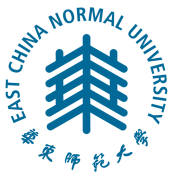
2022届研究生硕士学位论文

分 类 号：_____ 学校代码：10269

密 级：_____ 学 号：51194501005



華東師範大學

East China Normal University

硕士学位论文

MASTER'S DISSERTATION

论文题目：基于模型检测方法的安全协议形式化分析

院 系：软件工程学院

专 业：软件工程

研 究 方 向：可信软件

指 导 老 师：赵涌鑫副教授

学位申请人：蒋洪剑

2022 年 5 月 14 日

中国知网 https://www.cnki.net

Dissertation for master degree in 2022 University Code: 10269

Student ID: 51194501005

East China Normal University

Title: Formal Analysis of Security Protocols Based on Model Checking

Department: School of Software Engineering

Major: Software Engineering

Research Direction: Trustworthy Software

Supervisor: Asst Prof. Yongxin Zhao

Candidate: Hongjian Jiang

May , 2022

中国知网 https://www.cnki.net

中国知网 https://www.cnki.net

蒋洪剑 硕士学位论文答辩委员会成员名单

姓 名	职 称	单 位	备 注
朱惠彪	教授	华东师范大学	主席
郭建	副教授	华东师范大学	
毛宏燕	副教授	华东师范大学	

中国知网 https://www.cnki.net

摘　要

安全协议作为网络中数据交换的规则标准，在保障物联网、5G 网络的通信安全方面发挥了至关重要的作用。随着网络安全问题日益突出，作为网络安全重要组成部分的安全协议是以密码学为基础的安全信息交换协议，如果安全协议本身存在安全隐患，一旦遭受恶意攻击，网络用户财产隐私乃至于正常生活都将受到极大影响。如何设计出安全的网络安全协议成为学术界与工业界共同关注的一个问题，形式化方法从数学算法出发，帮助研究人员更全面完备地保证安全协议的安全性以及找到潜在攻击手段。形式化分析方法已经被研究者成功应用，并发现了大量实际应用的安全协议中的安全隐患，但是使用形式化分析技术对安全协议进行证明需要专业人员花费巨大的时间精力。

本文从模型检测角度出发，开展对安全协议的自动分析验证研究。目前业界广泛采用 Alice&Bob 语言描述安全协议中主体之间的信息交互过程，然而由于缺乏统一的规范和明确的语义定义，Alice&Bob 语言无法被直接应用于形式化验证工具。另外，由于模型检测难以解决状态空间爆炸的问题，我们提出一种约束的安全协议自动分析验证机制，很好解决了状态空间爆炸问题。最后能够全面地、完备地对安全协议进行分析设计。主要研究成果总结如下：

1. 提出了一套有效的 Alice&Bob 语言规范, 能够在保持简洁易懂的同时提供更强的描述能力，从形式语言视角刻画网络安全协议的执行过程。支持转发信道和代数推理的定制，以很好地描述多主体参与型 (如 Otway-Rees) 和密钥交换型 (如 Diffie-Hellman) 协议.

2. 提出了一种安全协议自动验证方法，采用明确的 Alice&Bob 语言规范进行建模，并基于此给出了一套模型转换和分析验证的方法论及其实现工具，能够将 Alice&Bob 规范模型转换到 Murphi 模型检测工具上进行验证。我们提出了第一个串空间操作语义来形式化地描述参与者的行为。在操作语义下，可以很好地保证中间格式与 Murphi 形式模型之间的语义一致性。此外，还提出了一个模式匹配过程来对接收消息动作的行为进行形式化表示，这有助于在操作语义上下文中进行形式化推理。

3. 提出了一种基于扩展串空间操作语义形式化的验证方案，在 Murphi 模型中对安全协议进行建模验证。其中包括对递归定义的消息进行编码、对正规主体代理和 Delvo-Yao 模型中的恶意入侵者的行为进行建模。我们采用特定的规约策略将模型约束为连结串和加密串，并只构造正规主体代理所接受

