

基于进程演算的安全协议形式化研究

Formal Research of Security Protocols
Based On Process Calculus

研究生： 李国强

导师： 傅育熙 教授

专业： 计算机软件与理论

完成日期： 2005年1月

基于进程演算的安全协议形式化研究

摘 要

随着网络技术应用与发展逐步渗透到许多关键部门，各种安全需求日渐复杂。然而，已有的安全协议往往被证实并不如它们的设计所期望的那样安全。同时，网络系统仍然面临着各种新、旧攻击手段的威胁。这种复杂的网络环境使得攻击者可利用安全协议自身的缺陷来实施各种各样的攻击，从而达到破坏网络安全的目的，因此，安全协议已成为网络安全的关键。

安全协议形式化分析技术可使协议设计者通过系统分析将注意力集中于接口、系统环境的假设、在不同条件下系统的状态、条件不满足时出现的情况以及系统不变的属性，并通过系统验证，提供协议必要的安全保证。目前，国内外形式化研究安全协议主要有基于知识与信念推理的模态逻辑方法，基于定理证明的方法和基于进程演算的方法三种。其中，进程演算对于协议的描述几乎接近协议的本身含义，可以很精确地刻画协议的运行过程。使用进程演算对安全协议分析和验证时，协议的每一个主体都被建模为一个单独的进程，这些子进程并发运行，并使用进程之间的共享通道进行同步通信，这样得到的并发系统将作为安全协议的基本模型。利用进程演算对安全协议进行分析和验证时，既借鉴了进程演算作为代数模型的基本验证理论与方法，又使用了进程演算作为抽象程序设计语言的程序分析方法。因此，基于进程演算的安全协议验证技术可分为模型检测技术、互模拟验证技术和程序分析技术三种。

本文在基于进程演算方法方面作了一定的研究，并且对本方向的研究

前景作了总结和展望。论文主要内容及创新点包括：

- (1) 在对 Spi演算的研究中，定义了一个原语，解决了原演算不能验证存在时间戳的安全协议的各种验证，并基于扩展了的 Spi演算，验证了 Kerberos协议的认证性。
- (2) 目前，安全协议形式化研究尚未有人验证协议的匿名性，本文基于 Applied pi演算，讨论了 iKP协议是否具有匿名性，同时也验证了它的认证性。

关键词 安全协议，进程演算， Spi演算， Applied pi演算， 认证性， 匿名性

FORMAL RESEARCH OF SECURITY PROTOCOLS BASED ON PROCESS CALCULUS

ABSTRACT

With the wide application and rapid development, computer network is gradually accepted by many key departments, which leads to various and complex requirements of security. However, many existing security protocols are proven not to be so reliable as what were expected. Meanwhile, the network systems still face up to various threats of new or old attacking methods. The complication of network environment makes the attacker possible to launch various attacks from the flaws in the protocols to break network security. So security protocols are crucial to network security.

Formal methods for protocol analysis can make the designer focus on, through system analysis, interfaces, hypotheses of the environment, system states under various conditions, exceptions when some conditions are not satisfied and invariant properties. Formal methods can also provide, through system validation, guarantees of securities of protocols. Recently, formal methods for protocol analysis are mainly of three kinds: methods based on modal logic of knowledge and belief, methods based on theorem proof and methods based on process calculus. Among them three, since process calculus descriptions approach the nature of protocols, it can depict the running process of protocols precisely. When using

process calculus for protocol analysis and validation, each principal is modelled as a single process, all these processes run concurrently and communicate in a synchronized manner through shared channels, and the resulting concurrent system is regarded as the model of the protocol. Process calculus methods can utilize both basic theorems and methods from algebra modelling and methods from abstract programming language analysis. So, methods based on process calculus can be divided into three aspects: model checking, validation through bisimulation and program analysis.

There exist some research results based on process calculus in this dissertation, solving some problems never be solved before. Research foregrounds have also been concluded and prospected. The main contents and contributions are as follows:

- (1) A primitive has been extended in Spi calculus so that it can verify security protocols with time stamp. And based on this extended calculus, the authentication of Kerberos protocol has been verified.
- (2) Till now, There are no other formal research of security protocols focus on anonymity property. In this dissertation, Whether iKP protocol has anonymity property is discussed based on applied pi calculus. And its authentication is also been verified at the same time.

KEY WORDS: security protocols, process calculus, Spi calculus, Applied pi calculus, authentication, anonymity

目 录

| | |
|--------------------------------|-----------|
| 1 绪言 | 1 |
| 1.1 安全协议 | 1 |
| 1.2 安全协议形式化分析的意义 | 2 |
| 1.3 安全协议形式化分析概述 | 4 |
| 1.3.1 基于知识与信念推理的模态逻辑方法 | 4 |
| 1.3.2 基于定理证明的方法 | 6 |
| 1.4 基于进程演算的安全协议形式化分析概述 | 7 |
| 1.4.1 进程演算 | 7 |
| 1.4.2 安全协议形式化分析的进程演算模型 | 10 |
| 1.4.3 基于进程演算的安全协议形式化分析技术 | 11 |
| 1.5 主要研究内容与成果 | 15 |
| 1.6 本文结构 | 16 |
| 2 基于 Spi演算的安全协议分析 | 17 |
| 2.1 引言 | 17 |
| 2.2 Spi演算的语法和语义 | 17 |
| 2.2.1 Spi演算的语法 | 17 |
| 2.2.2 Spi演算的语义 | 18 |
| 2.3 等价关系 | 21 |
| 2.3.1 测试等价 | 21 |
| 2.3.2 Framed互模拟关系 | 22 |
| 2.4 基于 Spi演算的形式化验证 | 24 |
| 2.4.1 Kerberos 协议 | 24 |
| 2.4.2 Kerberos协议形式化表示 | 26 |
| 2.4.3 Kerberos协议认证性的定义 | 27 |
| 2.4.4 Kerberos协议的规范 | 28 |
| 2.4.5 Kerberos协议的认证性及缺陷 | 28 |
| 2.5 结论与相关工作 | 29 |

| | | |
|----------|-------------------------------|-----------|
| 3 | 基于 Applied pi演算的安全协议分析 | 30 |
| 3.1 | 引言 | 30 |
| 3.2 | 语法和语义 | 31 |
| 3.2.1 | Applied pi演算的语法 | 31 |
| 3.2.2 | Applied pi演算的语义 | 32 |
| 3.3 | 等价关系 | 34 |
| 3.3.1 | 观察等价 | 34 |
| 3.3.2 | 静态等价 | 35 |
| 3.4 | 基于 Applied pi演算的形式化验证 | 35 |
| 3.4.1 | ikp 协议 | 35 |
| 3.4.2 | 1kp协议形式化表示 | 38 |
| 3.4.3 | 1kp协议认证性的验证 | 40 |
| 3.4.4 | 1kp协议匿名性的讨论 | 40 |
| 3.5 | 结论和相关工作 | 41 |
| | | |
| 4 | 工作的总结与展望 | 42 |
| 4.1 | 论文工作总结 | 42 |
| 4.2 | 未来工作展望 | 43 |
| 4.2.1 | 模型的建立 | 43 |
| 4.2.2 | 性质的形式化定义 | 44 |
| 4.2.3 | 形式化分析 | 45 |
| | | |
| | 参考文献 | 46 |
| | | |
| | 致 谢 | 55 |
| | | |
| | 发表的论文 | 56 |

第一章 绪言

随着网络技术应用与发展逐步渗透到许多关键部门，各种安全需求日渐复杂。近年来，许多安全协议在提出之初被认为是足够安全的，然而却在很短的时间内被证明有漏洞，目前越来越多的安全协议不断涌现，关于协议安全性方面的讨论正在成为学术热点，主要的研究集中在对安全协议的形式化验证方面。在众多形式化研究方法中，使用进程演算对安全协议分析和验证时，协议的每一个主体都被建模为一个单独的进程，这些子进程并发运行，并使用进程之间的共享通道进行同步通信，这样得到的并发系统将作为安全协议的基本模型。本文主要对已有的形式化验证安全协议的方法进行了综合比较，并且扩展了进程演算模型，证明了两个有代表性安全协议的安全性质。

1.1 安全协议

在理解安全协议这一概念之前，首先要了解什么是协议。所谓协议，就是两个或两个以上的参与者采取一系列步骤以完成某项特定的任务。这个定义包含三层意思：

第一，协议至少需要两个参与者。一个人可以通过执行一系列的步骤来完成一项任务，但它不构成协议。

第二，在参与者之间呈现为消息处理和消息交换交替进行的一系列步骤。

第三，通过执行协议必须完成某项任务，或达成某种共识。

可见，协议与算法的概念是不尽相同的。算法应用于协议中消息处理的环节。对消息不同的处理方式则要求不同的算法，而对算法的具体化则可定义出不同的协议类型。因此，可以简单地说，安全协议就是在消息处理环节采用了若干的密码算法的协议。由此可见，密码算法和安全协议处于网络安全体系的不同层次，是网络数据安全两个主要内容。具体而言，密码算法为网络上传递的消息提供高强度的加解密操作和其他辅助算法（如Hash函数等），而安全协议是在这些密码算法的基础上为各种网络安全性方面的需求提供其实现方案。

安全协议是建立在密码体制基础上的一种高级协议，它运行在计算机通信网或分布式系统中，为安全需求的各方提供一系列步骤，借助于密码算法来达到密钥分配、身份认证、信息保密以及安全地完成电子交易等目的。

在网络通信中最常用的、最基本的安全协议按照其目的可以分成以下四类：

（1）密钥交换协议

这类协议用于完成会话密钥的建立。一般情况下是在参与协议的两个或者多个实体之间建立共享的秘密，如用于一次通信中的会话密钥。协议中的密码算法可采用对称密码体制，也可以采用非对称密码体制。这一类协议往往不单独使用，而与认证协议相结合。

（2）认证协议

认证协议包括实体认证（身份认证）协议、消息认证协议和数据目的认证协议等，用来防止假冒、篡改、否认等攻击。

（3）认证和密钥交换协议

这类协议将认证协议和密钥交换协议结合在一起，先对通信实体的身份进行认证，在成功认证的基础上，为下一步的安全通信分配所使用的会话密钥，它是网络通信中应用最普遍的一种安全协议。常见的认证和密钥交换协议有互联网密钥交换（IKE）协议、分布认证安全服务（DASS）协议、Kerberos认证协议等。

（4）电子商务协议

电子商务协议是保证客户和商家之间完成正常、可靠、安全交易活动的规则。与上述协议最为不同的是，电子商务协议中主体往往代表交易的双方，其利益目标是不一致的，或者根本就是矛盾的，电子商务协议不仅要考虑基本的安全性（如保密性、认证性、完整性等），更要关注其特有的性质（如交易的可追究性、公平性、货币和商品的原子性、匿名性等）。常见的电子商务协议有SSL，SET，iKP协议等。

1.2 安全协议形式化分析的意义

随着网络技术应用与发展逐步渗透到许多关键部门，特别是政府、军事机构，及电子商务的兴起与广泛应用，各种安全需求日渐复杂。在一个分布式的互联网网

络环境中，人们通过安全协议来具体实现安全共享网络资源的需求。已有的安全协议往往被证实并不如它们的设计所期望的那样安全，同时网络系统仍然面临着各种新、旧攻击手段的威胁。这种复杂的网络环境使得攻击者可利用安全协议自身的缺陷来实施各种各样的攻击，从而达到破坏网络安全的目的，因此，安全协议已成为网络安全的关键。由于安全协议的重要性，近十几年来，世界各国学者对其设计和分析进行了较为广泛和深入的研究，尤其是美国和欧洲的一些国家在这一关键领域投入了大量的研究经费和力量。

安全协议的设计极易出错，即使我们只讨论安全协议中最基本的认证协议，其中参加协议的主体只有两三个，交换的消息只有3-5条，设计一个正确的、符合安全目标的、没有冗余的认证协议也十分困难。

安全协议设计与分析的困难在于：

- 安全目标本身的微妙性。例如，表面上十分简单的“认证目标”，实际上十分微妙，关于认证的定义，至今存在着各种不同的观点；
- 协议运行环境的复杂性。当安全协议运行在一个十分复杂的公开环境时，攻击者处处存在。我们必须形式化的刻划安全协议的运行环境，这是一项艰巨的任务；
- 攻击者模型的复杂性。我们必须形式化描述攻击者的能力，对攻击者和攻击行为进行分类和形式化分析；
- 安全协议的“高并发性”。由于安全协议具有“高并发性”的特点，因此，安全协议的分析变得更加复杂并具有挑战性。

由于安全协议的运行不是独立的，而是处于某种不安全的环境之中，因而它是易错的，并且其错误很难完全人工识别，必须要借助形式化的分析方法或工具来完成。

安全协议形式化分析技术可使协议设计者通过系统分析将注意力集中于接口、系统环境的假设、在不同条件下系统的状态、条件不满足时出现的情况以及系统不变的属性，并通过系统验证，提供协议必要的安全保证。通俗地讲，安全协议的形式

化分析是采用一种常规的、标准的方法对协议进行分析，以检查协议是否满足其安全目标。

因此，安全协议的形式化分析有助于：

- 界定安全协议的边界，即协议系统与其运行环境的界面。
- 更准确地描述安全协议的行为。
- 更准确地定义安全协议的特性。
- 证明安全协议满足其说明，以及证明安全协议在什么条件下不能满足其说明。

1.3 安全协议形式化分析概述

目前，国内外形式化研究安全协议主要有基于知识与信念推理的模态逻辑方法，基于定理证明的方法和基于进程演算的方法三种，在本节中我们简单介绍前两种，然后在下一节，详细介绍基于进程演算的方法。

1.3.1 基于知识与信念推理的模态逻辑方法

模态逻辑方法是分析安全协议最直接、最简单的一种方法。它们由一些命题和推理规则组成，命题表示主体对消息的知识或信念，而应用推理规则可以从已知的知识和信念推导出新的知识和信念。在这类方法中有许多逻辑的提出，其中最著名的就是 BAN逻辑和 BAN类逻辑，以及专门用于分析验证电子商务协议的 Kailar逻辑。

Burrows、Abadi和 Needham在1989年提出了 BAN逻辑 [19]，在形式化解决安全协议分析问题上迈出了一大步。BAN逻辑是分析安全协议的一个里程碑，至今在使用逻辑手段分析安全协议方面取得的进展大都以它为基础。BAN逻辑获得广泛的认可，它是关于主体信念以及用于从已有信念推出新的信念的推理规则的逻辑。这种逻辑通过对认证协议的运行进行形式化分析，来研究认证双方通过相互发送和接受消息能否从最初的信念逐渐发展到协议运行最终要达到的目的。如果在协议执行结束时未能建立起关于诸如共享通信密钥、对方身份等信念，则表明这一协议有安全缺陷。然而，BAN逻辑最大的问题是不够完备，它不能查出协议的所有问题。缺陷表现在以下几个方面：非标准的理想化协议进程；不合理的假设；无法检查协议运

行的违规现象。对 BAN 逻辑自身进行扩充导致了 BAN 类逻辑的产生，如 GNY 逻辑 [48]、AT 逻辑 [16]、SVO 逻辑 [76, 77] 等。

GNY 逻辑是在 1990 年由 Gong、Needham 和 Yahalom 提出的。由于 BAN 逻辑存在不可逾越的障碍使得其应用受到限制，GNY 逻辑则试图消除 BAN 逻辑中对主体诚实性的假设、消息源假设、可识别假设，并从以下几个方面对 BAN 逻辑进行了改进：取消了一些全局假设，增加了 BAN 逻辑的分析能力；引入了可识别性的概念，用于描述主体对其所期望的消息格式的识别能力；区分断言集合和符号集合；加入了若干 fresh 判断法则。但是，GNY 逻辑为了消除 BAN 逻辑中的假设，引入了多达 41 条的推理规则，因而由于过于复杂，使其应用受到很大的限制。

BAN 逻辑和 GNY 逻辑成功地分析了一些协议的缺陷，但它们都没有对逻辑系统自身进行形式化的语义分析。因此，Abadi 和 Tuttle 提出了 AT 逻辑。一方面，AT 逻辑从语义的角度分析了 BAN 逻辑，并进行了改进。另一方面，AT 逻辑给出了形式化语义，并证明了其推理系统的合理性。作为形式化的分析工具，AT 逻辑比 BAN 逻辑和 GNY 逻辑更加自然、简洁，分析能力也更强，但它没有提供基于公钥体制的分析方法。同时，有些公理也存在着缺陷。

