 CLDC 资源受限设备来说，这种先进的字节码验证仍不可承受，容易造成较大的内存空间消耗，因此不具可行性。

在这里对 CLDC 定义了另一种更有效的验证模式。此模式将 CLDC 应用的验证分为两个阶段：第一阶段在 Java2 标准平台或更高级的环境中执行，被称为

2.3 J2ME 安全体系结构

J2ME 安全体系主要包括两个层次：即 JVM 虚拟机层和应用程序层。具体结构如图 2-5 所示：

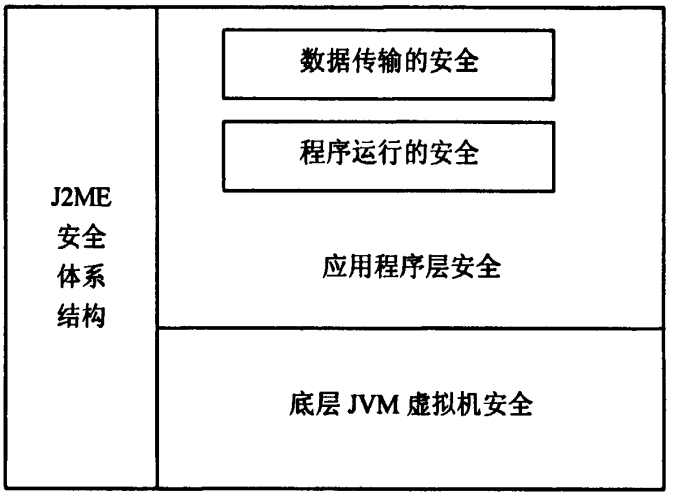

图 2-5　J2ME 安全体系结构简图

2.3.1 底层 JVM 虚拟机安全

虚拟机[21]安全性，完全由 CLDC 提供，MIDP 不定义任何额外的底层安全特性，它确保虚拟机和设备不会由于病态或者恶意编码的类文件导致系统崩溃。

J2ME 通过多种策略来保证底层系统的安全，如类文件检查，内存越界检测和 Java 平台本身独有的沙盒模型（即 CLDC/MIDP-Sandbox）机制，通过这些手段以确保应用程序能在一个较为封闭的环境中安全运行。

1. 字节验证模式[19]

J2SE 提供了包括许可权限、安全策略、保护域和接入控制等安全模式。同时字节码验证模式可以确保存储在 Java 类文件的字节码不会对 JVM 产生影响。然而对于 CLDC 资源受限设备来说，这种先进的字节码验证仍不可承受，容易造成较大的内存空间消耗，因此不具可行性。

在这里对 CLDC 定义了另一种更有效的验证模式。此模式将 CLDC 应用的验证分为两个阶段：第一阶段在 Java2 标准平台或更高级的环境中执行，被称为

预验证；第二阶段用于 CLDC 设备，被称为设备中验证。

1) 预验证

预验证模式[22]用于预校验 Java 类文件，它摒弃了 J2SE 中 Java 字节码验证所用的规范。为实现应用的安全性，此模式要求被下载的 Java 类在开发平台上，先通过类文件验证步骤来实现低级的虚拟机安全，同时确保位于受保护系统包中的类不能被应用程序中的类覆盖。预验证执行过程流程如图 2-6 所示。

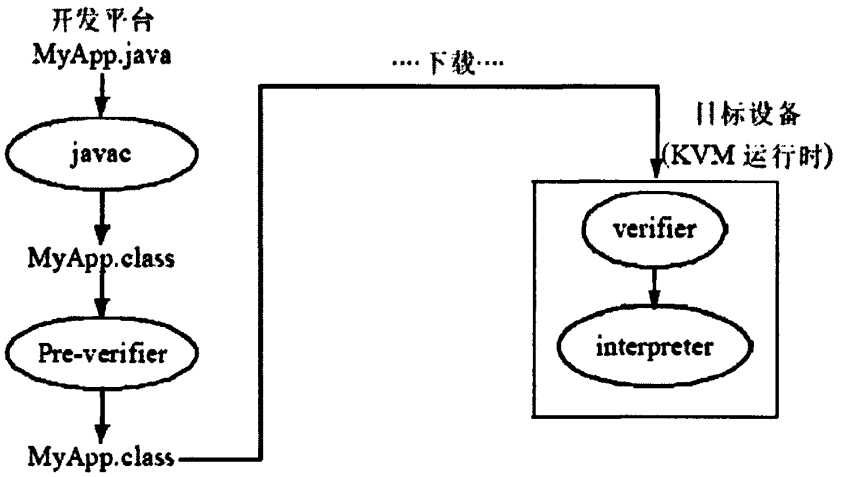

图 2-6　预验证执行流程[22]

在 CLDC 类验证机制中，要下载的 Java 类文件的每一个方法都包含一个堆栈映射（StackMap）属性，此属性为 CLDC 中新定义的，并未在 Java 虚拟机规范中定义。验证文件的字节码被相应的字节码替换，即预验证将所有 jsr、ret、wide ret 字节码从类文件中去除，并且在类文件中增加堆栈映射。替换后，堆栈映射就被预验证工具插入到预校验的 Java 类文件中。预验证类文件本身还是 Java2 平台的类，但是预验证工具将堆栈映射插入其中后，它就能被 J2SE 虚拟机忽略，从而使得 J2ME CLDC 的预验证类文件向上与 J2SE 完全兼容。通常预验证工具会分析类文件中的每一个方法。预验证堆栈映射属性会使此类文件尺寸增加约 5%。与标准的 Java 虚拟机验证过程相比，这个属性可以使与 CLDC 兼容的 Java 虚拟机验证 Java 类文件的速度更快，且虚拟机代码和动态 RAM 消耗更少，而它们的安全级别却相同。

2) 设备中验证

设备中验证[22]在 CLDC 中由 KVM 负责执行，KVM 字节码验证充分利用了由预验证工具产生的信息。

在设备中验证主要包括以下步骤：

首先，验证工具会分配足够的空间用于存储所有变量的类型和操作数堆栈项目。这里内存的尺寸取决于本地变量和代码属性中堆栈的大小，而此存储空间用于存放通过验证器的字节码的原始类型。

其次，验证器将初始化这些原始类型，如方法、空操作数堆栈，使之成为 this 指针。

接着，验证器将进行验证操作，将原始类型与堆栈映射入口记录的指令进行配对，若不匹配则抛出错误。

此外，设备中验证还要执行一项额外检查：验证器必须区分对象是新分配的对象还是构造器调用的对象，它必须确保被新指令分配的对象可以方便调用构造器，而且必须是合法的新指令分配的对象才可以调用构造器。

2. 沙盒模型

在 J2SE 中，对应用程序外部资源的访问是由安全管理器和访问控制以及安全策略来实现的，但是由于 CPU 的能力和内存的有限，CLDC 中定义了另一种安全模式——沙盒模型[18]（Sandbox Model）。

在沙盒模型中一个应用必须运行在一个封闭的环境里，在其中应用仅可以访问设备所支持的配置、描述和原始设备制造商（OEM）特定类所定义的那些库。在这个封闭的环境中，应用程序不可介入那些安全敏感的资源，即 Java 应用不能从这个沙盒里逃出来或者访问不属于预先定义的功能的那些库或资源。Java 应用程序能够安全运行的封闭沙盒环境是通过严格规范 Java 语言特性得到的。

从技术角度来说，沙盒模型[19]意味着：

a) 类文件被正确地检查并且保证是合法有效的 Java 应用；
b) 一个封闭的预定义的 Java API 集合可以被程序员使用。这些 API 由 CLDC 描述（例如 MIDP）和 OEM 特定类所定义；
c) 在设备上 Java 应用的下载和管理是在虚拟机内部的本地代码级上实现的，而且并不提供用户定义的类加载器，其目的是为了阻止程序员替换虚拟机中标准的类加载器；
d) 可以访问的本地函数集合也是封闭的，这就意味着应用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 CLDC 描述（如 MIDP）或 OEM 特定类所提供的 Java 库一部分的本地函数。

CLDC 沙盒模型将 Java 语言中与安全较敏感的问题相关的部分排除在外，保证了不可信赖且可能含有恶意代码的应用程序不能获得系统资源的访问权，因此通过减少 Java 语言特性来获得应用的安全性，同时也使 JVM 尺寸更小。

3. Java 本地接口（JNI）

在 CLDC 中不支持 J2SE 中采用的 Java 本地接口[19]。因为如果执行 JNI，则会造成 CLDC 目标设备开销大，并致使本地功能调用（Native Function Calls）被排除在沙盒模型定义的规范之外，因此 CLDC 应用运行本地代码非常不安全。

CLDC 不再支持 JNI 的另一个原因是为了减少虚拟机的尺寸，因为 JNI 的执行将极大地增加 KVM 的尺寸。本地功能调用可在 KVM 中执行，但是它们并不在 CLDC 和 MIDP 定义的规范内。

4. 用户自定义类加载

CLDC 不支持用户自定义类加载[19]，CLDC 所使用的类加载为内置的类加载器，不能被用户覆盖、替换或重新定义，从而保证了类的完整性，确保 CLDC 的安全。

5. 其它安全措施

为了确保沙盒模式的有效性，以下的 Java 语言特性[19]也是在 J2SE 中定义过，却在 CLDC 中予以删除了。

a) CLDC 不支持 java.lang.reflect 类。KVM 没有映射的特征，因此不能使用与映射相关的功能，如远程方法调用 RMI 和对象串行化（Object Serialization）。
b) CLDC 可以执行多线程，但它既不支持 java.lang.Thread-Groups，也不支持守护线程（Daemon Thread）。为了聚集和管理线程，必须执行其本身的线程组机制。
c) CLDC 不支持 java.lang.Object.finalize () 方法。在 J2SE 中，当系统判断应用不再对对象有用时，java.lang.Object.finalize() 方法就被垃圾收集器调用。在 J2ME 的 CLDC 平台下 java.lang.Object.finalize() 方法是从来不会被调用的。
d) 弱的引用在 CLDC 平台也是不允许的。

2.3.2 应用程序层安全

MIDP2.0 推出之前，无线 Java 仅仅通过取消自定义类装载器、限制可用的 API、禁止覆盖系统类等措施来实现应用级安全，但这些措施都是静态的，对所有的无线 Java 应用都一样。为了适应无线网络中较高的安全需求，MIDP2.0 在增加新的 API、特别是新的网络连接功能的同时，也引入了新的安全措施——基于保护域的访问控制和 PKI 技术，主要为了限制无线 Java 应用对敏感的 API 的访问。这些敏感的 API 通常存在潜在的安全隐患，比如可能泄露设备中用户的私有信息，让用户为网络连接付费等。

1. 数据传输的安全性

数据传输安全性，就是要在无线通信应用中保证传输数据和信息的完整性、保密性和不可否认性等。

1) 通过 HTTPS 和安全套接字实现端到端的安全

在普通环境中，我们大多使用标准安全网络协议来保证网络通信的完整性和保密性。其中最通用的协议就是 SSL 和 TLS 协议。它们为应用数据在开放的非安全网络上提供了加密、源端认证和完整性保护。但是鉴于 SSL 和 TLS 协议不适合无线环境中移动设备计算能力差和存储空间有限这些缺点，J2ME 引入了 KSSL 协议。

KSSL 协议实质上也是通过 SSL 这个协议来实现数据的安全传输的。有了 KSSL 协议，就可以通过 HTTPS 和安全套接字实现端到端的安全性。为此，MIDP2.0 中定义了两种安全网络接口：HttpsConnection 和 SecureConnection。

HTTPS 又称安全的 HTTP，访问这种安全的 HTTP 连接，是要受到防止非授权的数据传输和数据接受的限制。在 MIDP2.0 也增加了对 Socket、Datagram 串行通信等网络协议的支持，其中 SecureConnection 继承自 SocketConnection 这个类，也同样能够保证网络传输的安全。与 HTTPS 相同的是它也支持 TLS、SSL、WAP (TM) 等协议规范。

在现在的电子商务领域，HTTPS 已经得到了最广泛的应用，J2ME 对 HTTPS 的支持也必将推动无线商业应用的发展。

例如：我们要访问一个安全的网页：https://www.mycompany.com。下面的代码可以帮助我们实现安全的访问：

```
String myurl = "https://www.mycompany.com" ;
try{
  //建立 HTTPS 连接
  HttpsConnection sc = (HttpsConnection) Connector. open(myurl);
    //用于数据的输入
  InputStream is = sc. openInputStream();
  //用于数据的输出
  OutputStream os = sc. openOutputStream();
  ……
} catch (CertificateException ce) {
  …… //获得证书的相关信息
} catch (Exception e) {
  ……//其它异常信息
}
```

在这个过程中，HTTPS 连接就会帮助我们完成与服务器端的握手会话、身份认证、证书验证和数据加密传输等工作，一切正确后就可以保证我们安全地访问网页了。

但是基于 SSL 协议的 HttpsConnection 和 SecureConnection 这也不能保证绝对的安全，因为在 SSL 协议握手阶段，会用系统生成的随机数值作为会话密钥，但这些随机数值的产生是有规律可寻的。还有，MIDP2.0 支持 HTTPS，这为端到端的加密提供了一个良好的解决方案，但它并不适合为以后广播等多点传输应用提供加密通信。而它正是未来无线应用的重点。

2) 支持第三方的安全方案保障传输的安全

目前，XML 已经广泛应用于数据交换领域，具有优秀的跨平台特性。J2ME 上的开发很多时候也都涉及到 XML。MIDP2.0 现在还不支持 XML 和数字签名技术，但它支持第三方工具来实现传输 XML 数据的安全。

MIDP2.0 支持第三方 XML 解析器，如 kxml、nanoxml 等。在加密和签名方面可以使用 BouncyCastle API 等。BouncyCastle API 是一个开放源码的 Java 加密项目,提供一个适用于 J2ME 的轻量级的加密包。它支持很多通用的块加密和流加密算法，还支持摘要生成、密钥交换、数字签名等，这可以保证无线 Java 网络的数据传输的安全。

例如，我们可以采用如下流程实现了 XML 数据的安全传输。

```
a. 生成密钥对
void   generateKeys()
b. 生成消息摘要
String getDigest (String xmldatas)
c. 服务器端进行数字签名
String getSignature (String xmldatas)
d. 客户端验证签名
boolean verifySignature (String message, String signature),
```

无庸质疑，采用加密技术是保障网络通信安全的一个最基本也是最重要的手段。然而即使在 MIDP2.0 中，J2ME 也没有明确提供任何加密系统 API，甚至连最常用的 RSA 算法也不支持。所以建议最好采用第三方的轻量级的加密工具包，来满足实际应用中我们较高级别的安全需求。

2. 程序运行的安全性

运行安全性就是要保证系统的稳定性、程序本身不可篡改、根据权限的访问控制以及网络本身不能带来额外的费用开支等。MIDP2.0 主要采用基于 PKI 技术的访问控制、保护域、代码签名和验证以及证书管理等方法来保障 MIDlet 套件的源码一致性和真实性。

1) 确保程序没有被篡改

MIDP2.0引入了可信任 MIDlet套件(trusted MIDlet suite)的概念。在MIDP1.0中，所有的MIDlet无差别的对待，都"一视同仁"的在沙箱（sandbox）中执行，且只能访问和使用少数几个系统资源，只能使用不危险的APIs。MIDP2.0 可以对MIDlet suite 进行数字签名，并且这些签名可以被验证。对于签名的MIDlet suite，可以验证签名者的真伪，可以校验MIDlet suite的完整性。J2ME系统对可信任的MIDlets 开放了更多的APIs，例如网络访问、消息(message)和PIM APIs等。对MIDlet suite 的签名和验证签名都是基于已经存在的Internet 标准——X.509 公钥体系（X.509 PKI）。每个移动信息设备都保存了一套根证书。MIDP 中采用的X.509 证书是基于WAP Certificate Profile 的。MIDP 按照PKCS#1 标准为MIDletsuite 签名，采用的签名的算法是RSA，计算消息摘要的算法是SHA-1，签名的结果以base64 格式编码并保存在MIDlet 的应用程序描述文件（jad）中。

在用户下载 JAR 包，并安装或者升级应用程序时，系统必须强制对所下载的 JAR 文件进行验证，只有验证通过才能以正常的可信任方式安装。如果验证失败，根据情况选择不能安装或者以不可信任的方式来安装。

移动信息设备根据描述文件中的证书链一直验证到根证书。验证主要包括以下几点：①证书路径是否有效；②签名是否吻合，同时验证MIDlet suite 是否被篡改过；③证书是否在有效期内。MIDP2.0 规范并没有强制要求更新过期的证书，而是将这一点留作虚拟机的可选实现；如果设备实现了相关的函数，那么将采用OCSP(online certificatestatus protocol)在线验证证书状态。

验证过程如图 2-7 所示。

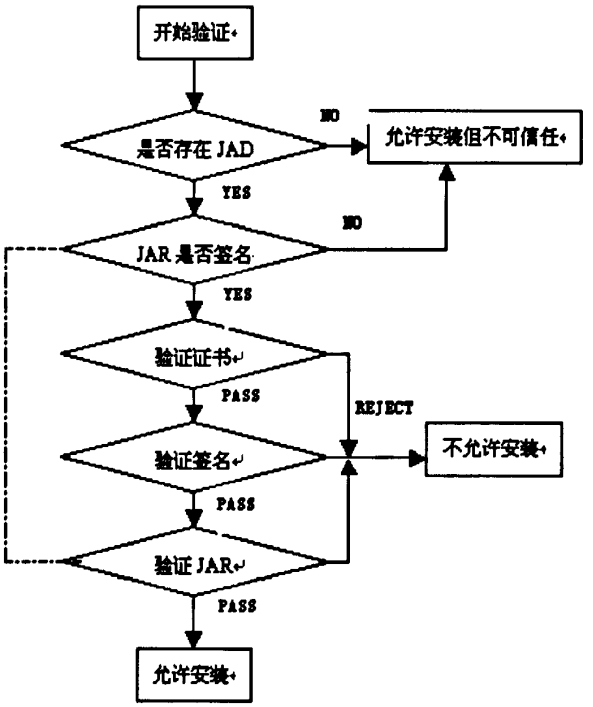

图 2-7　MIDlet 套件的验证过程

2) API 权限管理策略[20]

MIDP1.0 的 MIDlet suite 被限制使用敏感的、对设备的软硬件系统构成威胁的 API——在这里，我们不妨称这些 API 是受保护的 API；也就是说，MIDP1.0 对待受保护的 API 的策略是通通不让用，这虽然加强了系统的安全性，但是对程序所能完成的功能限制过多了。MIDP2.0 对此进行了改进，MIDP2.0 为受保护的 API 赋予一定的权限，MIDlet suite 要使用这些受保护的 API，必须先申请相应的权限并能获得准许使用的权限。

MIDP2.0 有两种类型的权限，每种都有和用户不同的交互模式：其一是“被

准许”（allowed），当受保护的 API 的权限被标识为“被准许”（allowed），MIDlet suite 请求使用这些 API 时不再需要经过用户明确的授权就可以使用；另外一种是“用户模式”（user），MIDlet suite 请求使用这些权限标识的 API 都需要经过用户明确的直接授权，授权的模式有以下 3 种：①“终身制”（blanket authorization），这个 MIDlet suite 每次调用第一次被授权的受保护的 API 都不需要再次明确被授权，除非这个 MIDlet suite 被卸载或者受保护的 API 的权限被修改了；②“临时制”（session authorization），在这个 MIDlet suite 当前执行期内，系统对这个 MIDlet 申请使用的受保护的 API 的授权一直有效，当这个 MIDlet suite 重新执行后，调用这里的受保护的 API 需要重新请求权限；③“一次性”（oneshot authorization），此次授权仅仅在当前调用有效，当受保护的 API 再次被调用时需要重新申请权限。

