摘 要

目前，以 WAP 为平台的电子交易类业务作为一种全新的交易模式，正逐渐成为移动增值服务发展的一个趋势。WTLS 作为 WAP 中的传输层安全协议，对基于 WAP 的移动业务安全起着至关重要的作用。

本文系统地研究了 WTLS 协议栈，并对 WTLS 协议栈中的握手协议进行了详细分析，指出了 WTLS 握手协议中存在的四个安全漏洞：缺乏向前安全性，不能防止未知密钥共享攻击、中间人攻击，并且不能提供对用户身份的保护。在广泛分析国内外相关研究现状之后，指出现有解决方案的不足，提出了一种无线密钥交换和双向认证协议 FS-MAKEP。该协议能快速、安全地实现密钥交换，证书认证和隐式认证相结合的双重认证，提供向前安全性，防止未知密钥共享攻击和中间人攻击，并且具有很少的在线计算开销。基于 FS-MAKEP 在密钥交换和身份认证上的优势，本文把 FS-MAKEP 的思想应用到 WTLS 协议中，提出了一种基于 FS-MAKEP 的 WTLS 握手协议，很好地解决了原有 WTLS 协议中存在的 4 个安全漏洞；用 Rubin 逻辑证明了改进后 WTLS 握手协议的安全性；并在 Linux 环境下实现了改进后 WTLS 握手协议，证明了改进后协议的可行性。

可是移动电子交易对用户身份保护提出了更高要求——用户匿名性保护，因此本文引用了基于陷门散列函数的在线/离线签名算法，设计了一种基于匿名通行证的用户身份认证方案 AL-SA。方案的主要特点是通过第三方认证中心为用户办理匿名通行证，服务器端不直接对用户证书认证，而是对匿名通行证认证。本文在 FS-MAKEP 的基础上，结合 AL-SA 方案对 WTLS 握手协议的客户端身份认证过程作出进一步改进。改进后的协议不仅同样解决了原有协议的 4 个漏洞，而且提供了用户匿名性保护，减轻了客户端的计算量，也提高了服务器端对用户身份认证的效率。

最后，分析了两种改进方案的计算开销和通信载荷量，阐述了两种方案的优势。并对 WTLS 安全性研究进行总结，指出了将来进一步的研究方向。

关键词 WTLS，匿名性，椭圆曲线密码系统，在线/离线签名

ABSTRACT

Nowadays, a novel electronic transaction mode based on WAP is becoming more popular in Mobile Enhanced Message Service. As the Transport Layer Security Protocol in WAP, WTLS has the important effect on mobile service security.

This paper systematically introduces the WTLS protocol stack. Through doing some analysis on handshake protocol of WTLS protocol stack in detail, four security leaks in WTLS handshake protocol were indicated: lack of forward security, unknown key-share attack, man-in-the middle attack and no protecting of user identity. After analyzing the existent status of research all over the world, pointing out the deficiency of solving schemes in existence, a mutual authentication and key exchange protocol FS-MAKEP is proposed. The protocol can implement key exchange promptly and safely. It provides forward security, prevents unknown key-share attack and man-in-the middle attack, and realizes dual authentication which combines certificate authentication and anonymous authentication. Furthermore, it decreases the amount of online calculation of clients. Base on the advantage of FS-MAKEP on key-exchange and identity authentication, the paper applies the idea of FS-MAKEP in WTLS protocol and proposes an improved WTLS handshake protocol, which primely solves four security leaks of previous WTLS protocol; Then we use Rubin logic to prove the security of improved WTLS handshake protocol; The feasibility of the modified protocol is proved under Linux environment on which the modified WTLS handshake protocol is implemented.

But mobile E-business requires more on protecting of user identity—user anonymity protection, thus, a user anonymity license authentication scheme AL-SA is designed based on trapdoor hash function online/offline signature scheme. The main feature of the scheme is to produce anonymous License for users through the third part authentication centre, and the server verifies anonymous License instead of direct authentication of user certificate. Then we combine AL-SA

with FS-MAKEP to improve the user authentication process of WTLS handshake protocol. The improved protocol not only solves four leaks of previous protocol, but also provides users anonymity, decreases the amount of calculation of clients and enhances the efficiency that the server verifies user identity.

According to analyze calculation costs and communication loads of both improved schemes, the advantage of both schemes are discussed. Finally, provide a summary of WTLS security research and an outlook of the further research direction.

KEY WORDS WTLS, anonymity, elliptic curve cryptosystems, online/offline signature

原 创 性 声 明

本人声明，所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了论文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得中南大学或其他单位的学位或证书而使用过的材料。与我共同工作的同志对本研究所作的贡献均已在在论文中作了明确的说明。

作者签名：何毅俊　　日期：2007 年 5 月 20 日

关于学位论文使用授权说明

本人了解中南大学有关保留、使用学位论文的规定，即：学校有权保留学位论文，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以采用复印、缩印或其它手段保存学位论文；学校可根据国家或湖南省有关部门规定送交学位论文。

作者签名：何毅俊　导师签名 李志义　日期：2007 年 5 月 20 日

第一章 绪论

在当今飞快发展的信息领域中，有两支突飞猛进的支柱产业。一是移动通信；二是 Internet。随着移动用户数量的不断增加，移动市场对移动业务的需求也在不断上升，移动通信网和现有 Internet 之间的融合将日渐广泛。而把这两大产业连在一起的就是无线应用协议(Wireless Application Protocol，WAP)。

1.1 研究背景

无线应用协议(WAP)[1]是 WAP 论坛经过不断努力得到的成果，它规定了适用于多种无线设备的网络协议和应用程序框架，这些设备包括移动电话、寻呼机、个人数字助理(PDA)等。利用 WAP，数字无线电话以及其他无线设备的用户将能快速、安全地访问 Internet 和 Intranet。通过手机显示屏，WAP 用户便可浏览 Internet 信息、接收电子邮件、选用增值服务，如：银行帐目查询、手机月费查询、款项存取、股市行情等。尤其是手机购物、手机银行、手机证券等以 WAP 作为平台的交易类业务由于大幅度降低了交易成本，打破了时间和空间的限制，正逐渐成为移动通信增值服务发展的一个趋势。据最新的数据显示，中国移动电话用户超过 4.16 亿，其中 WAP 用户数量为 8000 万到 1 亿，预计到 2008 年将达到 2.5 亿。可以看到目前 WAP 业务已经拥有了广泛的用户基础。

毫无疑问，WAP 协议为现有的无线通信网和 Internet 之间的通信架起了桥梁，但也产生了新的风险。这是因为移动环境中所有以电子形式表达的与商业和私人有关的信息，如：文件、账款、商品等都存在被篡改、窃取、窃听、冒充或者交易否认的威胁，这就在数据保护方面提出了全面的诸如机密性、完整性、身份认证、不可否认性方面的安全需求。

相对于有线通信，无线数据网络的通信环境受到的约束更多。无线数据网络存在以下问题：

1. 带宽更窄。
2. 时延较大。
3. 连接稳定性差。

这些限制使得无线通信更容易受到攻击者的攻击。加之 WAP 通信的特殊性(即跨越无线和有线网络)，任何个人、企业、银行及政府如果在一个不安全的网络进行机密信息流通或信息获取，就可能会导致机密信息泄漏和利益损失。因此，如何采取安全措施保证无线网络上数据传输的安全，实现安全的无缝连接，保证

无线终端到 Internet 之间端到端传输的安全[2]，成为国际上计算机网络和无线通信领域中一个重要的研究课题。

1.2 WAP 技术分析

WAP 体系结构为移动通信设备提供了一个层次化的、可扩展的应用开发环境。研究 WAP 体系结构是研究 WAP 安全的基础。

1.2.1 WAP 协议栈

整个 WAP 协议栈的分层设计实现的，如图 1-1 所示。WAP 体系结构的每一层都为上一层提供接入点，并且还可以接入其他服务和应用程序。

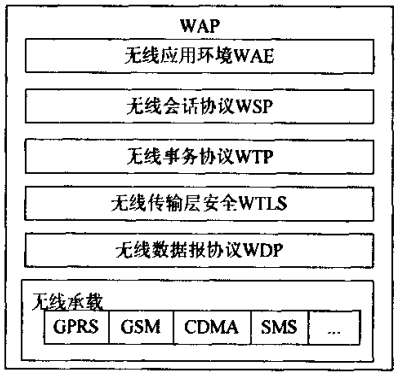

图1-1 WAP协议栈

1. 无线应用环境(Wireless Application Environment，WAE)[3]是一个融合了 WWW 和移动电话技术的通用的应用开发环境。WAE 的主要努力目标是建立一个兼容的环境，以便让运营商和服务的提供者能够在各式各样的无线平台上高效和实用地建立应用程序和服务。

2. 无线会话协议(Wireless Session Protocol，WSP)[4]为两种会话服务提供了一致的接口。第一种会话服务是面向连接的服务，它工作在事务层协议 WTP[5]之上；第二种会话服务是无连接的服务，它工作在数据报服务 WDP[6]之上。

3. 无线事务协议(Wireless Transaction Protocol，WTP)[5]运行在数据报服务之上，是一种轻量级的面向事务的协议，可以在无线数据报网络上高效地运行。

4. 无线传输层安全(Wireless Transport Layer Security，WTLS)[7]协议是一种基于工业标准的传输层安全(Transport Layer Security，TLS)协议。WTLS 专门设计与 WAP 传输协议配套使用，并针对窄带通信信道进行了优化。WTLS 可以用于终端之间的安全通信，如电子商务卡兑现时的鉴权。

5. 无线数据报协议(Wireless Datagram Protocol，WDP)[6]，它工作在有数据承载能力的各种类型的网络之上。作为一种通用的传输服务，WDP向上层的WAP协议提供统一的服务，并对承载业务提供透明的通信能力。

6. 承载

WAP 协议能工作在各种不同的承载业务之上，包括短报文业务、基于电路交换的数据业务和分组数据业务。由于对吞吐量、误码率和延迟的要求不同，承载业务具有不同级别的服务质量。WAP 协议能够适应各种不同质量的服务。

1.2.2 WAP 模型

WAP 模型[1]至少包括 WAP 客户端、WAP 网关和 WAP 服务器。

1. WAP 客户端：

具有 WAP 浏览器的 WAP 无线终端，包括移动电话、寻呼机、个人数字助理(PDA)等。通过这些无线设备，用户便可使用 WAP 业务。

2. WAP 网关：

它是联系 GSM 网与万维网的桥梁，使得 WAP 电话可以访问 WAP 服务器上的资源。WAP 网关在 WAP 设备与 WAP 服务器之间的连接中起到了以下功能：

(1) 将无线标记语言(WML)从文本转换成可以被 WAP 设备读懂的编码(二进制/压缩的)格式；

(2) 将 WAP 设备的请求转换成 Web 中的 HTTP 请求；

(3) 在 Web 和 WAP 之间实现 SSL 加密和 WTLS 加密的转换；

(4) 在 Web 和 WAP 的传输层之间实现 TCP 和 WDP 的转换。

3. WAP 服务器：

WAP 服务器是支持 WML 语言的虚拟主机，它存储着大量的信息，以供 WAP 手机用户访问、查询、浏览等。WAP 客户端可以直接通过 WAP 协议访问该主机上 WAP 站点。

在图1-2中，WAP应用模型采用了与WWW应用模型相似的结构，即C/S(客户端/服务器端)的应用模型。

图1-2 WAP模型

WAP 设备的用户代理将编码的请求以 WML 格式发送到网关，网关对请求进行解码，即将请求转换为服务器端可以识别的 HTML 格式，然后把请求转发到服务器端；服务器端对请求进行处理，把应答发送给网关，网关首先对应答进

行编码，即将应答转换成 WAP 设备可以识别的 WML 格式，然后把应答发送给 WAP 设备的用户代理。

在 WAP 中，WTLS 和 SSL 分别在 WAP 通信不同阶段起到安全保护作用，WAP 客户端到 WAP 网关之间的安全完全依赖 WTLS 协议的保护，而 WAP 网关到 WAP 服务器之间的安全完全依赖于 SSL 协议的保护。在有线安全中，SSL 协议的研究已经非常成熟，因此，本文的研究重点在于解决无线安全协议 WTLS 中的安全问题。

1.3 WAP 中无线传输层安全 WTLS 分析

1.3.1 WTLS 协议栈

WTLS 层运行于传输协议层之上，它是模块化的。WTLS 的主要目的是在两个进行通信的应用间提供保密性、数据完整性以及鉴权。

1. 数据完整性(Data integrity)：WTLS 可以确保终端和应用程序服务器之间传送数据的正确性。

2. 私有性(Privacy)：WTLS 可以确保在终端和应用程序服务器之间传送数据的私有性，任何中途试图截获数据流的设备均无法破译。

3. 鉴权(Authentication)：WTLS 可以在终端和应用程序服务器之间建立鉴权机制。

4. 拒绝服务保护(Denial-of-service protection)：WTLS 可以检测和拒绝那些要求重传的数据或未成功检验的数据。WTLS 使许多常见的拒绝服务攻击更难以实现，从而保护了上层协议。

WTLS 协议栈是分层结构的，共分为两层，如图 1-3 所示。

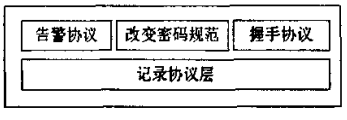

图 1-3 WTLS 协议栈

1. 记录协议层：记录协议获取需要传送的消息，压缩这些数据，使用一个 MAC 加密，然后传送结果。接收到的数据被解密，验证，然后解压缩，最后分发到握手协议层。

2. 握手协议：负责进行安全对话的协商，产生安全对话的加密参数，这些参数包括：协议版本号、使用的加密算法、鉴别的信息和由公开密钥技术生成的密钥等。

3. 改变密码规范协议：该协议只包含一个简单消息，此消息在双方同意安

全参数发送之后，而最后校验信息尚未发送前被发送。该消息发送后，消息发送者和接受者将当前写状态改为未决状态。

4. 告警协议：告警类型是 WTLS 记录层支持的内容类型之一，告警信息传送的内容为信息错误的严重程度及告警描述。

1.3.2 WTLS 协议中安全认证级别

WTLS 根据服务器端和客户端相互认证的情况把 WTLS 应用分为三类[7]：

1. Class 1：服务器端和客户端不需要相互认证。

2. Class 2：服务器端需要被客户端认证，但客户端不需要被服务器端认证。

3. Class 3：服务器端和客户端相互都进行认证。

WTLS 允许通过匿名方式或证书对客户端与服务器端进行认证，一般需要客户端或服务器端在建立会话信息中提供他们的公钥。在这三类 WTLS 中，有些特性是必备的(M)，有些是可选的(O)，还有些是不需要的。具体区别如表 1-1：

表 1-1 WTLS 三类应用区别

	公钥交换	加密	MAC	客户端认证	服务器端认证	压缩	共享密钥握手	智能卡接口
Class 1	M	M	M	O	O	—	O	—
Class 2	M	M	M	O	M	O	O	O
Class 3	M	M	M	M	M	O	O	O

本课题的研究意义在于提高 WAP 在移动电子交易中的安全性，在移动电子交易中，要求的认证等级是 Class 3。这也是最高安全级别，它在服务器端鉴别的基础上，又增加了客户端鉴别。

1.3.3 WTLS 协议中安全问题根源

WTLS 协议的设计思想来源于 TLS，但是根据无线网络低带宽和无线终端计算能力弱、存储量小的特点，对 TLS 进行了优化和压缩。

WTLS 协议同 TLS 协议最显著的不同是：

1. TLS 需要一个可靠的传输层，也就是 TCP。TLS 无法在 UDP 上工作。WAP 协议栈没有提供可靠的传输层，而且在分组网络上优先选择了 UDP。它只在协议栈的上层通过 WTP 和 WSP 实现了可靠性。因此，这也是设计另一套安全协议的动机，使得 WTLS 工作在 WDP 和 UDP 之上。

2. WTLS 帧中定义了序列号，而这在 TLS 中是不存在的。该序列号确保

WTLS 可以工作在不可靠的传输层上。

3. WTLS 不支持数据的分组和重装，它将这个工作交给了下层协议处理。与此不同的是，TLS 可以对上层协议的数据包进行分组。

所以，WTLS 中与 TLS 的差别导致了它不如 TLS 强壮和有效，不能给 WAP 提供足够的安全保障，更引发了众多的安全问题，如文章[8]中提到：IV 的可预测性可引起选择明文攻击；DES 加密存在安全隐患；PKCS#1 攻击等等。

在整个 WAP 通信过程中，包括通信链路的建立、信息的传输，如用户身份信息、位置信息、用户输入的用户名和密码、语音及其它数据流，存在被第三方截获的可能，从而给用户造成损失。另一方面在移动通信系统中，移动用户与网络之间不像固定电话那样存在固定的物理连接，商家如何确认用户的合法身份，如何防止用户否认已经发生的商务行为都是急需解决的安全问题。

1.4 研究目的和意义

本课题的研究目的是：构建一个安全的，有效率的，强壮的，可操作的 WTLS 握手协议，可以无线环境中实现密钥交换，且计算开销小，安全性高，能防止向前安全攻击，未知密钥共享攻击和中间人攻击等典型的主动攻击；而且在不泄漏客户端身份的基础上，服务器端和客户端可以成功实现双认证过程。

本课题研究的重要意义在于它对移动电子交易将产生重大影响。手机已经成为一种全世界普遍使用的通讯工具，而在无线移动通讯领域中，WAP 是国际上流行的通用的无线应用协议标准。从商业的角度出发，提高 WTLS 的安全性和有效性能够大大加速 WAP 的应用和发展。一个更加安全的无线交易环境必将带动移动电子银行、移动贸易、移动购物、移动证券、移动缴费的飞速发展，用户再也不用担心诸如身份泄漏，信息偷窃，交易抵赖等影响移动电子交易的安全问题出现，移动电子商务才有可能进一步成功推广。

1.5 论文结构

本论文共分成六章，各章的主要内容说明如下：

第一章：概括性地介绍了论文的研究背景，及 WAP，WTLS 相关知识。通过分析 WTLS 协议设计上的欠缺，而指出了安全问题产生的根源，并叙述了 WAP 安全漏洞研究的重要性及深远意义，最后给出了论文结构。

第二章：通过分析 WTLS 握手协议的详细通信过程，指出了现有 WTLS 中存在的 4 个安全漏洞，分析了现有解决方法的局限性。

第三章：提出了 FS-MAKEP 协议，描述了相互认证及密钥交换过程在

FS-MAKEP 中的实现，分析了它在无线环境中的优势。并且把该方案应用到了WTLS 握手协议中，修补了原有握手协议中存在的安全漏洞。最后，用 Rubin 逻辑证明了改进后协议的安全性，并且在 Linux 环境下实现了改进后的协议。