SVO 逻辑吸收了上述逻辑的优点，将他们集成在一个逻辑系统中，在形式化语义方面，SVO 逻辑对一些概念作了重新定义，取消了上述逻辑的一些限制。因此 SVO 逻辑是 BAN 类逻辑中的佼佼者，它的理论基础更加坚实，在实用方面仍然保持了 BAN 逻辑简单、易用的特点，因此被广泛接受。最近，也有国内外学者利用 SVO 逻辑来分析电子商务协议。

随着电子商务的兴起，Kailar 首先注意到对非否认性和可追究性进行形式化分析的重要，他指出 BAN 逻辑和 BAN 类逻辑不适于分析非否认协议。其根本原因是，信念逻辑的目的是证明某个主体相信某个公式，而非否认协议的最终目的是某个主体能向公众证明某个公式。为此，Kailar 提出了新的逻辑分析方法，即 Kailar 逻辑 [53]。运用 Kailar 逻辑对协议进行形式化分析一般有下列几个步骤：首先明确协议的非否认性或可追究性目标；然后将协议的语句转化成为逻辑公式，并初始化协议假设条件；最后运用已有的推理规则进行形式化分析，看协议是否达到既定目标。Kailar 逻辑的主要缺陷表现为：首先，Kailar 逻辑只能分析协议的非否认性和可追究

性，不能分析协议的公平性。这是其最主要的缺陷，也是 Kailar逻辑进一步改进的方向；其次，Kailar逻辑在解释协议语句时，只能解释那些签过名的明文消息，这就限制了它的使用范围；第三，Kailar逻辑在推理之前需要引入一些初始化假设。而这一过程是一个非形式化的过程，往往不恰当的初始化假设将导致协议分析的失败。这也是一切逻辑方法存在的一个难题。

此外，基于知识与信念推理的模态逻辑方法还有很多，诸如分析与时间相关的协议的CS逻辑 [39]，以及为突破主体信仰与知识的单调性而提出的 Nonmonotomic逻辑 [70]等等。

1.3.2 基于定理证明的方法

这种方法可分为两类，一类是推理构造方法，另一类是证明构造方法。

在推理构造方法中，重要的工作包括：Meadows的NRL协议分析器方法 [58]，它发现了许多已知的和未知的安全协议漏洞，并成功的用于分析IETF标准-Internet密钥交换协议IKE [59]。此外，还有Cervesato等学者的基于线性逻辑的协议验证方法 [32]，Millen等学者的基于逻辑规则的协议验证方法 [62]。推理构造方法的优点是可以发现攻击，并可以证明协议在多回合运行下的正确性，其缺点是需要用户介入，防止状态爆炸。

Kemmerer等人使用了一阶谓词逻辑扩展，一种名为 Ina Jo的形式化说明语言 [54]来试图解决问题。Ina Jo和后来研制的 ITP是证明构造方法的典型代表。

此外，证明构造方法还包括 Bolignano提出的 human-readable证明法 [27]，可进行脆弱性分析或形式化代码验证。他将重点放在明确区分主体的可信度上，以及对他们所扮演的角色、信仰、控制结构、增强排序约束、使用临时特征表示认证属性等，并运用强有力的不变式技术和攻击者知识公理，使得认证过程类似于基于模型的验证方法。

Paulson的基于归纳的定理证明方法 [68, 69]，是将协议归纳定义为所有可能事件路径的集合，每条路径是一个包含多轮协议通信的事件序列。这种方法可以模拟所有攻击和密钥丢失。部分分析过程可用 Isabelle定理证明器通过对路径的归纳来自动证明性质的成立。

Thayer, Herzog和 Guttman提出了一种新的概念——串空间 (strand space) [79–81], 该方法集 NRL协议器, Schneider的秩函数以及 Paulson的归纳法思想之大成, 是一种新型有效的协议形式化方法。

证明构造方法的优点是可以分析无限大小的协议, 不限制主体参与协议运行的回合。其缺点是证明过程不能全部自动化, 需要人工进行“专家式”的干预, 在适用范围上受到一定的限制。

1.4 基于进程演算的安全协议形式化分析概述

1.4.1 进程演算

进程演算是一种刻画并发计算的代数模型。

上世纪80年代初, Robin Milner提出了通信系统演算 CCS(Calculus of Communicating Systems)[60, 61]。CCS通过通信来刻画并发计算, 所有的计算都由进程之间的通信来完成, 而进程之间的并发理解为通信动作的交错加上不确定性。CCS的语法非常简单, 它只含有前缀、并置、选择、限制、换标号和递归这几个算子, 但具有较强的描述能力。

CSP(Communicating Sequential Processes)模型是 Hoare提出的 [49, 50], 在时间上略早于 CCS。但对这两个模型的研究是平行地进行的。正如它的名字所表示的那样, CSP允许我们把系统描述为由许多组件组成的整体。这些组件也就是进程, 它们的操作是独立的, 并且通过命名的通道在互相之间进行通信。(值得注意的是“进程必须是顺序的”限制在1978年到1985年之间被取消掉了, 但是该模型的名称保持不变。) CSP的方法非常适合许多问题的结构, 从而使得它成为一种强有力的对并发系统建模的工具。

CCS和 CSP都能刻画并发系统的非确定性、通信、递归、抽象、分流和死锁等行为, 但是它们所采用的刻画方法各不相同。譬如在刻画非确定性方面, CSP 有两种选择算子: 外部选择算子“ \square ”和内部选择算子“ \sqcap ”。前者依赖环境作出选择从而具有确定性, 而后者具有非确定性。在 CCS中只有一种选择算子“+”。非确定性并不是该算子的一个属性, 而是两个候选进程的共同属性。当环境不参与选择时, 就表现出完全的非确定性; 而当环境参与选择时, 就表现出确定性的一面。又譬

如在刻画通信行为方面，对于通信和交错，CSP分别有两个算子：“ \parallel ”和“ $\|\!\!\|$ ”。而CCS只有一个并发子“ $|$ ”。当两个CCS的进程分别有互补的动作时，才可以发生通信，通信的结果是一个内部动作“ τ ”，由于通信的结果是不可见的，所以通信在CCS中起的是同步的作用。总的来说，CSP更接近于实际的程序设计语言，而CCS则是一种演算。

基于互模拟概念的行为等价理论的建立和完善，是CCS能成功地用于并发系统建模和并发程序验证的主要原因。互模拟等价是进程演算中最为重要的等价关系。进程演算中大部分的研究就是围绕着互模拟等价这个概念展开的。根据是否忽略表示系统内部通信的 τ -动作，互模拟等价又可以分为强互模拟等价和弱互模拟等价。强互模拟等价关系要求两个系统互相模拟所有的动作，包括内部通信；而弱互模拟等价关系则忽略 τ -动作。基于弱互模拟的观察同余关系是实际应用中最重要关系，因为它所刻画的等价关系最接近人的直观。

在进程演算的研究中，通信一直是并发计算模型的中心。纯CCS中的通信是指进程之间的同步，而不描述进程间传递的数据，数据的传递通过同步来描述；在传值CCS中，进程间可以传递数据，但是这就显示处理CCS的局限性，因为CCS中传递的数据不是CCS中固有的元素，即CCS变成了不封闭。CCS的另一个局限是它只能描述静态结构的并发系统，对于具有动态通信拓扑结构的系统无法进行描述。

上世纪90年代初，Robin Milner等人又提出了 π 演算 [66]。在CCS中，进程在通信过程中接收到的数据不能在以后的通信中用作通道名，而 π 演算是一个封闭的系统，在通信中只能传输通道名，这使得 π 演算的表达能力得到增强，能表示通信拓扑结构可以动态变化的系统（所谓的移动进程），还可以解释 λ 演算这一经典的顺序计算模型。许多CCS中的概念在 π 演算中得到了继承和发展，如互模拟等价；同样，CCS中的互模拟等价关系也在 π 演算中都得到了进一步的发展。如强互模拟和弱互模拟、早互模拟和迟互模拟、Barbed互模拟和符号互模拟。测试等价 [28]也有CCS和 π 演算两个版本。而开互模拟则是在 π 演算中提出的一种新的互模拟。

随着对 π 演算研究的深入，人们又提出了 π 演算的多种变体。其中包括多维 π 演算 (Polyadic π calculus)、带不等名测试的 π 演算、 π I 演算、异步 π 演算、L π

演算、Fusion演算、chi演算 [44]等。这些演算在语法和操作语义上与 pi演算存在着差异：多维 pi演算允许多个名的同时输入和输出；带不等名测试的 pi演算是在 pi演算中加入了不等名测试操作子，以方便建立等式公理系统；在 piI 演算中则是只允许受限名的输出；在异步 pi演算中输出动作是异步的，即不用等待输出动作的完成，进程即可继续进行，这与实际中的通信是一致的；Fusion演算和 chi演算在通信机制上与 pi演算不同：pi演算通过输入输出动作传递信息来完成通信，而这两个演算中统一了输入和输出操作，通过变量值的扩散来实现通信，另外，它们也统一了 pi演算中的两类受限名。pi演算中的各种互模拟在它的变体中依然是研究的对象，并且，这些新的演算中产生了许多新的互模拟关系，如在异步 pi演算中的异步互模拟关系及其三种等价定义方式、HT-互模拟等；而在 chi演算中，傅育熙提出了 L-互模拟关系，即只要求进程在某些种类的动作上是互模拟的，并在此之上构造出了互模拟格；在 Fusion演算则是提出了超互模拟的概念。

上述代数理论统称为进程演算，其共同特征为：

1. 均使用通信，而不是共享存储，作为进程之间相互作用的基本手段，表现出面向分布式系统的特点。

2. 在语法上，用一组算子作为进程构件。算子的语义用结构化操作语义方法定义。进程可看作标号迁移系统。

3. 把并发性归结为非确定性，将并发执行的进程的行为看作是各单个进程的行为的所有可能的交错合成，即所谓交错式语义。

由于进程演算对于协议的描述几乎接近协议的本身含义，可以很精确地刻划协议的运行过程。使用进程演算对安全协议分析和验证时，协议的每一个主体都被建模为一个单独的进程（在一些研究方法中，攻击者也被建模成为一个单独的进程），这些子进程并发运行，并使用进程之间的共享通道进行同步通信，这样得到的并发系统将作为安全协议的基本模型。

利用进程演算对安全协议进行分析和验证时，既借鉴了进程演算作为代数模型的基本验证理论与方法，又使用了进程演算作为抽象程序设计语言的程序分析方法。因此，基于进程演算的安全协议验证技术可分为模型检测技术、互模拟验证技术和程序分析技术。

上世纪90年代末, Cardelli和 Gordon提出了一种新的并发计算模型 ambient演算[36]。ambient演算有明确的位置和区域的概念, 计算的能力主要是基于进程的移动。目前对 ambient演算的研究主要集中在它的类型系统和安全性的相互作用方面, 基于ambient演算的网络安全和资源控制的研究也做了许多工作, 但 ambient演算适合于网络系统安全的研究。

1.4.2 安全协议形式化分析的进程演算模型

1.4.2.1 CSP+FDR

CSP是最早用于安全协议的描述与分析的进程演算模型。其方法是模型检测技术, 就是将分析和验证安全协议的问题归约为 CSP进程是否满足其 CSP说明的问题。它将所分析的协议性质与具体的协议形式加以区分, 并在 CSP总体框架下对协议的性质进行分析与验证。CSP具有良好的语义使其可以较好地描述协议是否满足其安全属性这一问题。另一方面, CSP对协议的描述极为接近协议的本身含义。1996年, 英国学者 Gavin Lowe首先启用 CSP模型检测工具 FDR(Failures Divergences Refinement Checker) [55], 来分析并发现了 Needham-Schroeder公钥协议的一个以前从未发现的漏洞。Schneider做了一些基于 CSP模型的安全协议分析工作 [73, 74]。

1.4.2.2 CryptoSPA

1999年, R.Focardi等人提出了一个传值 CCS的变体 CryptoSPA (Cryptographic Security Process Algebra) [43] 用于分析安全协议。在传值 CCS中为进程提供了一些消息处理的原语。特别是, 进程能执行加密解密操作, 也可以通过组合一些简单的消息构造复杂的消息。

CryptoSPA的 agents语法定义如下:

$$E ::= 0 \mid c(x).E \mid \bar{c}\langle e \rangle.E \mid \tau.E \mid E + E \mid E|E \mid E \setminus L \mid E[f] \mid A(m_1, \dots, m_n) \mid [e = e']E; E \mid \langle e_1 \dots e_r \rangle \vdash_{rule} x]E; E$$

1.4.2.3 Spi和类Spi演算

1997年, Abadi和 Gordon在 pi演算的基础上建立了 Spi演算 [9], 它提供了数据加密和解密相应的原语, 可以很方便的描述网络安全协议。这种方法根据 Dolev-Yao模

型，假定协议执行的每一步都可能与攻击者的执行步骤交叉，来验证协议的各种性质。

应用 Spi演算分析安全协议有一些重要的工作，如 Amadio和 Lugiez分析了对称密钥安全协议 [13]，Amadio和 Prasad在分析安全协议时对攻击者的消息构造能力进行了刻画和限制 [14]。Spi演算的缺点是目前还缺乏相应有效的工具支持，而且也只是局限于保密性和认证性的研究。我们定义了一个新的算子，解决了原演算不能验证存在时间戳的安全协议的各种验证，并基于这个扩展了的 Spi演算，验证了 Kerberos 协议的认证性。

2001年，Abadi等学者参照了 lambda演算到 Applied lambda演算的扩展，将 pi演算扩展成 Applied pi演算 [6]。Applied pi演算继承了 pi演算的通信和并发构造，增加了函数和等式原语。消息不仅仅是原子名，还有通过名和函数构成的值。这个演算能很容易地处理标准数据类型（如整型、数组等），而且对安全协议的形式化描述也很方便。譬如，用新名表示新的通道、随机值和密钥，用函数表示密码学原语。相对于 Spi演算，这个演算不需要为特定的密码系统操作构造新的原语，从而有很好的通用性，也就能表达和分析相当复杂的协议（混合了多个密码学原语的协议，如加密、解密、Hash函数、签名等操作原语）。以 Applied pi演算为模型，他们分析了一些新的安全协议 [5]。我们利用 Applied pi演算中定义的等价关系，第一次定义了安全协议的匿名性，并且验证了 ikp协议的部分匿名性。

1.4.3 基于进程演算的安全协议形式化分析技术

1.4.3.1 模型检测技术

模型检测技术是一种验证有限状态系统的自动化分析技术，它对协议的自动验证和协议的工程化设计具有重要意义。

使用模型检测技术进行验证时，对象系统将使用某种建模语言被建模成（有限）状态转换图（称为实现，即 Implementation）。同时，该对象系统需要满足的性质将使用某种规范语言进行描述（一般采用某种时态逻辑约束，称为规范，即 Specification），模型检测技术将使用搜索算法来确定系统实现的状态转换图是否满足某种规范描述。模型检测技术需要搜索整个状态空间，因此，在状态急剧增大

时，模型检测技术的效率也将下降。

使用模型检测技术验证安全协议方面的代表性工作包括 G.Lowe等的研究。G.Lowe将安全协议的参与者看作是并发的 CSP进程，攻击者也被建模为具有多种攻击操作能力（例如窃听、冒充、重放等）的进程。并发的参与者进程与攻击者进程共同组成了安全协议的系统实现描述。G.Lowe同样也将协议的安全性质（规范）描述为CSP进程。例如：保密性可以描述为踪迹中不泄漏有关保密消息的进程；而认证性的定义采用了与 Woo和 Lam所提出的对应性类似的“一致性”定义，它被描述为如下的 CSP进程：当在其踪迹中出现表示主体 B作为响应者完成与主体A的一次协议运行的事件时，在此事件之前必定出现表示 A（作为协议发起者）完成与 B的一次协议运行的事件。