MIDP 中受保护的 API 数目众多，如果对每一个函数都赋予权限，那么用户执行一个 MIDlet suite 的时候，绝大部分时间都用来给每一个函数调用进行授权认可，即使授权设置为“终身制”只需要第一次调用时设置，那也是不可忍受的。因此，MIDP2.0 将这些受保护的 API 分组(group)管理，然后针对每一个组进行权限管理，MIDlet suite 请求授权的权限都是以组为单位的。例如，MIDP2.0 为 GSM 和 UMTS 的设备定义了以下几个组：打电话（phone call）、访问网络（network access）、使用短消息和多媒体消息（messaging）、程序自动执行（application auto invocation）、使用本地连接（local connectivity）、记录多媒体信息（multimedia recording）、读写用户数据（user data access）。

MIDP2.0 应用程序非常丰富，每个移动信息设备上存贮的 MIDP2.0 的应用程序数目众多，对每一个 MIDlet 进行权限的管理也是繁琐和困难的，因此 MIDP2.0 针对不同的 MIDlet suite 进行分组管理。这就是下面要介绍的“保护域”（protection domain）。

3) 程序必须在保护域内运行[20]

保护域是 MIDP2.0 中的一个很重要的新概念。一个保护域有一套使用受保护的 API 的权限，这些权限被自动赋给属于这个域中的 MIDlet suite。每一个根证书——前面提到的用于验证 MIDlet suite 的数字签名，都属于一个惟一的确定的保护域。一个移动信息设备拥有一个或者几个保护域，每一个保护域都拥有一套相关的根证书。

当一个 MIDlet suite 被验证为可信任的 MIDlet 时，根据这个 MIDlet suite 的签名追溯到的根证书，这个 MIDlet suite 被划分到与这个根证书对应的惟一的确定的保护域中。一个 MIDlet 不可能既属于这个保护域，又拥有另一个保护域的

访问受保护的 API 的权限。MIDP2.0 为 GSM/UMTS 的设备定义了以下的保护域：

① 厂商域（manufacturer domain），这是功能最强大的一个保护域，包含预安装的厂商的应用程序。这个域中的所有权限都被设置为“被准许”（allowed），属于这个域的 MIDlet 可以毫无限制的使用 MIDP2.0 以及 JVM 的所有功能。

② 使用者域（operator domain），包含了由移动设备的使用者签名的应用程序。这个域中的所有权限都被设置为“被准许”（allowed）。

③ 第 3 方域（third-party domain），包含除厂商和使用者之外的签名的程序。这个域中的所有权限都被设置为“用户模式”（user），调用受保护的 API 需要使用者明确的批准。

④ 非信任域（untrusteddomain），包含所有没有数字签名的程序或者数字签名验证没有通过的程序。所有 MIDP1.0 的程序都属于这个域。

每一个支持 MIDP2.0 的移动信息设备都支持几个保护域，并且都精确的规定了每个保护域的权限集合，保存了和具体的 MIDlet suite 相关的授权和授权交互的模式。权限和保护域的使用，在不损失系统的安全性的前提下大大增强了系统的功能性，并且对 MIDlet 的签名本身就降低了系统受恶意程序攻击的风险，增强了系统的安全性。

如果对签名过的 MIDlet 套件的检验没有通过，或者套件本身没有被签名，则该 MIDlet 套件被称为非可信的 MIDlet 套件。非可信 MIDlet 套件在非可信的保护域中运行，该保护域限制了 MIDlet 对敏感 API 的访问或者根本不允许其访问。

通过保护域以及 MIDlet 套件模型。在实际应用中，J2ME 就可以满足对安全性很高的移动应用程序和商业模型的需要。

4) 网络连线时必须提示让用户知道

随着网络时代的到来，手机上网用户越来越多。但手机上网的费用暂时也是比较高的。如果由于用户不小心操作导致连上了网络，那么势必给用户造成一定的损失。而 J2ME 就必须在用户上网时，给出提示来减少潜在的危险。

但这也存在着一些隐患，如著名的 Phenoelit 黑客组织曾经利用一些手机本身的系统程序漏洞，成功地绕过了 J2ME 的这一重要提示。他们主要利用用户所看到的用户界面（UI）与实际执行的程序的分离来屏蔽掉了系统的提示界面，这类似于 URL 的重定向技术一样。

建议对于来源不明或者以不可信方式安装的程序，在执行网络连接等需要付费的功能时，一定要谨慎对待。

2.3.3 J2ME 安全与 Java Card

Java Card [23] 是一种可以运行 Java 程序的 CPU 智能卡，也可以理解为是 Sun 为智能卡开发平台而制定的一个开放的标准。

MIDP2.0 主要是利用 PKI 来保障了程序运行时的安全，而这些包括敏感数据（如用户的私钥、公钥证书）以及个人信息的存储都变的非常重要，因为它们是实施 PKI 的基础。现有的无线设备都依赖于比如 Java Card 等"安全元素"的交互来达到存储安全和执行安全的目的。这一环的安全性至关重要，可以说它在很大程度上决定了整个无线 Java 平台的安全性。

为此，MIDP2.0 特别提供了可选的 API 开发包 JSR-177，也就是 Security and Trust Services API（SATSA）。SATSA 考虑到用于 Java Card 应用程序的两个模型：APDU-消息传递模型和 Java Card RMI 面向对象分布式模型。SATSA 为了这些实现模型，在通用连接框架（Generic Connection）中新增了两个 Connection 连接类型（接口）：APDUConnection、JavaCardRMIConnection。我们利用他们来访问 Java Card。更重要的是 SATSA 还为我们提供了两个可选的工具包：PKI 包和加密算法包。他们可以帮助我们完成密钥对的生成、消息摘要、数字证书管理，也可以利用对称和非对称的加密算法加密数据等。

在 SATSA 出现之前，为了实现利用移动设备对这些 Java Card（也称为 Smart Card）进行管理的操作，我们一般采用图 2-8 所示的方式。这种方式存在的问题是：我们必须提供一个 Java Card 到移动设备的网关。而实现这个网关本身就比较复杂，并且也不能很好地保证通信的安全。

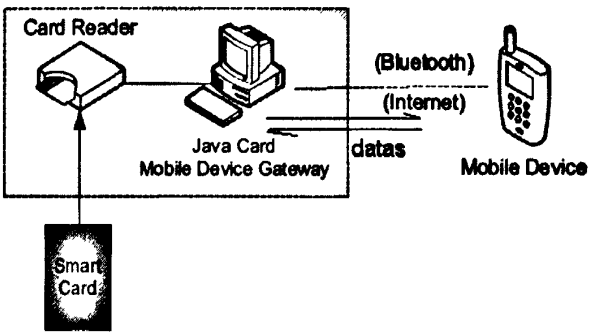

图 2-8 移动设备访问 Java Card 的一般方式

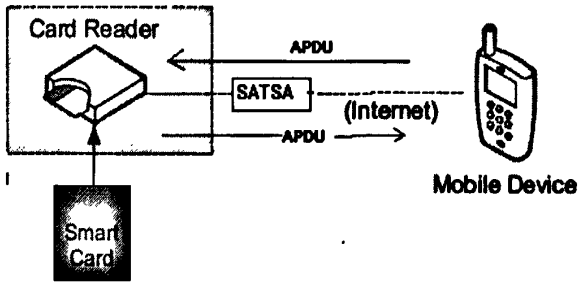

图 2-9 利用 SATSA 访问 Java Card 的方式

SATSA 恰好可以解决上述问题。如图 2-9 所示。

下面几个简单步骤就可帮助我们完成移动设备与 Java Card 之间交换数据并执行加密功能实现简单而又安全通信的目的。

a) 首先给出加密明文、加密算法和加密公钥：

```
byte[ ] plainText = "Wireless Java Security";
byte[ ] cipherText = new byte[256];
String keyAlg = "RSA";
byte[ ] publicKey = "Xu Qiting";
```

b) 从公钥中生成一个 X.509 格式的密钥，并从指定的加密算法的密钥库中生成一个密钥对象。

```
X509EncodedKeySpec pks = new
        X509EncodedKeySpec (someonesEncodedPublicKey);
KeyFactory kf = KeyFactory.getInstance (keyAlg);
PublicKey publickey= kf. GeneratePublic (pks);
```

c) 下面我们就可以对明文执行加密了。

```
Cipher cipher = Cipher.getInstance (publicKey. getAlgorithm());
cipher.init (Cipher.ENCRYPT_MODE, publicKey);
cipher.doFinal (plainText, 0, plainText.length, cipherText, 0);
```

d) 随后就可以把得到密文 cipherText 打成 APDU 包在无线网络上进行安全传输，并在另一端解密得到明文信息。

第三章 椭圆曲线密码算法实现的优化设计

1976 年，Diffie 和 Hellman 提出了公钥密码学（Public-key Cryptography），离散对数问题（DLP）和密码学意义上的重要性开始逐渐为人们所重视起来。ElGamal 首先描述出如何将离散对数问题应用到公钥加密和数字签名方案中去。随后，ElGamal 的这种方法被进一步改善提高，并且被各种各样的应用中的各种各样的协议所采纳。其中之一就是美国政府的数字签名算法（DSA）。

尽管当年 Diffie 和 Hellman 提出离散对数问题是要应用到他们的密钥协商协议中，这时离散对数问题被精确定义为寻找模 p 乘法群中生成元的对数问题，但是现在这个问题可以被推广到任意群众。椭圆曲线密码系统正是基于有限域上的椭圆曲线点所构成的群的离散对数问题。

椭圆曲线已经被广泛研究了 100 多年，而且从中得到了非常广泛的研究课题。椭圆曲线现在已经成为很多重要应用领域的工具，如编码理论、伪随机比特生成以及数论算法（素性证明、整数分解）等。

椭圆曲线密码系统自 1985 年由 Neal Koblitz 和 Victor Miller 分别独立地提出，其后人们对它的安全性和实现的有效性进行了广泛和深入的研究。椭圆曲线密码系统是建立在椭圆曲线点群的离散对数问题上的。在有限域上椭圆曲线点群中，还没有关于寻找离散对数的诸如“Index-calculus”的亚指数时间算法出现。因此，利用规模更小的椭圆曲线群来达到相同的安全级别，可以拥有更小的密钥长度、更小的带宽需要，以及更快的实现。这些特性对于那些计算能力和集成芯片空间受限的安全应用特别具有吸引力，例如，在智能卡、PC 卡，以及无线设备上。

3.1 椭圆曲线的基本概念

1. 椭圆曲线的定义[27]

椭圆曲线方程的一般形式为：

$$E:\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

其中 a_1，a_2，a_3，a_4，a_6 为数域 K 中的元素，域 K 可以是有理数域、实数域、复数域，还可以是有限域。

域 K 上的点集为：

$$E:\ \{(x,y)\,|\,y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}$$

式中，a_1，a_2，a_3，a_4，a_6 $\in K$；$\{O\}$为无穷远点，叫做域 K 上的椭圆曲

线。

椭圆曲线方程可写成齐次形式：

$$Y^2Z + a_1XYZ + a_3Y = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

式中，a_1，a_2，a_3，a_4，a_6 $\in K$。这时，无穷远点 O 就是齐次坐标点 [0:1:0]。

在对域 K 上的椭圆曲线 E 的研究中，通常取如下形式的椭圆曲线方程：

2）当域 K 的特征不为 2、3 时，椭圆曲线方程为

$$y^2 = x^3 + a_4x + a_6$$

3）当域 K 的特征为 2 时，椭圆曲线方程为

$$y^2 + xy = x^3 + a_2x^2 + a_6 \text{ 或 } y^2 + a_3y = x^3 + a_4x + a_6$$

4）当域 K 的特征为 3 时，椭圆曲线方程为

$$y^2 = x^3 + a_2x^2 + a_6 \text{ 或 } y^2 = x^3 + a_4x + a_6$$

2. 椭圆曲线加法规则

设 E 是定义的域 K 上的椭圆曲线

$$E: \{(x,y) \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}$$

定义 E 上的运算法则，记为 $\oplus$：

设 P 和 Q 是 E 上的两个点，L 是过 P 和 Q 的直线（如果 $P = Q$，L 是过 P 点的切线），R 是 L 与曲线 E 相交的第三点。设 L'是过 R 和 O 的直线，则 $P \oplus Q$ 就是 L'与 E 相交的第三点。

E 上的运算法则 $\oplus$ 具有以下性质：

1）如果直线 L 交 E 于点 P，Q 和 R（不必是不同的），则

$$(P \oplus Q) \oplus R = O$$

2）对任意 $P \in E$，$P \oplus O = P$。

3）对任意 $P, Q \in E$，$P \oplus Q = Q \oplus P$。

4）设 $P \in E$，存在一个点，记作 $-P$，使得

$$P \oplus (-P) = O$$

5）对任意 $P, Q, R \in E$，有

$$(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$$

这就是说，E 对于运算法则 $\oplus$ 构成一个交换群。更进一步，如果 E 定义在 K 上，则

$$E(K) := \{(x,y) \in K \times K \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}$$

是 E 的子群。

3. 实数域 R 上的椭圆曲线

因为实数域 R 的特征不为 2、3，所以实数域 R 上椭圆曲线 E 的方程可设为

$$E: y^2 = x^3 + a_4x + a_6$$

其判断式 $\Delta = -16(4a^3 + 27b^2) \neq 0$。这时，$E$ 在 R 上的运算规则为：

设 $P_1 = (x_1, y_1)$，$P_2 = (x_2, y_2)$为曲线 E 上的两个点，O 为无穷远点，则

1）$O + P = P + O$

2）$-P_1 = (x_1, -y_1)$

3) 如果 $R = (x_3, y_3) = P_1 + P_2 \neq O$

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$

其中，$\begin{cases} \lambda = \dfrac{y_2 - y_1}{x_2 - x_1} & x_1 \neq x_2 \\ \lambda = \dfrac{3x_1^2 + a_4}{2y_1} & x_1 = x_2 \end{cases}$

运算法则的几何意义是：设 $P_1 = (x_1, y_1)$，$P_2 = (x_2, y_2)$ 为曲线 E 上的两个点，O 为无穷远点，则 $-P_1$ 为过点 P_1 和点 O 的直线 L 与曲线 E 的交点。换句话说，$-P_1$ 是点 P_1 关于 x 轴的对称点，而点 P_1 与点 P_2 的和 $P_1 + P_2 = P_3 = (x_3, y_3)$ 是过点 P_1 与点 P_2 的直线 L 与曲线 E 的交点关于 x 轴对称点 $P_3 = -R$。

图 3-1 表示当 $P \neq Q$ 时，$P + Q = R$ 的几何解释；图 3-2 表示当 $P = Q$ 时，$P + P = R$ 的几何解释，即倍加。

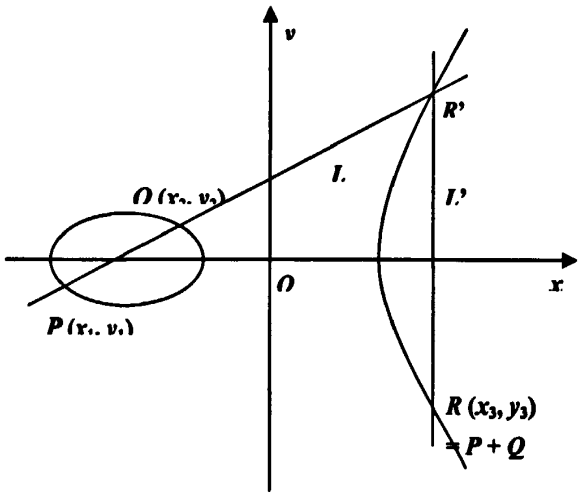

图 3-1　椭圆曲线上点加的几何解释

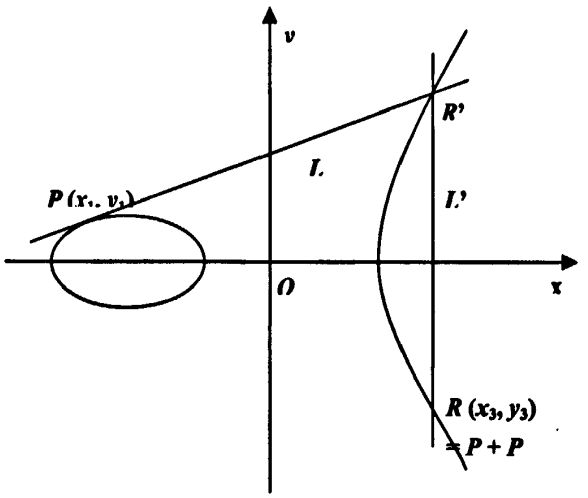

图 3-2　椭圆曲线上倍加的几何解释

4. 素域 Fp (p>3) 上的椭圆曲线 E

因为素域 F_p 的特征不为 2、3，所以素域 F_p 上椭圆曲线 E 的方程可设为

$$E:\ y^2 = x^3 + a_4x + a_6$$

其判断式 $\Delta = -16\,(4a^3 + 27b^2) \neq 0$。这时，$E$ 在 F_p 上的运算规则为：

设 $P_1 = (x_1, y_1)$，$P_2 = (x_2, y_2)$为曲线 E 上的两个点，O 为无穷远点，则

1）$O + P = P + O$

2）$-P_1 = (x_1, -y_1)$

3) 如果 $R = (x_3, y_3) = P_1 + P_2 \neq O$

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$

其中，$\begin{cases} \lambda = \dfrac{y_2 - y_1}{x_2 - x_1} & x_1 \neq x_2 \\ \lambda = \dfrac{3x_1^2 + a_4}{2y_1} & x_1 = x_2 \end{cases}$

5. 建立在有限素数域上的椭圆曲线

有限域上元素的表示方法对椭圆曲线密码系统的可行性、费用和速度都有很大的影响。这个问题很重要，必须强调。但是，有意思的是，某个有限域 F_p 的表示方法并不影响其上的椭圆曲线密码系统的安全性。为了尽可能减小模乘法的运算时间，素数 p 可以是形如 $p = ak - 1$ 的 Mersenne 素数。F_p 的元素为介于 0～p-1 之间的整数，用二进制表示。令 $m = [log_2 p]$ ，$t = m / 32$。

如前所述，F_p 域上的椭圆曲线运算公式如下：

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$

其中，$\begin{cases} \lambda = \dfrac{y_2 - y_1}{x_2 - x_1} & x_1 \neq x_2 \\ \lambda = \dfrac{3x_1^2 + a_4}{2y_1} & x_1 = x_2 \end{cases}$

6. 建立在有限二进制域上的椭圆曲线

$GF(2^m)$ 域上的元素表示 $GF(p)$ 域不同，最简单的元素表示方法为标准基，有限域 F_2^m 可以被看成是 F_2 上的 m 维的向量空间。也就是说，这里存在 F_2^m 上包含 m 个元素的集合 $\{\alpha_0, \alpha_1, ..., \alpha_{m-1}\}$，使得对于任意一个 $\alpha \in F_2^m$，可以唯一地表示成如下的形式：

$$\alpha = \sum_{i=0}^{m-1} a_i \alpha_i,\ a_i \in \{0, 1\}$$

可以将元素 α 表示成二进制向量 $(\alpha_0, \alpha_1, ..., \alpha_{m-1})$。域上的加法可以由逐比特的 XOR 异或计算来实现。

F_2^m 有很多不同的基，下面分别进行介绍。

1) 三项式基（Trinomial bases）

如果 $f(x)$ 是一个 F_2 上的次数为 m 的不可约多项式，那么域 F_2^m 可以表示为 F_2 上的次数小于 m 的多项式的集合。其中，多项式的乘法运算需要进行模 $f(x)$ 运算。也就是说，取 α_i 为 x^i，$0 \le i \le m-1$。这样的表现形式成为多项式基表示方法。

三项式基表示方法是多项式基表示方法的一种，其中多项式 $f(x)$ 具有 $f(x) = x^m + x^k + 1$ 的形式。

这样的表示方法可以简化模 $f(x)$ 运算，有效地提高性能。

2) 最佳正规基（Optimal normal bases）

F_2^m 上的一组正规基可以表示为 $\{\beta, \beta^2, \beta^{2^2}, ..., \beta^{2^{m-1}}\}$，其中 $\beta \in F_2^m$，这样形式的基总是存在的。因为平方运算在 F_2^m 域上是线性运算，所以有

$$\alpha^2 = \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} a_{i-1} \beta^{2^i} = (a_{m-1}, a_0, ..., a_{m-2})$$