第四章：依据在线/离线签名的理论和原理，提出了基于匿名通行证的签名认证方案 AL-SA。结合 AL-SA 和 ECC，提出了改进 WTLS 握手协议安全性的第二种方案。这种方案的特点是保证了用户身份匿名性，并且减少了客户端计算开销。

第五章：对第三章，第四章提出的两种方案与原有的 WTLS 协议作综合性对比分析，包括计算开销和通行载荷量的分析。然后，分别阐述了两种改进后的WTLS 握手协议的优势所在，及各自的适用领域。

第六章：对全文进行了总结，展望了本领域今后的研究方向，并提出相关建议。

1.6 本章小结

本章首先介绍了无线应用协议 WAP 的应用领域及发展趋势，然后介绍无线通信相对于有线网络的局限性，讨论了 WAP 协议，特别是 WTLS 的协议栈，及 WTLS 要求的三种安全级别。通过分析 WTLS 协议设计上的欠缺，指出了安全问题产生的根源及急待解决的问题。最后说明了论文研究目的，意义与结构。

第二章 WTLS 握手协议及漏洞分析

WTLS 协议为两个通信实体提供通信机密性、数据完整性和认证服务，但是快速发展的移动电子支付等增值业务对移动通信安全有更高的要求，这也对作为移动网络和有线网络之间通信安全保障的 WTLS 协议的安全能力提出了更高要求。

本文主要研究 WTLS 握手协议中存在的安全隐患。WTLS 握手协议是 WTLS 协议栈中已定义的较高层客户端之一，该协议被用来在安全会话中就安全属性进行协商，当 WTLS 客户端和服务器端建立通信后，双方就协议版本达成一致，选择加密算法，互相鉴权，并且使用公共密钥加密技术产生双方共享的密码。最后，握手过程中的消息被提供给 WTLS 记录层。

2.1 WTLS 握手协议流程

在WTLS握手协议中定义了ECDH，RSA，DH作为可选的密钥交换算法，本文中选择了安全性最高的基于ECDH算法的WTLS握手协议作为分析对象，以利于更好地和后面改进的WTLS握手协议协议对比。

基于 ECDH 的 WTLS 握手协议的完整流程如图 2-1 所示[7],[9]：

SID：会话标识号，用于识别激活或重起的安全对话

V：WTLS 协议版本号

E：实体 E(客户端或者服务器端)

r_E：实体 E 产生的随机数

$SecNeg_E$：实体 E 支持的信息，如密钥交换方案，密码组，压缩方法和密钥更新等

K_P：预主密钥

K_m：客户端与服务器端共享的 20 bytes 主密钥

h：单向哈希函数

f：计算主密钥的函数

$Cert_E$：实体 E 身份证书

x_E：实体 E 的私钥

P_E：实体 E 的公钥

P：椭圆曲线上的点

$u_{m(c/s)}$：中间杂凑值

$\pi(\cdot)$：一个方程，结果可以输出 X 坐标值，如：$\pi(x,y)=x$

$\oplus$：异或

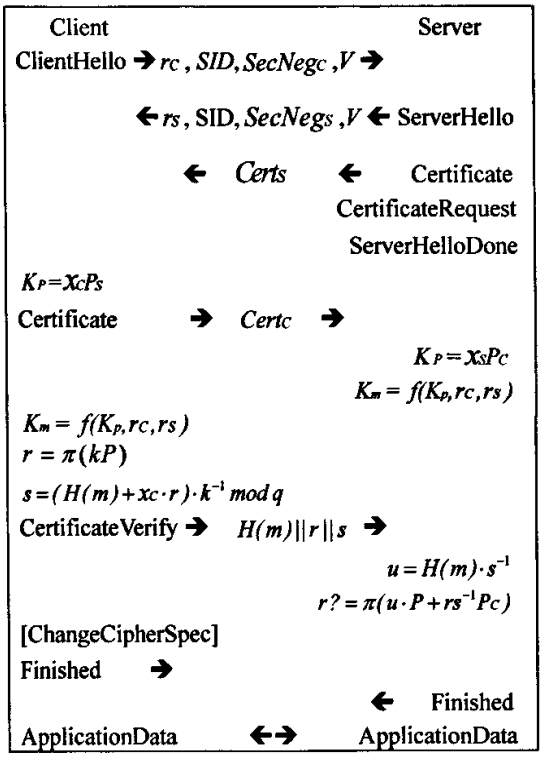

图 2-1 基于 ECDH 算法的完整的 WTLS 握手过程

1. 客户端问候消息 ClientHello

客户端向服务器端发送一个问候消息。该消息包含了客户端的随机数 r_C，客户端希望使用的 WTLS 协议版本 V，客户端所支持的密钥交换规则 $SecNeg_C$，另外，还有安全连接所希望使用的安全会话标识号 SID。当 SID 不可得或客户端希望生成新的安全参数时，该域应该为空。

2. 服务器端问候消息 ServerHello

服务器端在接到客户端的问候信息后，返回服务器端的随机数 r_S，WTLS 协议版本 V，与安全连接一致的安全会话标识号 SID，从客户端的问候消息中所选出的密钥交换套件，解码套件及压缩规则。

3. 服务器端授权消息 Certificate

服务器端发送授权消息，其中授权类型必须适用于所选的密钥交换套件的规则。例如：对于 RSA，所有的授权消息已经带了用 RSA 签署的 RSA 密钥。对于 ECDH_ECDSA，第一个授权消息包含了一个由 ECDSA 签署的 ECDH 密钥，随后的授权消息则带了由 ECDSA 签署的 ECDSA 密钥。

4. 服务器端的密钥交换消息 ServerKeyExchange

只有当服务器端的授权消息无法包含足够的数据去允许客户端交换预主密钥时，服务器端的密钥交换消息才被服务器端发送。如在：ECDH_anon，RSA_anon，DH_anon 中。但不在 ECDH_ECDSA，RSA 中发送。

5. 服务器端授权请求消息 CertificateRequest

服务器端可以向客户端请求一个授权消息，以确认客户端身份。

6. 服务器端的问候已做消息 ServerHelloDone

该消息表明服务器端已经发送了消息去支持密钥交换，客户端可以处理密钥交换过程。在收到 ServerHelloDone 的基础上，客户端应该确认服务器端提供了有效的授权消息，并检验服务器端的问候消息参数是否可接受。

7. 客户端授权消息 Certificate

如果服务器端请求一个授权消息，则客户端向服务器端发送授权消息。

8. 客户端的密钥交换消息 ClientKeyExchange

如果客户端的授权消息中没有包含足够的密钥交换信息或根本没有发送，则需要发送 ClientKeyExchange 消息，消息内容必须基于 ClientHello 和 ServerHello 所协商选用的公钥算法。在这个消息后开始设置主密钥。

9. 客户端授权确定消息 CertificateVerify

授权消息发送后，客户端立即发送授权确定消息，用来提供对客户端授权消息的明确的确定。

10. 客户端改变密码规范消息 ChangeCipherSpe

客户端可能会发送一个改变密码规范消息，这时需要将预生效的密码规范复制到当前密码规范中。

11. 客户端结束消息 Finished

在发送改变密码规范消息后，客户端立即发送一个基于新规则和密钥的结束消息，在收到了服务器端有效的结束消息后，客户端就可以在安全连接上开始发送和接收应用数据。

12. 服务器端结束消息 Finished

服务器端接收到 ChangeCipherSpec 后的响应是发送服务器端的改变密码规范消息，并同客户端一样将当前的密码规范调整为预生效的密码规范，并在新规范的基础上发送 Finished 消息。

至此整个握手完成，客户端和服务器端可以开始应用层数据的交换。

2.2 WTLS 握手协议安全漏洞分析

2.2.1 WTLS 握手协议不能防止几种主动攻击

1. 缺乏向前安全性

向前安全性[10]的思想是密钥泄漏前签名文件的安全性，有别于密钥泄漏后签名文件的安全性，其作用是保证长期密钥的受损不会危及先前协议运行期间所建立的会话密钥，也就是保护先前的行为不受当前密钥的影响。

进行向前安全攻击的前提是：攻击者拥有通信一方私钥。前提条件具备后，攻击者就可以由此推断出双方协商的主密钥。在原始 WTLS 协议中，并不能提供向前安全性。对 WTLS 的向前安全攻击过程如下：

(1) 攻击者一直在监听通信会话，并保存了所有会话数据，如客户端随机数 r_C，服务器端随机数 r_S，服务器证书等。

(2) 当服务器端的私钥 x_S 泄漏后，攻击者可以通过服务器证书中的公钥 P_S 和泄露的私钥 x_S 计算出预主密钥 $K_P = x_S P_C$。

(3) 根据主密钥计算公式计算出以前的会话主密钥 $K_m = f(K_p, r_C, r_S)$。

(4) 拥有主密钥后，攻击者就可以解密先前会话中用主密钥加密的所有会话数据。

2. 未知密钥共享攻击

未知密钥共享攻击[11]是这样一种攻击：攻击完成后，实体 A 确信和实体 B 共享了一个秘密会话密钥，而实体 B 则错误地认为他和另一个实体 E($E \neq A$)共享了该秘密会话密钥。

成功进行未知密钥共享攻击的前提是：攻击者有自己的长期私钥 x_E，并有自己的证书 $Cert_E$。一旦前提条件全部具备，攻击者就可以对 WTLS 协议进行未知密钥共享攻击，攻击过程如下：

(1) E 截获客户端发送给服务器端的证书 $Cert_C$，用自己的证书 $Cert_E$ 代替，并发送给服务器端。

(2) 客户端计算预主密钥和主密钥。

(3) 服务器端验证客户端身份，如果合法，则计算预主密钥和主密钥。

通过上述过程后，客户端确信和服务器端共享了主密钥，但是服务器端则认为他和攻击者 E 共享了主密钥。

3. 中间人攻击

中间人攻击[12],[13]是指攻击者用自己的值替换来自客户端的值或服务器端的值，从而与客户端和服务器端分别共享不同的密钥，而不被客户端和服务器端发

现。WTLS 中的中间人攻击如下：

(1) 攻击者 E 截获客户端发送给服务器端的随机数 r_C，用自己产生的随机数 r_E 代替，然后发送给服务器端。

(2) 客户端计算预主密钥 $K_P=x_CP_S$，服务器端计算预主密钥 $K_P=x_SP_C$，双方并没有发现随机数被替换的事实。

(3) 客户端和服务器端计算主密钥 $K_m=f(K_p,r_C,r_S)$，尽管预主密钥相同，但是双方得到的主密钥不一样，客户端计算的主密钥为 $K_m=f(K_p,r_C,r_S)$，而服务器端计算的主密钥为 $K_m=f(K_p,r_E,r_S)$，但是双方并没有发现。

2.2.2 WTLS 握手协议中缺乏对用户身份保护

身份认证[14]是计算机系统的用户在进入系统或访问不同保护级别的系统资源时，系统确认该用户的身份是否真实、合法和唯一的手段，其目的是防止非法人员进入系统。身份认证是判明和确认交易双方真实身份的重要环节，也是电子商务过程中最薄弱的环节。对服务提供商来说，特别是在 WAP 手机银行[15]的应用上，对用户的身份认证显得犹为重要。他们不希望将服务提供给没有证书的用户或者是完全不可信任的用户，这必将带来很大的安全隐患。

在本文中，我们把 WAP 业务的不同需要，把对客户端的身份认证安全分为 2 个等级：

1. 普通用户身份保护

对于普通 WAP 业务，如新闻浏览，图铃下载，天气预报等，用户身份只需要满足不被泄漏给恶意攻击者即可，而用户身份对服务器端是可见的，服务器通过验证用户身份证书来验证用户身份。

2. 用户匿名性保护

在移动电子商务活动和移动通信中，用户匿名性[16]被认为是一个重要的安全属性。在无线互联网中，用户匿名性必须保证无线互联网用户在享受无线互联网服务提供商提供的服务时，能够保留自己的隐私，实现对用户身份的机密性保护，从而避免不必要的麻烦。因此，这就意味着用户身份信息对服务器端也是不可见的。

通过对 WTLS 握手协议进行分析可以发现，协议缺乏用户身份保护，更不能提供用户匿名性保护。在客户端 Certificate 消息流中，发送给服务器端的用户证书 $Certc$ 没有经过任何加密处理，因此如果用户证书被攻击者截获，攻击者可以从证书中计算出有用的用户信息，如用户的公钥，证书发行者信息及版本信息等，最终攻击者可以利用这些用户信息对用户和服务提供商之间的通信造成更大的威胁。身份泄漏可能造成的危险性分析如下：

在正常情况下，客户 U 向服务提供商发送客户证书以供服务提供商验证身份，客户证书中保存有客户的公钥 e。如果服务提供商打算给客户发送秘密信息，那么服务提供商从客户证书得到客户的公钥，用它对消息加密，然后将密文发送给客户。图 2-2 描绘了这么一个网络。

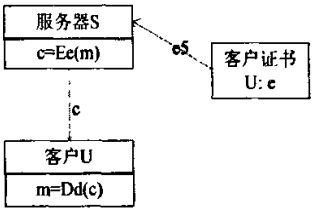

图 2-2 正常的证书传递及应用

但是，当人们必须关注一个主动攻击，当该攻击能够修改含有公钥的证书时，这种传递方式就变得不再安全可靠。图 2-3 说明了主动攻击时怎样危害上面的消息传递过程的。图中，攻击者修改了证书，用自己的公钥 e'来代替 U 的公钥 e。现在，发给 U 的用证书中的公钥加密的任何消息，只能由攻击者解密。在解密并读取了消息之后，攻击者再将它重新用 U 的真正公钥加密，把密文向前发给 U。而 S 仍会相信只有 U 能够解密密文 c。

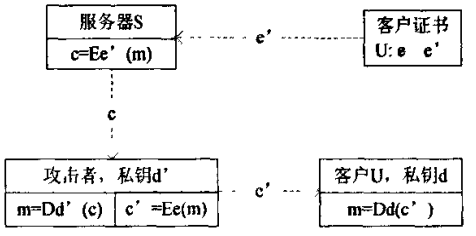

图 2-3 主动攻击者以公钥 e' 对 U 的假冒

显然，匿名的服务无法保证服务提供商的信任要求，而提供证书的服务则会透露用户信息。因此我们需要找到一种方案，既能满足服务器端对客户端必要的身份认证，也能提供用户身份保护，保证用户身份不被泄漏给恶意攻击者；而且在移动电子交易这类对安全有更高要求的 WAP 业务中，能够提供用户匿名性，使服务器端都不能获得用户的私密信息。

2.3 国内外研究现状

目前，对 WAP 安全研究主要集中在三个方面：智能卡的 WIM 规范研究[17],[18]，WAP 网关安全研究[2],[19],[20]，WTLS 协议栈安全研究。

智能卡的WIM规范在WAP1.2版本中提出，智能卡有自己的处理器，可以在

卡上的芯片中实现加解密算法和哈希功能，因此智能卡的WIM规范将安全功能从手机转移到防篡改设备中。

WAP 网关起到了在WTLS协议和SSL协议间进行协议转换的作用。从无线终端到网关的部分由WTLS加密传输，而由无线网关到服务器端的HTTP通信部分由TLS加密传输。WAP 网关需要将来自于无线终端的数据解密，把WML与WMLScript翻译为二进制代码，然后加密后送给服务器端。所以在网关可以看见所有的数据明文，这样就给WAP通信带来了严重的安全隐患。目前，网关泄漏问题已经在WAP2.0版本中得到解决，实现了真正的端到端安全[2],[19],[20]。

WTLS 协议栈的安全研究包括对 WTLS 仿真研究[21][21][23]，握手协议安全研究[9],[24], [25],[26]，本文的重点主要集中在研究 WTLS 握手协议的密钥交换和身份认证过程。

余堃描述了 ECDH 算法及其在 WTLS 中的应用[24]，但是并没有指出 ECDH 在 WTLS 中进行密钥交换过程时存在的安全问题，只是简单描述了 ECDH 算法的应用过程。

WTLS中存在的向前安全性和用户匿名性的安全漏洞[9]最早是在2003年由Dongjin Kwak等人指出。随后，他提出了改进后的基于ECDH密钥交换的WTLS握手协议和改进后的基于RSA密钥交换的WTLS握手协议。Dongjin Kwak指出这两种方法都能为WTLS提供向前安全性和用户匿名性。但是在2004年，崔媛媛等人指出Dongjin Kwak提出的改进后的基于RSA密钥交换的WTLS握手协议并不能提供所预期的向前安全性[25]，然后在文章中对该协议进行了改进，使得改进后的握手协议可以提供向前安全性。在2004年，邹学强提等人出了一个基于DH 密钥交换方案的改进的WTLS握手协议[16][15]，这个协议同样改进了WTLS的向前安全性和用户匿名性。

虽然已有研究者提出了对 WTLS 握手协议的向前安全性和用户匿名性的改进方法，但是这些方法都存在三个缺点就是：

1. 在改进 WTLS 协议的同时，使客户端和服务器端的计算开销和通信载荷量都有所增加，这对于计算能力有限的无线客户端来说是一个无形的威胁。

2. 因为在 WTLS 握手协议中采用的证书体制，使用证书进行身份认证失去了匿名性，目前所采用的解决方法全部是用主密钥加密用户身份证书，然后再发送给服务器端。这种做法只能满足对用户身份保护的第一安全等级，不能满足第二安全等级。在现有解决办法中，服务器端仍然能够通过解密得到完整的用户信息，用户身份会暴露给服务器端，所以还是不能很好地满足用户匿名性的要求。

3. 目前的 WTLS 握手协议的安全研究中并没有涉及到对未知密钥共享攻击和中间人攻击的讨论。 这也是本文中要解决的问题。

2.4 本章小结

详细描述了 WTLS 握手协议的通信流程，指出了 WTLS 当中存在的 4 个安全隐患：缺乏向前安全性，不能防止未知密钥共享攻击，不能防止中间人攻击，和缺乏对用户身份保护；分别给出了对这 4 个漏洞的攻击方式，最后分析了国内外 WAP 安全研究现状，指出了现有解决方法的局限性。

第三章 基于 FS-MAKEP 的 WTLS 握手协议

密钥交换、双向认证是 WTLS 握手协议的重要组成部分，作用在于实现两个通信实体之间身份认证，建立有效的会话密钥。可是在无线移动通信中，基于公钥的双向认证和密钥交换协议的计算开销以及无线移动设备计算及存储能力的有限性，通常成为把该类协议付诸于实践的主要问题。