中国知网 https://www.cnki.net

的复合消息，从而实现了串空间操作语义在 Murphi 中的编码形式，对安全协议进行模型验证分析。

关键词: 安全协议，模型检测，Alice&Bob，Murphi，串空间理论，操作语义

中国知网 https://www.cnki.net

ABSTRACT

Security protocols, as one of the standards for data exchange in networks, play a crucial role in ensuring the communication security of the Internet of Things and 5G networks. With the increasingly issues of network security, the security protocol, as an important part of network security , is the security information exchange protocol based on the crytptography. If a security protocols itself has security risks, once a malicious attack occurs, the property, privacy and even normal life of network users will be greatly affected. How to design a secure network security protocol has become a common concern in the academic and industrial area. Formal methods, starting from mathematical algorithms, is able to help researchers to guarantee the security property of security protocols more comprehensively and completely by finding potential attacks. Formal methods has been successfully applied by researchers, and a large number of potential security risks in practical security protocols have been found. However, it takes a lot of time and efforts for experts to verify security protocols by using formal analysis technology.

From the perspective of model checking and theorem proving, this paper conducts research on automatic analysis and verification of security protocols. At present, Alice&Bob language is widely used in the industry to describe the message exchange process between principals in security protocols. However, due to the lack of unified specifications and clear semantic definitions, Alice&Bob language cannot be directly applied to Formal Verification tools. In addition, since model checking is hard to solve the problem of state space explosion, we propose a constrained automatic analysis and validation mechanism for security protocols, which solves the problem of state space explosion efficiently. Finally, it can analyze and design the security protocol comprehensively and completely.The main research achievements are summarized as follows:

1. An effective Alice&Bob specification is proposed, which can provide customized language description capabilities while keeping simplicity and consistency, and formally realizes the process of pictures. Network protocols that support steering and algebraic reasoning, both in the form of multi-agent participation (eg Otway-Rees) and protocol key exchange (eg Diffie-Hellman) that can be easily described.
2. An automatic verification method of security protocol is proposed, which adopts explicit Alice&Bob language specification for modeling security protocols. Based

中国知网 https://www.cnki.net

on this method, a set of methodologies and implementation tools for model transformation, analysis and verification are proposed, which can convert Alice&Bob specification model into the Murphi model checking tool for verification. We propose the first operation semantics of Strand Space theorem to formally describe the behavior of principals. Within the operational semantics, the semantic consistency between the intermediate format and the Murphi formal model can be well guaranteed. Furthermore, a pattern matching procedure is proposed to formalize the behavior of receiving message actions, which facilitates formal reasoning in the operational semantic context.

3. A formally verified scheme based on the operation semantics of extended Strand Space theorem is proposed, and the security protocol is modeled and verified in the Murphi model checker, which includes encoding recursively defined messages as well as modeling the behavior of regular principals and malicious intruders in the Delvo-Yao model. We use a specific reduction strategy to constrain the model to concatenated messages and encrypted messages, and only construct the compound messages accepted by the regular principals, so as to realize the encoding form of Strand Space operation semantics in Murphi, and conduct model verification analysis of security protocols.

Keywords: Security Protocols, Model Checking, Alice&Bob，Murphi，and Strand Space

中国知网 https://www.cnki.net

目　录

摘　要 i
Abstract iii
目录 v
插图目录 vii
表格目录 viii
第一章　绪论 1
1.1　研究背景和意义 1
1.2　国内外研究现状 2
1.2.1　形式化方法 2
1.2.2　安全协议分析工具 4
1.2.3　安全协议分析技术应用 6
1.3　技术路线与主要研究内容 7
1.3.1　研究目标 7
1.3.2　研究方法与技术路线 8
1.4　本文组织结构 8
第二章　相关技术基础 10
2.1　安全协议的形式化描述 10
2.2　模型检测方法 11
2.3　本章小结 15
第三章　安全协议的模型检测分析方法 16
3.1　安全协议自动验证方法 16
3.2　Alice&Bob 语言的形式化规范 17
3.3　扩展串空间及操作语义 21
3.3.1　串空间理论 21
3.3.2　串空间操作语义 22
3.4　AnB2Murphi 模型转换器 28
3.4.1　Alice&Bob 规约与 Murphi 模型检测器 29
3.4.2　扩展串空间与 Murphi 模型检测器 31
3.4.3　模型转换过程 37
3.4.4　模型形式化验证 39
3.5　安全协议自动分析验证机制 39
3.5.1　枚举量定义 39