因此，一组正规基的表示方法在进行平方运算的时候非常有优势，只要进行

一个简单的循环右移就可以实现。

正规基的另一个优势在于 F_2^m 中元素的平方根也可以很快地求出来。

3) 利用子域（Using subfields）

如果 $m = lr$，其中，l 是一个小整数，如 8 或者 16，那么有限域 F_2^m 可以被看成 F_2^l 上的 r 次扩域。如果 $\{\alpha_0, \alpha_1, \ldots, \alpha_{r-1}\}$ 是上 F_2^l 的 F_2^m 的一组基，那么。对于任意一个元素 $\alpha \in F_2^m$ 可以惟一地确定为

$$\alpha = \sum_{i=0}^{m-1} a_i \alpha_i,\ a_i \in F_2^i$$

F_2^m 的域乘法涉及到 F_2^i 上的一些运算。因为 l 很小，所以 F_2^m 上的计算可以运行得非常快。比如，可以预计算对数或者反对数来提高运算速度。但是这种方法的缺点在于这些表格的空间要求很大。

3.2 安全椭圆曲线

当需要使用椭圆曲线密码系统的时候，首先要建立一条有限域上的椭圆曲线。Schoof 的方法被认为是生成椭圆曲线密码系统的椭圆曲线的一种有效的方法。

1. 计算椭圆曲线上点的个数

设 $K = F_q$ 是 $q = p^n$ 元有限域。设 E 是定义在 F_q 上的椭圆曲线，当 $p > 3$ 时，椭圆曲线方程可写成

$$y^2 = x^3 + a_4 x + a_6$$

易知，F_q 上椭圆曲线 E 中的点数 $\#(E(F_q)) \le 2q + 1$。事实上，对每个 $x \in F_q$，至多有两个 $y \in F_q$ 使得 $P(x, y) \in E$，再加上无穷远点 O，有 $\#(E(F_q)) \le 2|F_q| + 1 = 2q + 1$。

2. 求取椭圆曲线上的点

考虑消息 m ($0 \le m \le M$) 到椭圆曲线 E 上的对应关系，以及椭圆曲线 E 上点的求取。为简单起见，考虑素域 F_q ($p > 3$) 上的椭圆曲线 E。

设素域 F_q 上椭圆曲线 E 的方程为

$$y^2 = x^3 + ax + b$$

其中 $16(4a^3 + 27b^2) \ne 0$。

如何确定 E 上的点 (x_0, y_0)？通常先选定 $x = x_0$，然后求解 $y = y_0$ 使得

$$y^2 = x_0^3 + ax_0 + b \quad (\mathrm{mod}\, p)$$

也就是说，将求取 E 上的点 (x_0, y_0) 转化为求解二次同余方程

$$y^2 = c \quad (\mathrm{mod}\, p)$$

当 c 不能被 p 整除时，方程或者有两个解 $\pm y_0 \bmod p$，或者无解。这样，可以将数字信息 m 与点 $P_m(x_m, y_m)$ 的对应转化为 m 与第一个 x_m 和符号 ± 的对应。

但并不是对所有的 $x = x_0$ 都有 $y = y_0$，使得 (x_0, y_0) 为 E 上的点，所以需要讨论 m 与 x_m 的对应方法。

假设有一个正整数 k 满足 $p > Mk$，使得对任意的 m，$0 \le m \le M-1$，都存在一个整数 j，$1 \le j \le k$，满足

$$y^2 = (mk + j)^3 + a(mk + j) + b \quad (\mathrm{mod}\, p)$$

则可以建立消息 m，$0 \le m \le M$，到椭圆曲线 E 上 $P_m(x_m, y_m)$ 的对应关系。事实上，根据假设条件，对于 m，$0 \le m \le M-1$，可以找到 $x_m = mk + j$，$1 \le j \le k$，使得同余方程

$$y^2 = x_m^3 + ax_m + \mathrm{b} \quad (\mathrm{mod}\, p)$$

有解，取其中的一个解为 y_m，则 $P_m(x_m, y_m)$ 是 m 对应的 E 上的点。反过来，对于点 $P_m(x_m, y_m)$，令 $m = [(x_m - 1)/k]$（其中 $[x]$ 表示数 x 的整数部分），则得到点 P_m 与 m 的对应关系。

显然，上述 k 是存在的，具体取多大，要视所选定的曲线而定。

3. 构造一个有限域上的椭圆曲线

一条安全的椭圆曲线，首先要满足以下几个条件：

1) $\#E(F_q)$ 应该能够被一个足够大的素数 n 整除（例如，$n > 2^{160}$）。
2) $\#E(F_q) \ne q$。
3) $n^x q^k - 1$，$1 \le k \le C$，其中，C 是一个足够大的整数，使得寻找 $F_{q^k}^*$ 上的离散对数在计算上不可行。

随机选取参数 $a, b \in F_q$，满足

$$\begin{cases} 4a^3 + 27b^3 \ne 0, & q = p \\ b \ne 0, & q = 2^m \end{cases}$$

然后可以计算 $u = \#E(F_q)$，且分解 u，判断是否满足安全椭圆曲线的 3 个条件，直到找到为止。针对不同的有限域，这种随机挑选椭圆曲线的方法又可以分为 SEA 和 Satoh 方法。这两种方法针对各自的有限域对随机挑选方法进行了优化，使得在目前椭圆曲线点的计算方面，这两种方法成为应用的主流。

3.3 椭圆曲线密码体制

3.3.1 椭圆曲线上的离散对数问题

椭圆曲线密码系统的安全是基于椭圆曲线离散对数问题（ECDLP）的，椭圆曲线离散对数问题可以描述如下：

给定一条定义于有限域 F_q 上的椭圆曲线 E，n 阶点 $P \in E(F_q)$，

1）对任意整数 l，$0 \le l \le n-1$，计算点 $Q = lP$ 很容易；

2）对于点 Q，求解整数 x，$0 \le x \le n-1$，使得 $xP = Q$ 是很困难的。

3.3.2 椭圆曲线密码算法

将椭圆曲线离散对数问题应用到密码系统中，可以得到椭圆曲线密码算法。

a) 准备工作

选择有限域 F_q、椭圆曲线 E、基点 (x, y)，把明文编码到曲线上的点 $P_m(x_m, y_m)$，即每个明文都对应一个二维点，选择一个私钥 n，计算公钥 $P = n(x, y)$。

对于 A：私钥 n_A，公钥 $P_A = n_A(x, y)$；

对于 B：私钥 n_B，公钥 $P_A = n_B(x, y)$。

b) 加密

对于任何想加密消息并发送给 A 的人，选择一个随机整数 k，生成密文 $C_m = \{ k(x, y), (x_m, y_m) + kP_A \}$。

c) 解密

A 用私钥恢复明文，计算 $n_A(k(x, y))$，计算 $(x_m, y_m) + kP_A - n_A(k(x, y)) = (x_m, y_m) + k(n_A(x, y)) - n_A(k(x, y)) = (x_m, y_m)$

3.3.3 椭圆曲线密码体制的安全性分析

1. 攻击原理

如前所述，椭圆曲线密码体制的安全是基于椭圆曲线离散对数难解问题的。Pohlig-Hellman 算法将确定 l 的问题归约到确定 l 模 n 的每一个素数因子上，减少计算复杂度。目前，对于 ECDLP 的最好算法就是 Pollard Rcho 方法，采用这个方法，经过 $\sqrt{\pi n}/2$ 次椭圆曲线加法以后可以解决 ECDLP 问题。还有人通过将

ECDLP 问题归约到 DLP 问题上来进行算法的改进。

同时，如果有限域 F_p 上的椭圆曲线上椭圆曲线点的个数 $\#E(F_p)=p$，这条曲线就称为素数域不规则（prime-field-anomalous）。理论证明，存在快速有效的方法可以找到这种 $\#E(F_p)$ 的同构，继而解决了 ECDLP 问题。但是这种攻击方法对于其他结构的椭圆曲线并非有效。

这些方法都要求研究者具有相当水平的数学基础。

2. 软件攻击

假设一台每秒能够执行 100 万条指令（MIPS）的计算机每秒钟可以进行 4×10^4 次椭圆曲线加法。也就是说，大约每年可作 2^{40} 次椭圆曲线加法运算。名词“MIPS 年”表示一台运算能力为 MIPS 的计算机一年的计算能力。

表 3-1 表示在不同的 n 下利用 Pollard Rho 方法计算一个离散对数问题所需要的计算能力。举例来说，如果 10000 台计算机中每一台机器都可以达到 1000MIPS 的计算能力，当 $n\approx2^{160}$，那么单独一个椭圆曲线离散对数问题需要用 85000 年来解决。

域规模 / bit	n 的大小	$\sqrt{\pi n}/2$	MIPS 年
163	160	2^{80}	8.5×10^{11}
191	186	2^{93}	7.0×10^{15}
239	234	2^{117}	1.2×10^{23}
359	354	2^{177}	1.3×10^{41}
431	426	2^{213}	8.5×10^{51}

表 3-1　计算能力[27]

3. 硬件攻击

对于一个装备很好的攻击者而言，利用 Pollard Rho 方法并行搜索可以很快地实现对某种特殊用途的硬件进行攻击。根据 1994 年的研究，如果 $n\approx10^{36}\approx2^{120}$，那么耗资 10000 万美元的具有 325000 个处理器的机器破解一个离散对数问题需要花费 35 天。

需要注意的是，以上的分析仅仅用于破解一个端用户的私钥，而真正破解一个系统还要花费同样的时间来破解另一个端用户的私钥。

4. 安全性比较

ECC 的安全性依赖于 ECDLP 的困难性。与 FIP 和 DLP 一样，对 ECDLP 还没有有效的攻击算法，但事实上，ECDLP 是比 IFP 和 DLP 更困难的问题，这使

得 ECC 成为当今强度最高的密码系统。图 3-3 比较了用最好算法对不同模数攻击 ECC 和攻击 RSA 或 DSA 所需的时间（以 MIPS 年为单位），

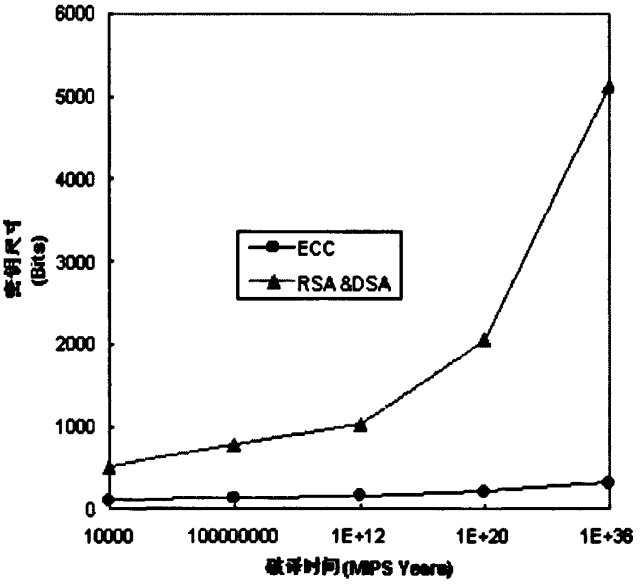

图 3-3　ECC 与 RSA 和 DSA 的安全性比较[26]

从上图可以看出，为获得可信的安全性，RSA 和 DSA 需要 1024 比特的模数，而 160 比特的模数对 ECC 已经足够。不仅如此，ECC 与 RSA 和 DSA 的安全性差别也随着密钥长度的增长而增大。例如，300 比特的 ECC 甚至比 2000 比特的 RSA 或 DSA 安全得多。目前可接受的安全级别是 10^{12}MIPS 年。

3.3.4 椭圆曲线密码体制的有效性分析

公钥密码体制的有效性需考虑 3 个因素：计算开销、密钥长度和带宽，对不同系统的有效性进行比较应基于相同的安全级。我们选择密钥长为 160bit 的 ECC、1024bit 的 RSA 和 DSA，由上一节可知，这些密钥长度为各自系统提供了彼此相当的安全级。

1. 计算开销

RSA 一般选择 $e = 65537 = (2^{16} + 1)$，这样 e 的二进制表达式中只含两个"1"，可大大减少计算量。对于 DSA 和椭圆曲线数字签名算法（ECDSA）或椭圆曲线

加密方案（ECES），大部分数字签名和加密操作能进行预计算。假设一次椭圆曲线加法大约花费 10 次模乘的开销，一次 1024 bit 模乘运算花费一个单元时间，所有应用于离散对数加密系统的预计算技巧同等地应用于椭圆曲线系统中。个系统的计算开销如下表 3-2 所示，其中 q 为 160 bit 长，表中数据表示完成给定操作所需的时间单元数。

项　目	基于 F_q 的 ECDSA 或 ECES	RSA (n = 1024 bit, $e = 2^{16} + 1$)	离散对数系统 / 1024 bit
加密	120	17	480
解密	60	384	240
签名	60	384	240
验签	120	17	480

表 3-2　不采用优化措施时，各系统计算开销的大致比较[26]

ECC 可在很短的时间里产生符合条件的密钥，即使一个计算能力非常有限的智能卡也能产生满足要求的密钥对，其他公钥体制由于产生密钥所需的计算非常复杂，在计算能力受限的情况下很难产生合适的密钥。另外由于 ECC 的基域及其元素表示法能被选择（虽然基于域 F_q 和 F_2^m 的 ECC 在安全性和标准化上没有区别，但在实际的应用上其性能和成本还是有区别的），从而域运算（域加/域乘/域求逆）能被优化，基于离散对数和整数因式分解的公钥密码系统不能做到这一点。

2. 密钥长度

密钥长度决定存储密钥对和系统参数需要的比特数，ECC、RSA/DSA 的系统参数和密钥对长度的比较如表 3-3 所示。可以看出，ECC 所用的密钥对和系统参数比 RSA/DSA 要求的短。

项　目	系数参数 / bit	公钥 / bit	私钥 / bit
RSA	-	1088	2048
DSA	2208	1024	160
ECC	481	161	160

表 3-3　计算能力[26]

3. 带宽

带宽是指传送一个加密消息或一个签名所需传输的比特数。当 3 类公钥系统用于加密或对长消息进行数字签名时，具有相似的带宽要求。当传送短消息时带

宽的要求值得注意，因为公钥密码系统经常用于传送短消息（例如为对称密码系统传送会话密钥）。为了进行具体的比较，假设待签名消息长度为 2000 bit，待加密消息长度为 100 bit，几种情况下签名和加密消息的长度比较如表 3-4、表 3-5 所示。可以看出，当对短消息加密时，ECC 比其他公钥系统节省带宽。而且 ECC 的点压缩技术进一步节省了存储密钥、证书的空间和带宽。

项 目	签名长度 / bit
RSA	1024
DSA	320
ECC	320

表 3-4　对长消息(2000 bit)签名的长度

项 目	加密消息长度 / bit
RSA	1024
ElGamal	2048
ECC	321

表 3-5　对短消息(100 bit)加密的长度

3.3.5 椭圆曲线密码体制的性能总结

综合上述分析，ECC 与其他公钥加密系统相比能提供更好的加密强度、更快的执行速度和更小的密钥长度，因此 ECC 可用较小的开销（所需的计算量、存储量、带宽、软件和硬件实现的规模等）和时延（加密和签字速度高）实现较高的安全性，特别适用于计算能力和集成电路空间受限（如 IC 卡）、带宽受限（如无线通信）要求高速实现的情况。

3.4 椭圆曲线密码体制的应用

3.4.1 椭圆曲线 Diffee-Hellman 密钥交换

1994 年，Scheidler、Buchmann、Williams 利用一个非群的结构，也就是实二次数域，来实现 Diffie-Hellman 密钥交换（ECDH）协议。

1）用户 A、B 已经商定好的参数：E 表示椭圆曲线，n 表示椭圆曲线群上基点 P 的阶，F_q 表示有限域。

2）用户 A 随机选择 $d_A \in [1, n-1]$，计算 $Q_A = d_A P$，发送 Q_A 给 B。

3）用户 B 随机选择 $d_B \in [1, n-1]$，计算 $Q_B = d_B P$，发送 Q_B 给 A。

4）用户 B 计算 $K_B = d_B Q_A$，用户 A 计算 $K_A = d_A Q_B$。

5）得到的密钥为：$K = d_A d_B P = K_A = K_B$

宽的要求值得注意，因为公钥密码系统经常用于传送短消息（例如为对称密码系统传送会话密钥）。为了进行具体的比较，假设待签名消息长度为 2000 bit，待加密消息长度为 100 bit，几种情况下签名和加密消息的长度比较如表 3-4、表 3-5 所示。可以看出，当对短消息加密时，ECC 比其他公钥系统节省带宽。而且 ECC 的点压缩技术进一步节省了存储密钥、证书的空间和带宽。

项 目	签名长度 / bit
RSA	1024
DSA	320
ECC	320

表 3-4　对长消息(2000 bit)签名的长度

项 目	加密消息长度 / bit
RSA	1024
ElGamal	2048
ECC	321

表 3-5　对短消息(100 bit)加密的长度

3.3.5 椭圆曲线密码体制的性能总结

综合上述分析，ECC 与其他公钥加密系统相比能提供更好的加密强度、更快的执行速度和更小的密钥长度，因此 ECC 可用较小的开销（所需的计算量、存储量、带宽、软件和硬件实现的规模等）和时延（加密和签字速度高）实现较高的安全性，特别适用于计算能力和集成电路空间受限（如 IC 卡）、带宽受限（如无线通信）要求高速实现的情况。

3.4 椭圆曲线密码体制的应用

3.4.1 椭圆曲线 Diffee-Hellman 密钥交换

1994 年，Scheidler、Buchmann、Williams 利用一个非群的结构，也就是实二次数域，来实现 Diffie-Hellman 密钥交换（ECDH）协议。

1）用户 A、B 已经商定好的参数：E 表示椭圆曲线，n 表示椭圆曲线群上基点 P 的阶，F_q 表示有限域。

2）用户 A 随机选择 $d_A \in [1, n-1]$，计算 $Q_A = d_A P$，发送 Q_A 给 B。

3）用户 B 随机选择 $d_B \in [1, n-1]$，计算 $Q_B = d_B P$，发送 Q_B 给 A。

4）用户 B 计算 $K_B = d_B Q_A$，用户 A 计算 $K_A = d_A Q_B$。

5）得到的密钥为：$K = d_A d_B P = K_A = K_B$

3.4.2 椭圆曲线数字签名算法

椭圆曲线数字签名算法（ECDSA）是对美国政府数字签名算法（DSA）在椭圆曲线上的模拟。ECDSA 已经为 ANSI 标准、ISO 标准和 IEEEP1363 标准所接纳。

1. ECDSA 密钥生成

E 是有限域 F_q 上的椭圆曲线，P 是椭圆曲线 $E(F_q)$ 上阶位 n 的一点，这些都是系统参数。为了描述方便，假设 q 就是一个素数，尽管在实际应用中 q 还可以是素数的幂，但是这种假设不失一般性。

那么，对于每一个端用户 A 来说，密钥生成的步骤是：

1）随机选择整数 $d \in [1, n-1]$；

2）计算 $Q = dP$；

3）用户 A 的公钥为 Q，私钥为 d。

2. ECDSA 签名生成

为了对消息 m 进行签名，用户 A 需要：

1）随机选择整数 $k \in [1, n-1]$；

2）计算 $kP = (x, y)$，$r = x_1 \bmod n$；

3）其中，x_1 需要介于 0 和 $q-1$ 之间。如果 $r = 0$，退回到第一步；

4）计算 $k^{-1} \bmod n$；

5）计算 $s = k^{-1}\{h(m) + dr\} \bmod n$，其中，$h$ 是安全哈希函数 (SHA-1)。如果 $s = 0$，退回到第一步。

6）消息 m 签名就是这样的一对整数 (r, s)。

3. ECDSA 签名验证

为了验证用户 A 关于 m 的签名 (r, s)，用户 B 需要：

1）得到用户 A 的公钥 Q 的可靠副本；

2）验证 r 和 s 世界与 1 和 $n-1$ 之间的整数；

3）计算 $w = s^{-1} \bmod n$ 和 $h(m)$；

4）计算 $u_1 = h(m)\, w \bmod n$ 和 $u_2 = rw \bmod n$；

5）计算 $u_1 P + u_2 Q = (x_0, y_0)$ 和 $v = x_0 \bmod n$；

6）当且仅当 $v = r$ 时，接受这个签名。

DSA 和 ECDSA 之间最大的区别在于 r 的生成。DSA 通过随机取得元素 $\alpha^k \bmod p$，然后将其进行模 p 运算来达到同样的目的。