在这章中，提出了一种能够提供向前安全性的无线密钥交换和双向认证协议 FS-MAKEP (Forward Security-Mutual Authentication Key Exchange Protocol)，该协议的思想起源于 ES-MAKEP。ES-MAKEP 是 Fuw-Yi Yang 等人提出的一种适用于无线移动场景中的密钥交换双向认证协议[27]，具有效率高，安全性好的的特点，但是不足的是：ES-MAKEP 缺乏向前安全性。在 ES-MAKEP 中，一旦服务器端的私钥泄漏，会话密钥就很可能被计算出来。

本文提出的 FS-MAKEP 对 ES-MAKEP 协议进行了改进，能安全地，完整地完成客户端和服务器端的密钥交换过程，实现客户端和服务器端的相互认证，并且经过证明，该协议能够提供向前安全性，防止未知密钥共享攻击，中间人攻击等主动攻击。

3.1 ES-MAKEP 协议介绍及漏洞分析

为了和文章[27]中的概念保持一致，我们采用以下的符号来描述 ES-MAKEP 的实现过程。

$\varepsilon_{PK}()$：不对称加密函数

$\delta_{SK}()$：不对称解密函数

$E_K()$：对称加密函数

$D_K()$：对称解密函数

SK_S：服务器端 S 的私钥

PK_S：服务器端 S 的公钥

ID_U：客户端 C 的身份证书

ID_S：服务器端 S 的身份证书

p, q：客户端 C 的私有密钥对

g, n：客户端 C 的公有密钥对

x ‖ y：x 和 y 的级联

$|n|$：n 的长度

l：会话密钥长度

r_{UK}, r_{UF}, r_{UR}：客户端C产生的三个随机数

r_{SK}：服务器端S产生的随机数

假设p，q，p'，q'是四个大素数，且满足$p = 2p' + 1$，$q = 2q' + 1$。$g \in Z_n^*$，是一个$\lambda(n)$阶的生成元，$n = pq$，$\lambda(n) = \mathrm{lcm}(p - 1, q - 1)$。ES-MAKEP的具体工作流程如图3-1：

1. 客户端选3个随机数r_{UK}，r_{UF}，r_{UR}，且满足$r_{UK}, r_{UF} \in_R \{0,1\}^l$，$r_{UR} \in_R Z_{\lambda(n)}$。然后用$\varepsilon_{PKS}()$及服务器端公钥$PK_S$加密$r_{UK}$，预先计算出$C1_{r_{UK}}$和$CMT$值，最后发送由$C1_{r_{UK}}$，$CMT$和$ID_U$构成的M1消息给服务器端。

2. 服务器端收到M1后，用私钥SK_S解密$C1_{r_{UK}}$，得到r_{UK}。然后用选取的随机数$r_{SK} \in_R \{0,1\}^l$和r_{UK}计算出会话密钥$\sigma_{SU} = r_{SK} \oplus r_{UK}$。最后用$E_K()$加密$r_{UK}$得到$C2_{r_{UK}}$，并和$r_{SK}$一起组成$M2$传给客户端。

3. 客户端收到$M2$后，计算会话密钥$\sigma_{US} = r_{UK} \oplus r_{SK}$，解密$C2_{r_{UK}}$得到$r'_{UK}$。如果$M1$和$M2$都能成功到达对方的话，$r_{UK} = D_{\sigma_{US}}(C2_{r_{UK}}) = r_{UK}$是成立的。这样，客户端就成功认证了服务器端S，因为只有服务器端才能计算出$C2_{r_{UK}}$。服务器端身份确认无误后，客户端计算杂凑值$S_F = h(r_{UK}, r_{SK}, ID_U, ID_S)$，并返回一个响应消息$M3$，该消息由加密$ID_U$后的值$C3$，以及$S_R$组成。$S_R$由方程

$$2^{l/4} r_{UF} + r_{UR} = 2^{l/4} S_F + S_R \bmod \lambda(n) \qquad \text{公式(3-1)}$$

计算得

$$S_R = 2^{l/4}(r_{UF} - S_F) + r_{UR} \bmod \lambda(n) \qquad \text{公式(3-2)}$$

4. 服务器端S计算杂凑值$S_F = h(r_{UK}, r_{SK}, ID_U, ID_S)$和$CMT'$，通过比较CMT? = CMT'来验证客户端的身份。

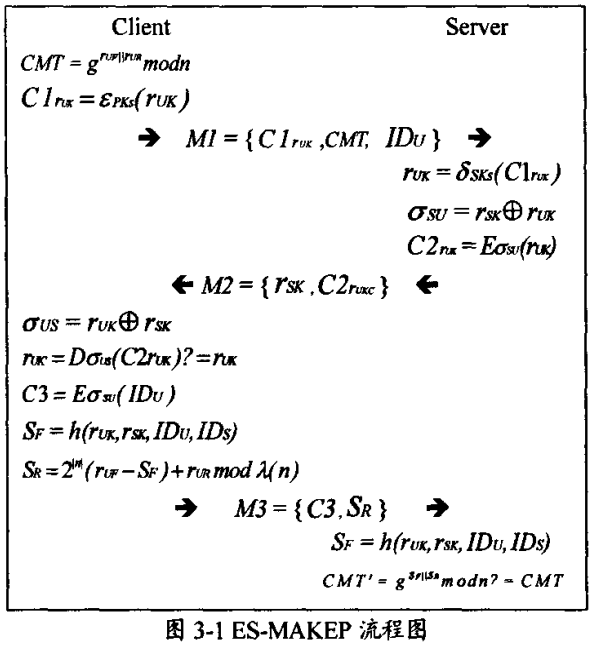

图3-1 ES-MAKEP流程图

向前安全性要求保证长期密钥的受损不会危及先前协议运行期间所建立的会话密钥，而 Fuw-Yi Yang 指出 ES-MAKEP 不提供向前安全性[27]。ES-MAKEP 中存在的向前安全攻击过程如下：

1. 假设攻击者 E 一直在窃听 ES-MAKEP 的会话，保存了以前会话的所有信息(合理假设)，包括 $C1_{r_{UK}}$，r_{SK}。

2. 一旦服务器端的私钥 SKs 泄漏，就可以通过 $r_{UK}=\delta_{SKs}(C1_{r_{UK}})$ 得到客户端随机数 r_{UK}。

3. 攻击者用从 M2 中获得的 r_{SK}，通过 $\sigma_{SU}=r_{SK}\oplus r_{UK}$ 计算出会话密钥 σ_{SU}。

从上述分析可知，ES-MAKEP 不能提供向前安全性，也就是说服务器端私钥 SKs 的泄漏使攻击者计算出会话密钥 σ_{SU} 成为可能，这为 ES-MAKEP 带来了安全隐患。

3.2 ES-MAKEP 的改进协议 FS-MAKEP 协议

在提供向前安全性的无线密钥交换和双向认证协议 FS-MAKEP 中，我们结合了椭圆曲线密码系统(ECC)来改进 ES-MAKEP 协议的安全性。椭圆曲线密码系统的使用可以降低密钥长度，故可提高加/解密、签名/验证签名的效率及减少储存的成本，非常适合运用在较小的内存及有限运算能力的装置上，如智能卡手机或 PDA 等。改进后的 FS-MAKEP 协议不仅能满足 ES-MAKEP 缺乏的向前安全性，而且能抵抗未知密钥共享攻击，中间人攻击等主动攻击，并且在性能上也具有一定优势。

3.2.1 椭圆曲线密码系统

椭圆曲线密码系统[28],[29]是由 Koblitz 和 Miller 两位学者所提出来的，整体的安全性是建立在解椭圆曲线离散对数问题(Elliptic Curve Discrete Logarithm Problem；ECDLP)的困难度上。椭圆曲线离散对数问题是指：在有限场 F_p 之下，对于椭圆曲线 E 上的两个点 P 及 Q，给定 x 和 P，计算 $Q=x\cdot P$ 比较容易，而给定 Q 和 P 计算 x 是很困难的。

相对于使用大指数运算的 RSA 机制而言，ECC 的运算时间少了很多。在 RSA[30]中需要使用 1024 位的模数，才能达到足够的安全等级，而 ECC 只需要使用 160 位的模数即可[31],[32]。

表 3-1 描述了 ECC 与 DSA，RSA[30]算法抗攻击强度的比较。由表 3-1 中可看出，椭圆曲线密码系统所使用的密钥长度相对较短，但是却可以达到相同的安全等级[31],[32]。

表 3-1 ECC 与 RSA/DSA 抗攻击性能比较

RSA/DSA 密钥长度 /bit	ECC 密钥长度 /bit	RSA 和 ECC 密钥长度比率	所需工作量 MIPS 年
512	106	5:1	10^4
768	132	6:1	10^8
1024	160	7:1	10^{11}
2048	210	10:1	10^{20}
21000	600	35:1	10^{78}

椭圆曲线密码系统的数学原理[28]简单介绍如下：

在有限场 F_p 之下，其中 p 为一个大质数，则椭圆曲线可定义成所有满足方程式 E：$y^2 = x^3+ax+b$ $(x,y,a,b \in F_p)$的点(y, x)所构成的集合。若方程式 x^3+ax+b 没有重复的因式或是 $4a^3+27b^2 \neq 0$，则 E：$y^2 = x^3+ax+b$ 能成为群(group)，G 为阶为 r 的生成元。

性质：

1. 椭圆曲线中存在着一个无限远的点 O，但它并不在椭圆曲线上。

2. 若椭圆曲线上点 P 序(Order)为 n，则 n 是最小的正整数使得 $n \cdot P = O$。

3. 椭圆曲线上的生成数(Generator)可以生成椭圆曲线上所有的点，但点 O 除外。

4. E(F_p) 为在 F_p 之下，由椭圆曲线 E 上全部的点所构成的集合。

利用 ECDH（椭圆曲线 Diffie-Hellman）实现密钥交换的过程如下：

1. 设 A 和 B 分别选取随机数 a 和 b 予以保密。

2. A 计算 aG，B 计算 bG，并发送给对方。

3. A 计算 $bG*a$，B 计算 $aG*a$。

4. 则 A 和 B 间通信用的密钥为 abG。

3.2.2 FS-MAKEP 协议流程设计

FS-MAKEP 中保留了大部分 ES-MAKEP 中原有的参数 (可参考 3.1 小节)，也增加了一些参数，表示如下：

E：椭圆曲线方程式 $y^2 = x^3+ax+b$，其中 $4a^3+27b^2 \neq 0$

p_1：大质数

q_1：p_1-1 的大质因子

F_{p1}：p_1 个元素的有限域

E(F_{p1})：E 在 F_{p1} 上的点及无穷远点构成的集合，表示成 $E \cup \{O\}$

O：椭圆曲线上的一个特殊点，称为无穷远点

B：序为 q_1 的点，即椭圆曲线上的生成数

下面给出了 FS-MAKEP 完整的，详细的工作流程，如图 3-2 所示。

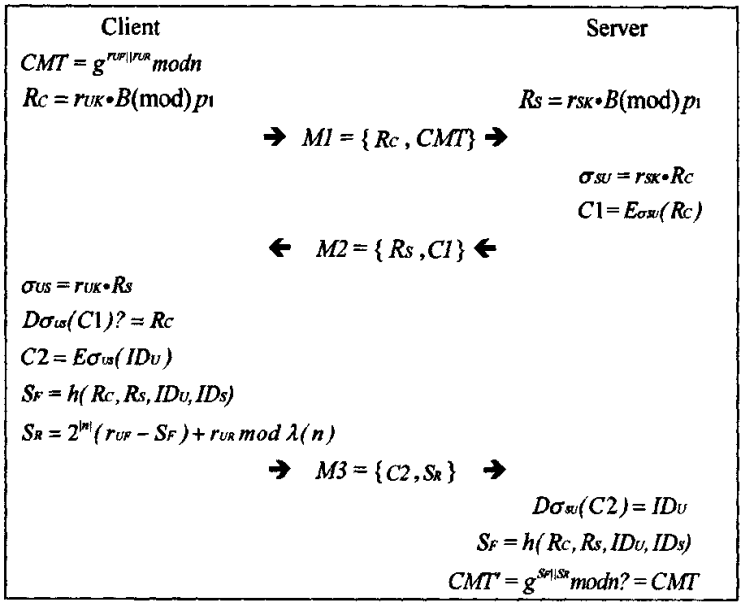

图 3-2 FS-MAKEP 流程图

FS-MAKEP 与 ES-MAKEP 的区别在于：

1. 第 1 步中，从客户端传给服务器端的是 $R_C = r_{UK} \bullet B(\mathrm{mod}) p_1$，而不是 $\varepsilon_{PKS}(r_{UK})$，这样保证了即使服务器端私钥泄漏，也无法计算出 r_{UK}。

2. 为了保证客户端身份不被泄漏，ID_U 不再在 *M1* 中传送，而是在 *M3* 中被加密后得到 $C2 = E_{\sigma_{SU}}(ID_U)$，再传给服务器端。

3. 在 *M2* 中，服务器端传给客户端的随机数是预计算值 $R_S = r_{SK} \bullet B(\mathrm{mod}) p_1$，而不是 r_{SK}。会话密钥的计算修改为 $\sigma_{SU} = r_{SK} \bullet R_C$。

4. R_C，R_S 取代了 r_{SK}，r_{UK}，被用于后面计算 $S_F = h(R_C, R_S, ID_U, ID_S)$ 的哈希函数中。

FS-MAKEP 详细描述如下：

1. 初始化阶段：

p，q，p'，q'是四个大素数，且满足$p = 2p' + 1$，$q = 2q' + 1$。$g \in Z_n^*$，是一个$\lambda(n)$阶的生成元，$n = pq$，$\lambda(n) = \mathrm{lcm}(p - 1, q - 1)$。挑选一个大质数 p_1 以及满足等式 $y^2 \mathrm{mod} p_1 = (x^3 + ax + b) \mathrm{mod} p_1$ 的椭圆曲线参数a和b，这里 p_1 为素数或是形为 2^m 的整数。由此可以定义出点的椭圆群 $E_{p1}(a,b)$；其次，在 $E_{p1}(a,b)$ 中挑选基点 $B = (x_1, y_1)$，B的阶为 q_1。椭圆曲线上的B的阶 q_1 是使得 $q_1 B = O$成立的最小整数。$E_{p1}(a,b)$ 和B是该FS-MAKEP中通信双方均已知的参数。

2. 密钥交换机制：

客户端和服务器端之间完成密钥交换的过程如下：

(1) 客户端选择一个小于q_1，且与q_1互质的整数r_{UK}作为随机数，然后通过点积运算得到$R_C = r_{UK} \bullet B(\mathrm{mod}) p_1$，根据椭圆曲线密码系统性质，因此$R_C$也是$E_{p1}(a,b)$中的一个点。发送 M1={$R_C$}到服务器端。

(2) 服务器端可类似地选择小于q_1且与q_1互质的整数r_{SK}作为随机数，并计算$R_S = r_{SK} \bullet B(\mathrm{mod}) p_1$。

(3) 服务器端收到R_C后，产生会话密钥$\sigma_{SU} = r_{SK} \bullet R_C$。并以会话密钥$\sigma_{SU}$为加密密钥，用对称加密函数$E_K()$加密$R_C$，然后发送消息M2＝{$E\sigma_{SU}(R_C), R_S$}给客户端。

(4) 客户端计算会话密钥$\sigma_{US} = r_{UK} \bullet R_S$，然后以会话密钥$\sigma_{US}$为解密密钥，用对称解密函数$D_K()$解密$E\sigma_{SU}(R_C)$，得到$R'_C$。如果*M1*和*M2*都能成功到达对方的话，$R'_C = D\sigma_{US}(E\sigma_{SU}(R_C)) = R_C$是成立的。因为只有服务器端可以成功计算$E\sigma_{SU}(R_C)$，所以客户端隐式地认证了服务器端S。

步骤(3)和(4)中，会话密钥的两种计算结果是相同的，因为

$$\begin{aligned}\sigma_{SU} &= r_{SK} \bullet R_C \\ &= r_{SK} \bullet r_{UK} \bullet B(mod) p_1 \\ &= r_{UK} \bullet r_{SK} \bullet B(mod) p_1 \\ &= r_{UK} \bullet R_S \\ &= \sigma_{US}\end{aligned}$$

要破译会话密钥，攻击者必须由B和$r \bullet B(\mathrm{mod}) p_1$计算r，而基于解椭圆曲线离散对数难题ECDLP，这被认为是非常困难的。

3. 认证机制：

该认证机制是基于陷门散列函数h()的，此散列函数具有以下性质：

给定客户端公钥对(g,n)，以及私钥对(p,q)，私钥对也称为陷门密钥，两个随机信息 m，m'，一个随机数 $r \in Z_{\lambda(n)}$，则必定可以计算出一个 $r' \in Z_{\lambda(n)}$，满足$h_{(g,n)}(m,r) = h_{(g,n)}(m',r')$。

FS-MAKEP 和 ES-MAKEP 一样定义了散列函数$h() = g^{m||r} \bmod n$，为了计算碰撞信息，我们必须要找到一个 r'，使得

$$g^{m||r} = g^{m'||r'} \bmod n \qquad \text{公式(3-3)}$$

即

$$2^k m + r = 2^k m' + r' \bmod \lambda(n) \qquad \text{公式(3-4)}$$

由公式(3-4)计算得到

$$r' = 2^k (m - m') + r (\mathrm{mod}\ \lambda(n)) \qquad \text{公式(3-5)}$$

整个认证过程如下：

(1) 客户端选2个随机数r_{UF}，r_{UR}，且满足$r_{UF} \in R\{0,1\}^l$，$r_{UR} \in RZ_{\lambda(n)}$。然后根据陷门散列函数预先计算值$CMT = g^{r_{UF}||r_{UR}} \bmod n$，最后发送CMT给服务器端。

(2) 客户端和服务器端完成密钥交换过程后，计算出杂凑值 $S_F = h(R_C, R_S, ID_U, ID_S)$，并返回一个响应消息*M3*，该消息由加密 ID_U 后的密文*C2*，以及 S_R 组成。S_R 由方程

$$2^{|n|} r_{UF} + r_{UR} = 2^{|n|} S_F + S_R \bmod \lambda(n) \quad \text{公式(3-1)}$$

计算得

$$S_R = 2^{|n|}(r_{UF} - S_F) + r_{UR} \bmod \lambda(n) \quad \text{公式(3-2)}$$

(3) 服务器端S解密*C2*，得到 ID_U，然后计算杂凑值 $S_F = h(R_C, R_S, ID_U, ID_S)$ 和计算碰撞 $CMT' = g^{S_F \| S_R} \bmod n$。根据陷门散列函数的性质，如果前几步信息传递过程无误的话，*CMT*应该等于*CMT'*的值。由此，服务器端隐式地验证了客户端的身份。