G.Lowe使用了基于 CSP的通用模型检测工具 FDR。FDR接受两个 CSP进程，其中一个为安全协议的 CSP进程（作为实现），另外一个则为描述相应安全性质的 CSP进程（作为规范）。FDR将通过枚举系统实现的所有行为（踪迹），检查其是否包含在规范描述的行为当中（CSP中称之为系统实现为规范的精炼）。当系统实现的行为符合规范时，认为安全协议满足安全性质；当检查失败时，FDR将返回一个违反规范描述的行为（即导致攻击成功的系统踪迹）。

G.Lowe还设计了 Casper程序 [56]，它从安全协议的抽象描述（类似于消息序列的形式）半自动地产生 CSP描述，因此大大简化了建模和分析过程。

由于模型检测技术需要搜索整个状态空间，因此，在状态急剧增大时，模型检测技术的效率也将下降（称为“状态爆炸问题”）。在使用模型检测技术对安全协议进行形式化验证时也将面临这一问题。一般情况下，对这一问题通常采用了对安全协议模型进行限制的方法进行处理。这种限制是两方面的，即：

第一，对参与协议运行的主体个数进行限制。实际的网络环境下可能允许每个主体同时运行多个协议实体，但允许无穷多个协议实例的并发运行显然会带来无穷的协议模型状态空间。使用模型检测技术进行安全协议验证时一般都只允许有限个协议实例的运行。特别是关于保密性，G.Lowe曾经证明 [57]:如果构造一个运行协议的小系统模型（协议各方参与者都只有一个，这样的小系统通常都是一个有限状态转换系统），同时结合一个通常意义下的入侵者模型（即入侵者可窃听及中途拦截系

统中途传送的任何消息、可在系统中插入新的消息（其中可改变明文部分）、可运用它所知道的知识产生新的消息，并且还具有一般的正常用户的能力），如果在这样的协议模型中没有由于对协议的攻击而导致某种安全性破坏的情况，那么在任意系统上一定不会有由于攻击而导致安全性的破坏。

第二，对攻击者可生成的消息数目进行限制。如果攻击者可生成无穷多的消息（事实如此），协议的状态空间也将是无穷的。使用模型检测技术对安全协议进行验证时，一般将对攻击者可生成的消息进行限制，例如，限制地进行加密时所使用的密钥必须为原子密钥（而并非由其它的元素通过加密或其它的函数生成），要求攻击者可生成的消息与协议正常参与者可接受消息具有相同的结构。

为了进一步提高模型检测的效率，安全协议的模型检测技术还借鉴了在进程演算研究领域内非常重要的符号化方法与 Partial Order 方法，以便压缩状态空间。

1.4.3.2 互模拟验证技术

互模拟是进程演算领域的最重要的核心概念之一，基于互模拟概念的行为等价语义理论的成功也是进程演算能够广泛应用于并发系统建模和程序验证的重要原因之一。采用互模拟验证技术进行安全协议的形式化验证的主要思想是：如果一个安全协议进程（实现）与对应的特定理想化进程（规范）是测试等价的（即在各种环境中，协议实现与规范在外部的观察者看来是没有区别的），那么安全协议进程将与该特定理想化系统一样具有相同的安全性质。

采用互模拟验证技术的工作目前主要是在 Spi 演算上展开的。Abadi 和 Gordon 在 Spi 演算中引入了测试等价关系作为进程的行为等价标准，并将安全协议的安全性质（认证性、保密性）均用测试等价来加以刻画。例如，保密性定义为：仅当在安全协议进程使用不相同的保密信息进行通信，而外部观察到的协议进程行为完全相同（测试等价）时，安全协议满足保密性；认证性的定义稍有不同，它也是以测试等价关系来定义的：理想化的协议规范与安全协议实现的形式类似，但在协议规范中，消息接收子进程已经“魔术般地”预知了发送者子进程将要发送的消息，当协议规范与协议实现测试等价时，安全协议满足认证性。

对 Spi 演算的互模拟关系有许多学者作了研究，如 Abadi 和 Gordon 首先研究了

Spi演算互模拟方法 [12], Borgstrom也研究了 Spi上的互模拟关系 [29, 30], 其他学者为了互模拟的自动验证提出了多个等价关系 [23, 24, 38, 40, 42, 51]。

1.4.3.3 程序分析技术

程序分析 [65] 的目的是静态检查一个程序的性质是否在各种执行情况下能保持。而且, 静态分析提供了分析系统性质的自动方法和工具, 比起动态执行来验证程序更有效。最近几年, 用静态的程序分析来检查安全性质得到了很大的发展。程序分析技术主要有四种: 数据流分析、控制流分析、抽象解释和类型系统。

数据流分析是一个成熟的研究领域, 已经开发了许多形式化框架来解决不同的数据流问题。文献 [64] 综合了当前数据流分析的一些形式化框架。数据流分析用于 π 演算只有在文献 [52]中发现, 它讨论了进程间的因果关系和真并发问题。但是, 目前还没有发现安全相关的数据流分析。

控制流分析关注的是回答下列问题: 给定程序中的一个特定的点, 能从那个点出发到达的子程序、函数、命令的集合是什么。换句话说, 一个控制流分析试图记录程序的不同执行路径的信息。早期开发的控制流分析技巧主要用于函数程序, 后来也用于其它的程序(如面向对象的、并发的和逻辑程序)。为了达到一个更精确的控制流分析, 文献 [75]开发了一个 k -CFA概念, k 代表了在分析时要考虑的上下文信息的层次。因此, 0-CFA表示了上下文无关的分析。当 $k > 0$ 时, 其分析就是上下文有关的。动态上下文信息的提出允许区分不同的变量和程序块, 因此, 可以达到更精确的分析结果。现在有几个 k -CFA分析的变体, 如统一 k -CFA, 多态 k -CFA和笛卡儿积算法等。控制流分析还可以结合其它的方法(如数据流分析、抽象解释和流图等), 从而达到更好的分析结果。控制流分析用于安全相关的工作主要是由丹麦技术大学 Neilson夫妇和意大利 Pisa大学 Degano教授等学者来进行的 [20–22, 31], 最近他们合作的一个项目提出了 LySa演算。

设计一个抽象解释的第一步是决定语言的标准语义是否能足够表达所要考虑的性质。如果不能, 必须设计一个非标准语义来扩展或修改标准语义从而能够表达感兴趣的性质。非标准语义有时候被证明相对于标准语义是正确的。有不同的方法来证明正确性关系的存在。抽象分析器的应用已经和语言编译器结合, 用于程序优化,

并且证明程序相对于一定安全策略是安全的。安全相关的抽象解释应用可见文献 [17]。

类型系统已被广泛应用于程序语言，以避免程序运行时不需要的行为。这些行为有些可能对违反安全性质是关键，如私有信息的泄漏等。一般来说，一个类型可以被看作是保持程序实体分类的信息。如果把一个类型看作是值的集合，那么它的子集组成了一个子类型。子类型表示了不同类型之间的序关系，也可以看作子类型实体的信息的精炼。我们可以定义一个主类型作为一个表达式的一般类型。如，在 lambda 演算中， $\lambda x.x$ 有一个主类型， $a \rightarrow a$ ，这里 a 可以实例化为任意类型。语言表达式的主类型常常可以用 Robinson 算法从它们的子表达式类型计算出来。一个类型环境用于映射不同的程序实体（如语句、表达式、常量和变量等）到它们的类型。这样的环境可以依照一个公理和规则的集合手工构造。类型推理算法用来从语言实体的上下文环境中推出它们的类型。类型公理和规则的集合用来判断一个程序在这个类型环境中是否良好。算法中的归约可以用于生成主体类型。无论什么时候一个语言的所有程序被证明是良好的，那么这个语言是类型可靠的。定义一个类型系统应当从构造类型检查算法中分离出来。程序语言上的大多数类型系统是一阶的，就是没有类型参数和抽象的。否则的话，就是二阶的。类型系统可以用特别的评注，称为 effects，表示类型环境中定义的每一个类型的效果。Effects 系统常常作为类型推理算法的扩展来实现，并且用来提供每个程序表达式的内部计算步的信息。类型系统中一个有趣的主题是类型等价。名等价表示这些类型有同样的名匹配，结构等价表示有同样的结构具有相同的能力。实际中，常常是两个类型等价关系混合使用。在文献 [1] 中，Abadi 首先用类型系统研究了在 Spi 演算中的保密性，又在文献 [2] 中研究了不对称密码体制情况，和 Blanchet 合作开发了基于 Prolog 逻辑程序的自动验证工具 [3]。Gordon 和 Jeffrey 参考了 Woo-Lam 模型中对应性的概念，用类型系统研究了 Spi 演算中的认证性 [45]，又在文献 [46] 中研究了不对称密码体制情况，并开发了自动工具 CrypTyc。Cardelli 等研究了一般的方案，类型可以动态地创建 [37]。在文献 [33–35] 中，Cervesato 提出了带类型的 MSR (Multi-Set Rewriting)。

1.5 主要研究内容与成果

进程演算对于协议的描述几乎接近协议的本身含义，可以很精确地刻划协议的运

行过程。使用进程演算对安全协议分析和验证时，协议的每一个主体都被建模为一个单独的进程，这些子进程并发运行，并使用进程之间的共享通道进行同步通信，这样得到的并发系统将作为安全协议的基本模型。利用进程演算对安全协议进行分析和验证时，既借鉴了进程演算作为代数模型的基本验证理论与方法，又使用了进程演算作为抽象程序设计语言的程序分析方法。因此，基于进程演算的安全协议验证技术可分为模型检测技术、互模拟验证技术和程序分析技术三种。本文的内容主要集中在互模拟验证技术方面，主要的贡献有以下两点：

- (1) 本文提出了基于时间戳协议的一种解决方案，Abadi在[9]提出，基于时间戳的协议也可以通过进程代数来分析，但他并没有给出解决方案。本文在Spi演算中利用一个类 match算子给出了基于时间戳安全协议验证的一种解决方案，同时通过扩展了的Spi演算验证了Kerberos协议的认证性，并且指出了其认证的脆弱性。
- (2) 本文提出了匿名性的定义和证明的一种方法，目前尚未有其他基于进程代数对匿名性进行定义和证明的研究论文发表。并且将这种证明方法应用于Applied pi演算中，证明了ikp协议的部分匿名性。同时，在此论文中，笔者采用了一种新的方法来应用对应性断言而不是传统的构造协议规范的方法证明了ikp协议的认证性。

1.6 本文结构

本文共分为四章，除本章外，其余各章组织如下：

第2章定义了带有时间戳算子的扩展Spi演算，并用此演算来验证Kerberos协议的认证性，并且指出了协议的脆弱性所在。

第3章提出了定义和证明匿名性的一种方法，并且基于Applied pi演算，证明了ikp协议的部分匿名性；同时，在本章中，采用了一种新的证明方法应用对应性断言证明了ikp协议的认证性。

第4章简要总结了本文的主要贡献，并且阐述了对基于进程代数的安全协议分析方法的后续工作的一些设想。

第二章 基于 Spi演算的安全协议分析

2.1 引言

1997年, Abadi和 Gordon在 pi演算的基础上建立了 Spi演算 [9], 它提供了数据加密和解密相应的原语, 可以很方便的描述网络安全协议。这种方法根据 Dolev-Yao模型, 假定协议执行的每一步都可能与攻击者的执行步骤交叉, 来验证协议的各种性质。利用 Spi演算分析安全协议有一些重要的工作, 如 Amadio和 Lugiez分析了对称密钥安全协议 [13], Amadio和 Prasad在分析安全协议时对攻击者的消息构造能力进行了刻划和限制 [14]。

Kerberos协议是一种应用于分布式网络环境, 以共享密钥为基础, 对用户及网络联接进行认证的增强网络安全的服务。作为为 TCP/IP网络设计的第三方认证协议, Kerberos协议可以提供安全的网络认证, 检查是否允许客户访问网络中的应用服务器, 目前已经开发到了第5版。

本章通过增加一个时间戳算子扩展了 Spi演算, 利用其描述并验证 Kerberos协议的认证性, 并指出了其缺陷。

2.2 Spi演算的语法和语义

2.2.1 Spi演算的语法

Spi演算的项 (term)是如下定义的集合:

$$L, M, N ::=$$

| | |
|-----------|------------------------|
| n | 名(name) |
| x | 变量(variable) |
| (M, N) | 对(pair) |
| 0 | 零(zero) |
| $suc(M)$ | 后继算子(successor) |
| $\{M\}_K$ | 由密码K加密M的密文(encryption) |

在 π 演算中，名是唯一的项。在 Spi演算中增加描述对 (M, N) ，数字的结构和密文项 $\{M\}_K$ ，是为了便于在安全协议中描述它们。这些结构并不能增加 π 演算的描述能力，引入它们的目的仅仅在于简化安全协议的描述。

进程 (process) 则是由项通过如下规则定义的集合：

| | |
|---|---------------------------|
| $P, Q, R ::=$ | |
| $\bar{M}\langle N \rangle.P$ | 输出(output) |
| $M(x).P$ | 输入(input) |
| $P Q$ | 复合(composition) |
| $(\nu n)P$ | 限制(restriction) |
| $!P$ | 复制(replication) |
| $[M \text{ is } N]P$ | 匹配(match) |
| $\mathbf{0}$ | 空进程(nil) |
| $\text{let } (x, y) = M \text{ in } P$ | 拆对(pair splitting) |
| $\text{case } M \text{ of } 0 : P \text{ suc}(x) : Q$ | 整数分支(integer case) |
| $\text{case } L \text{ of } \{x\}_K \text{ in } P$ | 解密(shared-key decryption) |

其中，拆对进程 $\text{let } (x, y) = M \text{ in } P$ ，如果 M 是对 (N, L) 的话，则结果为 $P[N/x][L/y]$ ，否则为 $\mathbf{0}$ 。整数分支进程 $\text{case } M \text{ of } 0 : P \text{ suc}(x) : Q$ ，如果 M 是 0 的话，则结果为 P ；否则如果存在某个 N ， M 是 $\text{suc}(N)$ ，则结果为 $Q[N/x]$ ，其他情况为 $\mathbf{0}$ 。解密进程 $\text{case } L \text{ of } \{x\}_K \text{ in } P$ ，如果 L 是由 K 加密的密文 N_K 的话，则结果为 $P[N/x]$ ，否则结果为 $\mathbf{0}$ 。

在 $(\nu n)P$ 中， P 中的名 n 称为是受限的 (bounded)；在 $M(x).P$ 中， P 中的变量 x 是受限的；在 $\text{case } M \text{ of } 0 : P \text{ suc}(x) : Q$ 中，变量 x 在第二个分支 Q 中是受限的。如果项中的名 n 不是受限的，则称 n 是自由的 (free)。记进程 P 中所有的自由名的集合为 $f_n(P)$ ，所有自由变量的集合为 $f_v(P)$ 。如果一个进程中 $f_v(P) = \emptyset$ ，则称该进程为闭进程。

2.2.2 Spi演算的语义

我们来定义闭进程下的各种关系：

定义 2.1 (归约关系) 闭进程上的归约关系 (reduction relation) 是有如下规则定义的关系:

$$\begin{aligned}
 & !P > P | !P \\
 & [M \text{ is } M]P > P \\
 & \text{let } (x, y) = (M, N) \text{ in } P > P[M/x][N/y] \\
 & \text{case } 0 \text{ of } 0 : P \text{ suc}(x) : Q > P \\
 & \text{case suc}(M) \text{ of } 0 : P \text{ suc}(x) : Q > Q[M/x] \\
 & \text{case } \{M\}_K \text{ of } \{x\}_K \text{ in } P > P[M/x]
 \end{aligned}$$

定义 2.2 (结构等价) 结构等价 (structural equivalence) 是满足下列等式和规则的闭进程上的最小关系:

$$\begin{aligned}
 & P | \mathbf{0} \equiv P \\
 & P | Q \equiv Q | P \\
 & P | (Q | R) \equiv (P | Q) | R \\
 & (\nu m)(\nu n)P \equiv (\nu n)(\nu m)P \\
 & (\nu n)\mathbf{0} \equiv \mathbf{0} \\
 & (\nu n)(P | Q) \equiv P | (\nu n)Q \quad \text{if } n \notin f_n(P) \\
 & \frac{P > Q}{P \equiv Q} \quad \frac{P \equiv Q}{P \equiv P} \quad \frac{P \equiv Q}{Q \equiv P} \\
 & \frac{P \equiv Q \quad Q \equiv R}{P \equiv R} \quad \frac{P \equiv Q}{P | R \equiv Q | R} \quad \frac{P \equiv Q}{(\nu n)P \equiv (\nu n)Q}
 \end{aligned}$$