在 DSA 中，q 是一个 160bit 的 $p-1$ 的素因子，α 是域 F_q^* 上的 q 阶元素。

ECDSA 是对点 kP 的 x 坐标进行模 n 运算。为了达到和 DSA 一样的安全强度，参数 n 需要有 160bit 长，这样的话，DSA 和 ECDSA 具有相同的比特长度 320bit。

如果通信的两端都仅仅固定有限域 F_q 而没有系统参数，那么每个端系统都需要有自己的椭圆曲线 E 和点 $P \in E(F_q)$。在这种情况下，椭圆曲线 E 的定义、点 P、点 P 的阶 n 都需要包含在端系统的公钥中。

3.5 J2ME 环境下椭圆曲线密码算法的优化设计

3.5.1 Bouncy Castle 中的椭圆曲线密码算法

Bouncy Castle 是一种用于 Java 平台的开放源码的轻量级密码术包。它支持大量的密码术算法，并提供 JCE 1.2.1 的实现。因为 Bouncy Castle 被设计成轻量级，所以从 J2SE 1.4 到 J2ME（包括 MIDP）平台，它都可以运行。它是在 MIDP 上运行的唯一完整的密码术包。

在 Bouncy Castle 中，我们可以使用椭圆曲线密码算法进行加解密：

```
ECCurve.Fp curve = new ECCurve.Fp(
    new BigInteger
        ("6277101735386680763835789423207666416083908700390324961279"), // q
    new BigInteger
        ("fffffffffffffffffffffffffffffefffffffffffffffc", 16), // a
    new BigInteger
        ("64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1", 16)); // b

ECDomainParameters params = new ECDomainParameters(
    curve,
    curve.decodePoint
        (Hex.decode("03188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012")), // G
    new BigInteger(
```

```
        "6277101735386680763835789423176059013767194773182842284081")); // n

ECPrivateKeyParameters priKey = new ECPrivateKeyParameters(
    new BigInteger(
        "651056770906015076056810763456358567190100156695615665659"), // d
        params);

ECPublicKeyParameters pubKey = new ECPublicKeyParameters(
    curve.decodePoint
        (Hex.decode("0262b12d60690cdcf330babab6e69763b471f994dd702d16a5")), // Q
    params);

AsymmetricCipherKeyPair[] keyPairs = new AsymmetricCipherKeyPair[2];
keyPairs[0] = new AsymmetricCipherKeyPair(pubKey, priKey);
keyPairs[1] = new AsymmetricCipherKeyPair(pubKey, priKey);

IESEngine eng = new IESEngine(new ECDHBasicAgreement(),
        new KDF2BytesGenerator(new SHA1Digest()),
        new HMac(new SHA1Digest()));

byte[] d = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };
byte[] e = new byte[] { 8, 7, 6, 5, 4, 3, 2, 1 };
IESParameters p = new IESParameters(d, e, 64);

// 加密
eng.init(true, keyPairs[0].getPrivate(), keyPairs[1].getPublic(), p);
byte[] cipherText = eng.processBlock(message, 0, message.length);

// 解密
eng.init(false, keyPairs[1].getPrivate(), keyPairs[0].getPublic(), p);
byte[] plainText = eng.processBlock(cipherText, 0, cipherText.length);
```

在 Bouncy Castle 中，我们还可以使用椭圆曲线密码算法进行数字签名：

```
byte[] kData = BigIntegers.asUnsignedByteArray(new BigInteger(
    "61405070670650010630650655656674055600061615565656656656654"));

SecureRandom k = new FixedSecureRandom(kData);

ECCurve.Fp curve = new ECCurve.Fp(
    new BigInteger(
        "6277101735386680763835789423207666416083908700390324961279"), // q
    new BigInteger(
        "fffffffffffffffffffffffffffffffefffffffffffffffc", 16), // a
    new BigInteger(
        "64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1", 16)); // b

ECDomainParameters params = new ECDomainParameters(
    curve,
    curve.decodePoint
        (Hex.decode("03188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012")), // G
    new BigInteger(
        "6277101735386680763835789423176059013767194773182842284081")); // n

ECPrivateKeyParameters priKey = new ECPrivateKeyParameters(
    new BigInteger(
        "651056770906015076056810763456358567190100156695615665659"), // d
        params);

ParametersWithRandom param = new ParametersWithRandom(priKey, k);

// 签名
ECDSASigner ecdsa = new ECDSASigner();
ecdsa.init(true, param);
BigInteger[] sig = ecdsa.generateSignature(message);
```

3.5.2 Bouncy Castle 中算法的局限性

通过对开放源码的分析，在 Bouncy Castle 包中，椭圆曲线加密算法的具体实现机制如下所示：

基于 F_p 上的同余方程：

$$y^2 + xy = x^3 + ax + b \quad (\mathrm{mod}\, p)$$

上所产生的点群，即方程的所有解（x, y）$\in F_p \times F_p$，连同一个特殊点 O 即无穷点共同构成 F_p 上的椭圆曲线点群，记为 $E_p(a, b)$，其中 $a, b \in F_p$ 是满足 $(4a^3 + 27b^2) \pmod p \neq 0$ 的常量，p 是素数。

在 Bouncy Castle 中，此方程通过类 ECCurve.Fp 来实现：

```
ECCurve.Fp curve = new ECCurve.Fp (q, a, b);
```

其中 a, b, q 都是大整数，以保证椭圆曲线加密算法的安全性。

这里，定义一个大整数 b 如下所示：

```
BigInteger b = new BigInteger
        ("64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1", 16);
```

Bouncy Castle 包中，椭圆曲线加密算法[29]的加密运算和签名运算都依赖于椭圆曲线上的点在 $E_p(a, b)$ 上的运算。$E_p(a, b)$ 上的运算主要包括点的加法运算，点的乘法运算（即椭圆曲线上的点和一个大整数相乘）和点的逆运算。无论是加密运算还是签名运算，其时间开销主要由点的乘法运算的时间开销来决定。在 Bouncy Castle 中，点的乘法运算中使用的大整数乘法运算是在实数域上实现的，运算效率很低。比如，一个 160 比特的大数平方，可能得到一个 320 位比特的大数，这就直接增加了算法的空间复杂度。另外，Bouncy Castle 包提供的 190 位密钥的 ECC 算法在 J2ME 环境中的签名时间超过了一个小时，这么长的时间对于用户来说是不可承受的。

因此需要改进 Bouncy Castle 包中 ECC 算法的性能，使其真正适合于 MIDP 环境中的各种安全应用。

3.5.3 椭圆曲线密码算法的优化设计

目前对椭圆曲线加密算法的各种理论研究已经相当成熟，包括适合加密算法的曲线类型，各种运算域安全性，运算速度的比较等等。在 J2ME 受限环境中可以充分利用已有的理论成果，在不降低安全性的条件下，最大限度地降低算法的时间复杂度和空间复杂度。

1. 运算域的选择

基于 J2ME 的智能终端平台的受限性是选择安全椭圆曲线和有限域的重要考虑因素。因此，在不降低安全性的前提下，应该最大限度地降低算法的时间复杂度和空间复杂度。由于椭圆曲线加密算法的时间开销主要取决于在某一有限域上的点加和点乘运算，因此选择一个合适的有限域，对于从根本上降低该算法的时间复杂度和空间复杂度至关重要。

有两种有限域可供椭圆曲线密码体制选择，一种是简单数域 F_P，另一种是多项式域 $GF(2^m)$，它们均被纳入 ECC 标准。基于两种域的安全性相同，但其应用性能和实现成本有所不同。在一般软件环境中，如 Sun SPARC，HP Server 及低成本的 8 位 Card，选择 $GF(2^m)$ 能提供更好的性能。如果数学处理器支持带模指数运算，F_P 性能可得到提高，在某些情况下可超过 $GF(2^m)$，如 Pentium 系统。

由于手机上的 J2ME 运行环境资源有限，加上椭圆曲线密码体制的安全特点，本文选择有限域 $GF(2^m)$ 实现椭圆曲线密码体制。由 3.1 节所述可知，有限域 $GF(2^m)$ 上的运算法则非常适合椭圆曲线加密的大数处理，比如，在 $GF(2^m)$ 的最佳标准基上，两个 160 位比特的大整数的加和减运算只是对应位的异或，平方只需循环左移一位。在 J2ME 设备的处理器上，都只需要一个时钟周期就可以完成。因此在这里我们选择 $GF(2^m)$ 作为有限域。

2. 曲线类型的选择

有限域上的椭圆曲线资源非常丰富，但是并非所有的曲线都适合加密体系，这里我们选择了 $GF(2^m)$ 上的一类非超奇异椭圆曲线 E，其方程形式为：

$$y^2 + xy = x^3 + ax^2 + b$$

式中 $a, b \in GF(2^m)$，$b \neq 0$

在基于 J2ME 的智能终端平台中，可以根据方程定义令 $a = 0$，这不影响算法的安全性，但却简化了运算公式，也降低了算法的复杂度，

因此方程变为：

$$y^2 + xy = x^3 + b$$

其中 $b \in GF(2^m)$，$b \neq 0$

3. 伪随机数的产生

随机序列的目的在于产生某种密钥值或与密钥值有关的值。基本要求是：即使已知该序列许多以前的值，但对于下一个要产生的值仍是不可预测的。值得指出的是：使用同一个“种子”是不明智的，因此必须考虑种子的来源，它应来源

于可靠的随机过程，如抛硬币或随机噪声源等。

为保证更好的安全性，我们应使用多种不同的随机源。本文在实现时，是以时间为种子，对于指定长度的二进制数进行随机的置位与复位，直到满足给定要求为止。

4. 素数的测试

公钥密码体制的基本原理是大素数原理。怎样产生符合要求的安全素数，是公钥密码体制实现的关键。这里使用米勒-勒宾（Miller-Rabin）概率测试法，其基本思想是：

已知 $n-1=2^j m$

(1) 在{ 1, 2, ..., n - 1}中均匀地产生一随机数 b；

$j \leftarrow 0$，$z \equiv b^m \pmod n$

若 $z=0$ 或 $z=n-1$，则 n 通过测试，结束；

(2) 若 $j=1$，则 n 为非素数，结束。否则转(3)；

(3) $j \leftarrow j+1$，$z \leftarrow z^2 \pmod n$

若 $z=-1$ 则通过测试，结束。否则转(2)。

5. 构造最佳标准基上的点乘法器

椭圆曲线密码系统最基本的运算是标量乘运算，它是椭圆曲线密码系统中最耗时的运算。如何高效地实现标量乘运算，一直是椭圆曲线密码系统的一个研究重点。在 $GF(2^m)$ 域上，椭圆曲线上点的乘法和逆元运算是加法和平方运算的混合，点的乘法运算占据了绝大部分的时间开销。

$GF(2^m)$ 域上的乘法器构造方法有，IEEE-P1363 给出的“addition-subtraction”算法，二进制算法（比如 k-ary 算法），有符号窗口算法 (signed window method) 等等。

本文采用了优化的 Montgomery 乘法[30]。

算法输入：大整数 $k \geq 0$ 和椭圆点 $P=(x, y) \in E(F_p)$，

算法输出：$Q=kP$

算法描述：

声明两个点 P_1、P_2，并且用 P 和 $2P$ 分别对其进行初始化；

$(k_{l-1}k_{l-2}\cdots k_1k_0)_2$ 为 k 的二进制序列；

for i from $l-2$ down to 0 do

if $k_i=1$ then

$P_1=P_1+P_2$，$P_2=2P_2$

else

$P_2 = P_1 + P_2$，$P_1 = 2P_1$

return P_1

3.5.4 具体实现

在 J2ME 环境中，我们用以下类实现基于有限域 $GF(2^m)$ 的椭圆曲线加密算法：

Fp 类：在有限域 Fp 中，实现椭圆曲线的加、减、乘和逆元运算。

F2m 类：继承自 Fp 类，定义椭圆曲线 $y^2 + xy = x^3 + b$。

ECPoint 类：定义有限域 Fp 中椭圆曲线上的点。

ECPointF2m 类：继承自 ECPoint 类，定义椭圆曲线 $y^2 + xy = x^3 + b$ 上的点。

CLDC 的类库可以分为两种：一种是从 J2SE 标准类库中继承的；另一种是专门为 CLDC 设计的。对于第一种 CLDC 类库，包括了 J2SE 的 3 个最核心的包 java.io，java.lang 和 java.util，且这 3 个包和 J2SE 相比，也只是其相应包的一个很小的子集。所以，在 J2ME 中实现椭圆曲线密码体制时必须提供一些基础类的实现，包括对安全随机数接口 SecurityRandom 和大整数类 BigInteger 的实现。

3.5.5 实现中的进一步优化

对于 J2ME 应用程序（MIDlets）来说，实现椭圆曲线密码算法还需考虑两个很重要的因素：性能与代码尺寸。在我们的实际工作中，对于这两方面的优化也使用了一些不错的技巧。

1. 性能

各种性能优化的有效与否往往都取决于底层虚拟机或者说 JRE 的性能。如今也已经有很多通用的技巧可以用来改善 Java 程序的性能，很多论文或书籍中都总结了不少这方面的内容。对于 J2ME 而言，其中的一些技巧同样有效，在这里就介绍一些我们在实现椭圆曲线密码算法过程中使用的比较有效的技巧。

最方便的优化方法是使用 Java 编译器的优化开关（Sun 的 javac 中为 -O）。这种编译器优化既可以提高性能，又可以减小代码尺寸。

优化中另一个很重要的步骤是 Application Profiling。Profiling 可以识别出那

些经常被调用并值得被优化的方法。有很多工具可被用来进行 Java 应用程序的 profiling，如 Sun 的 JRE 中就有一个开关-prof 可用来打开或关闭 profiling。这里还应考虑那些常见的经典优化方法，如对象回收，避免串连 String，短方法的内联以及用本地方法来代替短小的类库方法（如 math.max）。

实现公钥算法时往往需要处理多字节的数值或者查找数据表，我们通常会使用 Java 提供的数组，但值得注意的是，数组的访问常常意味着索引范围检查所带来的一定开销。如果数组元素为静态数值，即该数组的元素均为常数，那么我们应该将数组分开为一系列独立的变量。应用程序中被广泛使用的变量一定要声明为 static final，这样可以获得更好的性能并降低代码大小。

另一个问题就是冗长的初始化（initialization）代码。这样的初始化可以转移到类的静态初始化区域，在启动时被载入。这样，就可以通过牺牲一定的 MIDlet 加载时间来获得更好的执行性能。还有种方法，把初始化放在一个单独的后台线程。如果 MIDlet 需要等待用户输入，那么这种方法将非常有效，因为初始化可以在这段等待时间中进行。在椭圆曲线密码算法的应用中，这种方法可以极大地提升 MIDlet 的性能。

2. 代码尺寸

减小代码尺寸的第一步就是去除那些未被使用的类。如果所有的 MIDlet 类均为开发者所写，那么他就可以根据自己的特殊要求来做到这点，这样代码就可以被保持在相对较小的尺寸。但如果我们要使用一个现成的加密算法类库，那么就会有很多未被使用的类。如今有不少工具可以用来分析 MIDlets 并去除那些没有必要的类，如 ProGuard。

大多数 J2ME 设备都要求类文件与附加的资源（如图片）被放在一个 JAR（Java Archive）文件中。JAR 文件遵循被广泛使用的 ZIP 格式规范，并另外提供了一些附加特性。最重要的是 JAR 文件能对一堆文件进行压缩。压缩是可选的，但是对于 MIDlets 来说是必须要使用的。

还有种减小代码尺寸的方法是混淆（obfuscation）。一般而言，混淆被用于防止 Java 类文件的反编译。混淆工具通常会用较短的名称来替代变量名、方法名和类名，删除调试信息并压缩常数。这些措施可以使字节码（bytecode）更小。但是混淆有时也会导致程序不能正常运行，更严重的是，类的重命名会使其不能被显式动态加载。当然，混淆工具通常可以被配置为只执行部分代码的修改，但我们建议一定要谨慎选择那些需要被混淆的代码部分。前文所提的 ProGuard 同样也是一个很不错的混淆工具。

3.6 对优化后算法的测试分析

3.6.1 可靠性分析

输入明文，对优化后的椭圆曲线加密算法进行加密解密测试，可以发现优化后的加密解密算法仍能得到正确的结果，并未影响原算法的可靠性。

3.6.2 性能分析

我们利用 Bouncy Castle 包提供的椭圆曲线密码算法进行测试，使用密钥长度为 174 位的 ECC（比密钥长度为 512 位的 RSA 算法的安全性要高）在 J2ME Wireless Toolkit 2.5.2 和普通手机、智能手机上分别进行了测试。表 3-6 给出了具体的实验数据（这里取的是二十次测试数据的平均值）。

	加密（msec）	解密（msec）	签名（msec）
WTK 2.5.2	6037	6054	6098
Nokia 6300	6824	6850	6885
SE K790c	7306	7367	7412
Nokia N73	6212	6231	6272
Motorola E6	6108	6145	6168

表 3-6　Bouncy Castle 包的 ECC 运行时间

注：Wireless Toolkit（WTK）是 Sun 提供的无线开发工具包，是 J2ME 开发应用程序事实上的标准工具包，运行于 Win32 平台，可作为 J2ME 应用程序的模拟器。

普通手机具有相对较弱的 CPU 与较小的系统内存，这里我们选择了 Nokia 主流 S40 手机 Nokia 6300 和 Sony Ericcson 手机 K790c。

智能手机具有相对较强的 CPU 与较大的系统内存，这里我们选择了 Nokia 主流 S60 手机 Nokia N73 和 Motorola 主流 Linux 手机 E6。

在采用优化设计后，同样使用密钥长度为 174 位的 ECC 在 Wireless Toolkit 2.5.2 和普通手机、智能手机上分别进行了测试。表 3-7 给出了具体的实验数据（这里取的是二十次测试数据的平均值）。

3.6 对优化后算法的测试分析

3.6.1 可靠性分析

输入明文，对优化后的椭圆曲线加密算法进行加密解密测试，可以发现优化后的加密解密算法仍能得到正确的结果，并未影响原算法的可靠性。

3.6.2 性能分析

我们利用 Bouncy Castle 包提供的椭圆曲线密码算法进行测试，使用密钥长度为 174 位的 ECC（比密钥长度为 512 位的 RSA 算法的安全性要高）在 J2ME Wireless Toolkit 2.5.2 和普通手机、智能手机上分别进行了测试。表 3-6 给出了具体的实验数据（这里取的是二十次测试数据的平均值）。

	加密（msec）	解密（msec）	签名（msec）
WTK 2.5.2	6037	6054	6098
Nokia 6300	6824	6850	6885
SE K790c	7306	7367	7412
Nokia N73	6212	6231	6272
Motorola E6	6108	6145	6168

表 3-6　Bouncy Castle 包的 ECC 运行时间

注：Wireless Toolkit（WTK）是 Sun 提供的无线开发工具包，是 J2ME 开发应用程序事实上的标准工具包，运行于 Win32 平台，可作为 J2ME 应用程序的模拟器。

普通手机具有相对较弱的 CPU 与较小的系统内存，这里我们选择了 Nokia 主流 S40 手机 Nokia 6300 和 Sony Ericcson 手机 K790c。

智能手机具有相对较强的 CPU 与较大的系统内存，这里我们选择了 Nokia 主流 S60 手机 Nokia N73 和 Motorola 主流 Linux 手机 E6。