中国知网 https://www.cnki.net

3.5.2 结构化数据定义 41
3.5.3 协议运行规则模型 42
3.6 本章小结 44
第四章 案例研究与分析 **46**
4.1 5G EAP-TLS 协议分析 46
4.2 Murphi 协议模型的生成 48
4.3 Murphi 协议的验证分析 58
4.4 安全协议验证结果分析 60
4.5 本章小节 61
第五章 总结与展望 **62**
5.1 工作总结 62
5.2 工作展望 62
参考文献 **68**
致谢 **69**
研究成果 **70**

中国知网 https://www.cnki.net

插 图 目 录

1.1 本文技术路线图 8

2.1 模型检测过程示意图 12
2.2 Murphi 语言的基本组织架构 15

3.1 安全协议自动验证方法架构 17
3.2 Otway-Rees 协议的正规束形式 19
3.3 Otway-Rees 协议在串空间规约中的具体实现 20
3.4 AnB2Murphi 总体架构图 29
3.5 Alice&Bob 语言规范与 Murphi 模型映射 30
3.6 扩展串空间与 Murphi 之间的映射 31

4.1 5G 网络框架示意图 46
4.2 5G EAP-TLS 认证协议在串空间规约中的具体实现 47
4.3 Murphi 模型消息结构定义 49
4.4 Murphi 模型主体结构定义 50
4.5 Murphi 模型信道结构定义 50
4.6 Murphi 模型正规主体发送消息结构定义 51
4.7 Murphi 模型正规主体接受消息结构定义 52
4.8 Murphi 模型恶意攻击者接受消息结构定义 53
4.9 Murphi 模型恶意攻击者连结消息结构定义 54
4.10 Murphi 模型恶意攻击者发送消息结构定义 55
4.11 Murphi 模型弱认证性结构定义 56
4.12 Murphi 模型代理初始化结构定义 56
4.13 Murphi 模型模式集初始化结构定义 57
4.14 Murphi 模型代理消息集结构定义 58
4.15 Murphi 模型构造模式集结构定义 58
4.16 弱一致性反例串分析 59
4.17 Prekey 的私密性反例串分析 59

中国知网 https://www.cnki.net

表 格 目 录

3.1 迁移规则表 26

4.1 Murphi 模型总体结构 48

4.2 实验结果分析 60

中国知网 https://www.cnki.net

第一章　绪论

1.1　研究背景和意义

现今社会，人们的生活愈来愈离不开网络。网络在大幅便利人们的衣食住行的同时，也面临着遭受不怀好意的黑客攻击现象，一旦网络受到威胁，政府国家乃至于群众的生命安全都会受到极大的利益受损[1,2]。而网络安全协议作为互联网的重要组成部分，它规范了网络中的所有实体该按照何种秩序去顺序执行。网络中的各种设备如服务器、移动用户终端，交换机等都是基于网络安全协议进行通信交换的。因此正确无误的网络协议可以保证在受到恶意攻击的情况下网络的有序进行。资深的网络安全专家对于不同的应用场景和产品需求进行精细的统筹设计以及数学推导，从而确保网络的正确通信。

与此同时，安全作为网络协议中的最重要部分，在设计之初却被人们轻易忽视。这种安全设计漏洞往往被攻击者所发现利用，从而轻易获得网络中的所有信息。网络安全攻击不仅会威胁到个人的财产安全和泄露个人的隐私安全，更严重者会威胁到国家的网络空间安全。为了有效避免这种情况的发生，研究人员提出了各种密码学网络安全协议来保证其安全性，从而使得网络数据通信的安全得到极大的提升。为了实现用户相互认证、密钥分配和私密性等目标，以消息交换为形式的安全协议是基于密码学而提出的，其在网络中实体之间的加密通讯、消息和实体的认证、密钥分配等被广泛应用，很好地保障了系统之间的通信和交易安全。

在复杂的网络环境中，除了合法的通信实体之外，还存在大量的恶意攻击者，攻击者利用安全协议设计中的缺陷实施各种各样的攻击。这种网络攻击一旦成功了，该网络系统遭到破坏，网络用户的所有信息将对攻击者透明。如何保证安全协议在设计阶段和运行时始终满足安全属性是协议设计者的目标。这对研究人员甚至专家都是极大的挑战。如作为安全协议基石的 Needham-Schoreder 协议[3] 从设计以来都被认为是安全的，通常被后来的网络认证协议作为设计的初稿，1996 年，Lowe 等人[4] 通过形式化分析技术发现其协议设计存在了中间人攻击缺陷。从此以后，协议研究人员将形式化分析技术引入至安全协议设计领域。