定义 2.3 (交互关系) 交互关系 (reaction relation) 是 Milner 在 pi 演算 [66] 中引入的一种简单的关系, 交互关系由组合在一起的两个进程 $\overline{M}\langle N \rangle.P$ 和 $M(x).Q$ 发生的一次交互引起, 它的推导和规则如下:

$$\begin{aligned}
 & \overline{M}\langle N \rangle.P | M(x).Q \rightarrow P | Q[N/x] \\
 & \frac{P \equiv P' \quad P' \rightarrow Q' \quad Q' \equiv Q}{P \rightarrow Q} \quad \frac{P \rightarrow Q}{P | R \rightarrow Q | R} \quad \frac{P \rightarrow Q}{(\nu n)P \rightarrow (\nu n)Q}
 \end{aligned}$$

Spi演算还有另外一种操作语义，叫做委托关系 (commitment relation)[11]。为了定义这种关系，首先要定义几个句法项。

定义 2.4 (抽象 (abstraction)) 抽象记作 $(x)P$ ，其中 x 是受限变量， P 是进程。如果 F 是一个抽象 $(x)P$ ， M 是一个项的话，我们用 $F(M)$ 表示 $P[M/x]$ 。

定义 2.5 (具化 (concretion)) 具化记作 $(\nu m_1, m_2, \dots, m_k)\langle M \rangle P$ ，其中， P 是进程， m_1, m_2, \dots, m_k 是 M 和 P 中的受限名。常用 $(\nu \vec{m})$ 来表示 $(\nu m_1, m_2, \dots, m_k)$ ，所以具化也记作 $(\nu \vec{m})\langle M \rangle P$

由上述的抽象和具化的概念，我们定义交互 (interaction) 关系 $F@C$ 和 $C@F$ 如下：

$$F@C \triangleq (\nu \vec{n})(P[M/x] \mid Q) \quad C@F \triangleq (\nu \vec{n})(Q \mid P[M/x])$$

所以，委托关系定义如下：

定义 2.6 (委托关系) 委托关系 (commitment relation) 写作 $P \xrightarrow{\alpha} A$ ，其中 P 是闭进程， A 是闭项，它由两条规约关系和一系列规则构成：

$$\begin{array}{c} m(x).P \xrightarrow{m} (x)P \\ \vec{m}\langle M \rangle.P \xrightarrow{\vec{m}} (\nu)\langle M \rangle P \\ \hline \frac{P \xrightarrow{m} F \quad Q \xrightarrow{\vec{m}} C}{P \mid Q \xrightarrow{\tau} F@C} \quad \frac{P \xrightarrow{\vec{m}} C \quad Q \xrightarrow{m} F}{P \mid Q \xrightarrow{\tau} C@F} \\ \hline \frac{P \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} A \mid Q} \quad \frac{Q \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} P \mid A} \\ \hline \frac{P > Q \quad Q \xrightarrow{\alpha} A}{P \xrightarrow{\alpha} A} \quad \frac{P \xrightarrow{\alpha} A \quad A \notin \{m, \vec{m}\}}{(\nu m)P \xrightarrow{\alpha} (\nu m)A} \end{array}$$

仅仅有如上的定义，还不足以清楚的来表达一个较为复杂的安全协议。于是我们对上述部分进程语义进行了扩展，定义了多元组和时间戳的概念。

在 Spi演算的项中，只定义了对 (pair)，而对于一个复杂安全协议的消息往往是超过二元的，因此我们使用多元组 (tuple)： $P = (x_1, x_2, x_3, \dots, x_n)$ ，多元组可以由

多个二元组嵌套而成： $P = (\dots((x_1, x_2), x_3), \dots, x_n)$ ，因此并没有改变 Spi演算的表达能力，只是为了表达方便。

同时，我们也扩展了拆对进程 (pair splitting)的语义： $let(x_1, x_2, \dots, x_n) = M in P$ ，含义为如果 M 是一个多元组 (N_1, N_2, \dots, N_n) ，则整个进程的结果是 $P[N_1/x_1][N_2/x_2] \dots [N_n/x_n]$ ，否则进程停止，等价于空进程 (nil)。

为了对多元组的表述更为简洁，我们使用了下面几个简写来描述拆对进程的输入和解密：

$$c(x_1, x_2, \dots, x_n).P \triangleq c(y).let(x_1, x_2, \dots, x_n) = y in P$$

$$case L of \{x_1, x_2, \dots, x_n\}_K in P \triangleq case L of \{y\}_K in let(x_1, x_2, \dots, x_n) = y in P$$

许多安全协议利用时间戳来保证协议的安全。我们往往需要描述时刻 (t)是否在某个有效起止时间 (v)中。在[10]中，作者提出了定义新的原语来描述时间戳，但他没有定义这一原语的语法和语义。其实，我们可以通过定义一个类匹配进程 (match)的算子来完成时间戳的验证： $[t is v].P$ 表示：如果 t 在 v 内，则可以进行 P 进程，否则进程阻塞。

2.3 等价关系

2.3.1 测试等价

为了定义测试等价 (Testing equivalence)，首先需要定义一个操作子，描述进程可以与外界发生交互的能力。我们定义闭进程 $P \Downarrow \beta$ ，如果 m 是自由名，并且 m 是 P 可以与外界交互的通道，即：

$$m(x).Q \Downarrow m \quad \bar{m}\langle M \rangle.Q \Downarrow \bar{m}$$

如果 P 作了若干的动作以后变成 P' ，并且 $P' \Downarrow \beta$ ，则有 $P \Downarrow \beta$ 。因此，显然有：

$$\frac{P \Downarrow \beta}{P \Downarrow \beta} \quad \frac{P \rightarrow Q \quad Q \Downarrow \beta}{P \Downarrow \beta}$$

我们将二元组 (R, β) 定义为一个测试，这个二元组是由一个闭进程 R 和一个通道 β 组成，则有：

定义 2.7 (测试等价) 测试等价 $P \simeq Q$ 可定义为对于任意的测试 (R, β) , $(P \mid R) \Downarrow \beta$ 当且仅当 $(Q \mid R) \Downarrow \beta$

2.3.2 Framed互模拟关系

测试等价的验证不能自动实现, 为了实现互模拟的验证, Abadi定义了一种新的互模拟关系: framed互模拟[12]。这种互模拟关系通过构造框架和理论来定义进程 P 和 Q 之间的等价关系, 比测试等价要强一些。空间和理论并没有刻画两个进程之间的关系, 而是表达了它们能够使环境得到知识的程度。

- (1) 框架是一个有限的名 (name) 集合, 表达了环境可以访问到 P 和 Q 的所有名的集合。我们使用 fr 表达框架的集合。
- (2) 理论是一个有限的项对集合, 一般来说, 一个理论中的项对 (M, N) 表达环境无法区分 P 中的数据 M 和 Q 中的数据 N 。我们用 th 表达理论的集合。

对于断言 $(fr, th) \vdash M \leftrightarrow N$, 是通过下列的规则得出的:

$$\begin{array}{c}
 \frac{}{(fr, th) \vdash 0 \leftrightarrow 0} \text{Eq Zero} \quad \frac{}{(fr, th) \vdash x \leftrightarrow x} \text{Eq Variable} \\
 \\
 \frac{n \in fr}{(fr, th) \vdash n \leftrightarrow n} \text{Eq Frame} \quad \frac{(M, N) \in th}{(fr, th) \vdash M \leftrightarrow N} \text{Eq Theory} \\
 \\
 \frac{(fr, th) \vdash M \leftrightarrow M' \quad (fr, th) \vdash N \leftrightarrow N'}{(fr, th) \vdash (M, N) \leftrightarrow (M', N')} \text{Eq Pair} \\
 \\
 \frac{(fr, th) \vdash M \leftrightarrow M'}{(fr, th) \vdash \text{suc}(M) \leftrightarrow \text{suc}(M')} \text{Eq Suc} \\
 \\
 \frac{(fr, th) \vdash M \leftrightarrow M' \quad (fr, th) \vdash N \leftrightarrow N'}{(fr, th) \vdash \{M\}_N \leftrightarrow \{M'\}_{N'}} \text{Eq Encrypt}
 \end{array}$$

为了定义互模拟关系, 需要首先定义一个概念, 如果下述两个条件满足, 则称 $(fr, th) \vdash ok$:

- (1) 一旦 $(M, N) \in th$, 则:

- (a) M 是闭的, 且存在两个项 M_1 和 M_2 使得 $M = M_1 M_2$, 并且不存在 N_2 , 使得 $(fr, th) \vdash M_2 \leftrightarrow N_2$;
- (b) N 是闭的, 且存在两个项 N_1 和 N_2 使得 $N = N_1 N_2$, 并且不存在 M_2 , 使得 $(fr, th) \vdash M_2 \leftrightarrow N_2$;
- (2) 对于 $(M, N) \in th$, 以及 $(M', N') \in th$, 如果 $M = M'$, 则 $N = N'$ 。

我们定义四元组 (fr, th, P, Q) 为一个 framed 进程对, 在这个四元组中, P 和 Q 是闭进程, fr 是框架, th 是理论。并且用 $(fr, th, P, Q) \in \mathcal{R}$ 来表示 $(fr, th) \vdash PRQ$ 。其中, \mathcal{R} 表示一种 framed 进程的关系, 即, 如果有 $(fr, th) \vdash PRQ$, 则有 $(fr, th) \vdash ok$ 。其中, 我们构造一种 framed 进程关系如下:

定义 2.8 (framed 模拟关系) framed 模拟关系 \mathcal{S} 定义如下: 对于 $(fr, th) \vdash PSQ$ 有下列三个条件成立:

- (1) 如果 $P \xrightarrow{\tau} P'$:
- 则存在一个进程 Q' , 使得 $Q \xrightarrow{\tau} Q'$, 并且有 $(fr, th) \vdash P'SQ'$ 。
- (2) 如果 $P \xrightarrow{c} (x)P'$, 并且 $c \in fr$:
- 则存在一个抽象 (abstraction) $(x)Q'$, 使得 $Q \xrightarrow{c} (x)Q'$, 并且对于所有与集合 $fn(P) \cup fn(Q) \cup fr \cup fn(th)^*$ 不相交的集合 $\{\vec{n}\}$ 和所有的闭 M 和 N , 如果 $(fr \cup \{\vec{n}\}, th) \vdash M \leftrightarrow N$, 则有 $(fr \cup \{\vec{n}\}, th) \vdash P'[M/x]SQ'[N/x]$ 。
- (3) 如果 $P \xrightarrow{\vec{c}} (\nu \vec{m})(M)P'$, $c \in fr$, 并且集合 $\{\vec{m}\}$ 与集合 $fn(P) \cup fn(\pi_1(th)) \cup fr$ 不相交:
- 则存在一个具化 (concretion) $(\nu \vec{n})(N)Q'$, $Q \xrightarrow{\vec{c}} (\nu \vec{n})(N)Q'$, 集合 $\{\vec{n}\}$ 与集合 $fn(Q) \cup fn(\pi_2(th)) \cup fr$ 不相交, 并且存在一对 (fr', th') , $(fr, th) \preceq (fr', th')$ [†], $(fr', th') \vdash M \leftrightarrow N$, 且 $(fr', th') \vdash P'SQ'$ 。

定义 2.9 (framed 互模拟等价) framed 互模拟关系是指存在一个 framed 关系 \mathcal{S} , 使得 \mathcal{S} 和 \mathcal{S}^{-1} 都是 framed 模拟关系。而 framed 互模拟等价 (记作 \sim_f) 是指最大的 framed 互模拟关系。

* 对于一个理论 th , 我们令它的自由变量集合 $fn(th) = \cup\{fn(M) \cup fn(N) \mid (M, N) \in th\}$ 。

† framed 的序关系请见参考文献[12]

2.4 基于 Spi演算的形式化验证

在本节中，我们利用上述改进后的理论来验证一下 Kerberos协议的认证性。

2.4.1 Kerberos 协议

Kerberos协议[67, 72]是为 TCP/IP网络设计的第三方认证协议。它可以提供安全的网络认证，检查是否允许客户访问网络中的应用服务器，目前已经开发到了第5版。

Kerberos协议很简明，客户从 Kerberos服务器请求 TGS (Ticket Granting Service)票据，作为访问 TGS服务器的凭证。Kerberos服务器将 TGS票据用 TGS服务器的私有密钥加密后，发送给客户。客户将该票据提供给 TGS服务器，然后从 TGS服务器请求应用服务器票据。申请成功后，客户将持有的应用服务器票据发送给应用服务器，如果票据内的信息得到应用服务器确认，应用服务器便会让客户访问该服务。

为了便于描述 Kerberos协议，本文使用了一些缩写，如表 2-1所列： Kerberos协议的过程如下：

(1) 客户到 Kerberos服务器

客户向 Kerberos服务器发送请求消息，该消息包括客户名及 TGS服务器名： (c, tgs) 。

(2) Kerberos服务器到客户

Kerberos服务器如果鉴别客户合法，便产生一个在客户和 TGS服务器之间使用的会话密钥，并且用客户的私有密钥将之加密；同时产生 TGS票据，用来向 TGS证实客户的身份，并用 TGS的私有密钥对其加密。TGS票据的格式如下： $T_{c,tgs} = (c, a, v, K_{c,tgs})$ 。Kerberos服务器将这两种加密的消息发送给客户： $(\{K_{c,tgs}\}_{K_c}, \{T_{c,tgs}\}_{K_{tgs}})$ 。

(3) 客户到 TGS服务器

客户用自己的私有密钥解开传来的第一个加密消息，得到 Kerberos服务器传来的它与 TGS服务器的会话密钥；接着产生鉴别码，并用会话密钥加密。产

表 2-1: Kerberos协议的缩写

| 缩写 | 代表内容 |
|-----------|---------------------|
| c | 客户机名 |
| tgs | TGS服务器名 |
| s | 应用服务器名 |
| a | 客户的网络地址 |
| v | 票据的有效起止时间 |
| t | 时间标记 |
| key | 备用密码 |
| K_x | x 的私有密钥 |
| $K_{x,y}$ | x 与 y 的会话密钥 |
| $\{m\}_K$ | 以密钥 K 加密 m 后的密文 |
| $T_{x,y}$ | 使用 y 的 x 的票据 |
| $A_{x,y}$ | 从 x 到 y 的鉴别码 |

生的鉴别码格式如下： $A_{c,tgs} = \{c, t, key\}$ 。同时将 Kerberos传来的第二种加密消息一起传给 TGS服务器，请求获取应用服务器票据。（这一项工作一般由程序自动地完成，对于客户来说是透明的），它发送给TGS服务器的消息如下：

$(\{A_{c,tgs}\}_{K_{c,tgs}}, \{T_{c,tgs}\}_{K_{tgs}})$ 。

(4) TGS服务器到客户

TGS服务器接收到请求后，用自己的私有密钥解密 TGS票据，然后再用票据中的会话密钥解密鉴别码。最后，TGS比较鉴别码中的信息与 TGS票据中的信息，如果两者之间的客户名、客户地址吻合，并且鉴别码中的时间标记在票据有效起止时间范围内，便允许处理该请求。它产生应用服务器票据，并用服务器的私有密钥加密，返回给客户。应用服务器票据的格式如下： $T_{c,s} = (c, a, v, K_{c,s})$ 。同时，TGS还为客户和应用服务器产生一个新的会话密钥，此密钥由客户和 TGS 共享的会话密钥加密。然后将应用服务器名、加密后的会话密钥和应用服务器票据一起返回给用户： $(s, \{K_{c,s}\}_{K_{c,tgs}}, \{T_{c,s}\}_{K_s})$ 。

(5) 客户到应用服务器

客户再次产生一个鉴别码，鉴别码由客户名、客户网络地址和时间标记组成，格式如下： $A_{c,s} = \{c, a, t\}$ 。用客户和应用服务器会话密钥对其加密后，连同应用服务