在采用优化设计后，同样使用密钥长度为 174 位的 ECC 在 Wireless Toolkit 2.5.2 和普通手机、智能手机上分别进行了测试。表 3-7 给出了具体的实验数据（这里取的是二十次测试数据的平均值）。

	加密（msec）	解密（msec）	签名（msec）
WTK 2.5.2	2903	2934	2988
Nokia 6300	3840	3853	3865
SE K790c	3923	3967	4002
Nokia N73	3231	3247	3273
Motorola E6	3012	3056	3087

表 3-7　优化后的 ECC 运行时间

从实验数据来看，优化后的椭圆曲线密码算法在 J2ME 平台上运行的性能与使用 Bouncy Castle 包相比提高了近一倍，而所耗用的时间也在可被用户所接受范围内。

考虑到此处椭圆曲线密码算法的实现没有考虑硬件对算法的优化，因此在速度和使用资源方面，没能做到最优，仍有可提高余地。但这需要根据不同的平台进行修改和优化，因此可以作为本文下一步的研究方向。

第四章 移动支付模块的设计与实现

4.1 移动支付模块的设计

移动支付系统涉及到银行的参与，且交易额一般都比较大，所以其安全性要求都比较高。移动支付系统受无线信道、手持设备自身特有因素等的影响，安全性相比有线环境和 PC 机来说不容易保证，这也是制约移动支付得以普遍实施的重要原因。因此，确保移动支付系统的安全性尤其是对客户端 (Client) 即手持设备的认证以及重要信息如银行卡卡号、密码等信息的保密是移动支付系统得以实施的关键所在。本章重点介绍移动支付模块的整体设计方案，而下一章则将结合一个具体的系统实例来介绍移动电子商务系统的开发。

4.1.1 移动支付模块安全模型设计

本文采用的移动支付模块的总体框架如图 4-1 所示，整个移动支付模块分为几个模块：消息处理模块、数据库访问模块等，同时提供两个接口分别与用户界面模块及网络通信模块相连。

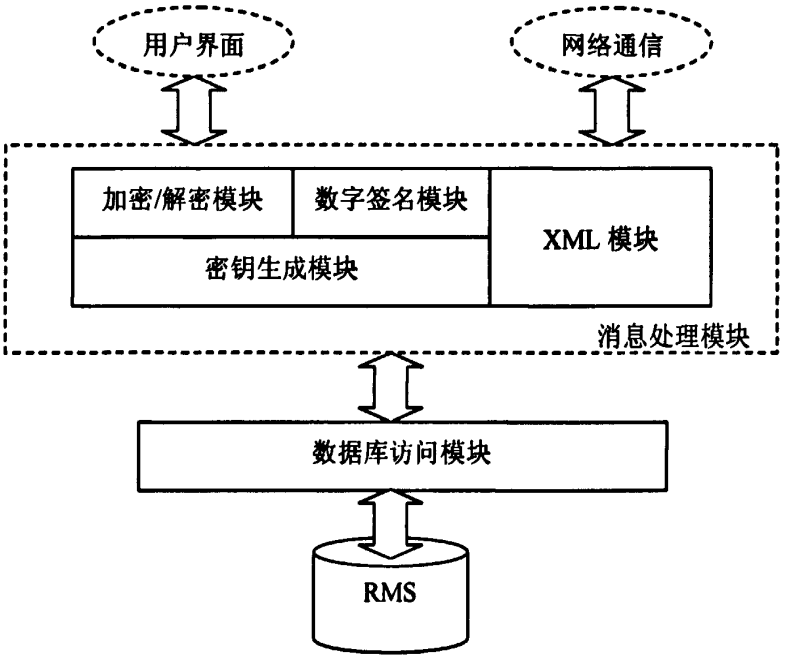

图 4-1　移动支付模块的总体框架

第四章 移动支付模块的设计与实现

4.1 移动支付模块的设计

移动支付系统涉及到银行的参与，且交易额一般都比较大，所以其安全性要求都比较高。移动支付系统受无线信道、手持设备自身特有因素等的影响，安全性相比有线环境和 PC 机来说不容易保证，这也是制约移动支付得以普遍实施的重要原因。因此，确保移动支付系统的安全性尤其是对客户端 (Client) 即手持设备的认证以及重要信息如银行卡卡号、密码等信息的保密是移动支付系统得以实施的关键所在。本章重点介绍移动支付模块的整体设计方案，而下一章则将结合一个具体的系统实例来介绍移动电子商务系统的开发。

4.1.1 移动支付模块安全模型设计

本文采用的移动支付模块的总体框架如图 4-1 所示，整个移动支付模块分为几个模块：消息处理模块、数据库访问模块等，同时提供两个接口分别与用户界面模块及网络通信模块相连。

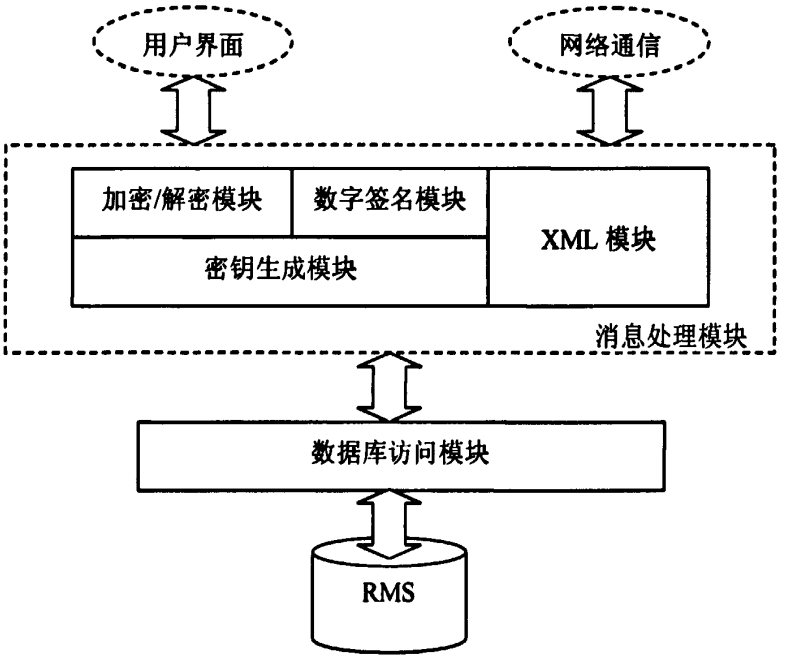

图 4-1　移动支付模块的总体框架

这里消息处理模块由几个子模块组成。其中 XML 模块用于传送 XML 格式的消息，从而更容易扩展；ECC 相关模块包括密钥生成、加密解密和数字签名，这些模块必须自己实现。

数据库访问模块负责调用 J2ME 的 RMS 数据库的功能接口，对用户界面模块的个性化设置，数字签名用的私钥和公钥对，网络通信模块用的 H'ITP 访问地址和设置等数据进行存取，而其它模块则通过访问数据库模块存取所需数据。

整个支付模块的流程为：

a) 从用户界面模块获得银行卡及密码信息。
b) 用户的银行卡信息经过加密传送给数字签名模块。
c) 数字签名模块对加密的信息进行签名确保信息在传输过程中的完整性和不可否认性。
d) 经过签名的信息发送给网络通信模块以与服务器端进行交互。
e) 由网络通信模块获取相应的支付成功或失败信息。

为了能实现终端用户在任何时候（在线或离线）都能对自己的历史交易进行查询，整个支付过程中其他模块都跟数据库访问模块相互通信。

4.1.2 移动支付模块安全设计方案

由于移动支付涉及到银行的参与，且交易额较大，因此对安全性要求比较高。特别是无线环境中的安全，它受信道、手持设备等本身特有因素的影响，因此安全性不容易保证，确保无线环境中的安全性特别是对用户即手持设备的认证是移动支付得以实施的关键。

本节的安全方案是针对移动支付而设计的。重点解决移动支付中端到端的认证问题，特别是对用户的认证。

该安全方案的设计目标是在不改变用户硬件、不改变底层通信协议如 GSM、CDMA 等的基础上通过应用层的解决方案来保证端到端之间能够认证对方，重点是对用户即手持设备的认证。

当通过用户界面模块获得客户（即终端用户）的银行卡及密码信息后，移动支付模块会对银行卡以及密码信息进行加密，以防止重要的信息被窃取而造成客户一定的损失，然后再对加密后的信息进行签名后通过网络通信模块建立 SSL 安全通道发送给服务器端，服务器端随后作相应的处理，或返回支付成功信息给

客户端，或返回具体的错误信息。

在移动支付模块的信息加密以及认证签名中分别用到了前面所介绍的椭圆曲线加密算法和数字签名技术，保证了数据的机密性、完整性以及客户的认证，有效地防止了信息被窃取和篡改。

4.2 移动支付模块的具体实现

本文的移动支付模块主要由以下几个模块组成：密钥生成模块、加密解密模块、数字签名模块、XML 模块、数据库访问模块。

下面对各个模块的主要功能及具体实现分别介绍：

4.2.1 密钥生成模块

密钥生成模块采用 IEEE-P1363 和 ANSl X9.63 标准推荐的 Ellipitc Curve Diffie-Hellman 算法（如 3.4.1 节所述），每个用户根据自己的私钥与对方的公钥来计算公用的密钥，在后面的对称加密中将使用此公钥。

```
public class ECC {
    ...
    public static Fp ECSVDP_DH (ECDomainParaeters params, BigInteger s, ECPoint Wi) {
        ECPoint P = params.E.mul(s, Wi);
        if (P.isZero())
            throw (new Runtime Exception("ECSVDP_DH: P is zero"));
        return P.x;
    }

  publics static BigInteger ECKAS_DH1 (ECDomainParameters params, BigInteger s,
          ECPoint Wi, int[] P) {
        Fp z = ECSVDP_DH(params, s, Wi);
        int[] Z = Utils.FE2OSP(z);
        int[] k = KDF2(Z, 16, P);   //128 比特
        BigInteger K = Utils.OS2IP(k);
        return K;
```

```
    }
    ...
}
```

4.2.2 加密/解密模块

加密模块采用 IEEE P1363a 和 ANSI X9.63 标准的 ECIES (Ellipitc Curve Integrated Encryption Scheme) 。ECIES 同时支持基于椭圆曲线公钥体制的非对称加密和采用 SHA-1 哈希算法进行数组签名、采用 AES 的对称加密。其中每个 ECIES 对象包含了三个信息：EC 公钥 V，密文 C 和数字签名信息 T。

假设用户 A（私钥为 u，公钥 V）向用户 B（公钥为 W）发送消息 M：

a) 通过 ECDH 算法建立会话密钥 Z；
b) 利用 IEEE P1363 标准中的算法，把会话密钥转换为八进制的字符串 z；
c) 由字符串 z 生成 256 比特的哈希消息，前 128 比特的数据 K_1 作为对称加密的密钥，后面 128 比特 K_2 作为 MAC 密钥；
d) 使用 AES 算法和密钥 K_1 对消息 M 进行加密，获得密文 C；
e) 利用 SHA-1 算法和密钥 K_2，获得 MAC T。

用户 B 收到用户 A 的消息，不仅可以根据自己的私钥对消息进行解密，同时可以验证消息的数字签名：

a) 通过 ECDH 算法获得会话密钥 Z；
b) 利用 IEEE P1363 标准中的算法，把会话密钥转换为八进制的字符串 z；
c) 由字符串 z 生成 256 比特的哈希消息，前 128 比特的数据 K_1 作为对称加密的密钥，后面 128 比特 K_2 作为 MAC 密钥；
d) 使用 AES 算法和密钥 K_1 对密文 C 进行解密，获得明文 M；
e) 利用 SHA-1 算法和密钥 K_2，验证 MAC T。

```
public class ECIES {
    public ECPubKey V;
    public int[] C;
    public int[] ;
    ...
    public ECIES (ECPrivKay u, ECPubKey W, byte[] p) {
        int[] P1 = new int[0];
```

```
        int[] P2 = new int[0];
        Fp z = ECC.ECSVDP_DH(u.params, u.s, W.W);
        int[] Z = Utils.FE2OSP(z);
        int[] K = ECC.KDF2(Z, 32, P1);   //256 比特
        int[] K1 = new int[16];   //128 比特加密密钥
        int[] K2 = new int[16];   //128 比特数字签名密钥
        for (int i = 0; i < K1.length; i++) {
            K1[i] = K[i];
            K2[i] = K[i + K1.length];
        }
        V = new ECPubKey(u);
        C = ECC.AES_CBC_IV0_Encrypt(K1, Utils.toIntArray(p), 128);
        T = ECC.MAC1(K2, Utils.concatenate(C, P2));
    }

    public byte decrypt (ECPrivKay s, ECPubKey V, int[] C, int[] T)
            throws Exception {
        int[] P1 = new int[0];
        int[] P2 = new int[0];
        Fp z = ECC.ECSVDP_DH(s.params, s.s, V.W);
        int[] Z = Utils.FE2OSP(z);
        int[] K = ECC.KDF2(Z, 32, P1);   //256 比特
        int[] K1 = new int[16];   //128 比特加密密钥
        int[] K2 = new int[16];   //128 比特数字签名密钥
        for (int i = 0; i < K1.length; i++) {
            K1[i] = K[i];
            K2[i] = K[i + K1.length];
        }
        int[] M = ECC.AES_CBC_IV0_Encrypt(K1, C, 128);
        if (!Utils.compare(T,   ECC.MAC1(K2, Utils.concatenate(C, P2)))) {
            throw new Exception("ECIES: Authentication Tag Invalid");
        }
        return Utils.toByteArray(M);
    }
```

```
    ...
}
```

4.2.3 数字签名模块

数字签名模块采用椭圆曲线签名机制 ECDSA (Elliptic Curve Digital Signature Algoirthm) 。ECDSA 是 ANSI X9.62，FIPS 186-2 和 IEEE P1363 标准的一部分，它主要提供对消息摘要或哈希的数字签名功能。它由两个大整数组成。

```
public class ECDSA {
    public BigInteger c;
    public BigInteger d;
    private int[] f;
    static Biginteger ZERO = BigInteger.valueOf(0);
    static Biginteger ONE = BigInteger.valueOf(1);
    ...
    /* 获取数据 data 的签名 (c, d) */
    public void sign (ECPrivKey s, byte[] data) {
        SHA1Digest sha = new SHA1Digest();
        sha.update(data);
        c = BigInteger.valueOf(0);
        d = BigInteger.valueOf(0);
        f = Utils.revIntArray(Utils.toIntArray(sha.digest()));
        while ((c.compareTo(ZERO) == 0) || (d.compareTo(ZERO) == 0)) {
            ECPrivKey u = new ECPrivKey(s.params);  //临时私钥
            ECPubKey V = new ECPubKey(u);  //临时公钥
            c = Utils.OS2IP(Utils.FE2OSP(V.W.x)).mod(s.params.r);
            BigInteger uinv = u.s.modinverse(s.params.r);
            BigInteger temp = Utils.OS2IP(f).add(s.s.multiply(c).mod(s.params.r));
            d = (uinv.multiply(temp.mod(s.params.r))).mod(s.params.r);
        }
    }
```

```
/* 验证数据 data 的签名 (c, d) */
public boolean verify (ECPubKey W, byte[] data) {
    if (!(((c.compareTo(ONE) <= 0) && (c.compareTo(W.params.r) < 0)) &&
            ((d.compareTo(ONE) <= 0) && (d.compareTo(W.params.r) < 0))))
        return fales;
    SHA1Digest sha = new SHA1Digest();
    sha.update(data);
    f = Utils.revIntArray(Utils.toIntArray(sha.digest()));
    BigInteger h = d.modInverse(W.params.r);
    BigInteger h1 = Utils.OS2IP(f).multiply(h).mod(W.params.r);
    BigInteger h2 = c.multiply(h).mod(W.params.r);

    ECPoint P = W.params.E.add(W.params.E.mul(h1, W.params.G), W.params.E.mul(h2,
W.W));
    if (P.isZero)
        return false;
    BigInteger i = Utils.OS2IP(Utils.FE2OSP(P.x)).mod(W.params.r);
    if (c.compareTo(i) == 0)
        return true;
    return false;
}
...
}
```

4.2.4 XML 模块

XML 正成为 Web 服务世界中一个主要的数据交换协议。驱动 Web 服务的 XML 消息在到达目的地之前，通常需要经过多个中间环节。因此，我们保护从端到端的通信内容是重要的。完成这一任务的最好方法是，将 XML 文档及其安全性信息（如签名、摘要和密钥等等）都装运在一起，作为单个文档。

XML 模块用于生成并传送 XML 格式的消息，从而进一步增强服务端与客户端之间通信的安全，同时也使系统更容易扩展，比如可以增加对 SOAP 的支持等。目前有几个第三方的开源工具包可以使用，如 KXML。

4.2.5 数据库访问模块

数据库访问模块是所有其它模块都需使用的模块，这是因为整个 J2ME 客户端所需使用的程序配置和用户设置都通过它存取到了 J2ME 的数据库中。

在 MIDP2.0 中，定义了一个简单的基于记录的数据库管理系统（Record Management System. RMS）。Record Management System 的主要职责是存储和唯一标识记录，而表示数据的任务则由应用程序完成。这样就简化了 MIDP 的实现，使得 J2ME 的子系统尽量的小巧、灵活。毕竟移动信息设备的存储空间和处理器的能力都非常有限。

在 RMS 中，Record Store 是一系列记录的有序集合，记录是不能单独存在的，必须属于 Record Store。Record Store 保证记录的读写操作都是原子的，数据不会被破坏。在同一个 MIDlet suite 里面的 MIDlets 可以相互访问彼此的 Record Store。

虽然 RMS 保障了对 Record Store 的记录操作具有原子性、同步性和连续性，但是 MIDlet 仍需要负责对多个线程之间的访问顺序进行同步。并且，由于手机终端的硬件限制，在遍历 Record Store 的记录时，需要特别注意防止堆内存“抖动”（一种从内存的堆中连续不断地创建和释放对象的情况），以免造成内存溢出。

在 MIDP 2.0 中，RMS 位于 javax.microedition.rms 包中，包括：一个类（RecordStore），四个接口（RecordComparator，RecordEnumeration，RecordFilter，RecordListener）以及五个相关异常。

数据库访问模块使用这些类对 RMS 进行操作，并对外部模块提供了两个存取数据的接口：

1) 按名称保存数据到 RMS 的接口：

```
public void setDataToRecordStore (String name, String data)
```

2) 按名称从 RMS 获取数据的接口：

```
public String getDataFromRecordStore (String name)
```

上述两个接口在实现过程中涉及到对 Record Store 的打开、修改、读取等操作。

由于这里需要存取的数据并不是很多，因此只需要建立一个 Record Store。另外我们还创建了 RMSAccess 类，以实现 RecordFilter 接口，如此就可在 RecordStore 存储的记录中，按照给定的数据名称查找出所有对应的记录，从而实现数据库访问功能。

第五章 移动电子商务开发实例

本章将利用前文所述的相关技术及已实现的移动支付系统客户端，以一个移动订票系统作为移动电子商务的原型系统，阐述移动电子商务系统的设计与实现。

5.1 系统概述

5.1.1 系统模型概述

本文所设计系统是一个带有 J2ME/MIDP 无线前端和 J2EE 应用服务器后端的智能的移动订票系统，它模拟了客户通过移动终端，登录到在线票务系统进行移动订票的小额支付商务过程。该系统主要包括两个部分：基于 J2ME 的手机客户端和基于 J2EE 的服务器端。

移动订票系统采用三层体系结构。客户端是运行应用程序的客户手机，主要用于显示用户界面，将用户的购买指令发送至服务器端，显示服务器端的返回信息。服务器端是运行在 Tomcat 服务器容器上的一组 Servlets 和 JavaBean，它们封装了整个移动支付的业务逻辑和对数据库的操作。接受客户的购买指令，进行业务处理，将处理结果转化为客户端手机能处理的格式，发送给手机，操作数据库完成相应操作。图 5-1 为移动订票系统架构图。