3.2.3 FS-MAKEP 安全性分析

1. FS-MAKEP 成功实现了对服务器端的认证。

证明：当服务器端收到客户端传来的预计算结果 R_C 后，计算出会话密钥 $\sigma_{SU} = r_{SK} \cdot R_C$，并产生密文*C1*。客户端收到 R_S 后，用随机数 r_{UK} 与 R_S 共同计算出会话密钥 $\sigma_{US} = r_{UK} \cdot R_S$。如果整个信息交换过程是成功的，$\sigma_{SU}$ 和 σ_{US} 的值应该相等。如果攻击者在中途篡改，伪造了 R_C 或者 R_S 的话，那么 σ_{SU} 和 σ_{US} 不会相等，而且，客户端会发现用错误的 σ_{US} 解密*C1*得到的 R_C' 和客户端本身预计算结果 R_C 不相等，则由此可以判断服务器端身份是假的，客户端会马上停止与服务器端通信。如果相等的话，则表明服务器端的身份是合法的，因为只有服务器端才能计算出正确的 $\sigma_{SU} = r_{SK} \cdot R_C$。

2. FS-MAKEP 成功实现了对客户端的认证。

证明：客户端首先用私有密钥对(p,q)计算出 $\lambda(n)=lcm(p-1,q-1)$，然后求得 S_R，并把 S_R 发送给服务器端。服务器端计算 S_F，然后用客户端发送的 S_R 计算 $g^{S_F \| S_R} \bmod n$。因为根据陷门散列函数[33]定义和性质，公式(3-3)是成立的，由此得到等式 $g^{r_{UF} \| r_{UR}} = g^{S_F \| S_R} \bmod n$。因为只有客户端拥有陷门密钥$(p,q)$，所以有且仅有客户端可以计算出陷门信息 S_R。服务器端比较 $g^{S_F \| S_R} \bmod n$ 是否等于*CMT*值，如果相等，则说明陷门信息正确，客户端身份是真实的，反之，客户端就是不合法的。

3. 成功实现了数据完整性认证

证明：完整性主要是对于数据传输到接收者之后，对于发送者的数据可以做检验以保证与发送者的数据相符，而发送者也希望所传送的数据与接收者的数据是一样的，并没有差异性的存在。

FS-MAKEP 中客户端向服务器端发送 $S_F = h(R_C, R_S, ID_U, ID_S)$，由于 S_F 的计算包括了双方在整个通信过程中所交换、传递的所有参数，R_S、R_C、ID_S 和 ID_U。

如果通信过程中任何一个参数的传送出错，都会导致服务器端计算的 S_F 值与客户端的 S_F 值不相等，从而，使得 CMT' ≠ CMT，整个认证过程失败。所以 FS-MAKEP 成功实现了数据完整性认证。

4. 有效提供向前安全性

证明：如图 3-2 所示，在 FS-MAKEP 中，客户端传给服务器端的是 R_C，服务器端传送给客户端的是 R_S，然后双方各自计算出会话密钥 $\sigma_{US} = r_{UK} \cdot R_S$ 和 $\sigma_{SU} = r_{SK} \cdot R_C$。基于椭圆曲线离散对数问题(ECDLP)的难解性，即使攻击者能截获 R_C 和 R_S，但是他仍然计算不出 r_{UK} 或者 r_{SK}，而且在整个会话密钥计算过程中，没有涉及到双方的私有密钥，所以尽管攻击者拥有服务器端的私有密钥 SK_S，仍然没有办法计算出会话密钥。这样就使该方案满足了向前安全性。

5. 有效防止未知密钥共享攻击

证明：从图 3-2 可以看出，双方除了交换各自的身份证书进行显式的身份认证外，客户端还发送了 $C2 = E_{\sigma_{SU}}(ID_U)$，$S_F = h(R_C, R_S, ID_U, ID_S)$，这就防止了未知密钥共享攻击。因为 $C2$ 的传送以会话密钥加密客户端身份证，这保证了攻击者不能用自己伪造的证书取代真的客户端身份证，因为无法获知 σ_{SU}。而 S_F 的中包含了 ID_S，服务器端在收到由 S_F 计算出的 S_R 后，即可通过比较 CMT 值来确认 ID_S 是否是真实的、未被篡改的，因为假的 ID_S 是不可能计算出真确的 S_R 值的，也就不可能计算出真实的 CMT。

6. 有效防止中间人攻击

证明：在传统的 Diffie-Hellman 协议中，攻击者能用自己的值替换来自客户端的预计算值($g^a \bmod n$)或服务器端的预计算值($g^b \bmod n$)，从而与客户端和服务器端分别共享不同的密钥，而不被客户端和服务器端发现。在 FS-MAKEP 协议中，攻击者 E 在第 1 步截获到 $R_C = r_{UK} \cdot B \pmod{p_1}$，用自己产生的值 $R_E = r_{EK} \cdot B \pmod{p_1}$ 替代 R_C 发送给服务器端。服务器端计算得到错误的密钥 σ_{EU}，用它加密 R_C 得到 $C1 = E_{\sigma_{SU}}(R_E)$ 返回给客户端。客户端通过解密 $C1$ 就能立即发现 R_E 或者 σ_{EU} 的不真实性，因此协议能防止中间人攻击。

7. 其他安全性

(1) 已知会话密钥的安全性

证明：由协议的描述可知，执行协议一次至多生成一个会话密钥。由于会话密钥 σ_{SU} 计算中的 R_C 和 R_S 为本次协议执行过程中通过随机元素产生的预计算值，随机元素在每次通信过程中不同，则 R_C 和 R_S 会不同。所以某次会话密钥泄露不会导致以前或者下次协议运行生成的会话密钥泄露。

(2) 会话密钥的不可控性

证明：由于双方都需要根据临时的预计算信息 R_C 和 R_S，才能计算唯一的会话

密钥σ_{US}，故任何用户都没有单独预先计算出会话密钥的能力。

(3) 密钥泄露的安全性

证明：假设实体的私钥泄漏，攻击者就可以成功伪装成该实体。因为只有实体本身才能拥有私钥，所以私钥也就是实体身份的代表。攻击者如果得到客户端或者服务器端的私钥，就可以伪装成客户端和服务器端。在FS-MAKEP中，攻击者既不能伪装客户端，也不能伪装服务器端。因为在密钥交换过程中，没有涉及到双方的长期私钥，而是以预计算值R_C和R_S代替了长期私钥的使用。而且即使攻击者得到了长期私钥，因为协议满足向前安全性，攻击者依然无法计算会话密钥。

3.2.4 FS-MAKEP性能分析

表3-1，3-2，3-3显示了FS-MAKEP与ES-MAKEP的计算开销和通信载荷量的对比结果。计算开销对比分为两部分：离线计算开销和在线计算开销。因为加法，散列函数计算，以及对称加，解密算法的计算开销比非对称加，解密算法和模指数运算，椭圆点积运算要小得多，所以本文作性能评估时就忽略了这些算法开销。Fuw-Yi Yang 和Jinn-Ke Jan在文章[27]中已经对ES-MAKEP的计算开销和通信载荷量作出了评估，不必再重复计算。在此，我们只计算FS-MAKEP的计算开销和通信载荷量，作为和ES-MAKEP的比较依据。根据A. Lenstra和E.Verheul在文章[34]中建议的适合实际应用的密钥大小，我们设置$|n|$=1024bits，$|ID_U|=|l|=|r_{UF}|=|S_F|$=160bits。

对比结果如表3-1，3-2，3-3所示：

注：MMs表示一个模乘a*b mod n的计算开销。其中a,b,n均设置为1024bits。

表3-1 客户端计算开销比较(单位：MMs)

	FS-MAKEP		ES-MAKEP	
	Online	Offline	Online	Offline
Message M1	0	1805 [a]	0	1778
Message M2	0	0	0	0
Message M3	0.1 [b]	0	0.1	0
Total	0.1	1805	0.1	1778

表3-2 服务器端计算开销比较(单位：MMs)

	FS-MAKEP		ES-MAKEP	
	Online	Offline	Online	Offline
Message M1	0	29 [c]	1536	0
Message M2	0	0	0	0
Message M3	1776 [d]	0	1776	0
Total	1776	29	3312	0

表 3-3 通信载荷量比较(单位: bits)

	FS-MAKEP	ES-MAKEP
Message M1	1184 [e]	2208
Message M2	480 [f]	320
Message M3	1184 [g]	1184
Total	2848	3712

根据文章[35]、[36]的结论得知，在椭圆曲线中，$Rc = r_{UK} \bullet B(\mathrm{mod})\, p_1$ 的计算中$|p_1|$=160bits，其安全性和使用 1024 位密钥的 RSA 相同。此外，对于模指数运算 $g^x \bmod n$ 的计算，其中 n 是一个 1024bits 的大质数，x 是一个 160bits 的随机整数，一个模指数运算的开销相当于 $3|x|/2$ 个模乘[37]，近似等于 240MMs。当 $|p_1|$=160bits 时，点积运算速度比模指数运算速度快近 8 倍[9],[38]，于是可以推导出一个椭圆曲线上的点积运算大约等于 29MMs[35],[36]。

a. 对随机数 r_{UK} 进行椭圆曲线上的点积计算，计算 Rc 需要 29MMs。而模指数运算 $CMT = g^{r_{UF}||r_{UR}} mod n$ 需要 3/2*(160+1024)=1776MMs[37]。

b. 文章[27]指出计算 $S_R = 2^{|n|}(r_{UF} - S_F) + r_{UR} \bmod \lambda(n)$ 的复杂度是 0.1MMs。

c. 对随机数 r_{SK} 进行椭圆曲线上的点积计算，Rs 需要 29MMs。

d. 模指数运算 $CMT' = g^{S_F||S_R} mod n? = CMT$ 需要 3/2*(160+1024)=1776MMs。

e. 使用 ECC 算法来产生 Rc，需经过一个模数 p_1 的运算，故所产生的 Rc 大小表示为$|p_1|$=160bits。计算 $CMT = g^{r_{UF}||r_{UR}} mod n$ 需要经过一个模数 n 的运算，故产生的 CMT 大小表示为$|n|$=1024bits。

f. 同 e 所述，计算 Rs 需要经过一个模 p_1 的运算，故所产生的 Rs 大小表示为$|p_1|$=160bits。$C1$ 的大小近似表示为$|R_S| = |p_1|$=160bits。$|ID_S|$=160bits。

g. 与 ES-MAKEP 相同，$|C3| = |ID_U|$=160bits，$|S_R|$=1024bits。

从表 3-1 和表 3-2 可以看出，改进协议中服务器端和客户端的在线开销不但没有增加，而且服务器端的在线开销反而减少了 1536MMs。虽然客户端的离线运算开销轻微增大，但应当注意的是，计算开销的增加仅在离线计算阶段，并不产生在在线阶段，这保证了协议在运行中仍是高效的，且在线运算比 ES-MAKEP 更加快速。运算中增加的离线开销主要是因为选取随机数的变化，用于实现向前安全，同时实现相互认证，防止中间实体攻击，从而改进协议在明文信息传输上的减少可以降低数据受攻击和泄漏的可能。从通信载荷量上来看，改进的协议比 ES-MAKEP 减少了 864bits，这为本来带宽就窄的无线通信带来了很大的优势。

3.3 基于 FS-MAKEP 的 WTLS 握手协议

FS-MAKEP 协议是一种适用于无线移动场景的安全，有效的双向认证密钥

交换协议。把实现的 FS-MAKEP 嵌入到 WTLS 握手协议当中，完整地解决了第二章指出的 4 个漏洞。

基于 FS-MAKEP 的 WTLS 完整握手过程(见图 3-3)如下，图中所用的符号定义请见 3.1 和 3.2.2 节：

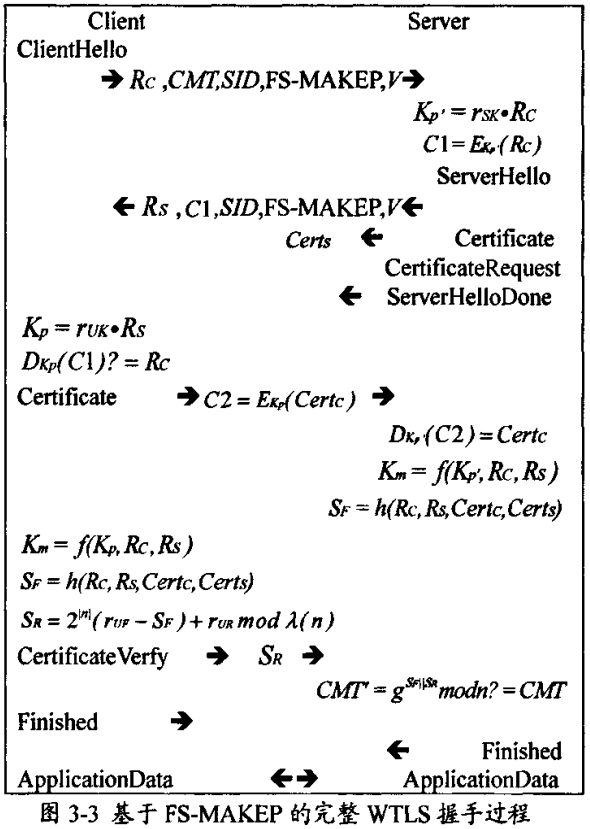

图 3-3 基于 FS-MAKEP 的完整 WTLS 握手过程

1. Client 发送 ClientHello 消息，它包含了提议的 FS-MAKEP 认证方案，客户端预计算的随机数 R_C 以及用于认证的 CMT 值。

2. Server 计算出预主密钥 $K_{P'}=r_{SK}\bullet R_C$，用预主密钥加密 R_C，得到密文 $C1$。

3. 如果 Server 同意使用 FS-MAKEP 认证方案，它将发送包含 FS-MAKEP 认证方案，预计算的随机数 R_S，和密文 $C1$ 的 ServerHello 消息。

4. 然后 Server 通过 Certificate 消息发送自己的身份证书 $Certs$，并发送 CertificateRequest 消息请求认证客户端的身份信息；最后，Server 发送 ServerHelloDone 消息，并等待来自 Client 的响应。

5. 客户端算出本次 WTLS 握手的预主密钥 K_P，然后以 K_P 为解密密钥，用对称解密函数 $D_K()$ 解密 $C1$，得到 R'_C。如果 $M1$ 和 $M2$ 都能成功到达对方的话，$R'_C=D_{K_P}(E_{K_P}(R_C))=R_C$ 是成立的。这样，客户端隐式地认证了 Server。

6. 然后 Client 用预主密钥加密自己的身份证书 $Certc$ 得到 $C2$，并通过 Certificate 消息发送给服务器端。

7. 服务器端 S 解密 C2，得到 $Certc$，然后计算杂凑值 $S_F = h(Rc, Rs, Certc, Certs)$，并根据 WTLS 密钥推导公式计算出主密钥 K_m。

K_m= PRF(pre_master_secret, "master secret", ClientHello.random+ServerHello.random)

公式(3-6)

8. Client 计算出杂凑值 $S_F = h(Rc, Rs, IDu, IDs)$，$S_R$ 和主密钥 K_m，并且通过 CertificateVerify 消息发送 S_R 给 Server。

9. Server 计算 $CMT' = g^{S_F \| S_R} \bmod n$。根据陷门散列函数性质，如果前几步信息传递过程无误的话，CMT 应该等于 CMT'的值。由此，Server 隐式地验证了 Client 的身份。

10. 最后，双方互相发送 Finished 消息，开始数据交换。

根据 3.2.3 小节中对 FS-MAKEP 的安全性分析可知，FS-MAKEP 是一种能够提供向前安全性，抵抗未知密钥攻击，中间人攻击等一系列主动攻击的安全的密钥交换和双向认证协议，把它应用到 WTLS 握手协议当中，解决了协议中原有的安全问题。

与原有 WTLS 握手协议在性能上的对比请参见第五章的详细分析。

3.4 基于 FS-MAKEP 的 WTLS 握手协议的形式化分析

WTLS 协议的安全性到底如何，必须通过形式化分析的方法，才能得到最令人信服的结论。本节以 Rubin 逻辑为基础，对基于 FS-MAKEP 的 WTLS 握手协议进行了形式化分析。

3.4.1 形式化分析的必要性

对 WTLS 协议的安全性分析，有两个方面的重要意义：

1. 可以严格地论证 WTLS 协议的安全性，使用户用得放心。

2. 形式化分析 WTLS 协议的经验，今后可以作为对其他不同类型的协议进行形式化分析的参考。

3.4.2 形式化分析工具 Rubin 逻辑简介

Rubin 逻辑[39]是 A.D.Rubin 在其博士论文中提出的一种分析安全协议的新方法，适用于分析非单调的密钥协议。Rubin 逻辑有以下特点：

1. 是第 1 个能非单调推理知识的方法；

2. 在规范协议时无需理想化过程；

3. 对协议的规范与协议的具体实现十分接近；

4. 将协议的规范与分析过程紧密结合。

而基于 FS-MAKEP 的 WTLS 协议正是这样一种非单调的密钥协议，具体地说，在建立一个新会话的情况下，当客户端和服务器端用预主密钥生成主密钥之后，二者都忘掉预主密钥。因此客户端和服务器端关于预主密钥的知识，开始时是增加，后来是减少。所以，Rubin 逻辑正是一个满足这种需求的形势化分析工具。

3.4.3 应用 Rubin 逻辑规范基于 FS-MAKEP 的 WTLS 协议

在分析协议之前，首先规范协议。Rubin 逻辑[39],[40]通过协议的局部集合和全局集合规范协议，全局集合中包括主体、推理规则、协议中出现的秘密和这些秘密的可能的观察者。Rubin 逻辑定义了“动作”，用于描述主体的操作。每个主体都有一个“动作”列表和两个局部集合：一个是由主体的知识组成的拥有集合；另一个是由主体的信念组成的信念集合。Rubin 逻辑用“动作”处理知识，用推理规则推理信念。各集合的关系如图 3-4 所示：

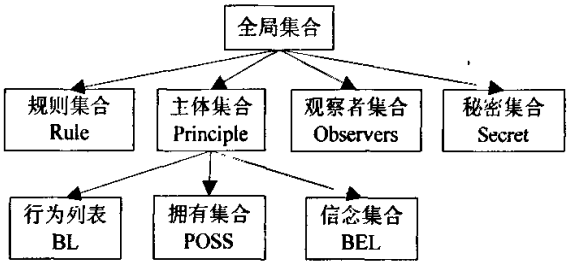

图 3-4 Rubin 逻辑规范结构图

主体集合：该集合包含参与协议运行的所有主体，在 WTLS 握手协议中，具体定义为 P={C, S}，C 代表客户端，S 代表服务器端。

规则集合：该集合包含所有推理规则，在 WTLS 握手协议中，具体定义为 R={RULES, Signature rule, Verify rule}。

秘密集合：该集合包含在任何给定时间中，在系统中存在的所有秘密，例如：会话密钥，在此具体定义为 S={}

观察者集合：这个集合里包含了知道秘密 S 的所有主体；它们可能是生成 S 的主体，也可能是通过监听网络而得到 S 的主体，在此具体定义为 Observers(S)={}。