器票据一起发给应用服务器： $(\{A_{c,s}\}_{K_{c,s}}, \{T_{c,s}\}_{K_s})$ 。应用服务器进行相关确认，通过客户认证后，就可以在应用服务器票据所规定的有效起止时间内用它与客户的会话密钥加密消息，进行会话了。

2.4.2 Kerberos协议形式化表示

定义 Kerberos服务器接受客户请求的通道为 c_{ker} ，服务器由此通道接收客户 c 一个消息，如果消息是二元组 (x_c, x_{tgs}) ，那么通过客户的 c_{x_c} 通道送出由两个密文组成的二元组 $(\{K_{x_c, x_{tgs}}\}_{K_{x_c}}, \{T_{x_c, x_{tgs}}\}_{K_{x_{tgs}}})$ 。

Kerberos服务器的形式化描述如下：

$$KERBEROS \triangleq c_{ker}(x_c, x_{tgs}).\overline{c_{x_c}}(\{K_{x_c, x_{tgs}}\}_{K_{x_c}}, \{T_{x_c, x_{tgs}}\}_{K_{x_{tgs}}})$$

其中 $T_{x_c, x_{tgs}}$ 是一个四元组，由客户名称（由通道 c_{ker} 传来的变量）、客户地址、票据有效起止时间和会话密钥组成： $(x_c, a_{x_c}, v_{x_c}, K_{x_c, x_{tgs}})$

TGS服务器接收到二元组 (x_{cipher}, y_{cipher}) ，它首先用自己的私有密钥解开第二个密文，从中取得会话密钥；再解开第一个密文，如果两个密文中的几个信息吻合（使用 match原语），则由 c_{x_c} 通道送出一个三元组：应用服务器名，加密的会话密钥和应用服务器票据： $(s, \{K_{x_c, s}\}_{K_{x_c}}, \{T_{x_c, s}\}_{K_s})$ TGS服务器的形式化描述如下：

$$TGS \triangleq c_{tgs}(x_{cipher}, y_{cipher}).case\ y_{cipher}\ of\ \{x_c, x_a, x_v, x_k\}_{K_{tgs}}\ in\ case\ x_{cipher}\ of\ \{y_c, y_t, y_{key}\}_{x_k}\ in\ [x_c\ is\ y_c].[y_t\ is\ x_v].\overline{c_{x_c}}(\{s, \{K_{x_c, s}\}_{x_k}, \{T_{x_c, s}\}_{K_s}\})$$

与 Kerberos服务器一样， $T_{x_c, s}$ 也是一个四元组，由客户名称（由密文中得知）、客户地址、票据有效起止时间和会话密钥组成： $(x_c, a_{x_c}, v_{x_c}, K_{x_c, s})$ 。

应用服务器接收到二元组 (x_{cipher}, y_{cipher}) ，它首先用自己的私有密钥解开第二个密文，从中取得会话密钥；再解开第一个密文，如果两个密文中的几个信息都吻合则可以验证通过，让客户来使用自己的服务。应用服务器的形式化描述如下：

$$S \triangleq c_s(x_{cipher}, y_{cipher}).case\ y_{cipher}\ of\ \{x_c, x_a, x_v, x_k\}_{K_s}\ in\ case\ x_{cipher}\ of\ \{y_c, y_a, y_t\}_{x_k}\ in\ [x_c\ is\ y_c].[x_a\ is\ y_a].[y_t\ is\ x_v].c_s(req_{cipher}).case\ req_{cipher}\ of\ \{x_{req}\}_{x_k}\ in\ F(x_{req})$$

如果客户通过了验证，应用服务器就可以按客户的需求处理事务，这里用 $F(x)$ 来表示处理事务。

客户首先向 Kerberos服务器发出请求，得到二元组后，用自己的私有密钥解开第一个密文，内容是与TGS服务器的会话密钥；然后使用这个密钥加密鉴别码，将鉴别码和得到的第二个密文发往TGS服务器，如果成功，得到另一个二元组；它使用与TGS服务器的会话密钥解开第一个密文，内容是与应用服务器的会话密钥；接着使用这个密钥加密新产生的鉴别码，将此鉴别码和第二次得到的第二个密文发往应用服务器，如果成功，则可以访问服务器。

$$C(REQ) \triangleq \overline{c_{ker}} \langle (c, tgs) \rangle . c_c(x_{cipher}, y_{cipher}) . case\ x_{cipher}\ of\ \{k_{c,tgs}\}_{K_c}\ in\ \overline{c_{tgs}} \langle (\{A_{c,tgs}\}_{k_{c,tgs}}, y_{cipher}) \rangle . c_c(x_s, z_{cipher}, w_{cipher}) . case\ z_{cipher}\ of\ \{k_{c,x_s}\}_{k_{c,tgs}}\ in\ \overline{c_s} \langle (\{A_{c,x_s}\}_{k_{c,x_s}}, w_{cipher}) \rangle . \overline{c_{x_s}}(REQ)$$

这里， $A_{c,tgs}$ 是一个三元组 (c, t, key) ，包括客户名 (c) ，当前时刻 (t) 和一个备用密码 key ，在我们描述的系统，备用密码并没有得到使用。 $A_{c,s}$ 也是一个三元组 (c, a, t) ，包括客户名 (c) ，客户地址 (a) 和当前时刻 (t) 。 REQ 代表客户对应用服务器的申请，通过应用服务器的通道发给应用服务器，以求得到应用服务器的响应。

除了各个密钥是保密以外，所有的通道都是公开的。因此，整个具有 Kerberos协议的系统可以有如下定义。假设这个系统需要处理 n 个客户请求 $REQ_1, REQ_2, \dots, REQ_n$ ，则系统表示如下：

$$Sys(REQ_1, REQ_2, \dots, REQ_n) \triangleq (\nu K_c)(\nu K_s)(\nu K_{tgs})(\nu K_{c,tgs})(\nu K_{c,s}) \\ (!KERBEROS|!TGS|!S| \prod_{i \in 1 \dots n} C(REQ_i))$$

在这里，我们已经简化多个 TGS服务器和应用服务器为一个（这一简化并不影响后面的验证）。

2.4.3 Kerberos协议认证性的定义

我们对协议的认证性 (authentication) 有如下定义：协议的接受方可以确认信息来自协议的发送方。用 Spi来验证一个协议的认证性，需要首先定义该协议规范 (specification)，然后证明对于任意的客户请求，形式化描述的协议和协议规范满足某种等价要求。在此，我们将采用测试等价 (testing equivalence) 关系来证明 Kerberos协议的认证性：

对于任意的请求 $REQ_1, REQ_2, \dots, REQ_n$, 都有

$$Sys(REQ_1, REQ_2, \dots, REQ_n) \simeq Sys_{spec}(REQ_1, REQ_2, \dots, REQ_n)$$

2.4.4 Kerberos协议的规范

对于Kerberos服务器和TGS服务器来说, 他们只负责认证的作用, 并没有信息的传递。因此规范与形式化描述相同:

$$KERBEROS_{spec} \triangleq KERBEROS$$

$$TGS_{spec} \triangleq TGS$$

对于应用服务器和客户双方有信息传递, 因此我们可以这样定义两者的规范, 客户并没有发送 REQ 到服务器, 它只是发送了一个私有通道 p ; 服务器收到信息以后, 从该通道发送一个信号 TRI , 客户接收此信号后就作 $F(REQ)$, 因此, 服务器和客户的规范如下:

$$\begin{aligned} S_{spec} \triangleq & c_s(x_{cipher}, y_{cipher}).case\ y_{cipher}\ of\ \{x_c, x_a, x_t, x_k\}_{K_s}\ in \\ & case\ x_{cipher}\ of\ \{y_c, y_a, y_t\}_{x_k}\ in\ [x_c\ is\ y_c].[x_a\ is\ y_a].[x_t\ is\ y_t]. \\ & c_s(ch_{cipher}).case\ ch_{cipher}\ of\ \{p\}_{x_k}\ in\ \bar{p}\langle TRI \rangle \end{aligned}$$

$$C_{spec}(REQ) \triangleq (\nu p)(C(p)|p(x).F(REQ))$$

$$Sys_{spec}(REQ_1, REQ_2, \dots, REQ_n) \triangleq (\nu p)(\nu K_c)(\nu K_s)(\nu K_{tgs})(\nu K_{c,tgs})(\nu K_{c,s})$$

$$(!KERBEROS_{spec}|!TGS_{spec}|!S_{spec}| \prod_{i \in 1 \dots n} C_{spec}(REQ_i))$$

2.4.5 Kerberos协议的认证性及缺陷

在这里, 我们仅对 Kerberos协议的认证性的证明作一简要的说明。

对于任意的测试 (R, β) , 如果有 $(Sys(REQ_1, REQ_2, \dots, REQ_n)|R) \Downarrow \beta$, 则 β 通道必定属于 $Sys(REQ_1, REQ_2, \dots, REQ_n)$ 或 R , 而不是两者作内部动作以后出现的新通道。这是因为有时间戳的验证 $[y_t\ is\ x_v]$, 任何非合法客户利用公有通道发出的消息都不能通过时间戳的检验, 而使得进程阻塞。由于

$Sys_{spec}(REQ_1, REQ_2, \dots, REQ_n)$ 可以和 $Sys(REQ_1, REQ_2, \dots, REQ_n)$ 有相同的外部通道, 则 $(Sys_{spec}(REQ_1, REQ_2, \dots, REQ_n)|R) \Downarrow \beta$ 。反之亦然, 因此有

$$Sys(REQ_1, REQ_2, \dots, REQ_n) \simeq Sys_{spec}(REQ_1, REQ_2, \dots, REQ_n)$$

但是, 上面的测试等价是建立在时间戳的确起作用的基础上, 如果 R 可以传递一个消息满足时间戳验证, 则 Kerberos协议是不安全的, 它无法阻止重放 (replay attack)。对于形式化而言, 如果我们去掉 $[x_t \text{ is } y_t]$ 匹配项, 则两者不是测试等价的, 下面定义的一个测试可以说明这个问题:

我们首先假设 $F(x) \triangleq \bar{c}_t(x)$, 其中 c_t 是一个新通道。令

$$R \triangleq c_c(u).\bar{c}_c\langle u \rangle.\bar{c}_c\langle u \rangle.c_t(x).c_t(y)[y \text{ is } x].\bar{d}\langle * \rangle$$

则有 $(Sys(REQ_1, REQ_2, \dots, REQ_n)|R) \Downarrow d$, 但没有 $(Sys_{spec}(REQ_1, REQ_2, \dots, REQ_n)|R) \Downarrow d$, 所以:

$$Sys(REQ_1, REQ_2, \dots, REQ_n) \not\approx Sys_{spec}(REQ_1, REQ_2, \dots, REQ_n)$$

2.5 结论与相关工作

Abadi在[9]提出, 基于时间戳的协议也可以通过进程演算来分析, 但他并没有给出解决方案。本章增加了一个时间戳算子, $[t \text{ is } v].P$ 来表示: 如果 t 在 v 内, 则可以进行 P 进程, 否则进程阻塞。

同时, 本章利用扩展的 Spi演算验证了 Kerberos协议的认证性, 并且指出了其认证的脆弱性: 认证性都集中在了时间戳的保证下, 一旦时间戳被攻破, 该协议将无法保证认证性。

第三章 基于 Applied pi演算的安全协议分析

3.1 引言

2001年, Abadi等学者参照了 lambda演算到 Applied lambda演算的扩展, 将 pi演算扩展成 Applied pi演算 [6]。Applied pi演算继承了 pi演算的通信和并发构造, 增加了函数和等式原语。消息不仅仅是原子名, 还有通过名和函数构成的值。这个演算能很容易地处理标准数据类型 (如整型、数组等), 而且对安全协议的形式化描述也很方便。譬如, 用新名表示新的通道、随机值和密钥, 用函数表示密码学原语。相对于 Spi演算, 这个演算不需要为特定的密码系统操作构造新的原语, 从而有很好的通用性, 也就能表达和分析相当复杂的协议 (混合了多个密码学原语的协议, 如加密、解密、Hash函数、签名等操作原语)。Applied pi演算 [6]提出了观察等价和静态等价两种等价关系, 使之更好的描述和验证具体的安全协议 [4, 5, 7]。

电子商务是以计算机互联网络和信息技术为平台的经济交易活动, 目前已经渗透到社会生活的诸多方面, 成为当今经济增长的重要力量。作为电子商务的关键, 电子支付协议对于安全的性质和功能有着特殊且更高的要求。除了保密和认证的安全要求以外, 性能不同的电子支付协议还有其他的安全要求, 如不可否认性、公平性、原子性、匿名性等。目前, 越来越多的电子支付协议正在不断的涌现, 如何保证它们的实现达到既定安全要求成为计算机科学家主要研究的问题。

本章利用 Applied pi演算来验证典型电子支付协议——IBM公司提出的 ikp协议族的认证性及其匿名性。

3.2 语法和语义

3.2.1 Applied pi演算的语法

我们首先定义 Σ 为一函数集合，其中包含有限个函数项。则给定一个 Σ ，Applied pi演算的项定义如下：

$$L, M, N ::=$$

| | |
|---------------------------|------------------------------|
| a, b, c, \dots | 名 (name) |
| x, y, z | 变量 (variable) |
| $f(M_1, M_2, \dots, M_l)$ | $f \in \Sigma$ 函数 (function) |

Applied pi演算的进程分为两种，一种是普通进程，其表达意义与 pi演算基本相同；另一种是增加了主动替换的扩展进程，增加了其对替换描述的灵活性。其中，我们定义Applied pi演算的普通进程 (plain processes) 如下：

$$P, Q, R ::=$$

| | |
|-------------------------------|------|
| 0 | 空进程 |
| $P \mid Q$ | 并行复合 |
| $!P$ | 复制 |
| $(\nu n).P$ | 受限 |
| $if\ M = N\ then\ P\ else\ Q$ | 条件 |
| $u(x).P$ | 输入 |
| $\bar{u}\langle M \rangle.P$ | 输出 |

普通进程所表达的意义与 pi演算基本相同，只是在条件进程 $if\ M = N\ then\ P\ else\ Q$ 中， $M = N$ 表达的是一种代数相等，而不是严格意义上的句法相等。若 $Q \equiv 0$ ，条件进程也可简化为 $if\ M = N\ then\ P$ 。

普通进程加入了主动替换 (active substitution) 原语，可以被扩展成为扩展进程

(extended processes), 定义如下:

$$\begin{array}{ll}
 A, B, C ::= & \\
 P & \text{普通进程} \\
 A \mid B & \text{并行复合} \\
 (\nu n).A & \text{名受限} \\
 (\nu x).A & \text{变量受限} \\
 \{M/x\} & \text{主动替换}
 \end{array}$$

主动替换是 Applied pi演算首次提出的, 其功能类似于其它演算的指派进程 $let\ x = M\ in\ \dots$, 但与之不同的是 $\{M/x\}$ 可以与任何一个进程复合, 所表达能力大于指派进程, 所以上述两个进程有以下的关系:

$$\nu x(\{M/x\} \mid P) \equiv let\ x = M\ in\ P$$

3.2.2 Applied pi演算的语义

我们来定义 Applied pi演算的各种形式语义。

3.2.2.1 结构等价

定义 3.1 (结构等价) 结构等价 (structural equivalence) \equiv 是定义在扩展进程上的最小的等价关系, 它在名和变量的 α -换元下是封闭的。结构等价定义如下:

$$\begin{array}{ll}
 \text{PAR-0} & A \equiv A \mid 0 \\
 \text{PAR-A} & A \mid (B \mid C) \equiv (A \mid B) \mid C \\
 \text{PAR-C} & A \mid B \equiv B \mid A \\
 \text{REPL} & !P \equiv P \mid !P \\
 \text{NEW-0} & (\nu n)0 \equiv 0 \\
 \text{NEW-C} & (\nu m)(\nu n)A \equiv (\nu n)(\nu m)A \\
 \text{NEW-PAR} & (\nu n)(A \mid B) \equiv A \mid (\nu n)B \quad \text{if } n \notin f_n(A) \\
 \\
 \text{ALIAS} & (\nu x).\{M/x\} \equiv 0 \\
 \text{SUBEST} & \{M/x\} \mid A \equiv \{M/x\} \mid A\{M/x\} \\
 \text{REWRITE} & \{M/x\} \equiv \{M/x\} \quad \text{when } \Sigma \vdash M = N
 \end{array}$$