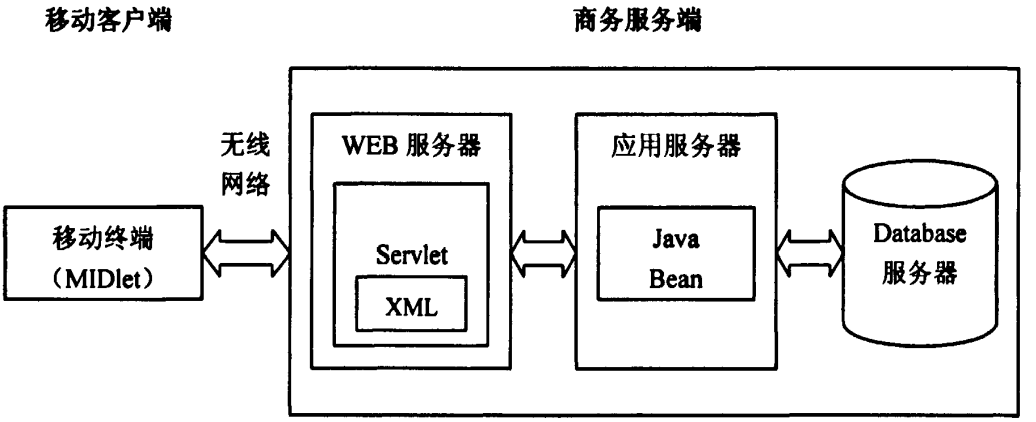

图 5-1　移动订票系统架构图

第五章 移动电子商务开发实例

本章将利用前文所述的相关技术及已实现的移动支付系统客户端，以一个移动订票系统作为移动电子商务的原型系统，阐述移动电子商务系统的设计与实现。

5.1 系统概述

5.1.1 系统模型概述

本文所设计系统是一个带有 J2ME/MIDP 无线前端和 J2EE 应用服务器后端的智能的移动订票系统，它模拟了客户通过移动终端，登录到在线票务系统进行移动订票的小额支付商务过程。该系统主要包括两个部分：基于 J2ME 的手机客户端和基于 J2EE 的服务器端。

移动订票系统采用三层体系结构。客户端是运行应用程序的客户手机，主要用于显示用户界面，将用户的购买指令发送至服务器端，显示服务器端的返回信息。服务器端是运行在 Tomcat 服务器容器上的一组 Servlets 和 JavaBean，它们封装了整个移动支付的业务逻辑和对数据库的操作。接受客户的购买指令，进行业务处理，将处理结果转化为客户端手机能处理的格式，发送给手机，操作数据库完成相应操作。图 5-1 为移动订票系统架构图。

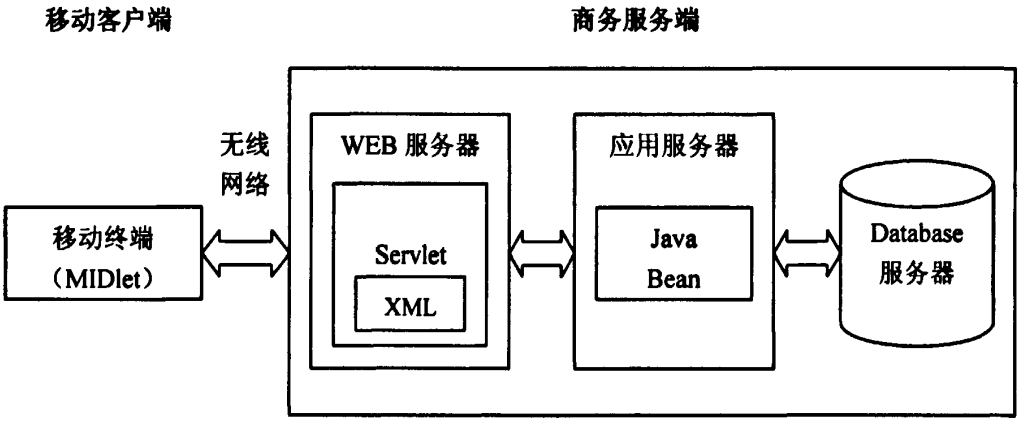

图 5-1　移动订票系统架构图

5.1.2 系统交易模型概述

手机客户端应用软件是一个运行在支持 J2ME 的手机上的 MIDlet，名为 TicketClientMIDlet。首先用户启动 TicketClientMIDlet，输入用户名和密码（合法的用户名和密码事先存储在服务器端的数据库中），通过 HTTP 协议连接到服务器端进行身份认证。如果是合法用户，则服务器端会返回登录成功信息，同时将电影放映表和相应票价等信息发送到手机客户端；如果是非法用户则给出相应的错误信息。用户根据返回的影片放映表选择感兴趣的场次，输入购买数量，然后通过一个 HttpConnection 向在线票务系统服务器发送购买请求，服务器收到该请求后，进行相应的业务逻辑处理，统计用户的购买信息，然后将用户的购买信息再次发送给用户，让用户进行交易确认。用户收到服务器返回的交易确认请求信息后，若确定要交易，则将确认信息，并输入银行卡号与密码，通过前文所设计的移动支付模块支付所需费用；如果用户想中止交易，则发送否认指令。在线票务系统服务器根据用户发送过来的信息进行相应业务逻辑处理：如果是确认信息，则将在客户支付成功后将本次交易详细信息写入数据库，同时将交易成功信息发送给用户，以便用户日后查询交易情况；如果是否认信息，则取消本次交易；如果客户支付失败，则提示客户再次尝试或取消交易。

以下是移动订票系统的交易流程：

1) 移动客户端通过 HTTP 协议将用户名和密码发送至服务器端进行登录身份认证。
2) 服务器端搜索数据库，验证客户是否为合法用户。如果是合法用户，则返回确认信息，同时将影片放映表和相应票价等信息发送给移动客户端；如果是非法用户，则发送非法用户的提示信息。
3) 移动客户端选择感兴趣的电影场次，输入购买数量，并发送给服务器端。
4) 服务器端接受到用户的购买请求，进行相应的业务逻辑处理，并将处理结果发送给移动客户端，请求移动客户端进行交易确认。
5) 移动客户端确认信息，并输入银行卡号与密码，通过前文所设计的移动支付模块支付所需费用。
6) 服务器端在移动客户端支付成功后，将本次交易的详细记录写入数据库，以便消费者日后查询。同时服务器端将交易成功信息发送给客户端，结束本次交易。

另外客户端还会提供一些其他附加功能，如电影评分、影片放映表缓存等。

a) 评分模块：用户可以对所观看的电影进行评分，此功能同时支持在线和离线两种模式。若客户端处于离线模式下，则评分数据将被缓存在移动设备上，并且可以在用户要求时同步到服务器上。这样就能使用户在手机离线模式下也可以为电影评分。同时同步功能是智能的，当同一个用户对同一部电影多次打分时，它通过在后端的数据库中只保持最新的评分来解决这个问题。当然，用户只能通过这个模块来对他所选择的电影进行评分。
b) 缓存影片放映表模块：此模块的主要功能就是将影院的放映时间表保存在移动客户端，这样用户就可以在离线模式下浏览缓存的时间表，从而通过减少网络往返传输提高了客户端性能，同时也节省了网络费用。当然，用户可以根据需要删除或重新下载时间表。另外，为了避免移动用户多次下载同一个放映表，缓存了放映表的影院的图标会有所不同。

5.2 客户端程序的开发

5.2.1 客户端程序的设计

本系统客户端应用程序整体遵循了模型-视图-控制器（MVC）模式，此模式将整个系统交互分为三个不同的角色。视图起 UI 的作用，通过一系列的显示和交互向用户展示信息。控制器扮演搭建用户需求和服务之间桥梁的角色，根据客户的需求调用特定的功能模块，处理相应的业务逻辑，并对视图进行调度。模型组件主要关注的是业务逻辑和计算机到计算机的交互（如数据库访问）。使用 MVC 模式有以下优点：

1) 降低耦合度、增强内聚性；
2) 使人员职能分工明确，有助于团队开发；
3) 有助于分布式开发；
4) 封装分解系统的复杂性。

此系统的 MVC 模式的实现如下：

a) 视图（View）：代表用户交互界面，每一个交互式的屏幕都由视图类表示。对于视图的处理仅限于视图上数据的采集和处理，以及用户的请求，而不包括在视图上的业务流程的处理。业务流程的处理会被事件处理器捕获，并被交予模型处理。

另外客户端还会提供一些其他附加功能，如电影评分、影片放映表缓存等。

a) 评分模块：用户可以对所观看的电影进行评分，此功能同时支持在线和离线两种模式。若客户端处于离线模式下，则评分数据将被缓存在移动设备上，并且可以在用户要求时同步到服务器上。这样就能使用户在手机离线模式下也可以为电影评分。同时同步功能是智能的，当同一个用户对同一部电影多次打分时，它通过在后端的数据库中只保持最新的评分来解决这个问题。当然，用户只能通过这个模块来对他所选择的电影进行评分。

b) 缓存影片放映表模块：此模块的主要功能就是将影院的放映时间表保存在移动客户端，这样用户就可以在离线模式下浏览缓存的时间表，从而通过减少网络往返传输提高了客户端性能，同时也节省了网络费用。当然，用户可以根据需要删除或重新下载时间表。另外，为了避免移动用户多次下载同一个放映表，缓存了放映表的影院的图标会有所不同。

5.2 客户端程序的开发

5.2.1 客户端程序的设计

本系统客户端应用程序整体遵循了模型-视图-控制器（MVC）模式，此模式将整个系统交互分为三个不同的角色。视图起 UI 的作用，通过一系列的显示和交互向用户展示信息。控制器扮演搭建用户需求和服务之间桥梁的角色，根据客户的需求调用特定的功能模块，处理相应的业务逻辑，并对视图进行调度。模型组件主要关注的是业务逻辑和计算机到计算机的交互（如数据库访问）。使用 MVC 模式有以下优点：

1) 降低耦合度、增强内聚性；
2) 使人员职能分工明确，有助于团队开发；
3) 有助于分布式开发；
4) 封装分解系统的复杂性。

此系统的 MVC 模式的实现如下：

a) 视图（View）：代表用户交互界面，每一个交互式的屏幕都由视图类表示。对于视图的处理仅限于视图上数据的采集和处理，以及用户的请求，而不包括在视图上的业务流程的处理。业务流程的处理会被事件处理器捕获，并被交予模型处理。

b) 模型（Model）：就是业务流程/状态的处理以及业务规则的制定。业务流程的处理过程对其它层来说是黑箱操作，模型接受控制器转交的视图请求数据，并返回最终的处理结果。模型层中的类包含了所有的业务逻辑。事实上，整个 J2EE 服务器组件、设备上的缓存和通信类都属于模型层。

c) 控制器（Controller）：可以理解为从用户接收请求，将模型与视图匹配在一起，共同完成用户的请求。控制层并不做任何的数据处理，它只是设计出每一种用户与程序之间可能的交互。在本系统中控制器类是 UIController 类。它对应每一种可能的操作都有一个方法。操作方法通常要启动两个线程：一个在后台执行操作，另一个向用户显示进度条。后台操作线程由 EventDispatcher 类表示。EventDispatcher.run()方法包含一系列 Switch 语句，通过调用模型层中相应的方法以执行这个操作。当模型方法返回时，控制器就调用相应的视图类显示下一个 UI 屏幕。

图 5-2 显示了模式在系统架构中的应用。

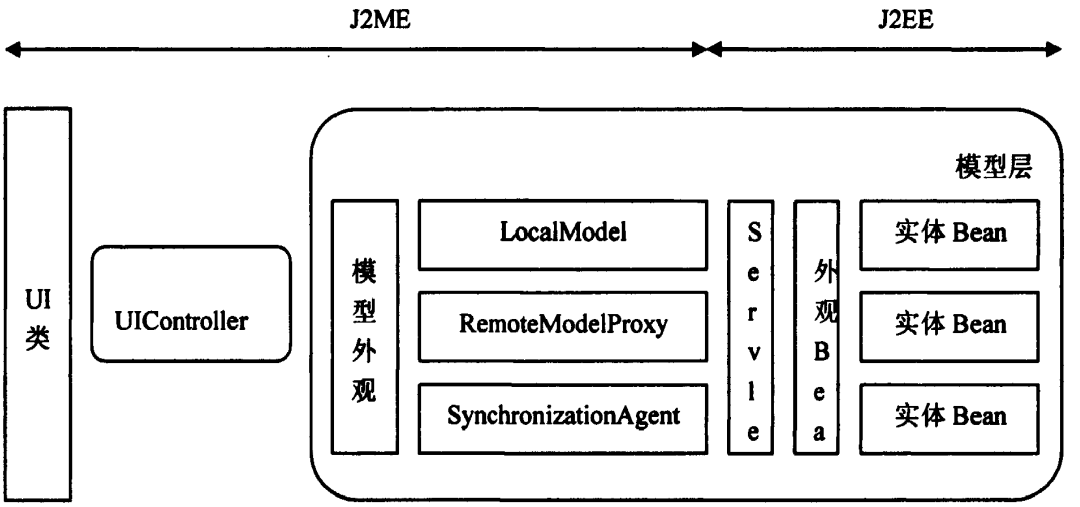

图 5-2　模式在系统架构中的应用

1. 客户端外观模式

外观模式[50]（Facade）定义了一个将子系统的一组接口集成在一起的高层接口，以提供一个一致的界面。通过这个界面，其他系统可以方便地调用子系统中的功能，而忽略子系统内部发生的变化。简而言之，就是为复杂的子系统提供简单接口的结构模型。

由于本系统支持离线和在线两种运行模式，尤其是对服务端数据的复制、维护、访问，以及移动客户数据的刷新、同步，增加了系统移动客户数据模型实现的复杂性。为了降低系统的复杂性，通常会将其分成若干个子系统，这样设计可以减少各个子系统之间的通信和依赖性。但是大多数客户端只需要复杂子系统提供一个简单接口，并没有必要了解子系统中某个类的具体实现。Facade 模式正是基于这样一种设计策略：定义一个高层接口，使外部能通过其访问内部子系统，从而隐藏了内部子系统的复杂性，使得复杂子系统具有简单的访问方式。

移动客户数据模型既包括本地缓存的数据，也包括必要时从服务器上获得的数据，使用哪类数据完全根据用户的需要和网络的状况。而 Facade 模式的使用，一方面可以使 MIDlet 访问数据相对简化，另一方面也可简化移动客户端显示层访问数据。

本系统模型层的客户端子系统中，Facade 类 ModelFacade 正是控制器到模型的入口点，LocalModel、RemoteModelProxy、SynchronizationAgent 类都隐藏于其后，如此 ModelFacade 类就为控制器提供了一个操作本地数据和远程服务器数据的统一接口（如图 5-2 所示）。

ModelFacade 类包含模型层中每个操作的方法，它按下列的方式将操作委托给子系统：

a) LocalModel 类处理访问本地设备上存储的操作。
b) RemoteModelProxy 类实现 RemoteModel 接口，处理需要访问远程 J2EE 服务器的操作。例如，用户购买电影票，那么事务将通过 RemoteModelProxy 在服务器端上执行并永久化到数据库中。RemoteModelProxy 类中的操作方法对服务器端的远程外观调用远程过程调用（RPC）。
c) SynchronizationAgent 类处理所有从本地数据存储到远程服务器的同步操作，本系统利用其处理电影评分的同步。它包含了两个方法：
synchronizeMovieRatings() 方法同步评分，commitMovieRatings() 方法将解析过的同步请求提交给后端并更新本地存储的内容。

2. 服务器外观模式

外观模式的一个最重要优点是它减少了远程系统之间的网络往返传输。一个设计得当的外观使我们可以在子系统中细化对象，同时又可以提供简单的网络接口。它对于移动商务系统尤其重要，因为无线网络的速度相对来说非常慢。当 RemoteModelProxy 向服务器端发出 RPC 请求时，Servlet 通过业务委派对象调用会话中的相应操作方法。根据操作的特性，它会进一步委派给会话 Bean。服务器端的应用程序数据通过 CMP（Container Managed Persistence，容器托管永久性）

实体 EJB 的一个数组永久化到数据库中。

5.2.2 开发环境

1. Eclipse 3.3.1

Eclipse 是一个开放可扩展的集成开发环境（IDE）。它不仅可以用于 Java 的开发，通过开发插件，它可以构建其他的开发工具。Eclipse 可以快速开发复杂企业级应用系统在内的各种 Java 程序，其中包括了独立运行程序、Applet-Servlet、JSP、EJB、Web Service 等等。

2. EclipseME 1.7.7

EclipseME 是用于开发 J2ME MIDlet 的 Eclipse 插件。它可以帮助用户轻松地把无线工具包整合到 Eclipse 开发环境中，使用户不必再担心 J2ME 开发有何特殊需求，而可以把所有精力集中在应用开发上。

3. Wireless Tool Kit 2.5.2

Wireless Toolkit（WTK）是 Sun 提供的无线开发工具包，其设计目的是为了帮助开发人员简化 J2ME 的开发过程。目前 WTK 已经成为 J2ME 开发应用程序的事实上的标准工具包。自从 2000 年初次亮相以来，该工具包一直在发展以赶上移动 Java 技术外观上快速改变的步伐。

4. ProGuard 4.2

由于 MIDlet 是需要下载到手机执行的，而 Java 类可以借助反编译工具被解析出源代码，因此 MIDlet 的版权安全受到了严重威胁。Java 类混淆器解决了这个问题。借助 ProGuard 我们可以在不影响 Java 程序正常运行的情况下，把类进行混淆处理，使得反编译器无法正常解析出正确的源代码，同时 ProGuard 还可以减小 MIDlet 的体积。

5. Bouncy Castle 1.38

Bouncy Castle 是一种用于 Java 平台的开放源码的轻量级密码术包。它支持大量的密码术算法，并提供 JCE 1.2.1 的实现。因为 Bouncy Castle 被设计成轻量级的，所以从 J2SE 1.4 到 J2ME（包括 MIDP）平台，它都可以运行。它是在 MIDP 上运行的唯一完整的密码术包。

上述除 Eclipse 外的所有工具都能以 Library 或 SDK 的形式整合到 Eclipse 中，这样便可以用 Eclipse 进行集成开发。

5.2.3 客户端程序主要模块的实现

1. MIDP 主类：

TicketClientMIDlet 类为 MIDP 客户端的主类，实现了父类（MIDlet 类）中定义的三个抽象方法 startApp、destroyApp、pauseApp，这是一种基于回调的机制，由 AMS（应用管理软件）负责在系统启动、停止以及销毁 MIDlet 时分别调用这三个方法。MIDlet 的生命周期是一种状态机模型，MIDlet 可处于以下三种状态中：暂停状态、活跃状态、销毁状态。图 5-3 展示了通过调用这三种方法来实现这三种不同状态间的转换。

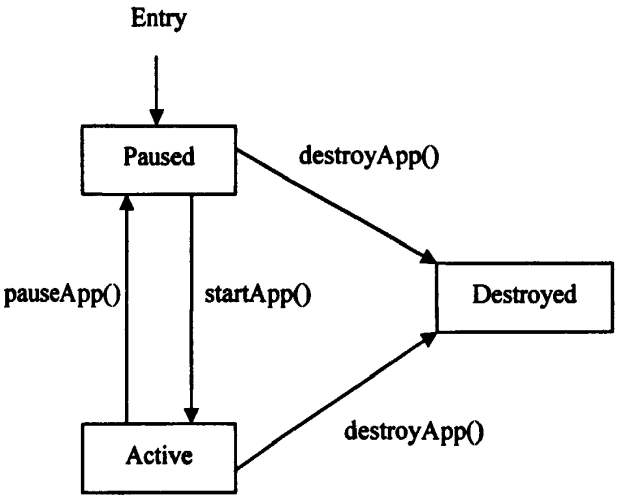

图 5-3　MIDlet 的生命周期

当一个 MIDlet 准备执行时，AMS 首先通过使用其公共的无参数构造函数来创建一个 MIDlet 实例，且该 MIDlet 进入到暂停状态。MIDlet 只保持少量资源，等待着被系统调度执行；当 MIDlet 被系统调度执行的时候，startApp()方法被调用，MIDlet 从暂停状态转换到运行状态，处于运行状态的 MIDlet 保持所有需要的资源，并被系统执行；当系统需要暂停 MIDlet 执行的时候，pauseApp()被调用，MIDlet 释放掉大部分的资源，从运行状态转换到暂停状态；当系统结束 MIDlet 执行的时候，destroyApp()方法被调用，MIDlet 从暂停状态或者运行状态转换到销毁状态，MIDlet 释放所有的资源，等待被垃圾收集程序回收。