拥有集合：该集合包含主体 P 所知道的或所拥有的与安全相关的所有数据，例如：加密密钥。记作 POSS(P_i)。

信念集合：该集合包含主体 P 所有相信的事务，例如：某一密钥是安全的信念、某个主体拥有某条消息的信念等。记作 BEL(P_i)。

行为列表：该集合包含零个或多个 P_i 执行的“动作”。这些“动作”包括加密解密、删除信息、调用函数等操作。

基于 FS-MAKEP 的 WTLS 握手协议中用到了 Rubin 逻辑定义的 12 个动作，如表 3-4 所示：

表 3-4 Rubin 逻辑中的行为列表

动作	动作描述
Encrypt(X,k)	如果主体拥有消息 X，并且知道密钥 k，则主体可以拥有 $\{X\}_k$
Decrypt($\{X\}_k$,k)	如果主体拥有消息 $\{X\}_k$，并且知道密钥 k，则主体可以拥有 X。
Generate-Nonce(N)	主体生成临时值 N，发出一个请求且等待一个响应。当主体收到一个响应时，将 LINK(N)从 BEL(P)中删除。
Generate-secret(s)	主体生成一个秘密数据项 s(例如密钥)。这时，将 s 加入秘密集合 S 之中，并同时更新观察者集合、拥有集合和信念集合。
Concat($X_1, X_2, ..., X_n$)	主体用子消息 $X_1, X_2, ..., X_n$ 构造消息 X。
Split(X)	主体将消息 X 拆分成子消息 $X_1, X_2, ..., X_n$。
Forget(X)	主体不再拥有 X。
Forget-secret(s)	主体不再知道秘密 s。
Apply(f,X)	主体对 X 应用函数 f 之后，主体就拥有 f(X)。
Abort	在协议规范中出现“不一致”或其他缺陷时，这个“动作”可能发生。
Generate-key-pair(k^+,k^-)	主体生成公钥和私钥对。
Applt-asymkey(X,k)	对 X 进行非对称加密操作。有两种情形：若 X 形如 $\{Y\}_{k'}$，且 k 与 k' 互逆，则此操作恢复出 Y；否则用密钥 k 加密 X。

为了更准确地应用 Rubin 逻辑分析基于 FS-MAKEP 的 WTLS 协议，本文对 Rubin 逻辑的“行为列表”进行了扩展，定义了以下函数专门用于分析 WTLS 协议，如表 3-5 所示：

表 3-5 基于 FS-MAKEP 的 WTLS 协议中扩展的行为列表

动作	动作描述
Generate-keys(X_1, X_2, X_3, X_4)	通过客户端的随机数，服务器端的随机数，客户端的身份证，服务器端的身份证生成杂凑值。
Choose-ciphersuite(X)	从客户端提供的 CipherSuite 列表中选择一个 CipherSuite。
Match(X_1, X_2)	检查 CipherSuite 是否被包含在 CipherSuite 列表中。
Finished(P, X_1, X_2)	对 master-secret 和已发送的其他消息进行 hash 运算，生成 Finished 消息。
$C_V(X_1, X_2)$	对 S_F 和已发送的其他消息进行运算，生成 Client Certificate Verify 消息。
Generate-msg(X)	通过两个随机数和客户端的公钥对生成临时陷门信息 X。
Compare(X_1, X_2)	比较两个值是否相等，不相等的话则中断协议。

根据以上分析说明，对基于 FS-MAKEP 的 WTLS 握手协议规范结果如下：

(1) 全局集合：

P={*C*, *S*}　　/*主体集合：C 和 S 分别表示客户端和服务器端*/

R={RULES, Signature rule, Verify rule}

/*规则集合：RULES 代表推理规则中的前 6 个*/

S={}　　/*秘密集合*/

Observers(*S*)={}　　/*观察者集合*/

Trust Matrix=$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$　　/*可信关系矩阵*/

(2) 局部集合：

注：以下，用 k^+ 和 k^- 分别标记公钥和私钥。

Principal *C*　　/*客户端的规范集合*/

POSS(*C*)={$k_{CA}^{+} \infty CA$, ID_U, $k_C^{+} \infty C$, k_C^{-}, ciphersuitesc, Finished(), Generate-keys(), Match(), $C_V()$, Generate-msg()}　　/*∞表示密钥的绑定*/

BEL(*C*)={#(k_{CA}^{+}), #(k_C^{-}), #(k_C^{+}), #(ID_U)}　/*客户端信念集合*/

Bindings(*C*)={$k_C^{+} \infty C$, $k_{CA}^{+} \infty CA$}　　/*客户端密钥绑定集合*/

BL(*C*)=　　/*客户端行为列表*/

Generate-Nonce(R_C)

Generate-Nonce(r_{UF})

Generate-Nonce(r_{UR})

Generate-msg(*CMT*)

Concat(R_C ,ciphersuitesc ,*CMT*)

Send(S,{ R_C ,ciphersuitesc ,*CMT* })

Update({ R_C ,ciphersuitesc ,*CMT* })

Receive(S,{ R_S ,ciphers, ID_S , $\{R_C\}_{PMS}$,CertificateRequest, HelloDones })

Split({ R_S ,ciphers, ID_S ,$\{Rc\}_{PMS}$,CertificateRequest, HelloDones })

Apply(Match, { ciphersuitesc , ciphers })

Generate-Secret(*PMS*)

Decrypt($\{Rc\}_{PMS}$,*PMS*)

Encrypt(ID_U ,*PMS*)

Generate-Secret(*MS*)

Forget-Secret(*PMS*)

Apply(Generate-keys,{ R_C , R_S , ID_U , ID_S })

Apply(Cv,{Generate-keys,SentM})

Apply(Finishedc ,{Client,MS,Sent-Messages})

Send(S,Concat($\{ID_U\}_{PMS}$,Cv(Generate-keys,SentM), ChangeCipherSpec, Finishedc))

Update({ $\{ID_U\}_{PMS}$,{Cv(Generate-keys,SentM),ChangeCipherSpec, Finishedc })

Receive(S,{ChangeCipherSpec, Finisheds })

Split({ChangeCipherSpec, Finisheds })

Principal S　　/*服务器端规范集合*/

POSS(S)={ $k^{+}_{CA} \infty CA$, ID_S , $k^{+}_{S} \infty S$, k^{-}_{S} ,Finished(),Generate-keys(),Choose-ciphersuite(),Generate-msg(),Compare()}　/*服务器端拥有集合*/

BEL(S)={#(k^{+}_{CA}),#(k^{-}_{S}), #(k^{+}_{S}),#(ID_S)}　/*服务器端信念集合*/

Bindings(S)={ $k^{+}_{S} \infty S$, $k^{+}_{CA} \infty CA$ }　/*密钥绑定集合*/

BL(S)=　/*服务器端行为列表*/

Receive(C,{ R_C , ciphersuitesc ,*CMT*})

Split({ R_C ,ciphersuitesc ,*CMT*})

Apply(Choose-ciphersuite(),{ ciphersuitesc })

Generate-Nonce(R_S)

Generate-Secret(*PMS*)

Encrypt(R_C ,*PMS*)

Concat(R_S , ciphers, $\{R_C\}_{PMS}$,CertificateRequest, HelloDones)

Send(C,{ R_S , ciphers, IDs, $\{R_C\}_{PMS}$,CertificateRequest, HelloDones })

Update({ R_S , ciphers, IDs, $\{R_C\}_{PMS}$,CertificateRequest, HelloDones })

Receive(C,{ $\{ID_U\}_{PMS}$, C_V (Generate-keys,SentM),ChangeCipherSpec, Finishedc })

Split({{ $\{ID_U\}_{PMS}$, C_V (Generate-keys,SentM),ChangeCipherSpec, Finishedc })

Decrypt({ ID_U } $_{PMS}$,*PMS*)

Generate-Secret(*MS*)

Forget-Secret(*PMS*)

Apply(Generate-keys,{ R_C , R_S , ID_U , *IDs* })

Generate-msg(*CMT'*)

Compare(*CMT*, *CMT'*)

Apply(*Finisheds* ,{Server,*MS*,Sent-Messages})

Send(C, {ChangeCipherSpec, Finisheds })

Update({ChangeCipherSpec, Finisheds })

3.4.4 应用 Rubin 逻辑分析基于 FS-MAKEP 的 WTLS 协议

协议规范完成后，就可以开始分析协议。协议分析方法和过程如图 3-5 所示：

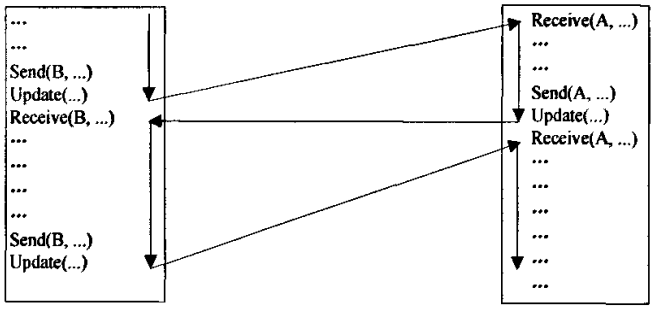

图 3-5 应用 Rubin 逻辑分析协议时的控制流程

在 Rubin 逻辑中，对协议的分析从协议发起者的“动作”列表中的第一个“动作”开始。这个“动作”执行完毕后，如果某个推理规则的条件被满足，则对这个主体的信念集合应用这个推理规则。接着，执行同一“动作”列表中的下一个“动作”。对每个 Send 消息操作，应用 Update 函数更新网上传送的秘密的观察者集合。然后，分析移至 Send 中接收消息主体的“动作”列表中的第 1 个未执行的 Receive 操作。随着分析的进展，拥有集合、信念集合、秘密集合、和观察者集合都被更新。分析结束后，所有的“动作”都应被标记为已执行过。在分析过程中的任何一步，都可检查出协议的安全缺陷。

1. 基于 FS-MAKEP 的 WTLS 握手协议分析过程

规范完成后，协议分析从 C 的“行为列表” BL(C)中的第 1 个“动作”开始。BL(C)的前 4 个“动作”执行完毕后，集合 POSS(C)和 BEL(C)中增加了新的元素。Update 操作使得 Observers(R_C)=W(W 表示所有的主体)。至此，没有使用任何推理规则。

POSS(C)={ $k_{CA}^{+}\infty CA$, ID_U , $k_C^{+}\infty C$, k_C^{-} ,CMT, ciphersuitesc , R_C , r_{UF} , r_{UR} , Finished() ,Generate-keys(),Match(), C_V (),Generate-msg()}

BEL(C)={#(k_{CA}^{+}),#(k_C^{-}), #(k_C^{+}), #(ID_U), LINK(R_C), LINK(r_{UF}), LINK(r_{UR}), #(CMT)}

BL(C)=

Generate-Nonce(R_C)

Generate-Nonce(r_{UF})

Generate-Nonce(r_{UR})

Generate-msg(CMT)

Concat(R_C ,ciphersuitesc ,CMT)

Send(S,{ R_C ,ciphersuitesc ,CMT})

Update({ R_C ,ciphersuitesc ,CMT})

Update 操作后，下一个要执行的“动作”是 Send 中指定的 S 的“行为列表” BL(S)中的 Receive。

Receive(C,{ R_C ,ciphersuitesc ,CMT})

BL(S)中的前 9 个“动作”执行完毕后，仍没有可以使用的推理规则。S 的局部集合的新的取值如下：

POSS(S)={ $k_{CA}^{+}\infty CA$, ID_S , $k_S^{+}\infty S$, k_S^{-} , R_C from C, ciphersuitesc from C, CMT from C, R_S ,PMS, $\{R_C\}_{PMS}$, { R_S , ciphers , $\{R_C\}_{PMS}$, CertificateRequest, HelloDones }, Finished(),Generate-keys(), Choose-ciphersuite()}

BEL(S)={#(k_{CA}^{+}),#(k_S^{-}),#(k_S^{+}),#(ID_S),LINK(R_S) ,#($\{R_C\}_{PMS}$),#(PMS)}

Bindings(S)={ $k_S^{+}\infty S$, $k_{CA}^{+}\infty CA$ }

BL(S)=

Receive(C,{ R_C ,ciphersuitesc ,CMT})

.

.

.

Update({ R_S , ciphers , ID_S , $\{R_C\}_{PMS}$,CertificateRequest, HelloDones })

下一个要执行的“动作”在 BL(C)中。

Receive(S,{ R_S , ciphers , ID_S , {R_C}$_{PMS}$,CertificateRequest, HelloDones })

接着，随后的 3 个“动作”被执行。

Split({ R_S , ciphers , ID_S , {R_C}$_{PMS}$,CertificateRequest, HelloDones })

Apply(Match, { ciphersuitesc , ciphers })

Generate-Secret(*PMS*)

ID_S , {R_C}$_{PMS}$ 和 R_S 加入到集合 POSS(*C*)之中，$k_s^+ \infty S$ 加入到集合 Bindings(*C*)之中。*C* 相信 *S*，因此有 TRUST[1,2]=1。下一个要执行的“动作”是：

Decrypt({Rc}$_{PMS}$,*PMS*)

此时，用于对称加密的子消息来源规则的条件被满足，所以{R_C}$_{PMS}$ from *S* 加入到集合 POSS(*C*)之中。并且，签名规则的条件也被满足，应用这一规则后，#({R_C}$_{PMS}$)(*S* 对 R_C 的加密是新鲜的)加入到集合 BEL(*C*)之中，LINK(R_C)从 BEL(*C*)集合中删除。然后，*C* 执行下面 2 个“动作”：

Encrypt(ID_U ,*PMS*)

Generate-Secret(*MS*)

PMS,*MS* 和{IDu}$_{PMS}$ 加入到集合 POSS(*C*)之中。*PMS* 和 *MS* 加入到秘密集合 *S* 之中。#(*PMS*)和#(*MS*)加入到集合 BEL(*C*)之中。下一个要执行的动作是：

Forget-Secret(*PMS*)

这个“动作”使 *PMS* 从集合 POSS(*C*)中删除，#(*PMS*)从集合 BEL(*C*)中删除，随后，执行以下 5 个“动作”：

Apply(Generate-keys,{ R_C , R_S , ID_U , ID_S })

.

.

.

Update({ {IDu}$_{PMS}$,{ C_V (Generate-keys,SentM), Finishedc , ChangeCipherSpec})

Generate-keys， C_V (Generate-keys,SentM)， Finishedc 加入到集合 POSS(*C*)之中。下一个将要执行的“动作”在 *S* 的“行为列表”之中。

Receive(*C*,{ {IDu}$_{PMS}$, C_V (Generate-keys,SentM),ChangeCipherSpec, Finishedc })

然后，执行下面 1 个“动作”：

Split({{{IDu}$_{PMS}$, C_V (Generate-keys,SentM),ChangeCipherSpec, Finishedc })

{IDu}$_{PMS}$ ， C_V (Generate-keys,SentM)， Finishedc 和 k_C^+ 加入到集合 POSS(*S*)之中，$k_C^+ \infty C$ 加入到集合 Bindings(*S*)之中。*S* 相信 *C*，因此有 TRUST[2,1]=1。下面要执行的“动作”是：

Decrypt({ IDu } $_{PMS}$,*PMS*)

Generate-Secret(*MS*)

执行此操作后，*MS* 和 ID_U 加入到集合 POSS(*S*)之中。 #(*MS*)加入到集合 BEL(*S*)中。下一个要执行的操作是：

Forget-Secret(*PMS*)

这个“动作”使 *PMS* 从集合 POSS(*S*)中删除，#(*PMS*)从集合 BEL(*S*)中删除。随后执行以下 6 个动作：

Apply(Generate-keys,{ R_C , R_S , ID_U , ID_S })

Generate-msg(*CMT'*)

Compare(*CMT, CMT'*)

Apply(Finisheds ,{Server,*MS*,Sent-Messages})

Send(*C*, {ChangeCipherSpec, Finisheds })

Update({ChangeCipherSpec, Finisheds })

Generate-keys，*CMT'*，Finisheds 加入到集合 POSS(*S*)之中。#(*CMT'*)加入到 BEL(*S*)中。

下一个将要执行的“动作”在 *C* 的“行为列表”之中。

Receive(*S*,{ChangeCipherSpec, Finisheds })

Split({ChangeCipherSpec, Finisheds })

Finisheds 加入到 POSS(*C*)之中。

2. 分析结论

至此，整个协议分析结束，所有的动作都被执行，说明协议自身不存在任何问题。在 Rubin 逻辑分析过程中，将协议的规范和分析结合在一起，随着协议的进展，代表系统当前状态的知识和信念集合不断更新，更新的过程符合 Rubin 推理规则，没有出现异常和数据不一致性。因此，从理论上分析，基于 FS-MAKEP 的 WTLS 握手协议是安全的。

3.5 基于 FS-MAKEP 的 WTLS 握手协议的设计与实现

本节完整地实现了改进后的 WTLS 握手协议，包括 WTLS 协议中的密钥交换到会话建立的整个通信过程。

3.5.1 WTLS 握手协议的实现特点及功能设计

本文所实现的 WTLS 协议在通信双方进行安全数据传输之前，通过一次完整的握手在客户端和服务器端之间建立一个端对端的会话(session)，确定本次安全连接的安全参数、数据加密密钥和算法。

实现过程具有以下特点：

1. 一个会话可支持一个或多个并发的安全连接，会话参数可以被重新使用，从而节约了重新握手时间。

2. 可以支持完整的握手和简化握手。

3. 存贮了私有数据，比如安全参数列表和会话记录等。

4. 定义了一系列函数完成信息封装和提取。

首先我们根据所提供的功能画出 Use Case 图，如图 3-6 所示：

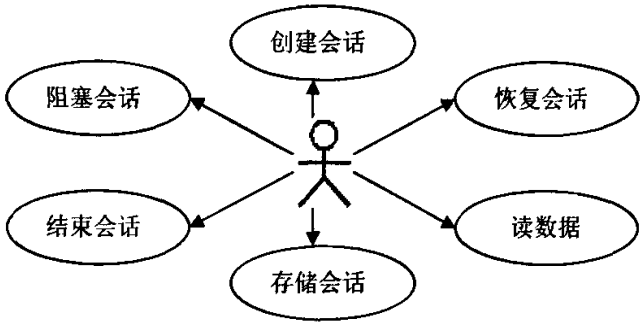

图 3-6 WTLS 协议 use case 图

由图可知，本文设计的 WTLS 协议通信部分主要具备六个功能：

1. new：创建会话

这个功能是 WTLS 中最复杂和最重要的，目的是建立服务器端与客户端之间的新连接，确定本次安全连接的安全参数（如双方随机数，选择的加/解密算法、密钥交换算法、压缩算法等），认证服务器端和客户端双方的身份，计算本次会话的主密钥。