由上述定义，任何一个闭扩展进程都可以写成一个主动替换和一个普通进程的并行复合，同时加上若干受限名：

$$A \equiv (\nu \tilde{n})\{\nu \tilde{M} / \nu \tilde{x}\} | P$$

3.2.2.2 内规约

定义 3.2 (内规约关系) 内规约关系 (internal reduction)是在结构等价和上下文中封闭的最小关系，它的定义如下：

$$\begin{aligned} \text{COMM} \quad & \bar{a}\langle x \rangle.P | a(x).Q \rightarrow P | Q \\ \text{THEN} \quad & \text{if } M = M \text{ then } P \text{ else } Q \rightarrow P \\ \text{ELSE} \quad & \text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \quad \Sigma \not\vdash M = N \end{aligned}$$

在 pi演算或类 pi演算中常见的规约关系可以由内规约关系和结构等价得出：

$$\begin{aligned} \bar{a}\langle M \rangle.P | a(x).Q &\equiv \nu x.(\{M/x\} | \bar{a}\langle x \rangle.P | a(x).Q) \\ &\rightarrow \nu x.(\{M/x\} | P | Q) \\ &\equiv P | Q\{M/x\} \end{aligned}$$

3.2.2.3 标号转移语义

Applied pi演算的标号转移语义是前两种语义必要的扩展，使我们可以对进程之间，进程与环境的交互进行推理。Applied PI演算的标号转移语义由如下几条规则构

成:

$$\text{IN} \quad a(x).P \xrightarrow{a(M)} P\{M/x\}$$

$$\text{OUT-TEAM} \quad \bar{a}\langle M \rangle \xrightarrow{\bar{a}\langle M \rangle} P$$

$$\text{OPEN-CHANNEL} \quad \frac{A \xrightarrow{\bar{a}(b)} A' \quad b \neq a}{\nu b.A \xrightarrow{\nu b.\bar{a}(b)} A'}$$

$$\text{OPEN-VARIABLE} \quad \frac{A \xrightarrow{\nu \tilde{u}.\bar{a}\langle M \rangle} A' \quad x \in fv(M) \setminus \{\tilde{u}\}}{\nu x.A \xrightarrow{\nu x.\tilde{u}.\bar{a}\langle M \rangle} A'}$$

x 可以从 $\nu \tilde{u}.\{M/x\} \mid A'$ 推导出

$$\text{SCOPE} \quad \frac{A \xrightarrow{\alpha} A' \quad u \text{不在} \alpha \text{中}}{\nu u.A \xrightarrow{\alpha} \nu.A'}$$

$$\text{PAR} \quad \frac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

$$\text{STRUCT} \quad \frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'}$$

3.3 等价关系

3.3.1 观察等价

如果 A 通过通道 a 送出一个消息, 也即, 如果存在某个上下文 (context)* $C[]$, 使得 $A \rightarrow^* C[\bar{a}\langle M \rangle.P]$, 我们可定义为 $A \Downarrow a$.

定义 3.3 (观察等价) 观察等价 (Observational equivalence) (\approx) 是闭扩展进程上关系 \mathcal{R} 的最大对称关系。 \mathcal{R} 的定义如下, 如果 $A \mathcal{R} B$ 则有:

*这里的上下文与传统意义上的不同, 它只表示在并行复合、受限和主动替换下的上下文。

1. 如果 $A \Downarrow a$, 则 $B \Downarrow a$;
2. 如果 $A \rightarrow^* A'$, 则对于某些 B' , 有 $B \rightarrow^* B'$, 并且 $A' \mathcal{R} B'$;
3. 对于所有的闭上下文, 有 $C[A] \mathcal{R} C[B]$ 。

观察等价基本上类似于 pi演算的 barbed同余关系, 只是其所定义的“barb”较 barbed同余关系弱。因此, 在 barbed同余关系上的一些性质可以对应的定义过来。

3.3.2 静态等价

静态等价 (\approx_s) 表示两个进程的任何框架 (frame) 都不能被任何其它进程所区分。其中, 框架是指由主动替换进程和 0 进程通过复合和受限构造出来的扩展进程。我们通常用 φ 和 ψ 来表示。一个框架 φ 中没有受限的主动替换进程 $\{M/x\}$ 中的变量 x 所组成的集合叫作框架的域 (domain), 记作 $dom(\varphi)$ 。

定义 3.4 (框架相等) 框架相等表示为 $(M = N)_\varphi$, 如果两个项 M 和 N 在框架 φ 下相等, 当且仅当存在名 \tilde{n} 和替换 σ , 有 $\varphi \equiv \nu \tilde{n} . \sigma$, $M\sigma = N\sigma$, 并且 $\{\tilde{n}\} \cap (f_n(M) \cup f_n(N)) = \emptyset$ 。

定义 3.5 (静态等价) 两个框架 φ 和 ψ 静态等价 ($\varphi \approx_s \psi$), 即 $dom(\varphi) = dom(\psi)$, 并且对于所有的项 M 和 N , $(M = N)_\varphi$ 当且仅当 $(M = N)_\psi$ 。

两个闭扩展进程静态等价 ($A \approx_s B$), 就是指它们的任何框架都是静态等价的。

3.4 基于 Applied pi演算的形式化验证

3.4.1 ikp 协议

ikp协议族[25, 26]由 IBM公司在1995年提出, 是一组基于公钥密码体制, 应用于 Internet上信用卡的交易。按照需要数字签名主体数量的不同分为 1kp、2kp、3kp三个协议, 分别满足不同的安全要求。参与协议的主体由购买方 (buyer), 销售方 (seller)和转账方 (Acquirer)组成。在这里我们主要来研究讨论 1kp协议的安全性。

为了描述 1kp协议, 我们首先定义加密原语: 我们用 SK_X 来表示 X 的秘密密钥, 使用 $PK(SK_X)$ 表示其对应的公开密钥。同时, 我们用 $H(\cdot)$ 来表示单向散列函数; $E_X(\cdot)$ 表示用公开密钥 PK_X 进行加密; $S_X(\cdot)$ 表示用秘密密钥 SK_X 进行数字签名。

表 3-1: 简化描述 ikp协议符号

| 符号 | 代表内容 |
|--------------|---------------------------------------|
| $Desc$ | 购买方的信息, 如地址, 电话, 信用卡账户名等 |
| $Salt_B$ | 购买方产生的随机数, 用来传递给销售方加密 $Desc$ |
| $Authprice$ | 购买方所购买商品的数量和价格 |
| $Date$ | 销售方产生的时间戳, 用来防止重复攻击 |
| $Nonces$ | 销售方产生的随机数, 用来防止重复攻击 |
| ID_S | 销售方的标识码, 用来区别不同的销售方 |
| TID_S | 销售方产生的交易标识码, 用来区别不同的交易 |
| Ban | 购买方的信用卡账号 |
| $Expiration$ | 购买方信用卡的期限 |
| R_B | 购买方用来产生自己标识码的随机数 |
| ID_B | 购买方的标识码, 产生方法: $ID_B = H_k(R_B, Ban)$ |
| $Despcode$ | 转账方的应答, 表示完成或未完成转账 |

我们定义了一些用来简化协议描述的符号, 和它们所代表的含义如表 3-1所列: 同时, 由于1kp协议的消息内容十分繁琐, 为了简明表示, 我们对其消息内容进行了如表 3-2的简写: 1kp协议是由六步组成: 购买方向销售方发起请求 (Initiate)、销售方响应销售方请求 (Invoice)、购买方向销售方提起支付 (Payment)、销售方向转账方申请转账 (Auth-Request)、转账方向销售方响应转账 (Auth-Response)、销售方向购买方确认转账结果 (Confirm), 协议流如下:

Initiate: $B \longrightarrow S : Salt_B, ID_B$
 Invoice: $B \longleftarrow S : Clear$
 Payment: $B \longrightarrow S : EncSlip$
 Auth-Request: $S \longrightarrow A : Clear, H(Salt_B, Desc), EncSlip$
 Auth-Response: $S \longleftarrow A : Respcode, Sig_A$
 Confirm: $B \longleftarrow S : Respcode, Sig_A$

(1) 购买方向销售方发起购买请求: 发送消息 $Salt_B, ID_B$ 。

购买方产生随机数 R_B , 使用公式 $ID_B = H_k(R_B, Ban)$ 计算出 ID_B , 连同产生的另一个随机数 $Salt_B$, 一起送往销售方, 发起购买请求。

表 3-2: ikp协议消息的简写

| 消息 | 代表内容 |
|----------------|--|
| <i>Common</i> | $Authprice, ID_S, TID_S, Date, Nonce_S, ID_B, H(Salt_B, Desc)$ |
| <i>Clear</i> | $ID_S, TID_S, Date, Nonce_S, H(Common)$ |
| <i>Slip</i> | $Authprice, H(Common), Ban, R_B, Expiration$ |
| <i>EncSlip</i> | $E_A(Slip)$ |
| <i>Sig_A</i> | $S_A(Respcode, H(Common))$ |

(2) 销售方响应购买方请求：发送消息 *Clear*。

销售方记录当前时间 $Date$ ，并且产生随机数 $Nonce_S$ ，这两个元素一起来标识信息的唯一性。随之产生 TID_S ，计算 $H_k(Salt_B, Desc)$ 。加上销售方的标识码 ID_S 和用户购买商品的数量和价格信息 $Authprice$ 建立起 *Common* 消息，并且用单向散列函数将之压缩。随后，将 ID_S 、 TID_S 、 $Date$ 、 $Nonce_S$ 和 $H(Common)$ 一起发给购买方。

(3) 购买方对销售方提起支付：发送消息 *EncSlip*。

购买方得到销售方的反馈后，通过得到的 ID_S 、 TID_S 、 $Date$ 、 $Nonce_S$ 和自己原有的信息得出 $H(Common)$ ，与销售方传来的 $H(Common)$ 作比较，若相等则可以提起支付。购买方将 $Authprice$ 、 $H(Common)$ 、用户的信用卡账号 Ban 、用来产生用户标识码的随机数 R_B 、信用卡使用期限 $Expiration$ 这些信息用转账方的公开密钥加密后发给销售方。

(4) 销售方向转账方申请转账：发送消息 *Clear*, $H_k(Salt_B, Desc)$, *EncSlip*。

销售方将消息 *Clear* 和 $H_k(Salt_B, DESC)$ ，连同购买方传来的消息一起发送给转账方，申请转账。

(5) 转账方向销售方响应转账：发送消息 *Despcode*, *Sig_A*。

转账方首先通过接收到的 *Clear* 得到 ID_S 、 TID_S 、 $Date$ 、 $Nonce_S$ ；接着采用自己的秘密密钥解开 *EncSlip*，得到 $Authprice$ 、 Ban 、 $Expiration$ 、 R_B 。然后用这些信息重构 $H(Common)$ ，并且同得到的 $H(Common)$ 进行比较。若相等，则将交易的结果和自己的数字签名反馈给销售方。

(6) 销售方向购买方确认转账结果：发送消息 $Despcode, Sig_A$

销售方将收到的信息送给购买方，告知交易是否成功。

3.4.2 1kp协议形式化表示

在 1kp协议的形式化描述中，我们需要定义它的项（包括变量、名和函数原语）如下：

| | |
|---|---|
| $T, U, V, V_0, \dots ::=$ | terms |
| A, B, S, x_1, x_2, \dots | variable |
| $C_{BS}, C_{SB}, C_{SA}, C_{AS}, \dots$ | name (for channel) |
| $Nonce_A, K_A, Date, \dots$ | name (for nonces, keys and time) |
| $f(T_1, \dots, T_n)$ | function application ($f \in \Sigma$) |

其中，函数集 Σ 包括了三类型的函数：

首先，是有关密码和检测的函数，其中 $H(u_1, \dots)$ 表示单项散列函数； $Pk(u)$ 表示公开密钥； $\{T\}_V$ 表示用公钥加密或者用私钥数字签名； $decrypt(w, u)$ 表示用私钥解密； $checksig(w, u)$ 表示用公开密钥检测数字签名； $checktime(Time, Expiration)$ 表示检测时间戳是否过期。

其次，函数集包括构造消息的函数，包括 $Initiate(u_1, u_2)$, $Clear(u_1, u_2, u_3, u_4, u_5)$, $Common(u_1, u_2, \dots, u_7)$, $Slip(u_1, u_2, u_3, u_4, u_5)$, $ARequest(u_1, u_2, u_3)$, $AResponse(u_1, u_2)$ 和 $Confirm(u_1, u_2)$ 。

此外，函数集还包括投影函数，即选择构造消息函数中第 j 个消息，比如 $Initiate.j(Initiate(u_1, u_2))$ 表示选择 $Initiate()$ 函数中第 j 个消息。这类函数还包括 $Clear.j(Clear(u_1, u_2, u_3, u_4, u_5))$, $Common.j(Common(u_1, u_2, \dots, u_7))$, $Slip.j(Slip(u_1, u_2, u_3, u_4, u_5))$, $ARequest.j(ARequest(u_1, u_2, u_3))$, $AResponse.j(AResponse(u_1, u_2))$ 和 $Confirm.j(Confirm(u_1, u_2))$ 。

由上述函数，我们有如下项的代数相等关系：

$$\begin{aligned}
 Decrypt(\{x\}_{Pk(z)}, z) &= x \\
 checksig(\{x\}_z, Pk(z)) &= true \\
 checktime(time, Expiration) &= \begin{cases} true & \text{if } time \in Expiration \\ false & \text{otherwise} \end{cases}
 \end{aligned}$$

$$\text{Tuple}.j(\text{Tuple}(x_1, \dots, x_i)) = x_j \text{ for } j < i, \text{ Tuple} \in \{\text{Initiate}, \text{Clear}, \text{Common}, \text{Slip}, \\ \text{ARequest}, \text{AReponse}, \text{Confirm}\}$$

有了项的定义和其代数项等关系，我们就可以形式化的描述 1KP 协议了。形式化描述分两步：消息定义和主体定义。

消息定义： 我们通过主动替换来定义协议的消息。

$$\sigma_1 \stackrel{\text{def}}{=} \{x_1 = \text{Initiate}(\text{Salt}_B, \text{ID}_B)\}$$

$$\sigma_2 \stackrel{\text{def}}{=} \{x_2 = \text{Clear}(\text{ID}_S, \text{TID}_S, \text{Date}, \text{Nonces}, H(\text{Authprice}, \text{ID}_S, \text{TID}_S, \text{Date}, \text{Nonces}, \\ \text{Initiate}.2(x_1), H(\text{Initiate}.1(x_1), \text{Desc})))\}$$

$$\sigma_3 \stackrel{\text{def}}{=} \{x_3 = \{\text{Slip}\}_{Pk(SK_A)}\}$$

$$\sigma_4 \stackrel{\text{def}}{=} \{x_4 = \text{ARequest}(x_2, H(\text{Initiate}.1(x_1), \text{Desc}), x_3)\}$$

$$\sigma_5 \stackrel{\text{def}}{=} \{x_5 = \text{AResponse}(\text{Respcode}, \{\text{Respcode}, H(\text{Common})\}_{SK_A})\}$$

$$\sigma_6 \stackrel{\text{def}}{=} \{x_6 = \text{Confirm}(\text{AResponse}.1(x_5), \text{AResponse}.2(x_5))\}$$