中国知网　https://www.cnki.net

形式化方法[5]是一门基于严格数学基础的学科，通过形式语言、语义以及推理证明等形式逻辑系统对计算机硬件和软件系统进行形式化的描述开发和验证。形式化分析技术与软件学科的许多领域不断交叉和融合，其形式验证可以帮助研究员对系统的规约进行验证，两种最常见的验证形式即定理证明和模型检测。定理证明将系统满足规约为主要的逻辑命题，通过不同的推理规则，以演绎推演证明其命题的成立，其中就包括著名的 Floyd-Hoare 逻辑[6]等。模型检测则与演绎式的推演证明略有不同，其基本思想是指检验系统是否满足设计规约，其比证明逻辑公式在任意条件下满足更简单，因此很好地用来证明了面向并发系统在有穷状态的模型上验证公式的可满足性。

在形式化分析技术中，代表性技术如 BAN 逻辑推理、软硬件模型检测以及定理证明技术在安全协议研究中发挥了重大作用

形式化分析技术在安全协议领域中产生了不同的方法，包括逻辑推理、模型检测以及定理证明等。研究人员还对各种方法进行改进，提出适用更复杂的协议设计和支持更复杂的密码算法。同时，还开发了一系列自动化分析的工具。但是这些工具各有优缺点，且各自有自己的使用规则和语言，需要研究人员花费大量时间精力学习，往往难以入手。另外，这些自动化工具仍旧可能存在设计上的缺陷。研究与设计形式化安全分析工具是非常有价值的一个方向。

1.2 国内外研究现状

1.2.1 形式化方法

一直以来形式化方法被用来降低软件开发复杂性和提高软件可靠性，它成功地应用于各种硬件系统和安全攸关的系统中，如高速列车控制系统、芯片设计、航天航空控制系统等。形式化方法的思想在软硬件开发过程中也得到了不同程度的体现，如基于模型的软件开发、建模软硬件系统、精化抽象模型以及对模型进行测试等。

近年来，形式化方法得到了很大的发展，国外的硬件制造商如 IBM、AMD 和 Intel 等巨头投入了大量的人力和物力从事形式化方法的研究，以确保所设计的芯片具有高并发性、高可信度、高处理性能等。NASA 航天局耗费巨大经费组织了形式化方法研究团队来保证航天器控制软件的可靠性，基于这些技术研发

中国知网 https://www.cnki.net

的软件的可靠性也愈来愈得到了保障。

此外在许多学科中随处可见形式化这三个词汇，但是在计算机领域中形式化方法是基于严格的数学逻辑，对计算机的软硬件系统进行形式规约、开发和验证的三者统一的技术。形式规约包括使用形式语言和形式语义对软硬件系统进行规约。其中形式语言是用符号化字母表和递归的语法规则来定义和生成句子的语言，形式语义在语言定义和系统软件开发中都得以应用。形式化方法的开发与基于模型的软件开发类似，通过需求分析定义初始系统的形式规约，随后通过逐步求精和转换得到逐步精化的形式规约，最后综合生成可实现的系统程序代码，精化生成的代码都是系统的功能抽象模型。

形式化方法是一门大学科，包括形式规约语言来定义软硬件的形式语义和模型理论、形式规约在开发中逐步精化模型和生成可执行代码、形式验证定理证明或模型检测等技术对模型进行验证、形式化工具统一的一个整体。以形式规约语言为基础，开发软件制品为形式规约，形式验证保证开发的正确性，形式语义和模型理论以严格的数学保证规约和验证之间的一致性，形式化工具则是系统设计中使用形式化方法的高度体现。