2. suspend：阻塞会话

用于暂时停止协议的执行，即挂起线程，直到另一个线程调用 resume()方法时该线程才会重新启动。

3. resume：恢复会话

resume 调用实际上是 new 的一种特殊情况，和 new 的调用关系及事件流类似。它的作用是恢复一个已经被阻塞的会话。

4. end：结束会话

调用者调用 end 用于断开 WTLS 连接，指示结束当前的 WTLS 会话。系统同时生成 e_alert 报警信息，并把 e_alert 报警信息用 send_alert()发送。

5. read：读数据

调用者调用 read 可能会有两种情况：可能是握手协议调用以读取握手报文；也可能是用户调用以读取系统命令。

6. session_record_op：存储会话状态

为保存一个 WTLS 连接的状态信息，引入一个 WTLS 状态对象：WTLS_SESSION_RECORD 。用户调用每个功能后都通过 WTLS_SESSION_RECORD 对象把新的 WTLS 会话状态存储到 hash 表中。

3.5.2 WTLS 握手协议流程设计

整个握手过程可以分为四个阶段：

Hello：客户端向服务器端提出连接请求，服务器端向客户端响应请求。

KeyExchange：密钥交换过程，生成主密钥。

Certificate：双方认证彼此身份信息。

Finished：双方交换提交协商信息，通知对方协商的算法和密钥开始生效。

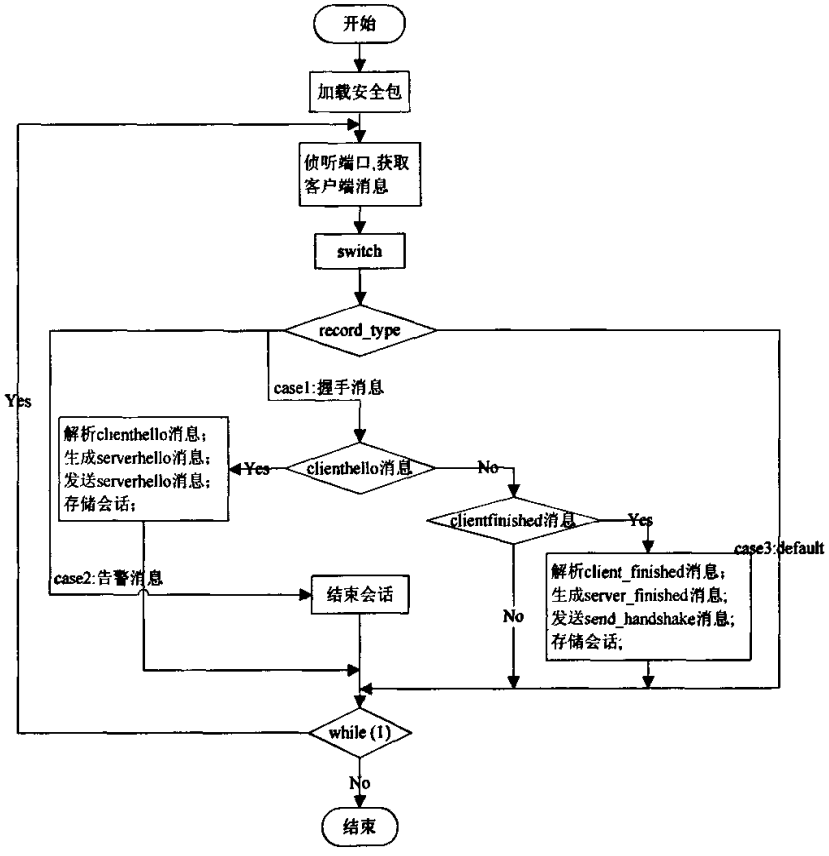

图 3-7 WTLS 握手协议服务器端流程图

由于整个通信流中调用的分支比较复杂，这里只分析主要分支的情况。

1. 服务器端事件流如下：

• 初始化当前连接状态和算法等。

• 提取当前系统时间作为随机数。

• 读取客户端请求。

• 从字节流中解码出消息类型。

• 如果是 ClientHello 消息，则选择加密算法，压缩算法，封装成 ServerHello 消息。

• 把 ServerHello 消息发送给客户端。

• 从证书管理对象 WTLS certificate 读取自己的证书发送给客户端。

• 收到 ClientFinished 消息后，解封该消息，得到新的规则和密钥。

• 生成 ServerHelloDone 消息。

2. 客户端事件流如下：

• 初始化当前连接状态和算法。

• 读取安全配置信息。

• 提取当前系统时间作为随机数。

• 生成 ClientHello 消息。

• 向服务器端发送请求。

• 读取服务器端响应。

• 从字节流中读取 ServerHello。

• 解析 ServerHello 消息。

• 从字节流中读取 Certificate。

• 调用 WTLS certificate 对象对服务器端证书进行验证。

• 调用 WTLS_SESSION_RECORD，把双方随机数，选择的加密，压缩算法等参数值存储至 hash 表。

• 生成 ClientFinished 消息。

• 发送 ClientFinished 消息。

• 读取服务器端响应。

• 从字节流读取 ServerHelloDone。

• 把候选连接状态和加密参数等切换为当前连接状态。

• 插入新的会话记录到 hash 表和 rpbuffer。

客户端中的一个分支 resume 事件流如下：

• 客户端通过 current_sessionID 查找等待恢复的会话。

• 提取当前系统时间作为随机数

• 生成 ClientHello 消息。

• 发送请求。

• 读取服务器端响应。

• 从字节流中读取 ServerHello。

• 解析 ServerHello，并选择算法。

•恢复会话，把会话的 suspend 状态转为 open 状态，调用 session_record_op 把新的会话参数存入 hash 表。

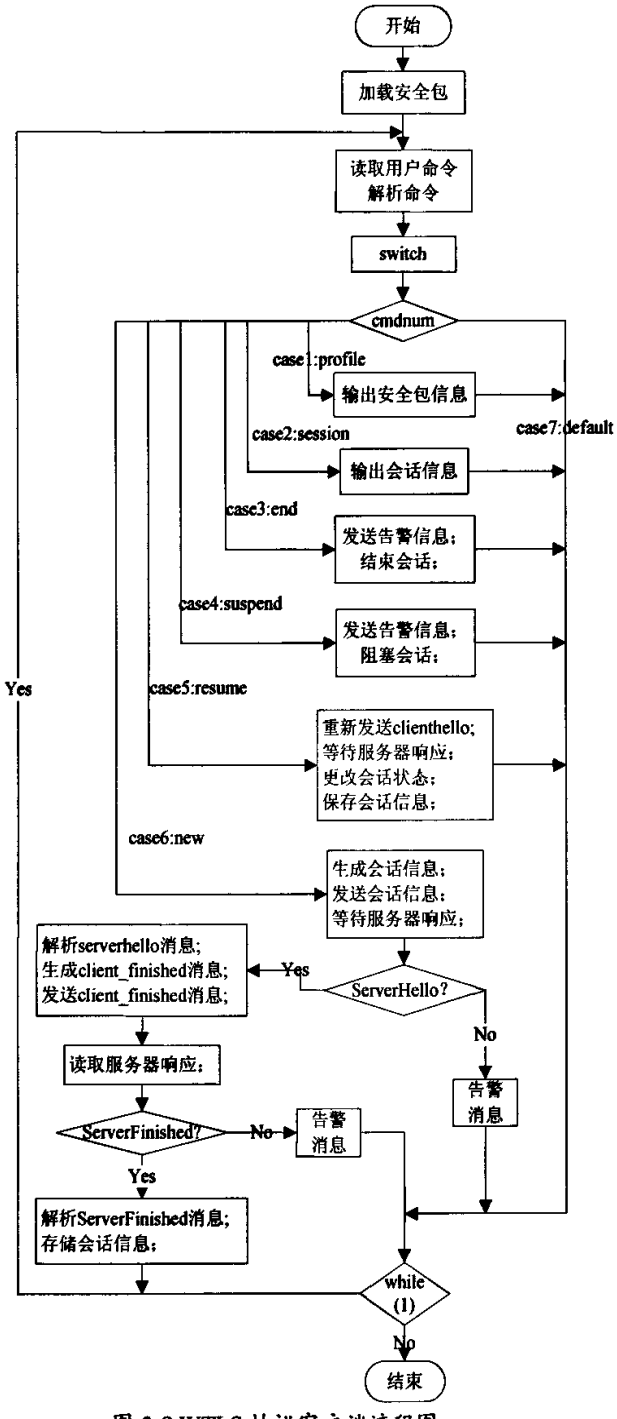

图 3-8 WTLS 协议客户端流程图

3.5.3 实验结果及分析

由于实验环境有限，无法在真正的无线网络环境中实现无线终端与服务器端的通信过程，只能在 PC 机上模拟 WTLS 协议的通信过程，具体运行环境为：P4 CPU 2.8GHz，1G 内存的 PC 机，Linux 操作系统，采用标准 C 代码实现。

由于整个协议是在 PC 机上实现的，PC 的计算能力比移动终端强，且有线网络性能比无线网络要好，所以在完成时间上会比实际的无线通信要短。经过多次实验，得到一个完全握手过程的平均完成对间为 71.26ms。协议的实现证明了改进后的协议是可行的。

3.6 本章小结

提出了一种名为 FS-MAKEP(Forward Security-Mutual Authentication Key Exchange Protocol)的双向认证密钥交换协议，详细阐述了其密钥交换和双向认证的过程和原理。经证明该协议不仅能提供很高的安全性，而且在计算开销和通信载荷上也比普通协议具有更大的优势，是一种适合应用于无线通信的密钥交换和双向认证协议。把 FS-MAKEP 应用到 WTLS 握手协议中，得到了一种更加安全的 WTLS 握手协议，该协议能提供向前安全性，并且能够防止未知密钥共享攻击，中间人攻击，并能提供用户身份保护的第一安全等级。

Rubin 逻辑是一种适用于分析非单调的密钥协议安全性的新方法，本章利用 Rubin 逻辑的特点，对 Rubin 逻辑进行了函数上的扩展，然后用扩展后的 Rubin 逻辑对改进后的 WTLS 协议进行了形式化分析。经证明，改进后的 WTLS 协议是安全的。最后，在 Linux 系统下，分析并简单实现了改进后的 WTLS 协议的通信过程，对 WTLS 的运行速度进行了测试。

第四章 基于 AL-SA 的 WTLS 握手协议

为防止用户匿名性攻击，并且更好的提供安全性与匿名性兼顾的安全服务，本文提出了一种基于匿名通行证的签名认证方案 AL-SA(Signature Authentication based on Anonymity License)，该方案在无线互联网中为用户提供服务时，既能实现对用户身份认证，也保证了用户信息不被服务器端获取，即满足了用户身份保护的第二安全等级。

在本方案中，基于无线网络的带宽有限，和无线设备的处理能力低的限制，我们引用了曾经在 2001 年被 Adi Shamir 和 Yael Tauman 提出的在线/离线签名算法[33]来实现认证中心对用户证书的签名认证过程。

4.1 基于陷门散列函数的在线/离线签名算法介绍

针对签名效率问题，1990 年，Even.Goldreich、MicaIi 首先提出了在线/离线签名[41]的概念。他们提出把签名分成两阶段：离线阶段和在线阶段。在离线阶段，待签名的消息确定前，先把计算量大的一部分运算进行预计算。在在线阶段，对要签名的消息结合预计算结果进行在线签名。这样可以有效地加快在线签名的速度。

陷门散列函数[42]由 Krawczyk 和 Rabin 提出，用来构造 Chameleon 签名算法。该函数与指定接收者的一对公钥/私钥有关，公钥参与散列值计算，私钥就是函数的陷门信息。任何知道接收者公钥的人都能计算散列值。但只有掌握陷门信息的人，也就是说只有指定接收者才能计算碰撞。

2001 年 Adi Shamir 和 Yael Tauman 对在线/离线算法作出改进[33]，使得在线签名速度仅为 0.1 个模乘，改进后的算法由下列有效算法组成：

1. 计算散列值算法：输入指定接收者 R 的需要签名的消息 m、一个随机整数 r、公钥 HK，输出散列值 $H=hash_{HK}(m,r)$。

2. 计算碰撞算法：输入 R 的私钥 TK、消息 m，另一消息 m'、随机整数 r、输出随机整数 r'，满足 $hash_{HK}(m',r')=hash_{HK}(m,r)$。

把该改进方案中的陷门散列函数运用到普通的签名算法中，就可以使普通的签名算法的在线计算签名速度提高为 0.1 个模乘。具体地说，Adi Shamir 提出的签名方案如下：

假设存在一个普通的签名认证算法，各符号定义如下：

SIG()：签名算法

V()：认证算法

(*SK,VK*)：一对签名认证密钥

(*HK,TK*)：陷门散列函数的一对公私钥

(*SK,HK,TK*)： 新算法的签名密钥

(*VK,HK*)：新算法的认证密钥

1. 离线阶段：

(1) 签名者 *S* 输入随机消息 *m'*、指定接收者 *R* 的公钥 *HK* 和随机数 *r'*，然后计算出散列值 $H=hash_{HK}(m',r')$。

(2) 签名者 *S* 用 *SK* 对散列值加密，得 $SIG_{SK}(H)$。

(3) 然后储存(*m',r'*)， $hash_{HK}(m',r')$，以及签名 $SIG_{SK}(H)$。

2. 在线阶段：

(4) 签名者 *S* 在收到接收者 *R* 要签名的真正信息 *m* 后，计算 *r*，满足 $hash_{HK}(m',r')=hash_{HK}(m,r)$，然后把(*r*， $SIG_{SK}(H)$)当作对消息 *m* 的签名。该签名的计算过程只要 0.1 个模乘。

3. 认证阶段

当签名者 *S* 使用签名方案为指定接收者对某消息进行签名后，因为陷门信息 *TK* 只有签名者知道，包括接受者在内的其他人都不可能通过数字签名获取陷门信息，因而只有签名者才可以利用陷门信息计算碰撞，由此保证了签名的有效性。

(5) 具体地说，当接收者接到签名者对消息的签名(*r*， $SIG_{SK}(H)$)后，通过认证密钥 *VK* 计算出 $hash_{HK}(m',r')$，同时通过 *HK*，*r*，*m* 计算 $hash_{HK}(m,r)$，比较 $hash_{HK}(m',r')$ =？ $hash_{HK}(m,r)$，由此验证了此签名的真实性。

Adi Shamir 最后对该改进算法和已存在的两个陷门散列函数在性能上作了对比，证明新算法的在线签名速度是其他算法的 10 倍，仅相当于 0.1 个模乘的计算量，而且能防止自适应性选择密文攻击[33]。

4.2 基于匿名通行证的签名认证方案 AL-SA

本文引用了基于陷门散列函数的在线/离线签名算法，提出了基于匿名通行证的签名认证方案 AL-SA。该方案必须满足如下的性质：

1. 安全性。方案安全性建立在WPKI(Wireless Public Key Infrastructure)的基础之上，用户证书和通行证都不能伪造。

2. 匿名性。通行证不会泄漏任何用户的身份信息。

3. 不可追踪性[43]。即使服务提供商与认证中心联合起来，也不能确定某用户是否接受了服务提供商提供的服务。

该方案中用到的匿名通行证与身份证书是有区别的，区别如下：

身份证书是经证书颁发机构 CA 认证后，颁发给用户的公钥证书。身份证书

包括证书申请者的信息和发放证书 CA 的信息。

匿名通行证是第三方认证中心通过认证身份证书后，给用户颁发的可以证明用户身份合法的证书，通行证中不包含证书申请者的信息。服务器通过匿名通行证只能判断用户的身份是否合法，不能获得用户的真实身份信息。

4.2.1 AL-SA 设计基本思路

该方案设计的基本思路是：定义一个用户端和服务提供商共同信任的认证中心 SA，S_r 与 V_r 分别是 SA 的签名算法和认证算法，V_r 向服务器端公布。认证中心核实用户证书信息，然后给用户颁发一个匿名通行证 License。服务提供商验证此通行证，然后向用户提供所需的服务。在该验证过程中，用户不必担心暴露自己的身份。图 4-1 描绘了上述条件下的用户网络。

注：‖ 表示级联。

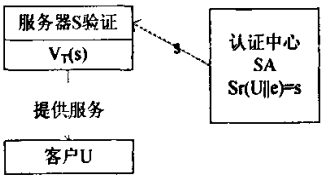

图 4-1 由认证中心签名的认证

新方案中，服务提供商如果要验证客户端的身份，不需要亲自验证客户端证书，只需要验证 SA 签名的真实性，但前提是必须先拥有 SA 的签名验证函数 V_r。为了简化问题，假定验证函数的真实性是以非密码方式来提供的，比如服务提供商亲自从 SA 处获得验证函数。然后服务提供商可以执行下列步骤：

步骤 1：从客户端获得 SA 的签名。

步骤 2：使用 SA 的验证函数来验证 SA 在客户证书上的签名。

步骤 3：若验证后签名是正确的，则可以给客户端提供服务；否则，拒绝客户端请求。

4.2.2 可信任认证中心 SA 的特点

在文章[44]中提到，认证中心的安全性和信任关系随着使用方式的不同而起变化，因而有如下分类。

定义 1：称一个 SA 是无条件可信的，如果它在所有事情上都可信。例如，它也许可以访问用户的秘密密钥和私钥，还承担着公钥与标识符的联系。

定义 2：称一个 SA 是功能上可信的，如果它是诚实且公正的，但是不可以访问用户的私钥。

在本文提出的方案中，采用的是一个功能上可信的 SA，从而保证在认证客户证书的同时，又不能访问客户端的私密信息。在为客户端生成一个证书签名之前，SA 必须采取适当的措施对客户端的身份进行验证。方法是要求客户端提交客户证书作为身份的证明，经过验证后，对其签名。服务提供商在验证了 SA 的签名后，对 SA 的签名的真实性信任就可以传递到对客户端身份真实性的信任。

使用认证中心来保证用户身份的保密性的优点有：

1. 阻止主动攻击者在网络上的假冒。

2. 认证中心不拥有监视通信的能力。实体需要确信的是认证中心对客户端身份的签名是有效的。

4.2.3 签名过程特点

本文引用了基于陷门散列函数的在线/离线签名算法思想，设计了一个基于可信任的认证中心 SA 的用户身份签名认证方案，既验证了用户身份的合法性，也保证了用户身份隐私不被泄漏。可信任的认证中心 SA 把匿名通行证的签发过程分成了在线和离线两个阶段，离线阶段是一个预计算过程，SA 存储计算出的信息，然后在线阶段利用离线阶段存储的信息对用户身份签名，目的是让在线计算阶段的计算更快速，以适用于计算能力弱的处理器需要。并且 Adi Shamir 和 Yael Tauman 证明了该签名认证算法的在线阶段计算速度为普通算法的 10 倍，而且可以有效防止自适应性选择密文攻击[33]。