主体定义： 下面的代码是来表示协议主体，它们分别形式化的定义了购买方 (P_B)，销售方 (P_S) 和转账方 (P_A)。

$$P \stackrel{\text{def}}{=} (\nu \text{Authprice})(\nu \text{Desc})(P_B \mid P_S \mid P_A)$$

$$P_B \stackrel{\text{def}}{=} (\nu \text{Salt}_B)(\nu R_B)(\overline{C_{BS}}\langle x_1\sigma_1 \rangle \mid (C_{SB}(x_2).if \text{Clear}.5(x_2) = H(\text{Authprice}, \text{Invoice}.1 \\ (x_2), \text{Clear}.2(x_2), \text{Clear}.3(x_2), H(R_B, \text{Ban}_B), H(\text{Salt}_B, \text{Desc})) \text{ then } (\overline{C_{BS}}\langle x_3\sigma_3 \rangle \\ \mid (C_{SB}(x_6).if \text{checksig}(\text{confirm}.2(x_6), Pk(SK_A)) \text{ then } F_B(x_6))))))$$

$$P_S \stackrel{\text{def}}{=} (\nu \text{ID}_S)(\nu \text{TID}_S)(\nu \text{Date})(\nu \text{Nonces})(C_{BS}(x_1).(\overline{C_{SB}}\langle x_2\sigma_2 \rangle \mid C_{BS}(x_3).(\overline{C_{SA}}\langle x_4\sigma_4 \rangle \\ \mid C_{AS}(x_5).(\overline{C_{SB}}\langle x_6\sigma_6 \rangle.F_S(x_5))))))$$

$$P_A \stackrel{\text{def}}{=} (\nu SK_A)!(C_{SA}(x_4).if \text{Slip}.2(\text{Decrypt}(\text{ARequest}.3(x_4), SK_A)) = \\ H(\text{Slip}.1(\text{Decrypt}(\text{ARequest}.3(x_4), SK_A)), \text{Clear}.1(\text{ARequest}.1(x_4)), \\ \text{Clear}.2(\text{ARequest}.1(x_4)), \text{Clear}.3(\text{ARequest}.1(x_4)), \text{Clear}.4(\text{ARequest}.1(x_4)), \\ H(\text{Slip}.3(\text{Decrypt}(\text{ARequest}.3(x_4), SK_A)), \text{Slip}.4(\text{Decrypt}(\text{ARequest}.3(x_4), \\ SK_A))), \text{ARequest}.1(x_4)) \wedge \text{checktime}(\text{datetime}, \text{Slip}(\text{Decrypt}(\text{ARequest}.3$$

$(x_4)))$ then $((\nu ARespcode)(\overline{C_{AS}}\langle x_5\sigma_5 \rangle.F_A(x_4)))$

在这些进程中，函数 $F_B(x)$ ， $F_S(x)$ ， $F_A(x)$ 各自表示购买方，销售方和转账方在协议结束后所作的的事情。

3.4.3 1kp协议认证性的验证

在这一节中，我们首先考虑 1kp协议的认证性。当一个协议开始运行的时候，如果转账方得到允许，开始将购买方账户上的钱转向销售方，那么这个允许一定是由购买方授权的。这就是我们所要考虑协议的认证性。因此，我们有如下定理：

定理 3.1 [转账方的认证性] 假设 $P \xrightarrow{\eta} P'$ ，如果在 η 中存在

$$\overline{C_{AS}}\langle AResponse(Respcode, \{Respcode, H(Common)\}_{PK_A}) \rangle$$

则在此项之前， η 中必定存在

$$\overline{C_{BS}}\langle \{Slip\}_{PK(SK_A)} \rangle \text{ and } \overline{C_{SA}}\langle ARequest(Clear, H(Salt_B, Desc), \{Slip\}_{PK(SK_A)}) \rangle$$

证明 我们可以用对应性断言 (correspondence assertion)[83]来解释这个定理。即一旦转账方开始转账，则必定是由购买方授权这次转账。

如果在 η 中存在 $\overline{C_{AS}}\langle AResponse(Respcode, \{Respcode, H(Common)\}_{PK_A}) \rangle$ ，则说明转账方成功的验证了购买方和销售方的身份。它通过 Ban_B 来验证购买方的身份，通过匹配 $H(Common)$ 来验证销售方的身份，因此我们必定有 $\overline{C_{BS}}\langle \{Slip\}_{PK(SK_A)} \rangle$ 和 $\overline{C_{SA}}\langle ARequest(Clear, H(Salt_B, Desc), \{Slip\}_{PK(SK_A)}) \rangle$ 。

同样，对于销售方和购买方的认证性，我们可以类似的构造并且证明定理，此处不再赘述。

3.4.4 1kp协议匿名性的讨论

本节讨论匿名性。在网络交易中，为了使购买方安全购物转账，需要对销售方和网络上其他偷听者保持匿名性。然而在ikp协议族并不完全提供购买方的匿名性，只是对与其帐户提供匿名服务。在下面的定理中，我们可以证明购买方对于销售方和网络偷听者可以保持它账户的匿名性：

定理 3.2 (购买方的匿名性) 对于购买方和网络偷听者, 我们有 $P_B(Ban_B) \approx_s P_B(Ban'_B)$, 则说明购买方在 Ban_B 具有匿名性。

证明 我们首先定义下面的框架:

定义 $\varphi_1(P_B(Ban_B))$ 为

$$\{x_1 = (Salt_B, H(R_B, Ban_B)) \mid x_3 = \{Authprice, H(Common), Ban_B, R_B, Expiration\}_{Pk(SK_A)}\}$$

定义 $\varphi_2(P_B(Ban'_B))$ 为

$$\{x_1 = (Salt_B, H(R_B, Ban_B)) \mid x_3 = \{Authprice, H(Common), Ban'_B, R_B, Expiration\}_{Pk(SK_A)}\}$$

很明显对于销售方和网络偷听者, 我们有 $\varphi_1(P_B(Ban_B)) \approx_s \varphi_2(P_B(Ban'_B))$, 因此有 $P_B(Ban_B) \approx_s P_B(Ban'_B)$, 表示购买方的账户不会泄漏给销售方和网络偷听者。

但是对于转账方, 由于 $Slip.3(Decrypt(x_3, K_A)) = Ban_B$ 在 $\varphi_1(P_B(Ban_B))$ 成立, 但是在 $\varphi_2(P_B(Ban'_B))$ 中不成立, 所以我们可以推出 $P_B(Ban_B) \not\approx_s P_B(Ban'_B)$, 表示购买方对于转账方不保持匿名性。

3.5 结论和相关工作

本章提出了匿名性的定义和证明的一种方法, 并且将这种证明方法应用于 Applied pi演算中。也即, 令包含身份敏感信息 id 的消息为 $M(id)$, 若对于任意的 id 和 id' , 我们有:

$$M(id) \approx_s M(id')$$

则认为该协议具有 m 上的匿名性。

本章利用上述方法证明了 ikp 协议据有部分匿名性, 即购买方对销售方具有匿名性, 而对转帐房不保持其匿名性。同时, 也采用了一种新的方法来应用对应性断言 (定理 3.1) 而不是传统的构造协议规范的方法证明了 ikp 协议的认证性。

第四章 工作的总结与展望

4.1 论文工作总结

本文在进程演算的框架里，做了一些对安全协议形式化的研究。主要的贡献有以下几点。

首先，对安全协议形式化领域做了一个比较完整的概述。这是我们对所在安全小组前段工作的一个总结，并也为以后的工作打下了一个较好的基础。最初小组进入这个研究领域的时候，并没有一本完整的专著或者论文来阐述每种方法目前所解决和尚未解决的问题、优势及其缺点。我们用两个月的时间查询了各种方法，并且进行了进行、学习和研究，得出一个基本完整的安全协议形式化分析的综述。随着这种研究的深入，相信我们的理解也会越来越透彻，而本文对于这部分的阐述，仅是目前研究的一个阶段性总结。

其次，提出了基于时间戳协议的一种解决方案，Abadi在[9]提出，基于时间戳的协议也可以通过进程演算来分析，但他并没有给出解决方案。这个问题是我们最先想解决的问题，在开题的时候提出了包括高阶进程、符号互模拟、类型理论等许多设想和解决方案，但其实并没有用到那么高深，虽然最后仅仅用了一个类 match 算子解决了最后的问题，但这种研究已经使我深入的了解了这个领域的研究内容。同时，通过扩展了的 Spi 演算验证了 Kerberos 协议的认证性，并且指出了其认证的脆弱性：认证性都集中在了时间戳的保证下，一旦时间戳被攻破，该协议将无法保证认证性。

第三，提出了匿名性的定义证明的一种方法，并且将这种证明方法应用于 Applied pi 演算中，证明了 ikp 协议的部分匿名性。同时，在此论文中，我们采用了一种新的方法来应用对应性断言而不是传统的构造协议规范的方法（构造协议规范是一项较为艰巨的任务，如何使得构造出来的规范具有我们所希望的性质特点，并且其要求不至于过高，是这个领域需要解决的问题之一）证明了 ikp 协议的认证性。而此前尚未有其他基于进程演算对匿名性进行定义和证明的研究论文发表。

最后，本文还对基于进程演算的安全协议分析方法进行了一些展望性的叙述，这是我在这一年研究当中的一些心得和体会，有些是自己想去做但却没有时间完成。同时，还与小组其他成员一起开展一些新的研究工作，比如模型的建立和性质定义方法，目前已经有了较为满意的结果，但这些结果还在继续研究当中，因此在本文没有叙述和陈列。

4.2 未来工作展望

自 1996年，英国学者 Gavin Lowe启用 CSP模型检测工具 FDR发现了 Needham-Schroeder公钥协议的一个以前从未发现的漏洞后，基于进程演算的安全协议形式化研究一直方兴未艾，各种模型和研究方法不断地被提出。不像模态逻辑将模型的建立与性质的定义及其证明方法混在一处（一个逻辑模型的建立，新的模态算子加入的目的就是为了一个性质的定义，而它们的语义就是此性质的证明），由于进程演算良好的刻划和证明能力，可以将安全协议研究所要解决的问题自然的分成三个部分：模型的建立、安全性质的定义和该性质的形式化分析，这三部分虽然有所联系，但泾渭分明，可以分别地进行研究，而这三方面都有较为广阔的研究前景。

4.2.1 模型的建立

形式化的分析安全协议，首先要使用一个适当模型来描述它。这个模型需要正确、直观、简洁，并且抽象程度适当。因为对于不同的安全性质，所需要描述协议的侧重点也不同：消息认证性和保密性侧重于协议间消息的传递，需要刻划环境的知识和能力；而非否认性、公平性等却是侧重于协议主体的知识和能力，考察一个主体是否可以通过自己得到的知识来验证其性质。我们需要抽象的协议应该有如下特点：各个主体将消息传递给环境，然后从环境中接受任意组织的消息，并且判断消息的合法性，从而决定协议是否继续进行。

π 演算和类 π 虽然可以很直观地表示协议的主体和消息的传递，但作为以共有通道传递消息的模型，对于消息的传递还是较“环境”稍显强了一些，需要仔细构造环境（或者说攻击者）模型来模拟环境的攻击能力，因此具有描述能力上的不足。比如对于一个广播的协议来讲，直观上的描述应该是一个协议主体发送消息，

多个协议主体接收:

$$c\langle M \rangle.P \mid c(x).Q \mid c(x).R$$

但这样的描述在类 Pi 演算中并不正确, 而事实上, 描述一个广播协议是非常困难的。同样, 刻划攻击者的窃听也存在着同样的困难和不直观。

此外, 对于多协议的并发运行, 已有的模型也存在着不足, 例如对于多个主体共用一个服务器进行认证, 通常是采用复制 (Replication) 来刻划的:

$$!S \mid A_1 \mid A_2 \mid \dots \mid A_n$$

而复制操作有 $!S \equiv S \mid !S$, 这样无限的进程是很难进行自动化验证。

因此, 如何构造一个适合描述协议, 同时又可以在其上定义各种安全性质的模型, 是此方向需要考虑的问题之一。

4.2.2 性质的形式化定义

有了合适的模型, 如何正确并且严格的定义各种安全性质是另一个需要考虑和研究的方向, 目前的研究主要集中在认证性和保密性两个性质上。然而随着电子商务的发展, 非否认性、公平性、原子性、匿名性等都需要形式化研究的问题。

互模拟等价关系是进程演算一个强有力的定义安全性质的方法, 在安全协议形式化研究中, 我们主要通过互模拟等价关系来刻划两个进程与环境之间的交互能力: 如果协议的规范 (Specification) 和其实现 (Implementation) 与环境的交互能力相同, 则可以证明协议具有其规范所具有的性质。目前这样的互模拟关系很多, 如测试等价, framed 等价, fenced 等价, 静态等价等等。我们可以通过利用已有的互模拟等价关系 (如本文利用静态等价定义并证明了协议的匿名性) 来定义性质, 对于目前的互模拟等价关系无法定义的安全性质, 我们也要研究定义出新的互模拟关系来定义它。其中, 研究新定义等价关系的演算语义也是一个需要研究的内容。

当然, 这方面的研究不仅仅局限于基于进程演算, 还可以在更抽象的层面上来形式化的定义安全协议的性质, 比如定义认证性的对应性断言 (correspondence assertion)[83], 就被运用于各种具体模型的认证性定义中。同样, 私有性 (privacy) 和公平性 (fairness) 都是被列为被研究的对象, 对于这些性质的深刻研究本身就是一个相当深刻的研究课题。

4.2.3 形式化分析

有了合适描述的模型，定义了正确并且可计算的安全性质，如何证明一个协议具有某个安全性质，或者通过寻找反例来说明该协议不具有某个性质是形式化分析安全协议最主要的任务。对于理论的研究，是要将之应用于实践，指导实践。因此，我们希望做出可以自动化验证的工具来指导新协议的设计和实现。所以，在证明方法中，我们除了保证其正确外，还需要考虑其可计算、可操作等性质。

目前基于进程演算的研究有三种形式化分析方法：模型检测技术、互模拟验证技术和程序分析技术，这三种方法都远未完善，并都有着作进一步研究的价值。

目前基于进程演算的模型检测技术比较单一，其主要方法是构造协议主体和攻击者可能行为的踪迹 (trace)，然后找出一条攻击者可以达到目的的路径。其实，作为成功应用于实践的理论之一的模型检测技术，在理论研究和实践算法上都有了比较深入的发展，所以，我们可以通过将进程演算模型转向模态逻辑模型的方法来验证协议的性质，充分运用模态逻辑上比较成熟的模型和检测方法。

互模拟验证技术和互模拟定义安全协议的方法是相辅相成的。它与另外两种方法的不同之处是用来证伪，而不能证明性质（也即，找到攻击方法说明协议不具有该性质，而没找到怎不能证明协议具有该性质）。但是互模拟技术一个缺点就是难以应用于自动化工具，大多数的互模拟是不可计算的，如何设计自动化验证工具是这个方向需要解决的问题。其实，实际问题并没有理论要求那样严格，因此可以将无穷的状态分类，使之成为有穷的状态空间，然后在其上进行搜索，并设计相关算法是一种解决办法，fusion演算的MWB工具就是这样一种设计准则。但这种转化是否会破坏安全性质的验证还不得知。

程序分析技术是一项非常有前景的技术，通过类型检测来判断协议是否具有安全性质是最贴近程序语言的一种方法。但目前这个方法的成功较少，只是检测了协议的认证性和保密性。其实，通过这项技术的发展，我们可以设计出一种安全性语言，来验证包括安全协议在内的各种安全模型的性质。

参考文献

- [1] M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 5(46):749-786, 1999
- [2] M. Abadi and B. Blanchet. Secrecy types for Asymmetric Communication. *Foundations of software Science and computation structures(FoSSaCS'01)*, LNCS 2030, Springer-Verlag, 2001
- [3] M. Abadi and B. Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. In 29th annual ACM SIGPLAN - SIGACT symposium on principles of programming languages (POPL 2002), 33-44, 2002
- [4] M. Abadi and B. Blanchet. Computer-assisted verification of a protocol for certified email. *Static Analysis, 10th International Symposium(SAS'03)*, LNCS 2694, pages 316–335. Springer-Verlag, Jun.2003.
- [5] M. Abadi, B. Blanchet and C. Fournet. Just Fast Keying in the Pi Calculus. In *13th European Symposium on Programming*, LNCS 2986, Springer-Verlag, 340–354, 2004.
- [6] M. Abadi and C. Fournet. Mobile Values, New Names and Secure Communication. In *28th ACM Symposium on Principles of Programming Languages(POPL'01)*, pages 104–115, Jan. 2001.
- [7] M. Abadi and C. Fournet. Hiding Names: Private Authentication in the Applied Pi Calculus. In *Proceedings of the International Symposium on Software Security (ISSS'02)*, LNCS 2609, Springer-Verlag, 317–338, 2003.
- [8] M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. In *Proceedings LICS'98*, pages 105–116, 1998.