TicketClientMIDlet 部分代码如下所示：

```
public class TicketClientMIDlet extends MIDlet {
    public static final String PROPERTY_SERVICE_URL = "Ticket_Servlet_URL";
    public static final String PROPERTY_ LOCALE = "Ticket_Locale";
    private UIController controller;
    private ModelFacade model;
    protected void startApp() {
        try {
            model = new ModelFacade (getAppProperty
                (PROPERTY_SERVICE_URL), getAppProperty(PROPERTY_LOCALE));
            controller = new UIController (this, model);
            controller.init();
        } catch (Exception e) { ... }
    }
    protected void destroyApp (boolean unconditional) {
        ......
    }
    protected void pauseApp() {}
}
```

应用开始时，在 startApp() 方法中初始化 ModelFacade 对象，调用应用程序流程控制器对象 (UIController)。

2. 控制器类 (UIController)

在该类中所有的事件处理通过 EventDispatcher 类的实例来实现，通过 switch 语句，来完成对不同事件的不同处理，而对于本地数据/远程数据处理等操作则通过 UIController 调用 ModelFacade 类（该类提供了所有信息处理接口）中的方法来实现。UIController 类为视图中的每个事件开启一个线程，避免了移动设备处理器限制产生的操作延迟。

```
public class UIController {
    public static class EventIds {
        public static final byte EVENT_ID_ACCOUNTSETUPREQUESTED = 0;
```

```
        public static final byte EVENT_ID_ACCOUNTSETUPSAVED = 1;
        ......
    }
    class EventDispatcher extends Thread {
        private int taskId;
        private Displayable fallbackUI;
        EventDispatcher (int taskId, Displayable fallbackUI) {
            this.taskId = tasked;
            this.fallbackUI = fallbackUI;
            return;
        }
        public void run() (
            try {
                switch (taskId) {
                    case EventIds.EVENT_ID_ACCOUNTSETUPREQUESTED: {
                        accountInfo = new AccountInfo();
                        accountSetupUI.init(accountinfo, preferences);
                        display.setCurrent(accountSetupUI);
                        break;
                    }
                    case EventIds.EVENT_ID_ACCOUNTSETUPSAVED: {
                        ......
                        break;
                    }
                    .......
                } catch (Exception e) {...}
            }
        }
}
```

3. 模型类

模型类包括 ModelFacade、LocalModel、RemoteModelProxy、SynchronizationAgent、RemoteModelRequestHandler 等类。

ModelFacade 类是控制器到模型的入口点，为控制器提供了一个操作本地数

据和远程服务器数据的统一接口，采用外观设计模式，具体的操作通过 LocalModel 类、RemoteModelproxy 类以及 SynchornizationAgent 类实现。

LocalModel 类提供了本地信息处理的接口，本地数据缓存、持久化、读或写等操作通过 RMSAdapter 类实现。

RemoteModelProxy 作为远程业务处理代理，将操作进一步委派给一个处理程序类的链，它们透明地完成 RMS 和 HTTP 串行化的烦琐工作。这是一种责任链设计模式[50]，将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理它为止。简而言之就是由一系列类处理一个请求，这些类之间是一个松散的耦合，唯一共同点是在它们之间传递请求，像一个链条一样传递下去，直到请求被链条上的某个类接收并被处理。

在本系统中的具体实现是通过 RemoteModelRequestHandler、RMSCacheHandler 和 HTTPCommunicationHandler 类来实现。

RemoteModelRequestHandler 为后二类的父类，当进行远程数据处理时，该请求先传递给 RMSCacheHandler 类处理，读取本地缓存数据，RMSCacheHandler 类提供了本地缓存数据处理的各种方法，如果缓存数据过期则传递给 HTTPCommunicationHandler 类处理，从远程服务器读取数据，此类提供了远程数据处理的各种方法。

下面的代码段展现了如何在两个具体的业务处理程序中实现 getMovie()方法。RMSCacheHandler 在设备上的缓存中查询请求的电影，如果请求的电影没有被缓存，那么 RMSCacheHandler 就调用基类的 getMovie() 方法，它会将控制传递给链中的下一个处理程序 HTTPCommunicationHandler 类，其 getMovie()方法会执行一些网络任务以从 J2EE 后端获得 movie 对象。

RMSCacheHandler 中的 getMovie()方法：

```
public class RMSCacheHandler extends RemoteModelRequestHandler {
    public Movie getMovie (String movieKey) {
        IndexEntry indexEntry = rmsAdapter.getIndexEntry
                        (movieKey, IndexEntry.TYPE_MOVIE, IndexEntry.MODE_ANY);
        if ( indexEntry != null) {
            return rmsAdapter.loadMovie (indexEntry.getRecordId());
        }
        return super.getMovie (movieKey);
}
```

HTTPCommunicationHandler 中的 getMovie()方法：

```
public class HTTPCommunicationHandler extends RemoteModelRequestHandler {
    public Movie getMovie (String movieKey) {
        HttpConnection connection = null;
        DataOutputStream outputStream = null;
        DataInputStream inputStream = null;
        try{
            connection = openConnection();
            updateProgress();
            outputStream = openConnectionOutputStream(connection);
            outputStream.writeByte(MessageConstants.OPERATION_GET_MOVIE);
            outputStre am.writeUTF(movieKey);
            outputStre am.close();
            updateProgress();
            inputStream = openConnectionInputStream(connection);
            Movie movie = Movie.deserialize(inputStream);
            updateProgress();
            return movie;
        } catch(Exception e) {
            ......
        } finally {
            closeConnection (connection, outputStream, inputStream);
        }
    }
}
```

4. 视图类

视图类承担着提供易用的输入接口给用户，并把操作结果以友善的界面形式表现在有限大小和色彩的手机屏幕上。每一个交互式的屏幕都是由视图类表示的。一旦用户生成一个 UI 事件，视图类的事件处理程序就会捕获这个事件并将它传递给控制器类。视图类对改善用户体验，提高操作的易用性等方面有着重大的意义。

J2ME 的一个最大优点就是其强大的可移植性，为了使 MIDlet 可以在不同的移动信息设备上任意运行，MIDP 在用户界面中使用了两种方法：抽象描述和运

行感知。

抽象描述通常也称为高级用户界面 API，其优点是可以提供很好的可移植性，缺点是使 MIDlet 变得复杂，因为 MIDlet 必须在内部将抽象描述变为实际的运行用户界面。其中包括 Display, Screen, Label, Command, Alert, Choice, ChoiceGroup, Form, List, TextBox, TextField 等等控件。

运行感知通常也称为低级用户界面 API，虽然可移植性不如抽象描述那么好，但是却能够提供非常强大的屏幕显示控制能力。其中包括了 Canvas、Image 等等。

视图类只要灵活、合理地运用这两种方法，就可以提供友善的易于操作的界面给用户。

凭借抽象描述提供的控件，我们可以按照以下方式运用：

Label：显示界面的文本信息。

TextField：提供文本输入框给用户进行输入。

List：配合数据库访问模块，显示最近用户输入过的数据如电影代码等，以便 用户可以不用重复输入。

Command：提供用户可以操作的命令按钮。

Alert：显示警告、操作成功/失败等提示信息。

Form：提供容纳上述控件的容器。

另一方面，为了提供富有观赏性的、友善的界面，我们可以通过使用运行感知方法提供的 Canvas 配合 Image，制造出动画的效果。通常，在向无线网络发起交易请求至接收到返回结果的之间往往需要一段不少的时间，如果在这段时间里应用程序界面没有任何响应，用户将会被误导以为应用程序僵死，因此动画效果在此时特别有用。

5. 网络通信模块

网络通信模块主要提供网络接入功能。由于 MIDP2.0 已经提供了对 Http、Socket、Datagram、Https 等连接的接口支持，因此客户端可采用 HTTP 协议与服务器端进行通信，封装发送用户的购买指令，接受服务器端的返回信息；而移动支付模块主要通过在应用层中对某些敏感数据进行加密和签名，可使用 HTTPS 连接。因此本系统不必对网络通信模块进行具体实现，而是采用 J2ME 提供的网络功能。

5.2.4 实现上的一些其他技巧

在 J2ME 开发的过程中，除了上述模块的主要实现部分外，还有许多各方面的技巧，这些技巧在开发一款优秀的 J2ME 应用程序的过程中，同样是非常实用的。

1. 双缓冲防止屏幕闪烁

双缓冲技术是编写 J2ME 游戏程序的关键技术之一。实际上，双缓冲技术是计算机动画的一项传统技术。造成屏幕闪烁的主要原因在于，画面在显示的同时，程序又在改变它，于是画面闪烁。解决办法就是在内存中开辟一片区域（可以用 Graphics 对象）作为后台画面，程序对它更新、修改、完成后再显示它。这样被显示的图像永远是已经完全画好的图像，程序修改的将不是正在被显示的图像。当然还有其它方法可以解决屏幕闪烁问题，但使用双缓冲技术是一种值得推荐的解决方案。

2. 减小 MIDlet 体积

MIDlet 越小，占用手机存储空间越少，下载的时间也越短。减小体积的方法有很多。例如可以通过把 MIDlet 用到的若干张小图预先合成为一张大图，这样可以省掉多个小文件的文件头的空间浪费，然后在 MIDlet 启动后通过编程实现分割为小图分别加载。此外，ProGuard 不但是保护 Java 源代码的利器，而且能够通过缩短类、方法、变量名的长度，使得混淆后的类减小 20%～30%。

3. 加快运行速度，减少内存占用

尽可能地使用数组，而不是 Vector 等向量（尽管它们可以提供编程的便利）能够最大化 MIDlet 的运行速度。而占用内存最多的除了对象的集合如数组、向量外，就是图片。因此在程序启动后，可以在适当的时机释放掉已经无用的对象所占用的内存空间，如 Splash 屏幕的图片对象等等。

4. 提高可移植性

可移植性是 Java 的一大特点，在理想的情况下，编写的 MIDlet 应该可以在所有厂商的终端上运行。但实际上由于各大厂商有各自的在 MIDP 基础上扩展的 API，因此无法做到绝对的可移植。解决方法是尽量不要使用厂商特定的 API 并且在代码中用常量对图片等文件的属性如长、宽等进行定义，这样可以在开发同一 MIDlet 的不同手机版本时，只需要对这些常量进行修改就能够完成一个新的

版本，把改动量控制在一个较低的水平。

5. 利用动画效果

在应用程序启动时，可以通过 splash screen 显示软件信息，同时也可防止用户因时间过长而误以为程序启动失败。同样，如前文所述，在程序向无线网络发起请求至接收到返回结果之间，或者程序在进行某些较为复杂的处理时，往往需要一段不少的时间，如果在这段时间里应用程序界面没有任何响应，用户将会被误导以为应用程序僵死，此时也可以设计一些动画效果来提示用户程序正在运行。

5.3 服务器端程序的设计

移动订票系统中服务器端的主要工作就是监听客户端的请求，收到客户端的请求后，调用数据库相关数据进行相应的业务逻辑处理，随后将处理结果发送回客户端，同时将交易信息写入数据库，以备日后消费者查询。

由于移动订票服务器端的主要功能较为简单，故服务器端程序只由三个Servlet组成：ServletApp_login.java，ServletApp_buy.java，ServletApp_db.java。在客户端程序设计之初专门设计了一个JavaBean来对移动订票系统所带的数据库moviedb进行操作，后来为简明起见，对JavaBean进行了优化，将对数据库操作的代码直接放进各个Servlet中，JavaBean被优化掉。其中各Servlet的功能如下：

ServletApp_login.java：该Servlet主要实现用户登录认证功能。接收客户端应用程序传递过来的用户名和密码参数，到数据库userdb进行用户合法性验证。如果用户名和密码与数据库中存储的一致，则向客户端返回登录成功信息，同时，向客户端发送在线票务系统提供的各项服务信息；如果用户名或密码与数据库中存储的不一致，则向客户端返回登录失败信息。该Servlet的程序流程图如图5-4 所示。

ServletApp_ticket.java：该Servlet主要实现处理用户购买订单功能。接收合法用户的订票信息，统计用户本次订购电影场次、订购数量及支付金额，将信息发送给用户，让用户核对。用户确认后，通过与银行接口获取用户支付成功或失败信息，若支付成功则调用ServletApp_db.java向数据库写入交易记录，同时向客户端发送交易成功信息。若用户输入错误信息、取消交易或支付失败则给出相应错误提示。该Servlet 的程序流程图如图5-5所示。

ServletApp_db.java：该Servlet主要实现将一次交易记录写入数据库功能。如果用户订票并支付成功，则将本次交易信息，包括电影场次、订购数量及支付金额等信息写入数据库，以备日后查询。

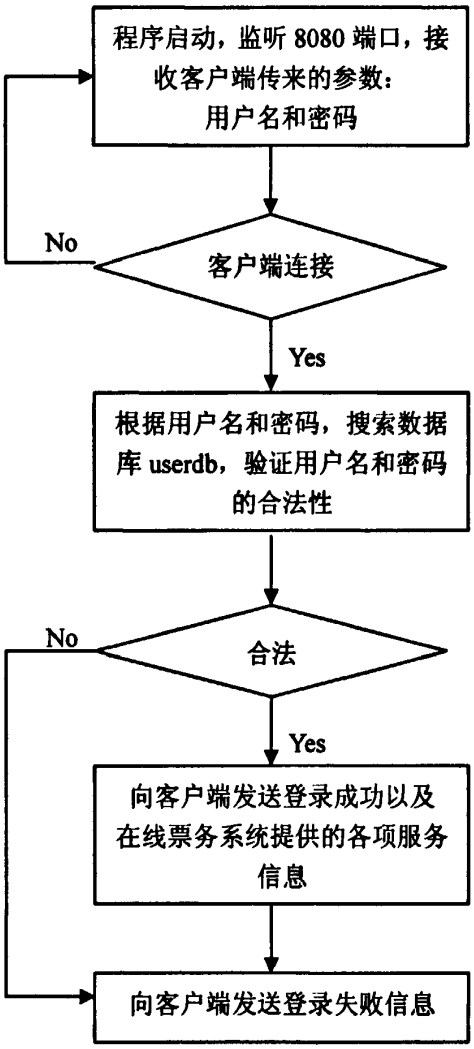

图 5-4　ServletApp_login.java 流程图

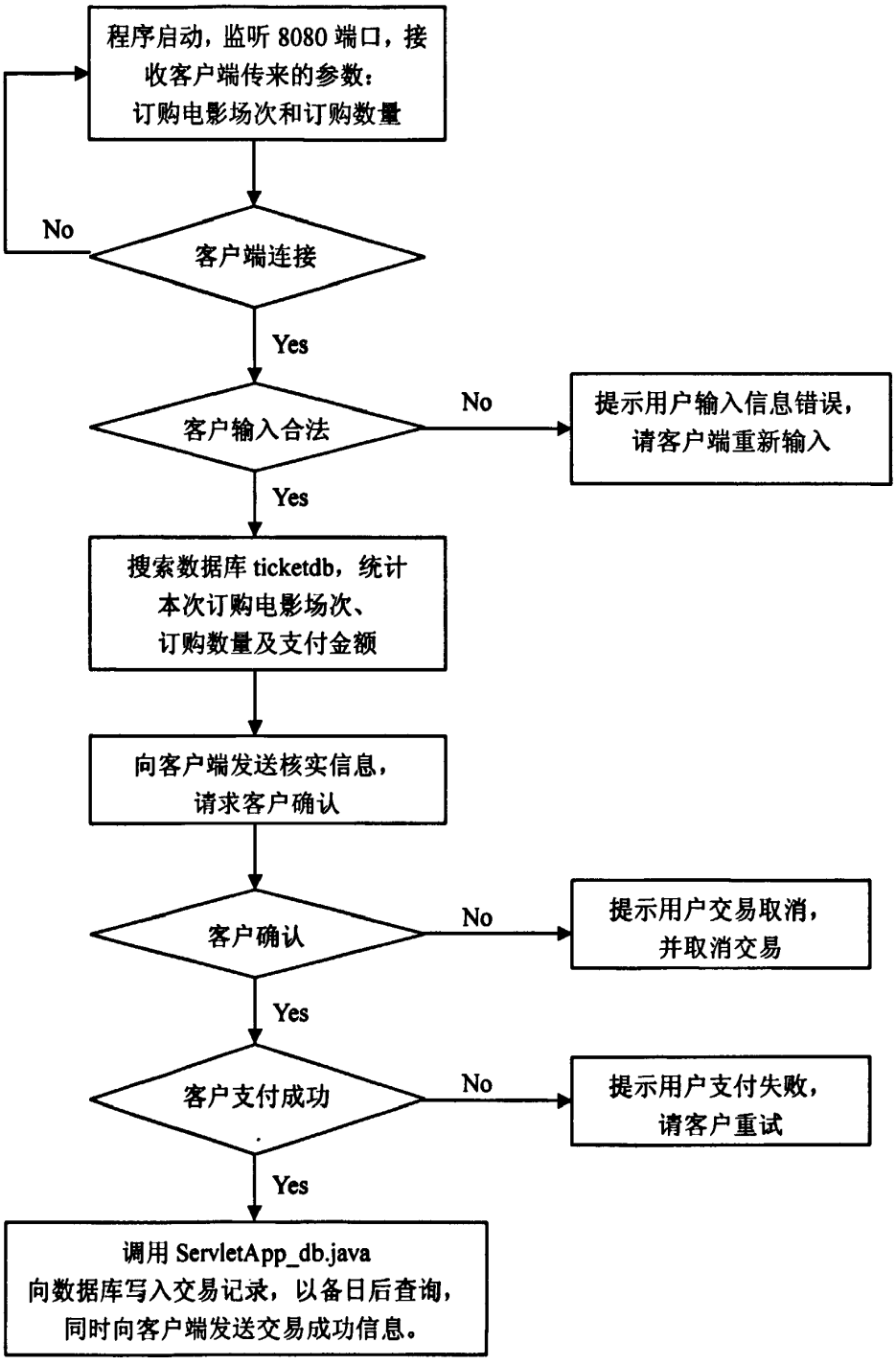

图 5-5　ServletApp_ticket.java 流程图

5.4 数据库设计

本系统数据库主要由七张表组成，resvervations (表 5-1)，shows (表 5-2)，seating (表 5-3)， cinemas (表 5-4)，movies (表 5-5)，userprefs (表 5-6)，tickets (表 5-7)。现分别说明各张表的结构，如下各表所示。

reservations	Who's assigned to each seat	
username	varchar(16)	references userprefs::username
show_id	integer	primary key
seat_id	integer	references seating::seat_id

表 5-1　resvervations

reservations	Who's assigned to each seat	
show_id	integer	primary key
cinema_id	integer	references cinema::cinema_id
movie_id	integer	references movies::movie_id
showtime	varchar(5)	Time(hhmm)

表 5-2　shows

seating	Seating chart for each show	
seat_id	integer	primary key
show_id	integer	references shows::show_id
row	integer	row of the seat
seat	integer	seat no.