4.2.4 AL-SA 详细流程

1. 符号定义：

Client：客户端

Server：服务器端

SP：服务提供商

SA：公共信任认证中心

PK：SA 的公钥

SK：SA 的签名密钥

VK：SA 的认证密钥

TK：陷门散列函数私钥

HK：陷门散列函数公钥

S：签名函数

V：认证函数

2. 初始化过程：

假设已经存在签名认证算法(S,V)，并已经产生一对签名认证密钥(SK, VK)。SA 将基于陷门散列函数的高效性，在原有签名算法的基础上作出改进，使新算法的在线签名速度为原有算法的 10 倍，具体过程如下：

SA 随机挑选两个大素数 $p, q \in \{0,1\}^{k/2}$，且满足 $p'=(p-1)/2$ 和 $q'=(q-1)/2$ 均为大素数，$n=pq$。$g \in Z_n^*$，且是一个 $\lambda(n)$ 阶生成元 $(\lambda(n)=lcm(p-1,q-1)=2p'q')$。SA 将公钥 $PK=(n,g)$，以及一个不带密钥的哈希函数 h()作为公共信息发布，私有密钥 $TK=(p,q)$被保存。(SK,HK,TK)为新算法的签名密钥，(VK,HK)为新算法的认证密钥，S 为签名函数，V 为认证函数。并定义带密钥的单向哈希函数 $hash_{HK}(m;r)=g^{m\|r}(modn)$。

3. 签名过程：

(1) 离线预计算：

a. SA 生成随机数 $r \in_R Z_{\lambda(n)}$，及 $t \in_{R\{0,1\}^l}$，并计算哈希函数 $hash_{HK}(t;r)=g^{t\|r}(modn)$。

b. SA运行签名算法S，用签名密钥SK对哈希值 $hash_{HK}(t;r)$ 签名。这里把签名后的值 $S_{SK}(hash_{HK}(t;r))$ 定义为C。

c. SA把(t,r)，哈希值 $hash_{HK}(t;r)$，以及C存储下来以避免在以后的阶段重复计算。

(2) 在线签名：

a. Client将身份证 Certc 发送给SA，请求签名。

b. SA验证 Certc，如果有效，则计算 Certc 的哈希值 $h(Certc)$，记为M。然后从存储空间中调用出预先计算的哈希值 $hash_{HK}(t;r)$，结合M，以及TK，计算出 $r' \in R$，使得 $hash_{HK}(M;r') = hash_{HK}(t;r)$，即满足

$$g^{t\|r} = g^{M\|r'} (mod\ n) \qquad \text{公式(4-1)}$$

也就是求

$$2^k t + r = 2^k M + r' mod\ \lambda(n) \qquad \text{公式(4-2)}$$

由公式(4-2)得

$$r' = 2^k (t-M) + r (mod\ \lambda(n)) \qquad \text{公式(4-3)}$$

c. SA把License $=(r',C)$当作对Client身份的签名，即Client的通行证，发送给Client。

4. 认证过程：

(1) Client 计算身份证书Certc 的哈希值 M，与 License 一起发送给 SP。

(2) SP 为了验证(r',C)是否确实是 SA 对 Client 的真实签名，首先计算出 $hash_{HK}(M;r')$，然后通过认证密钥(VK,HK)和认证算法 V 对 C 解密，将解密后的 $hash_{HK}(t;r)$ 和 SP 重新计算的 $hash_{HK}(M;r')$ 相互对比，如果两者一致，则说明 Client 是合法用户，SA 签名有效，SP 可以开始向 Client 提供服务。否则，则说明 Client

是非法的匿名用户。

上述通行证签发过程及认证过程可表示如下：

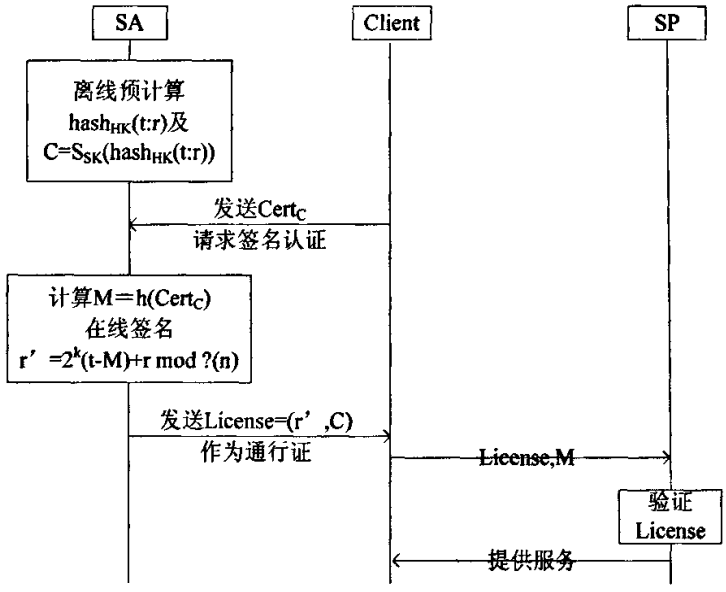

图 4-2 AL-SA 方案流程

在这里，我们使用了在线/离线签名技术。认证中心给用户颁发了代表用户合法身份的匿名通行证，而服务提供商无法从该通行证追踪到用户信息，从而保证了用户的隐私。同时，认证中心经过对用户身份证书的验证，保证了用户身份的合法性，满足了服务提供商所关心的安全性。

4.3 基于 AL-SA 的 WTLS 握手协议

4.3.1 协议流程

图 4-3 描述了基于 AL-SA 的 WTLS 完整握手过程，它与基于 FS-MAKEP 认证方案的 WTLS 握手过程类似，也采用了 ECC 算法实现密钥交换过程，不同的是：这里客户端的 Certificate 信息发送的是经过第三方认证的匿名通行证 License(r',C)，而不是客户端的身份证书；其次，不再在 ClientHello 中发送 *CMT* 值，服务器端也不用再计算 *CMT'*。服务器端通过认证 License 来认证客户端。关于匿名通行证的认证方案的具体描述可见 4.2 节。

1. Client 发送 ClientHello 消息，它包含了客户端预计算的随机数 R_C 以及会话标识号 *SID*，所支持的密钥交换规则 $SecNeg_C$ 和协议版本号 V。

2. Server 计算预主密钥 $K_{P'}=r_{SK}\cdot R_C$，用预主密钥加密 R_C，得到密文 $C1$。

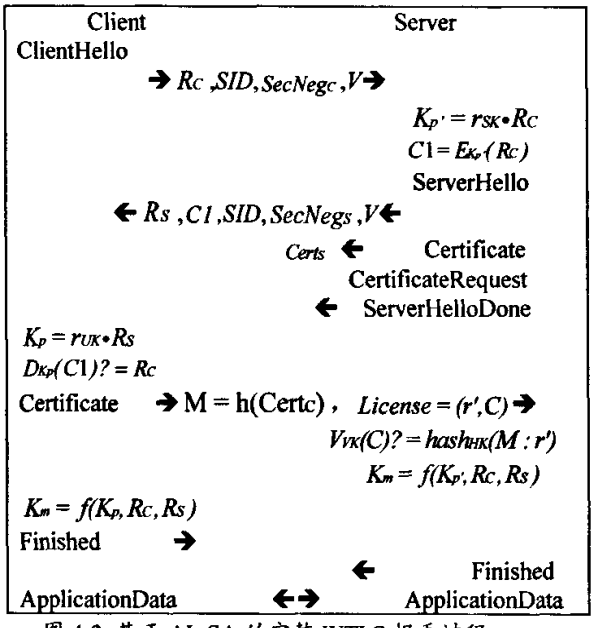

图 4-3 基于 AL-SA 的完整 WTLS 握手过程

3. Server 发送 ServerHello 消息，该消息包含预计算的随机数 R_S，密文 $C1$ 的，会话标识号 SID，所选择的密钥交换规则 $SecNeg_C$ 和协议版本号 V。

4. 然后 Server 通过 Certificate 消息发送自己的身份证书 $Certs$，并发送 CertificateRequest 消息请求认证客户端的身份信息；最后，Server 发送 ServerHelloDone 消息，并等待来自 Client 的响应。

5. 客户端计算本次 WTLS 握手的预主密钥 K_P，然后以 K_P 为解密密钥，用对称解密函数 $D_K()$ 解密 $C1$，得到 R'_C。如果 $M1$ 和 $M2$ 都能成功到达对方的话，$R'_C = D_{K_P}(E_{K_P}(R_C)) = R_C$ 是成立的。这样，客户端隐式地认证了 Server。

6. 然后 Client 用单向哈希函数计算自己的身份证书 $Certc$ 得到 M，从 SA 获得匿名通行证 License，通过 Certificate 消息发送给服务器端。

7. 服务器端验证 License，并根据 WTLS 密钥推导公式计算出主密钥 K_m。

K_m= PRF(pre_master_secret, "master secret", ClientHello.random + ServerHello.random) (3-6)

8. Client 计算主密钥 K_m。

9. 双方发送结束消息。

10. 最后，双方互相发送 Finished 消息，开始数据交换。

4.3.2 协议性能、安全性分析

1. 计算开销

离线预计算过程包括计算带密钥的单向哈希函数和一个签名算法，在线签名过程包括一个陷门信息计算和一个单向哈希函数，认证过程包含了一个带密钥的

单向哈希函数和一个认证算法。与普通的签名认证算法相比，在离线预计算阶段的运算总是发生在整个事件之前，不影响签名认证的速度，所以其开销可以忽略不计。而在线签名过程包含了一个单向哈希函数和一个陷门信息计算，带密钥的单向哈希函数的计算量是 1 个模指数运算，SA 在计算陷门信息 r'的复杂度仅相当 0.1 个模乘[33]。

2. 安全性

本方案采用基于因子分解的陷门散列函数，可以证明签名是不能伪造的[42]，因为攻击者无法得知陷门函数密钥 TK。而且方案可以防止自适应性选择密文攻击[33]。方案中的密钥交换过程采用了 FS-MAKEP 中的算法，所以也能提供向前安全性，防止未知密钥共享攻击和中间人攻击。方案通过可信认证中心 SA 对用户身份进行认证，认证中心只对满足服务提供商要求的用户颁发通行证，从而保证了服务提供商所要求的安全要求。

3. 匿名性

可以证明，即使攻击者或者服务提供商取得客户端的匿名通行证 License，他们仍然无法得到任何用户私人信息，因为 License 仅仅是 SA 对客户端的签名，并不是客户端的真实身份，除了公共信任中心 SA 以外的任何人都不能得知被签名的客户端的真实信息，因此保证了用户的匿名性，满足了用户身份保护的第二安全等级。

4. 高效性

用户在经过认证中心认证之后，每次向服务提供商请求服务时，只需要提交匿名通行证。另外，考虑到带宽或终端的计算能力，WAP 论坛[1]提出了采用 WTLS 证书，以减小证书大小。在本文提出的方案中，由于采用了通行证的方式，进一步减少了处理证书的时间，从而提高了效率。

4.4 本章小结

本章提出了一种基于匿名通行证的认证方式。整个认证过程的前提是存在一个客户端和服务提供商都信任的认证中心 SA，SA 给客户端颁发一个匿名通行证 License，服务器端无法得知关于客户端的任何信息，而是通过验证 License 的有效性来实现对客户端身份的认证。而且 SA 生成匿名通行证的过程仅为一次计算，也就是说用户在经过认证中心认证之后，以后每次向服务提供商请求服务时，只需要提交匿名通行证，SA 不需要重复计算。在计算开销上，该方案的在线计算具有比同类算法都小的计算开销，而且方案提供了安全性和保密性。

第五章 两种改进的 WTLS 握手协议综合分析

传统的 WTLS 协议规范提供的认证和密钥交换方案非常有限，它并不能完全满足移动电子交易安全要求。为了保障无线终端和服务提供商，即客户端与服务器端之间的双向认证安全，同时减轻无线终端的计算开销，本文提出了两种改进的 WTLS 握手协议：基于 FS-MAKEP 的 WTLS 握手协议和基于 AL-SA 的 WTLS 握手协议。

5.1 计算开销分析

为了更准确地评估计算开销，定义了各参数如下：

T_{ECC} ：计算 ECC 点乘运算所需的时间(如：$R_i = r_i \bullet B \bmod p_1$)

T_{MM}：计算模数乘法所需的时间

T_{ME}：计算模指数所需的时间

T_H：计算单向杂凑函数所需的时间

T_E：计算对称加密函数所需时间

T_D：计算对称解密函数所需时间

T_f：计算主密钥所需时间

根据A. Lenstra和E.Verheul在文章[34]中建议的适合实际应用的密钥大小，我们设置$|n|$=1024bits，$|l| = |ID_U| = |r_{UF}| = |S_F|$=160bits。根据文章[35]、[36]的结论得知，在椭圆曲线中，$R_C = r_{UK} \bullet B \pmod{p_1}$的计算中$|p_1|$=160bits，其安全性和使用1024位密钥的RSA相同。此外，对于模指数运算$g^x \bmod n$，其中n是一个1024bits的大质数，x是一个160bits的随机整数，一个模指数运算的开销相当于$3|x|/2$个模乘[37]($|x|$表示x的位数)，近似等于240MMs。当$|p_1|$=160bits时，点积运算速度比模指数运算速度快近8倍[38]，于是可以推导出一个椭圆曲线上的点积运算大约等于29MMs[35],[36]。

1. 原始WTLS握手协议计算开销如下表所示：

表 5-1 原始 WTLS 握手协议各阶段的计算开销

	Client	Server
初始化阶段(离线计算)	$T_{ECC} \approx 29 T_{MM}$	$T_{ECC} \approx 29 T_{MM}$
交换随机数和计算预主密钥阶段	$T_{ECC} \approx 29 T_{MM}$	$T_{ECC} \approx 29 T_{MM}$
验证身份阶段	$T_{ECC} + T_{MM}$	$2T_{ECC} + T_H$
计算主密钥阶段	T_f	T_f
总计	$88 T_{MM} + T_f + T_H$	$116 T_{MM} + T_f + T_H$

2. 基于 FS-MAKEP 的 WTLS 握手协议复杂度分析

在系统初始化阶段，客户端要计算临时公钥 $Rc = r_{UK} \cdot B \bmod p_1$ 与 $CMT = g^{r_{UF}||r_{UR}} \bmod n$，故其总计算开销为 $T_{ECC} + T_{ME}$，服务器端同样也要预先计算 $Rs = r_{SK} \cdot B(\bmod) p_1$，复杂度为 T_{ECC}，以上均在双方通信开始之前计算，为离线计算；

在交换随机数和计算预主密钥阶段，服务器端计算 ServerHello 中传送的 $K_{P'} = r_{SK} \cdot Rc$，$C1 = E_{K_{P'}}(Rc)$，故其总计算开销为 $T_{ECC} + T_E$。收到 ServerHello 消息后，客户端计算 $K_P = r_{UK} \cdot Rs$，$D_{KP}(C1)$，总复杂度为 $T_{ECC} + T_D$。

在验证身份阶段，客户端发送的 Certificate 消息，$C2 = E_{K_P}(ID_U)$ 计算开销为 T_E。服务器端计算 $D_{K_P}(C2) = ID_U$，$S_F = h(Rc, Rs, ID_U, ID_S)$，总计算开销为 $T_D + 2T_H$。客户端同时计算 $S_F = h(Rc, Rs, ID_U, ID_S)$，$S_R = 2^{y_1}(r_{UF} - S_F) + r_{UR} \bmod \lambda(n)$，总计算开销为 $T_H + 0.1T_{MM}$。服务器端收到 CertificateVerfy 消息后，计算 $CMT' = g^{S_R||S_R} \bmod n$，复杂度为 T_{ME}。

在计算主密钥阶段，服务器端和客户端各自计算 $K_m = f(K_{P'}, Rc, Rs)$，复杂度分别为 T_f。

表 5-2 基于 FS-MAKEP 的 WTLS 握手协议各阶段的计算开销

	Client	Server
初始化阶段(离线计算)	$T_{ECC} + T_{ME} \approx 1805T_{MM}$	$T_{ECC} \approx 29T_{MM}$
交换随机数和计算预主密钥阶段	$T_{ECC} + T_D \approx 29T_{MM} + T_D$	$T_{ECC} + T_E \approx 29T_{MM} + T_E$
验证身份阶段	$T_E + T_H + 0.1T_{MM}$	$T_D + 2T_H + T_{ME} \approx 1776T_{MM} + T_D + 2T_H$
计算主密钥阶段	T_f	T_f
总计	$1834.1T_{MM} + T_D + T_E + T_H + T_f$	$1834T_{MM} + T_E + T_D + 2T_H + T_f$

3. 基于 AL-SA 的 WTLS 握手协议复杂度分析

在初始化阶段，客户端要计算临时公钥 $Rc = r_{UK} \cdot B \bmod p_1$，服务器端同样计算 $Rs = r_{SK} \cdot B(\bmod) p_1$ 故其总计算开销为 $2T_{ECC}$，以上均为离线计算；

在交换随机数和计算预主密钥阶段，服务器端计算 ServerHello 中传送的 $K_{P'} = r_{SK} \cdot Rc$，$C1 = E_{K_{P'}}(Rc)$，故其总计算开销为 $T_{ECC} + T_E$。收到 ServerHello 消息后，客户端计算 $K_P = r_{UK} \cdot Rs$，$D_{KP}(C1)$，总复杂度为 $T_{ECC} + T_D$。

在验证身份阶段，客户端发送的 Certificate 消息 M = h(Certc)，复杂度为 T_H。服务器端计算 $V_{VK}(C)? = hash_{HK}(M : r')$，$K_m = f(K_{P'}, Rc, Rs)$，总计算开销为 $T_D + 2T_H$。

在计算主密钥阶段，双方各自计算 $K_m = f(K_{P'}, Rc, Rs)$，复杂度分别为 T_f。

表 5-3 基于 AL-SA 的 WTLS 握手协议各阶段的计算开销

	Client	Server
初始化阶段(离线计算)	$T_{ECC} \approx 29T_{MM}$	$T_{ECC} \approx 29T_{MM}$
交换随机数和计算预主密钥阶段	$T_{ECC}+T_D \approx 29T_{MM}+T_D$	$T_{ECC}+T_E \approx 29T_{MM}+T_E$
验证身份阶段	T_H	T_D+2T_H
计算主密钥阶段	T_f	T_f
总计	$58T_{MM}+T_D+T_H+T_f$	$58T_{MM}+T_D+T_E+2T_H+T_f$