- [9] M.Abadi and A.D.Gordon. A calculus for cryptographic protocols:The spi calculus. To appear in the Proceedings of the Fourth ACM Conference on Computer and Communications Security, 1997
- [10] M.Abadi and A.D.Gordon. A calculus for cryptographic protocols:The spi calculus. Technical Report 414, University of Cambridge Computer Laboratory, 1997
- [11] M.Abadi and A.D.Gordon. Reasoning about cryptographic protocols in the spi calculus. In CONCUR'97: Concurrency theory, volume 1243 of Lecture Notes in Computer Science:59-73, 1997
- [12] M.Abadi and A.D.Gordon. A Bisimulation Method for Cryptographic Protocols.Nordic Journal of Computing, 5(4):267-303, 1998
- [13] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. Proceedings of the CONCUR, 380-394, Springer-Verlag, 2000
- [14] R. Amadio and S. Prasad. The game of the name in cryptographic tables. Proceedings of the ASIAN'99, 15-26, Springer-Verlag, 1999
- [15] Anders, Michael, Hansand Kasper. Towards Automatic Bisimilarity Checking in the Spi Calculus. citeseer.nj.nec.com/232308.html, 1999
- [16] M. Abadi, M. R. Tuttle. A semantics for a logic of authentication. In Proceeding of the Tenth ACM Symposium on Principles of Distributed Computing. ACM Press, 201-216, 1991
- [17] B. Y. Y. Aziz. A static analysis framework for security properties in mobile and cryptographic systems. PhD thesis, 2003
- [18] Bruno Blanchet and Benjamin Aziz. A Calculus for Secure Mobility. In Vijay Saraswat,editor,Eighth Asian Computing Science Conference (ASIAN'03),Lecture Notes on Computer Science, Dec.2003
- [19] M.Burrows,M.Abadi and R.M.Needham. A Logic of Authentication. ACM Transactions on Computer Systems, Vol.8, No.1, 1990

- [20] C. Bodei, M. Buchholtz, P. Degano, F. Nielson and H. R. Nielson. Automatic validation of protocol narration. Proceedings of the 16th computer security foundations workshop, 126-140, IEEE computer society press, 2003
- [21] C. Bodei, P. Degano, F. Nielson, H. R. Nielson. Control flow analysis for the Π -calculus. Proceedings of CONCUR'98, LNCS 1466, 84-98, Springer-Verlag, 1998
- [22] C. Bodei, P. Degano, F. Nielson, H. R. Nielson. Flow logic for Dolev-Yao secrecy in Cryptographic Processes. Future Generation computer systems, 18(6):747-756, 2002
- [23] M. Boreale, R. D. Nicola and R. Pugliese. Proof techniques for cryptographic processes. Proceedings of LICS'99, 157-166, IEEE, Computer Society Press, 1999
- [24] M. Boreale, R. D. Nicola and R. Pugliese. Proof techniques for cryptographic processes. SIAM Journal on Computing, 31(3):947-986, 2002
- [25] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner. iKP - A Family of Secure Electronic Payment Protocols. Proceedings First USENIX Workshop on Electronic Commerce, USENIX, 1995
- [26] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, E. V. Herrevehgen and M. Waidner. Design, implementation, and deployment of the iKP secure electronic payment system. IEEE Journal of Selected Areas in Communications 18, 611-627, 2000
- [27] Bolignano D. An approach to the formal verification of cryptographic protocols. Proceedings of the Third ACM Conference on Computer and Communications Security, ACM Press, 106-118, 1996
- [28] M. Boreale, R. D. Nicola. Testing equivalence for mobile processes. Information and computation, 120(2):279-303, 1995
- [29] J. Borgstrom and U. Nestmann. On bisimulations for the Spi calculus. EPFL IC, Technical Report IC34, 2003

-
- [30] J. Borgstrom. On bisimulation for the spi-calculus. Term Project, EPFL, Switzerland, 2001
- [31] C. Bodei. Security issues in process calculi. PhD Thesis, Department of information, university Pisa, Italy, 2000
- [32] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell. A meta-notation for protocol analysis. Proceedings of the 1999 IEEE Symposium on Security and Privacy, Los Alamitos, IEEE Computer Society Press, 84-89, 1999
- [33] I. Cervesato. Typed multiset rewriting specifications of security protocols. First Irish Conference on the mathematical foundations of computer science and information technology(MFCSIT'00), 1-43, Elsevier ENTCS 40, 2000
- [34] I. Cervesato. A specification language for crypto-protocol based on multiset rewriting, dependent types and subsorting. 7th MFPS, 24-27, 2001
- [35] I. Cervesato. Typed MSR:Syntax and examples. LNCS 2052, 159-177, 2001
- [36] L. Cardelli and A. D. Gordon. Mobile Ambients. In LNCS 1378, Springer-Verlag, 1998
- [37] L. Cardelli, G. Ghelli and A. D. Gordon. Secrecy and group creation. Proceedings of the 11th international conference on concurrency theory, LNCS 1877, 365-379, 2000
- [38] V. Cortier. Observational equivalence and trace equivalence in an extension of spi-calculus. Application to cryptographic protocols analysis. extended version, Technical Report LSV-02-3, Lab. Specification and Verification. ENS de Cachan, 2002
- [39] T. Coffey, P. Saidha. Logic for verifying public-key cryptographic protocols. IEEE Proceedings Computers and Digital Techniques, 144(1):28-32, 1997
- [40] A. S. Elkjer, M. Hohle, H. Huttel, K. Overgard. Towards automatic bisimilarity checking in the spi calculus. Combinatorics, Computation Logic, volume 21(3):175-189, Springer-Verlag, 1999

- [41] C. Fournet and M. Abadi. Hiding Names: Private Authentication in the Applied Pi Calculus. citeseer.ist.psu.edu/573109.html, 2002
- [42] U. Frendrup, H. Huttel and J. N. Jensen. Two notions of environment sensitive bisimilarity for spi calculus processes. 2001
- [43] R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. Proceedings of world congress on formal methods, LNCS 1708, 794-813, Springer,1999
- [44] YX. Fu. The χ -calculus. Proceedings of the international conferences on advances in parallel and distributed computing, 74-81, IEEE Computer society press, 1997
- [45] A. D. Gordon and A. Jeffrey. Authenticity by Typing for Security Protocols. 14th IEEE computer security foundations workshop(CSFW-14), 145-159, 2001
- [46] A. D. Gordon and A. Jeffrey. Types and Effects for Asymmetric Cryptographic Protocols. 15th IEEE computer security foundations workshop-CSFW'01, 77-91, 2002
- [47] yonggen Gu, Guoqiang Li and yuxi Fu. Analyzing iKP Security in Applied Pi Calculus. CIS'04, LNCS 3314, 2004
- [48] L. Gong, R. Needham, R. Yahalom. Reasoning about belief in cryptographic protocols. In Proceedings of the 1990 IEEE Computer Security Symposium on Research in Security and Privacy. IEEE Computer Society Press, Los Alamos, California, 234-248, 1990
- [49] C. A. Hoare. Communicating sequential processes. Communications of the ACM, 21(8):666-677, 1978
- [50] C. A. Hoare. Communicating sequential processes. Prentice Hall, 1985
- [51] H. Huttel. Deciding framed bisimilarity. pre-proceedings of Infinity'02, 1-20, 2002

- [52] L. J. Jagadeesan and R. Jagadeesan. Causality and true concurrency: A data-flow analysis of the pi-calculus. Proceedings of the 4th International Conference in Algebraic Methodology and Software Technology, LNCS 936, 277-291, Springer Verlag, 1995
- [53] R. Kailar. Accountability in electronic commerce protocols. IEEE Trans. On Software Engineering, 22(5):313-328, 1996
- [54] R. Kemmerer, C. Meadows, J. Millen. Three systems for cryptographic protocols analysis. Journal of Cryptology, 7(2), 251-260, 1994
- [55] G. Lowe. Breaking and fixing the Needham-schroeder public-key using FDR. Tools and Algorithms for the construction and analysis of systems, LNCS 1055, 147-166, Springer-Verlag, 1996
- [56] G. Lowe. Casper:A compiler for the analysis of security protocols. Proceedings of 10th IEEE computer security foundations workshop, 1997
- [57] G. Lowe. Towards a completeness result for model checking of security protocols. The Journal of Computer Security, 7(2/3):89-146, 1999
- [58] C. Meadows. The NRL protocol analyzer: An overview. Journal of Logic Programming, 26(2), 1996
- [59] C. Meadows. Analysis of the Internet key exchange protocol using the NRL protocol analyzer. In Proceedings of the IEEE Symposium on Security and Privacy, Los Alamitos, IEEE Computer Society Press, 84-89, 1999
- [60] R. Milner. A calculus of communicating systems. LNCS, vol92, Springer-Verlag, 1980
- [61] R.Milner. Communication and Concurrency. Prentice-Hall International, 1989
- [62] J. Millen. The interrogator model. Proceedings of the 1995 IEEE Symposium on Security and Privacy, Los Alamitos, IEEE Computer Society Press, 251-260, 1995

- [63] A. Menezes, P. Van. Oorschot and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996
- [64] T. J. Marlowe and B. G. Ryder. Properties of data flow frameworks - a unified model. Acta Informatica, 28(2):121-163, 1990
- [65] F. Nielson, H. R. Nielson and C. Hankin. Principles of program analysis. Springer, 1999
- [66] R.Milner,J.Parrow, and D.Walker. A calculus of mobile processes, Parts I and II. Information and computation, 1992
- [67] B.C.Neuman and T. Ts'o. Kerberos: An Authentication service for computer network. IEEE Communications Magazine,1994
- [68] L. C. Paulson. Mechanized proofs for a recursive authentication protocol. Proceedings of the 10th IEEE Computer Security Foundations Workshop, Los Alamitos, IEEE Computer Society Press, 84-94, 1997
- [69] L. C. Paulson. The inductive approach to verifying cryptographic protocols. Journal of Computer Society (6), 85-128, 1998
- [70] Aviel D. Rubin, Peter Honeyman. Nonmonotonic cryptographic protocols. IEEE, 1994
- [71] Davide Sangiorgi. Expressing Mobility in Process Algebras:First-Order and Higher-Order Paradigms.Ph.D. thesis, Univeristy of Edinburgh, 1992
- [72] B.Schneier. Applied Cryptography Second Edition:Protocols, algorithms and source code in c.Wiley, 1996
- [73] S. A. Schneider. Security properties and CSP. IEEE Computer Society Symposium on Research in Security and Privacy, 1996
- [74] S. A. Schmeider. Verifying authentication protocols with CSP. CSFW10, IEEE Press, 1997

- [75] O. Shivers. Control flow analysis in scheme. Proceedings of the ACM SIGPLAN'88 Conference on Programming Language Design and Implementation, 23(7):164-174, ACM Press, 1988
- [76] P. F. Syverson, P. C. Oorschot. On unified some cryptographic protocol logics. In Proceedings of the 1994 IEEE Computer Society Press, 1994
- [77] P. F. Syverson, P. C. Oorschot. An unified cryptographic protocol logics. Manuscript, 1996
- [78] D.Sangiorgi and D.Walker. The π calculus: a Theory of Mobile Process. Cambridge University Press, 2001
- [79] F. J. Thayer, J. C. Herzog, J. D. Guttman. Strand spaces:Why is a security protocol correct. Proceedings of the 1998 IEEE Symposium on Security and Privacy, Los Alamitos, IEEE Computer Society Press, 160-171, 1998
- [80] F. J. Thayer, J. C. Herzog, J. D. Guttman. Strand spaces:Honest ideals on strand spaces. Proceedings of the 1998 IEEE Computer Security Foundations Workshop, Los Alamitos, IEEE Computer Society Press, 66-77, 1998
- [81] F. J. Thayer, J. C. Herzog, J. D. Guttman. Strand spaces:Proving security protocol correct. Journal of Computer Security, 66-77, 1999
- [82] Jan Vitek and G.Castagna. Towards a calculus of secure mobile computations. In Workshop on Internet Programming Languages (WIPL'98), 1998
- [83] T. Y. C. Woo and S. S. Lam. A semantic model for authentication protocols. In *14th IEEE Symposium on Research in Security and Privacy. IEEE Computer Society Press*, 178-194, 1993
- [84] Min Zhang, Guoqiang Li, yuxi Fu, Zhizhou Zhang and Lin He. Representation of Signal Transduction with Aberrance Using Ipi Calculus. CIS'04, LNCS 3314, 2004
- [85] 卿斯汉. 安全协议20年研究进展.软件学报, Vol.14, 2003

参考文献

- [86] 卿斯汉、冯登国. 信息系统的安全. 科学出版社, 2003
- [87] 范红、冯登国. 安全协议理论与方法. 科学出版社, 2003

致 谢

在这篇论文即将结束之际，我想对在这两年半硕士研究生的学习和生活中给予我帮助的老师、同学和朋友们表示衷心的感谢。没有他们的关心和支持，也就没有我的研究成果和这篇论文的撰写。

首先，我要特别感谢的是我的指导老师傅育熙教授。2002年，我有幸成为傅育熙老师领导下的BASICS实验室的一份子。在此期间，傅老师一直在学习和科研上给予我最大可能的帮助。在学术上，傅老师始终坚持高标准、严要求的指导方针，鼓励我们独立思考，大胆探索，勇于创新。支持我们向国内外最高级别的学术刊物和会议上投稿，并不时给我们提供大量的信息。同时，他还对我们严格要求，对每篇论文的内容和质量都亲自严格把关，保证其独创性和先进性。在工作条件上，傅老师为我们创造了一个非常舒服的研究环境，实验室软件和硬件设施齐备，可以让我安心的进行研究工作。

其次，我要感谢的是顾永跟师兄和张敏师姐。正是顾师兄牵头的安全协议形式化小组给了我研究的课题，在此课题的研究中，作为他的助手和合作者，顾师兄给了我许多帮助和建设性的意见。同时，顾师兄认真的治学态度和严谨的研究方法是我今后一直要学习的榜样；张师姐深厚的数学功底，积极的探索精神和谦虚的求知态度也是我学习的榜样。我和张师姐是同时进入实验室的，在这段时间里，我们一起讨论各方面的问题，积极交流看法和观点，让我从中受益匪浅。并且有幸和师姐合作，完成了一个很有趣的课题。

我要感谢实验室的董笑菊师姐，钟发荣师兄，李洋师兄和吴一新、肖会会、何超栋、袁国涛、董驻鹏、朱涵等师兄弟，是他们在学习和科研工作过程中给予我许多无私的帮助，提出了许多中肯的意见和建议。在此特别感谢董笑菊师姐对在此篇论文上给予我的莫大帮助和修改意见。

此外，我要感谢詹晓峰、林晨曦、陶云峰和吕昊，正是在他们的教导和帮助下，与他们的交流和探讨中，我从一个对计算机科学懵懂无知，充满迷惑的人，到了现在算是初窥门径，刚刚感觉到了其中的魅力与乐趣。

我还要感谢我的“粉丝”们，感谢 shoran、yj、eryelyn、Jammie、cynthia、candice、feilan、lindaith、panda、minicookie、pireia、inin和 dn、感谢她们陪我度过最枯燥，最无奈的研究生生活。在我最失意的时候给我鼓励，在我最绝望的时候给我勇气，在我最迷茫的时候给我智慧，在我最颓废的时候给我快乐。

最后，我要感谢我的父母，感谢他们一直以来对我默默的支持和鼓励着我，他们是我最大的精神来源！

发表的论文

- (1) 李国强, 顾永跟, 傅育熙. 基于Spi演算的Kerberos协议形式化研究. 计算机科学, 2004年, 第31卷(第11期), 7-10
- (2) Yonggen Gu, Guoqiang Li and yuxi Fu. Analyzing iKP Security in Applied Pi Calculus. CIS'04, LNCS 3314, 2004
- (3) Min Zhang, Guoqiang Li, yuxi Fu, Zhizhou Zhang and Lin He. Representation of Signal Transduction with Aberrance Using Ipi Calculus. CIS'04, LNCS 3314, 2004