表 5-3　seating

cinemas	Cinema infomation	
cinema_id	integer	primary key
zipcode	char(5)	theater zipcode
location	varchar(35)	Name/Location of Theater

表 5-4　cinemas

movies	Title, descriptions, rating and poster for each movie	
movie_id	integer	primary key
title	varchar(25)	Name of the movie
rating	integer	1-10
poster_url	varchar(30)	filename for the image

表 5-5　movies

userprefs	User account information	
username	varchar (16)	login id
password	varchar (16)	password(plaintext)
zipcode	char(5)	user's default zip code
creditcard	varchar(16)	user's credit card number

表 5-6　userprefs

tickets	Ticket information	
movie_id	integer	references movies::movie_id
show_id	integer	references shows::show_id
seat_id	integer	references seating::seat_id

表 5-7　tickets

5.5 移动订票系统的运行

下面将介绍本系统在模拟器 WTK2.5.2 下的运行情况。

移动订票系统在登陆时首先需要验证个人身份，输入用户名和密码（如图 5-6 所示）后，进入订票系统主界面（如图 5-7 所示）。

订票系统主要提供以下几个功能：选择电影，搜索电影，影片评分，订票历史记录，个人设置。接下来介绍本系统的一些主要功能。

1. 选择电影

点击进入选择电影主界面，用户选择影院（如图 5-8 所示）后就会出现该影院目前的播放影片列表（如图 5-9 所示），用户选择感兴趣的影片后就会出现时间选择界面（如图 5-10 所示），选定时间后，用户可以选择目前尚存的座位以及

所需的电影票数，确认后会出现影片明细界面（如图 5-11 所示）供用户核对信息。

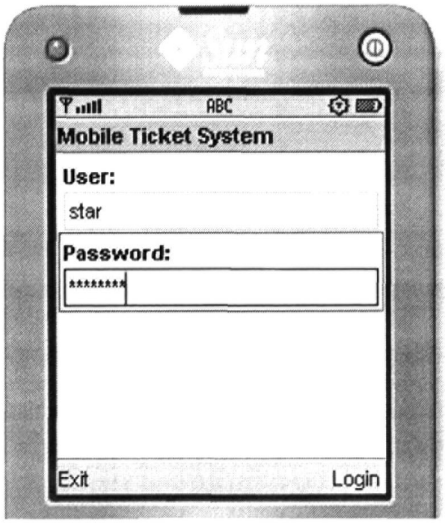

图 5-6　用户身份认证界面

图 5-7　系统主界面

图 5-8　影院选择界面

图 5-9　影片选择界面

用户核对信息无误后会出现确认订票界面（如图 5-12 所示），若用户确认，则会出现付费界面让用户输入信用卡号与密码在线付费；若用户取消订票，则将回到系统主界面。

图 5-10　时间选择界面

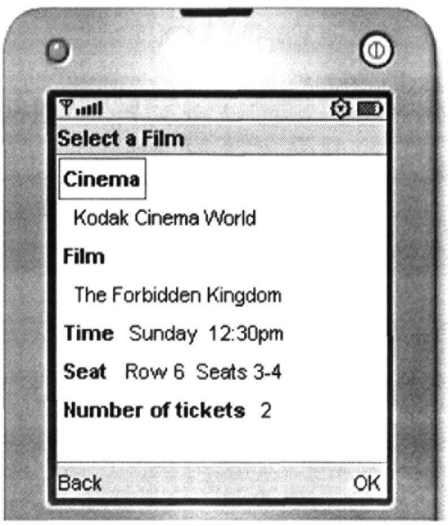

图 5-11　影片明细界面

图 5-12　确认订票界面

图 5-13　订票成功界面

当用户确认订票且付费成功后会出现订票成功界面（如图 5-13），通知用户订单号及取票等具体信息。

2. 搜索电影

点击进入搜索电影主界面，用户填入感兴趣的影片名（如图 5-14 所示）后，会出现目前放映该影片的影院列表（如图 5-15 所示），用户选择好影院后就会如前一样来到时间选择界面（如图 5-10 所示），此后订票操作与前相同，不再赘述。

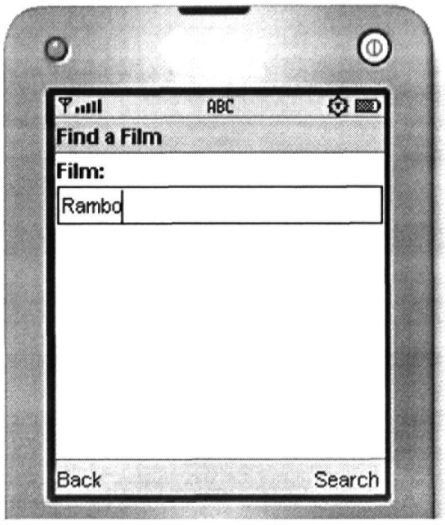

图 5-14　搜索电影界面

图 5-15　影院列表界面

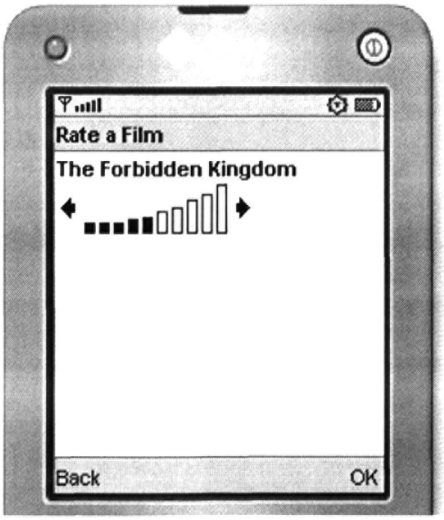

图 5-16　影片评分界面

3. 影片评分

用户在观看影片后，可以在在线或离线两种方式下给影片评分，离线方式下是把消息缓存在本地，然后在上网与系统连接时自动同步到后端的数据库中。用户也可以对同一影片多次评分，但是同步到数据库中时会智能地根据评分时间来选择最新的评分，从而保证了数据的一致性与完整性。评分界面如图 5-16 所示，其是通过导航键来完成的，是一个可视化的过程。

4. 订票历史记录

点击进入可查看以往订票的历史记录（如图 5-17 所示），若点击某条记录则可查看相应的订票详细信息（如图 5-18 所示）。

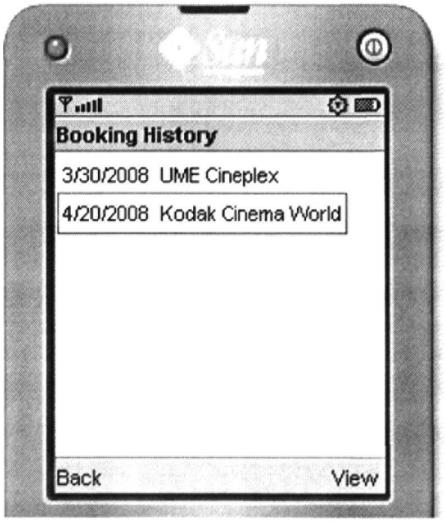

图 5-17　订票历史记录界面

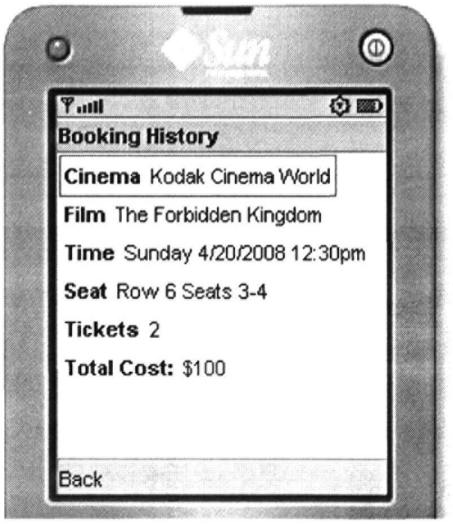

图 5-18　订票明细界面

5. 个人设置

这里为用户提供了修改个人设置的功能选项，如个人偏好、用户帐户、修改用户密码等等，个人设置界面如图 5-19 所示。用户帐户修改密码界面如图 5-20 所示。

图 5-19　个人设置界面

图 5-20　修改密码界面

第六章 总结与展望

6.1 本文的主要工作

移动支付技术是电子商务与移动通信技术相结合的产物，它的主要特点是简单、方便。目前在日本和欧洲等地，移动电子支付应用正逐渐走向成熟。而在我国还处于起步阶段。我国的银行正在与电信运营商、IC 卡制造商以及移动支付设备供应商开始构建移动支付网络，并在为最终用户带来方便的同时自身也获得了良好的效益。

安全方案对移动支付系统的安全性起着决定性作用，特别是无线环境中的安全是系统成败的关键。对其进行精心的设计和完善的分析是移动支付系统开发中必不可少的重要组成部分，也是移动支付系统得以实施的前提和保障。

本文通过对椭圆曲线密码体制的研究，结合现有的 J2ME 安全技术，针对移动通信的特点，特别是电子商务中较高的安全要求，提出了一个新的框架，并且在 J2ME 平台上实现了椭圆曲线密码体制的基本功能，为移动通信的安全研究提出新的方向。

最后本文还构建了一个移动订票系统，由此阐述移动电子商务系统的开发，在理论上和应用上都对企业构建高效率的移动电子商务系统具有一定的指导借鉴意义。

本文主要工作如下：

全面总结了移动支付技术的研究情况以及目前国内外移动支付业务的发展情况，并总结了其存在的主要问题。指出 J2ME 比较适合于移动支付的开发，基于 J2ME 的安全平台是移动支付系统的发展趋势。

1. 本文深入分析了 J2ME 系统的安全机制，从底层 JVM 虚拟机安全以及应用程序层安全两个方面对 J2ME 的安全体系结构进行了全面的研究与总结。
2. 本文研究了椭圆曲线密码体制的原理、优缺点及其应用，并在安全性与有效性等方面与其他密钥算法进行了比较，最后针对 J2ME 平台提出了一些优化设计。
3. 根据移动支付的特点和要求，以及移动支付系统的实际运行环境，同时结合 J2ME 平台的安全机制以及优化的椭圆曲线密码算法，提出了一个基于 J2ME 的移动支付安全方案，并给出了移动支付模块的设计与实现，为通信提供用户认证、密钥交换、数据加密和数字签名等服务。

4. 作为移动电子商务的一个具体应用，本文构建了一个带有 J2ME/MIDP 无线前端和 J2EE 应用服务器后端的移动订票系统，并给出了主要部分的设计与实现，由此阐述移动电子商务系统的开发。

6.2 下一步的研究方向

移动支付在技术上涉及到多方面的安全性，尤其是无线安全，并且没有现有的基于 J2ME 的系统可参考。本文所提出的安全方案在一定程度上能保证移动支付的安全，但是还有很多工作要进一步进行：

1. 本系统中的椭圆曲线密码体制的实现没有考虑硬件对算法的优化，因此在速度和使用资源方面，没能做到最优，这也需要根据不同的平台进行修改和优化。
2. 本系统主要的工作是在客户端的 J2ME 平台上实现移动支付功能，服务端的实现相对较为简单。因为服务端需根据不同的应用，进行不同的实现，同时涉及到实际的业务操作。

参考文献

[1] 周雪松．无线电子支付平台的设计[D]．华东师范大学，2006.

[2] Seema Nambiar Lu, C.-T. Liang, L.R.. Analysis of payment transaction security in mobile commerce[C]. Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference，2004.11：475-480.

[3] Mobile Payment Forum. Mobile Payment Forum White Paper [EB/OL]. 2002.12. http://www.mobilepaymentforum.org

[4] 通信世界周刊．全球移动支付发展现状——移动支付之综述篇 [EB/OL]. 2008.1.

[5] 百纳电信咨询．中国移动支付业务发展分析报告 [EB/OL]. 2006.1.

[6] Michael Juntao Yuan. 确保无线 J2ME 的安全．德州大学奥斯汀分校，2002.9. http://www.ibm.com/developerworks/cn/java/j-j2mesecurity/

[7] Roman Vichr. 提示和技巧：使用 J2ME 实现移动支付．2003.1. http://www.ibm.com/developerworks/cn/java/wi-tip29/

[8] Mobile Information Device Profile (JSR-118) Specification Version 2.0 [EB/OL].

[9] Kim Topley. Java™ 技术手册（J2ME in a Nutshell）[M]. O'REILLY & Associates, Inc 中国电力出版社，2004.2.

[10] 温尚书．J2ME 无线通信实用案例教程[M]．清华大学出版社，2003.10.

[11] 焦祝军，张威．J2ME 无线通信技术应用开发[M]．北京希望电子出版社，2002.8.

[12] 王森．Java 手机程式设计入门[M]．知诚数位科技股份有限公司，2001.8.

[13] James Keeogh. J2ME 开发大全[M]．清华大学出版社，2004.2.

[14] Paul Tremblett 著，王伯欣译. J2ME 无线 Java 应用开发[M]. 北京人民邮电出版社，2002.9.

[15] Bruce Eckel 著，侯捷译．Java 编程思想（Thinking in Java Second Edition）[M]．机械工业出版社，2002.9.

[16] Otto Kolsi, Teemupekka Virtanen. MIDP 2.0 Security Enhancements[C]. Proceedings of the 37th Annual Hawaii International Conference on 5-8，2004.1：287-294.

[17] Roger Riggs. Programming Wireless Devices with the Java 2 Platform Micro Edition, Second Edition[M]. 2004.

[18] 周赟，谢炜，高传善．基于 J2ME 的无线应用的安全性[J]．计算机应用与软件，2004，21(8)：100-102.

[19] 廖永刚，余冬梅，张秋余．J2ME 架构与安全机制的研究[J]．计算机工程与设计，2006，27(4)：575-577.

[20] 笪五三，杨维康. MIDP2.0 安全性分析[J]. 计算机工程与设计，2006，27(14)：2511-2515.

[21] Tim Lindholm，Frank Yellin. Java(TM) virtual machine specification, Second Edition [EB/OL].

[22] Sun Microsystems Inc.. Connected Limited Device Configuration 1.1 [EB/OL].

[23] 刘睿，熊璋等. Java 卡平台安全性研究与应用[J]. 计算机工程与设计，2004，(10)：1753-1759.

[24] 张华. 基于 PKI 的无线通信网络安全问题的研究与实现[J]. 电讯技术，2005，(1)：165-169.

[25] 卢开澄. 计算机密码学[M]. 清华大学出版社，2003.

[26] 杨义先，钮心忻. 应用密码学[M]. 北京邮电大学出版社，2005.6.

[27] 李建华主编. 现代密码技术[M]. 机械工业出版社，2007.5.

[28] 王衍波，薛通. 应用密码学[M]. 机械工业出版社，2003.8.

[29] 蔡家楣, 金利民等. J2ME 环境下椭圆曲线加密算法的优化设计[J]. 计算机时代，2006，(2)，17-18.

[30] 王榕，刘乃琦等. 智能终端上商务文档安全传输中 ECC 加密技术的探讨[J]. 信息安全与通信保密，2007，(4)：74-76.

[31] 罗皓，乔秦宝，刘金龙，黄双庆. 椭圆曲线签名方案[J]. 武汉大学学报（理学版），2003，49(11).

[32] 杨君辉，戴宗铎，杨栋毅，刘宏伟. 一种椭圆曲线签名方案与基于身份的签名协议[J]. 软件学报，2000，11(10).

[33] Neal Koblitz. Elliptic Curve Cryptosystems[J]. Mathematics of Computation / American Mathematical Society. 1987, 48(177)10：203-209.

[34] Koblia N.. The state of elliptic curve cryptography[J]. Designs, Codes and Cryptography，2000，(19)：173-193.

[35] Montgomery P.. Speeding the Pollard and elliptic curve menthods of factorization [J]. Mathematics of Comptutation，1985，(48)：209-224.

[36] N.Koblitz. Elliptic curve cryptosystems[J]. Mathematics of Computation，1987，48(17)：203-209.

[37] A.J.Menezs. Elliptic Curve Public Key Cryptosystems[M]. Kluwer Academic Publishers，1993.

[38] Stefan Tillich and Johann Großschadl. A Survey of Public-Key Cryptography on J2ME-Enabled Mobile Devices[C]. Computer and Information Sciences - ISCIS 2004，NCS 3280：935–944.

[39] R.Lidl and H.Niederreiter. Introduction to Finite Fields and Their Applications [M]. Cambridge University Press，1994.

[40] C-S.Park. On certificate-based security protocols for wireless mobile communicationsystems [J]. IEEE Network, 1997, 11(5).

[41] Michael Juntao Yuan. 保护您的 J2ME/MIDP 应用程序[EB/OL]. 德州大学奥斯汀分校, 2002.9. http://www.ibm.com/developerworks/cn/java/j-midpds/

[42] 万辉. 使用 EclipseME 开发 J2ME 程序 [EB/OL]. 2004.8. http://www.ibm.com/developerworks/cn/java/j-eclipse-me/

[43] J. Sairamesh, S. Goh, I. Stanoi, C. S. Li, S. Padmanabhan. Self-managing, disconnected processes and mechanisms for mobile e-business[C]. International Workshop on Mobile Commerce archive, Proceedings of the 2nd international workshop on Mobile commerce, 2002: 82-89.

[44] 王会进, 古鹏程. 一种基于 J2ME 的移动支付系统的设计与实现[J]. 微计算机信息, 2007, 23(1-1): 234-236.

[45] 张蓓, 张永平. 基于 Java 的移动商务安全解决方案设计[J]. 计算机工程与设计, 2006, 27(5): 875-878.

[46] 池瑞楠. 基于 J2ME 和 J2EE 的移动电子商务系统研究[J]. 微计算机信息, 2007, 23(4-3): 152-154.

[47] 张璞, 文登敏. 基于 J2ME 和 J2EE 的移动电子商务系统研究[J]. 成都信息工程学院学报, 2006, 21(4).

[48] 李焕. 基于 J2ME/J2EE 的移动应用系统研究[J]. 电脑知识与技术: 124-125.

[49] 蔡学军, 仵博. 基于 J2ME-J2EE 的移动电子商务平台的研究与开发[J]. 计算机工程与设计, 2006, 27(17): 3123-3125.

[50] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides 著, 李英军, 马晓星等译. 设计模式: 可复用面向对象软件的基础[M]. 机械工业出版社, 2000.6.

攻读硕士学位期间撰写的论文

[1] 王昕，李莹，高传善，记录管理系统 RMS 及其在手机通讯录中的应用，《微型电脑应用》 2008 年 第 24 卷第 4 期 P38-P41

[2] 李莹，王昕，毛迪林，高传善，J2ME MIDP 中 RMS 的设计实现与性能优化，《计算机工程》 2006 年 第 32 卷第 16 期 P52-P54, P69

[3] 徐其廷，王昕，高传善，无线 JAVA 安全体系结构的研究与应用，《计算机应用与软件》 2007 年 第 24 卷第 8 期 P170-P171

致 谢

本论文的工作是在导师高传善教授的悉心指导下完成的。在此，首先向高老师表示我衷心的感谢和崇高的敬意。在三年的研究生生活中，高老师严谨的治学态度、渊博的知识、敏锐的洞察力和平易近人的性格深深地影响了我，时时激励和鞭策着我不断前行。高老师非常关心我们平时的学习生活，在发表论文的事务上，给我提供了很多有用的信息和无私的帮助，在我找工作的过程中也给予了及时的关心和指导。在我完成毕业设计期间，高老师的无私关心、悉心教导和严格要求更是使我受益匪浅。

我还要衷心感谢孙慰迟老师、毛迪林老师和曹袖老师，在三年的学习生活中，我都得到了他们的帮助和支持。

这里还要特别感谢 J2ME 项目小组。J2ME 项目是我在三年实验室工作中参与时间最长的项目，前后大约两年的时间。在这两年时间里，我和小组的每个成员都有非常愉快的合作，屈光师兄、陆芸芸师姐、徐中礼师兄、谢丹铭师兄以及徐其廷师兄等都给了我很多无私的帮助，在项目中我们共同取得了令人欣喜的成绩，而我自己也在这个过程中取得了长足的进步。在整个工作过程中的团队精神和融洽气氛，让我觉得整个小组的每个成员之间都像兄弟姐妹一样的亲切，我们彼此也建立了深厚的友谊和信任。

另外，还要感谢我的实验室同门李冰峰、毛晓锋、薛平、孙冰、辛怀声，在这三年岁月里，他们在生活与学习中给了我很多帮助，陪伴我度过了美好的三年时光。

三年的研究生生活即将结束，点点滴滴，都已经铭刻在我的心上，相信此生都不会忘记。要感谢的人实在很多，然而千言万语也无法完全表达我的谢意，这里就将千言万语化作一句祝福：祝他们身体健康，万事如意。