因为散列函数计算T_H，以及对称加，解密算法的计算T_E，T_D开销比模指数运算T_{ME}和椭圆点积运算T_{ECC}要小得多，所以作性能评估时就可以忽略这些算法开销。WTLS 原始协议与两种改进的 WTLS 协议的总计算开销对比如表 5-4 所示：

表 5-4 总计算开销对比

	Client		Server	
	Offline	Online	Offline	Online
原始 WTLS 握手协议	$29T_{MM}$	$59T_{MM}+T_f$	$29T_{MM}$	$87T_{MM}+T_f$
基于 FS-MAKEP 的 WTLS 握手协议	$1805T_{MM}$	$29T_{MM}+T_f$	$29T_{MM}$	$1805T_{MM}+T_f$
基于 AL-SA 的 WTLS 握手协议	$29T_{MM}$	$29T_{MM}+T_f$	$29T_{MM}$	$29T_{MM}+T_f$

两种方案实现的密钥交换过程是基于 ECC 算法的，所需密钥长度较短，故可提高加/解密的效率及减少储存的成本。在计算开销上，因为基于 AL-SA 的 WTLS 握手协议把大部分计算量转移到了 SA 中，所以明显比原始 WTLS 协议和基于 FS-MAKEP 的 WTLS 握手协议具有较少的计算开销，是最高效的一种方案。

虽然基于 FS-MAKEP 的 WTLS 握手协议的总计算开销比原始 WTLS 协议要多，但应当注意的是，客户端计算开销的增加仅在离线计算阶段，而在线的计算开销比原始 WTLS 协议要小，保证了协议在运行中仍是高效的，且在线运算比原始 WTLS 协议更加快速。服务器端的在线运算量的增大主要是因为要对客户端进行双重身份认证，从而增加了运算量。客户端运算中增加的离线开销主要是因为选取随机数的变化，用于实现向前安全，同时防止中间实体攻击，从而减少协议的明文信息传输，降低数据受攻击和泄漏的可能。

5.2 信息载荷量分析

根据 WTLS 规范[7]中的要求，各参数设置如下，单位 bits：

$|rc|=|rs|=96$，$|Rc|=|Rs|=160$，$|v|=8$，$|SID|=8$，$|H()|=160$，$|E_K()|=160$，$|(r||s)|=320$。

由 4.3 节可知，M 的长度取决于所选哈希函数，License 的长度取决于所选签名算法，因此$|M+License|$的长度是不固定的。表 5-5 列出各阶段的信息载荷量。

表 5-5 各阶段的信息载荷量(单位：bits)

	原始 WTLS 握手协议	基于FS-MAKEP的WTLS握手协议	基于 AL-SA 的 WTLS 握手协议
ClientHello	112+|SecNegc|	176+|SecNegc|	176+|SecNegc|
ServerHello	112+|SecNegc|	176+|SecNegc|	176+|SecNegc|
Certificate	|Certs|	|Certs|	|Certs|
CertificateRequest	0	0	0
ServerHelloDone	0	0	0
Certificate	|Certc|	|Certc|	|M+License|
CertificateVerify	480	1024	0
Finished	0	0	0

由上表可以看出，基于 FS-MAKEP 的 WTLS 握手协议和基于 AL-SA 的 WTLS 握手协议在通信载荷上都有轻微增加。增加的主要原因是选取随机数的算法变化和对用户证书的加密以及对用户证书认证信息的增加。这些通信载荷的增加使得改进后的 WTLS 握手协议的明文信息传输减少，可以降低数据受攻击和泄漏的可能，随机数计算方法的改变和随机数长度的增加使攻击者无法破解随机数，从而无法计算出预主密钥。因此，为了提高协议的安全性，通信载荷量的轻微增加是可以允许和被接受的。

5.3 优势分析

基于 FS-MAKEP 的 WTLS 握手协议是一种安全的双向认证和密钥交换协议，能抵制向前安全攻击，未知密钥共享攻击，和中间人攻击等等，而且能够实现双向认证。

基于匿名通行证的签名认证方案，需要客户端和服务端具备可代表他们身份的数字证书。在整个认证过程中，对客户端的身份认证是基于对一个认证中心 SA 的信任，服务器端无法得知关于客户端的任何信息。

两种改进的 WTLS 握手协议各自具有独特优势，比原始的 WTLS 协议在安全性上和性能上有所改进。原始的 WTLS 协议不能提供向前安全性，而且不能防止主动攻击[7]，客户端身份认证阶段也存在身份泄漏危险。两种方案针对于这些安全隐患，作出了以下有效的改进措施：

1. 在交换随机数和计算预主密钥阶段，两种改进的 WTLS 握手协议都是基于 ECC 算法的，与同类密钥交换算法 RSA，DH 算法相比，ECC 密钥长度相对较短，但是却可以达到相同的安全等级。并且两种改进后的协议都提供了原始

WTLS 协议不具备的向前安全性，能够抵抗未知密钥共享攻击，中间人攻击等主动攻击，具体分析参见 3.2.3 和 4.3.2 节。

2. 在验证身份阶段，两种改进的 WTLS 握手协议具有不同的优势：

在基于 FS-MAKEP 的 WTLS 握手协议中，采取的是对双方身份的双重认证：通过双方身份证书认证和对双方的隐式认证[46](隐身认证的详细分析参见 3.2.3 节) 。这样即使用户的身份证书在传递过程中被攻击者修改，也会在协议中进行隐式认证的时候很快被发现，因此这种双重认证方式给 WTLS 的认证过程提供了双重保险。在计算开销上，客户端和服务器端仅仅增加了一个模指数运算，这是由于用来对客户端身份进行隐式认证的散列函数是一个模指数函数。

在基于 AL-SA 的 WTLS 握手协议中，采取的是对用户匿名通行证的认证，是基于对公共认证中心 SA 的信任。客户端提交客户证书作为身份的证明，经过 SA 验证后，对其签名。服务器端只需要验证 SA 给用户签发的匿名通行证来确认用户身份是否合法。该方案既保证了用户的身份信息不被攻击者知道，也保证了信息不被服务器端知道，而且在计算开销上比前一种方案少，因为对客户端的身份的认证和签名都在 SA 进行，大大减小了客户端的计算量，而且服务器端也只需要通过一个认证函数验证 SA 的签名即可。

5.4 应用领域

基于 FS-MAKEP 的 WTLS 握手协议对原始 WTLS 握手协议在安全性上做出了改进，使得用户不用再担心信息被篡改、窃听、身份被冒充等安全问题。该改进的协议应用很广泛，可以用于普通的 WAP 业务，如新闻浏览，天气预报等。

基于 AL-SA 的 WTLS 握手协议尽管在安全性上，计算开销和通信载荷上都比原始的 WTLS 握手协议和基于 FS-MAKEP 的 WTLS 握手协议有优势，但是，第三方认证中心 SA 的建立是一个比较复杂的过程，因为 SA 必须替数字证书的持有人作合法身份的保证和提供公钥的有效证明，所以 SA 需要维护密钥目录与用户身份证书，这会耗费凭证机构的储存空间与计算成本。因此基于建立第三方认证中心的安全性高，但是建立成本高，维护代价也高的特点，该方案更适用于对安全有特殊要求的移动电子交易上面，如移动电子银行，移动购物，移动证券。

5.5 本章小结

对基于 FS-MAKEP 的 WTLS 握手协议和基于 AL-SA 的 WTLS 握手 协议分析了计算开销和信息载荷量，并且阐述了两种改进方案各自的优势特点，指出了两种方案各自的适用领域。

第六章 结束语

6.1 总结

本文的工作主要包括以下几部分：

1. 了解现有 WTLS 握手的工作原理及流程，分析其在安全方面的不足，研究了 WTLS 相关理论，包括 WTLS 协议规范，加解密算法、签名认证算法，密钥交换算法，协议安全形式化证明等方面内容。

2. 提出了一种提供向前保密性的无线密钥交换和双向认证协议 FS-MAKEP。经证明，该协议能提供向前安全性，抵抗中间人攻击，未知密钥共享攻击等等，并能实现身份证书认证和隐式认证的双认证，保证了用户身份认证的第一安全等级。

3. 提出了基于 FS-MAKEP 的 WTLS 握手协议，改进了原有协议的 4 个漏洞，用 Rubin 逻辑证明了改进后的 WTLS 握手协议的安全性；并在 Linux 环境下实现了一个 WTLS 客户端和一个 WTLS 服务器端来模拟改进后的 WTLS 握手协议。

4. 根据 Adi Shamir 的在线/离线签名算法，设计了基于用户匿名通行证的身份认证方案 AL-SA。方案把客户端的大部分计算量转移到第三方 SA 中，并且通过匿名通行证的签发保护了客户端的真实身份信息。该方案既保护了用户的隐私，减轻了客户端的计算量，也提高了服务器端对用户身份认证的效率。

5. 提出了基于 AL-SA 的 WTLS 握手协议，不仅改进了原有协议的 4 个漏洞，而且保证了用户身份认证的第二安全等级，是高效性和安全性并重的一种方案。

6.2 下一步工作

本文虽然改进了 WTLS 协议，但是还需要在以下三个方面作进一步的研究：

1. 本文对改进后的 WTLS 握手协议的计算开销仅仅进行了数学理论上的分析，还需要在实际的运行环境中进行安全性和复杂性测试。

2. 为了避免公钥在传递的过程中被中间者所破坏， SA 将替数字证书的持有人作合法身份保证与提供公钥的有效证明，所以 SA 需要去维护密钥目录与管理证书，这会耗费 SA 的储存空间与计算成本。下一步工作还需要提出一种方案简化 SA 生成证书的过程，并且尽量减少计算成本。

3. 当实验环境允许的时候，在实际环境中对改进后的两种 WTLS 握手协议进行实现。

参考文献

[1] WAP Forum. Wireless Application Protocol Wireless Application Specification. Version 12-July-2001. http://www.wapforum.org/

[2] Juha Mynttinen, End-to-End Security of Mobile Data in GSM. Tik-110.501 Seminar on Network Security

[3] WAP Forum. Wireless Application Environment Specification. Version 7-Feb-2002. http://www.wapforum.org/

[4] WAP Forum. Wireless Session Protocol Specification. Approved Version 5-July-2001. http://www.wapforum.org/

[5] WAP Forum. Wireless Transaction Protocol. Version 10-Jul-2001. http://www.wapforum.org/

[6] WAP Forum. Wireless Datagram Protocol. Version 14-Jun-2001. http://www.wapforum.org/

[7] WAP Forum. Wireless Transport Layer Security Specification. Version 06-Apr-2001. http://www.wapforum.org/

[8] Markku-Juhani Saarinen. Attacks Against the WAP WTLS Protocol. http://www.freeprotocols.org/ harmofWap/wtls.pdf

[9] Dongjin Kwak, Jae Cheol Ha, Hoonjae Lee. A WTLS Handshake Protocol with User Anonymity and Forward Secrecy. Proceedings of Mobile Communications: 7th CDMA International Conference(CIC2002), LNCS2524, 2003. 219~230

[10] DongGook Park, Colin Boyd and Sang-Jae Moon. Forward Secrecy and Its Application to Future Mobile Communications Security. PKC2000, LNCS1751. Spring-Verlag, 2000. 433~445

[11] 周永彬，张振峰，冯登国. 一种认证密钥协商协议的安全分析及改进. 软件学报，2006，17(4)：868~875

[12] W. Diffie an M. E. Hellman. New Direction in Cryptography. IEEETrans. On Information Theory, 1979. 22(6)：644~654

[13] 吴刚，薛质. 针对 WLAN 的特定攻击手段与相关检测技术. 信息安全与通信保密，2006，117~119

[14] A. Shamir. Identity-based Cryptosystems and Signature Schemes. Advances in Cryptology: Crypto'84, Springer-Verlag, 1985. 47~53

[15] 翟好，贺惠萍，裘鸿林. WAP 手机银行安全体系的分析与应用. 中国金融电

脑，2005，No3，52~56

[16] 邹学强，冯登国. WTLS 握手协议的安全性分析及改进. 中国科学院研究生院学报，2004，21(4)：494~500

[17] WAP Forum. WAP Forum. Wireless Identity Module. Version 12-July-2001. http://www.wapforum.org/.

[18] Thomas Weigold. Java-Based Wireless Identity Module. 2002.

[19] Understanding Security on the Wireless Internet. Phone.com.2000, http://www.phone.com/products/publications.html

[20] 罗蕾，王庆，谭罗丽. WAP 安全构架研究及 WTLS 的实现. 电子科技大学学报，2002，31(4)：387~392

[21] Ian Herwono, Ingo Liebhardt. Performance of WTLS and its Impact on an M-Commerce Transaction. Xi'an China: ICICS'01, 2001. 167~171

[22] 谢湛. 无线传输安全协议 WTLS 仿真研究：[硕士学位论文]. 西安：电子科技大学，2006

[23] 熊万安，余 堃，龚耀寰. WTLS 协议性能仿真. 电子科技大学学报，2005，34(4)：489~492

[24] 余堃，周明天，杨光志. 椭圆曲线加密算法及其在 WTLS 中的应用. 计算机科学，2002，29(6)：94~101

[25] 崔媛媛，周永彬，丁金扣等. 一种具有用户匿名性和前向安全性的 WTLS 握手协议的安全性分析及其改进. 高技术通讯，2005，15(4)：6~10

[26] 叶润国，冯彦君，虞淑瑶等. 使用 ECMQV 密钥交换方案增强 WTLS 协议安全性. 计算机应用. 2005，25(4)：859~861

[27] Fuw-Yi Yang and Jinn-Ke Jan. A Secure and Efficient Key Exchange Protocol for Mobile Communications. Cryptology Eprint Archive 2004/167, July 2004, http://eprint.iacr.org

[28] N. Koblitz. Elliptic Curve Cryptosystems. Mathematics of Computation, 1987. 48(17)：203~209

[29] V.S. Miller. Use of Elliptic Curves in Cryptography. Advances in Cryptology: Crypto'85, Springer-Verlag, 1986. 417~426

[30] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. Communications of the ACM, 1978. 21(2)：120~126

[31] W. Caelli, E. Dawson, and S. Rea. PKI. Elliptic Curve Cryptography and Digital Signatures. Computer & Security, 1999. 18(1)：47~66

[32] S. Vanstone. Elliptic Curve Cryptosystem-the Answer to Strong, Fast Public-key Cryptography for Securing Constrained Environments. Information Security Technical Report, Elsevier, 1997. 2(2): 78~87

[33] A. Shamir and Y. Tauman. Improved Online/Offline Signature Schemes. Advances in Cryptology-CRYPTO'01, LNCS 2139, 2001. 355~367

[34] A. Lenstra and E. Verheul. Selecting Cryptographic Key Size. The Third International Workshop on Practice and Theory in Public Key Cryptography, LNCS 1751, 2000. 446~465

[35] 林祝兴，李正隆. Elliptic-Curve Undeniable Signature Schemes. 第 11 届信息安全会议，2001，331~338

[36] A. Jurisic, and A.J. Menezes. ECC Whitepapers: Elliptic Curves and Cryptography. Certicom corp. http://www.certicom.com/research/weccrypt. html

[37] Niels Fegruson, Bruce Schneier. Practical Cryptography.电子工业出版社. 2005.141

[38] N. Koblitz, A. Menezes, S. Vanstone. The State of Elliptic Curve Cryptography. Designs, Codes, and Cryptography. 2000, 19(2-3): 173~193

[39] Rubin A D. Nonmonotonic Cryptographic Protocols. Phd thesis, University of Michigan, Ann Arbor. 1994

[40] 何德全. 安全协议. 北京：清华大学出版社，2005. 243~256

[41] Shimon Even, Oded Goldreich, and Silvio Micali. Online/offline Digital Signatures. In Advances in Cryptology: Crypto '89. August 1990. Springer. 263~277

[42] Hugo Krawczyk and Tal Rabin. Chameleon Signatures. In Symposium on Network and Distributed Systems Security (NDSS '00), Internet Society, 2000. 143~154

[43] 宋宗宇，苏铭，陈铖. 不可追踪的离线电子货币方案. 微计算机信息，2005. 21(10)：35~36

[44] Alfred J.Menezes, Paul C.van Oorschot and Scoot A.Vanstone，Handbook of Applied Cryptography. CRC Press. ISBN 0849385237. 2005. 32

[45] 赖溪松，韩亮，张真诚. 近代密码学及其应用. 台湾：松岗图书数据公司，1998

[46] M. Girault. Self-certified Public Keys. Advances in Cryptology: EuroCrypt'91, Lecture Notes in Computer Science, Springer-Verlag, 1991. 491~497

致 谢

攻读硕士学位期间，导师李杰教授在学习和工作上对本人给予了悉心的指导和亲切的关怀。导师渊博的知识、严谨的治学态度和对学术问题的敏锐洞察力，使我逐渐领悟出进行科学研究的方法和途径，使我的科研工作和试验进展得以顺利进行。在校期间，导师给我以谆谆教导和亲切关怀，并为本人的学习和工作提供了良好的软硬件条件。毕业论文从开始开题到论文的最后完成，导师都倾注了大量的心血。在此谨向我的导师致以最真挚的敬意和最衷心的感谢！

感谢高琰博士，马征博士，马佩勋师兄三年来对我的关心和帮助，告诉我如何查阅英文资料，和撰写英文论文！

感谢信息科学与工程学院的各位老师！他们在我读研期间给予我无微不至的关怀和无私的帮助！

最后，还要感谢我们实验室同年级的其他同学：杨萍，李金，夏媛，曹龄兮，郝丽波，许甸，丁昭华！他们在论文的撰写过程中也给我提供了很大的帮助！

攻读硕士学位期间主要的研究成果

发表论文情况：

[1] HE Yijun, XU Nan, LI Jie. A Secure Key Exchange and Mutual Authentication Protocol for Wireless Mobile Communications. The Second International Conference on Availability, Reliability and Security. IEEE Computer Society Press. 2007. 558~563

[2] 何毅俊，李杰，徐楠. 提供向前保密性的无线密钥交换和双向认证协议. 计算机应用(已录用)

[3] 何毅俊，李杰. 基于Liferay的企业信息门户研究与实现. 计算机应用研究(已录用)

[4] 徐楠，何毅俊，陈松乔. 基于Agent的分布式入侵检测系统框架设计. 计算机测量与控制，2007，15(4)：421~422

参加的科研项目：

[1] WAP 网站设计与实现

[2] 开源门户软件包评估

[3] 湖南移动通信有限公司门户网站