形式验证视作为验证上一阶段的形式规约，可以划分为推理“系统模型规约是否满足性质规约”和推理“系统的一个模型规约是否与另一个模型规约有精化或等价的关系”两部分。演绎定理证明和算法模型检测两部分组成了形式验证。演绎定理证明以“系统满足规约”为逻辑命题，通过一系列的推理规则，对该逻辑命题进行展开证明。基于定理证明的验证可以通过程序逻辑和程序的操作语义来表达程序的执行。其中最经典的逻辑验证系统当属 Floyd-Hoare 逻辑，通过一组与程序语言语句对应的公理和规则，将程序验证转化到一组数学命题的证明上来，从而实现顺序程序的验证。基于定理证明的验证还可以细划分为自动定理器验证和基于交互的半自动验证。自动定理证明其包括常见的 Dafny[7]、Why3[8] 和 VeriFast[9] 等。另外，微软所开发的 Z3 证明器[10] 是目前应用最广泛的证明器。而本文所采用的 Isabelle[11] 以交互式半自动化的方式完成命题的验证，由人手工和计算机进行交互，将正确性的检查转换为类型的检查，其验证的结论也比自动化验证更为可信。

模型检测以显式或隐式方法在有穷状态空间上通过遍历策略算法进行遍历。

中国知网 https://www.cnki.net

系统状态爆炸则是模型检测的最大问题，学者一般通过三种不同的方法来完成在有限的状态空间和时间内搜索：1、结构化方法，通过定义系统的语法表达来缓解状态空间爆炸问题；2、符号化方法，将模型状态和迁移过程编码为逻辑公式进行压缩；3、抽象方法将复杂系统的状态空间等价规约到较小的模型系统中，在规约后的系统上实现验证。模型检测的方法输入形式规约的语义模型和性质规约，通过遍历系统模型中的有穷状态来检测两者是否满足，即输入规约模型和规约属性，经过不同的模型检测器给出检验结果是否满足；如果不满足，模型检验算法则会给出反例路径，帮助用户对系统进行修改设计，最后使得系统满足其规约的属性。

1.2.2 安全协议分析工具

Interrogator[12]：一个 Prolog 程序用于搜索自动密码学密钥分布的网络协议中的安全漏洞。给定一个协议的形式化规约，它可以通过寻找活跃的攻击者来击败协议对象。图形化用户界面的 Interrogator 通过 LM-Prolog 在 Lisp 计算机上实现，

FDR[13]：由牛津大学 Bill Roscoe 教授开发的精化软件检查工具，设计检查通过顺序通信过程表达的形式化模型。FDR 往往被视作为模型检测器，但它实际上是一个精化检查器。可以将两个 CSP 进程转化到对应的标记迁移系统中，决定某个进程是另一个进程的精化。FDR2 利用不同的状态空间比较算法处理标记迁移状态来减少精化过程中的状态空间。Gavin Lowe 利用 FDR 检测器分析 Needham-Schroeder 公钥加密协议，将协议和入侵者在 CSP 中进行编码，随后使用 FDR 发现安全协议漏洞。

Murphi[14]：协议验证分析工具，已经成功地应用于不少工业级硬件协议中，特别是多处理器缓存一致性协议和多处理器内存模型领域。Murphi 验证协议需要先建模至 Murphi 模型语言，在模型上设计相应不变式属性的规约，Murphi 通过显式状态深度优先或者广度优先算法遍历对模型进行自动检查，将所有遍历到状态的进行枚举，保存在一张全局的哈希表上。同时 Murphi 验证器支持通过特定语言构造对模型进行对称规约。

AVISPA[15]：AVISPA 是一个按钮式工具，用于自动验证互联网安全敏感协议和应用程序。它提供了一种模块化的表达形式语言，用于指定协议及其安全属

中国知网 https://www.cnki.net

性，并集成了实现各种最先进的自动分析技术的不同后端。没有其他工具能够在享受相同性能和可伸缩性的同时，展现出相同级别的范围和健壮性。

OFMC[16]：支持对安全协议的证伪以及通过探索迁移系统的有界验证，集成符号化表示技术，支持密码学操作的代数性质规约，标记以及无标记协议模型。

CL-AtSe[17]：应用约束求解、简化、启发式和冗余消除技术。基于模方法来处理密码学的代数性质，支持标记错误检测和消息连结的关联性问题。

SATMC[18]：建立了命题公式可以将有界展开的迁移关系通过 IF 语句进行编码，初始状态和状态集可以表示安全属性是如何违背的，随后，命题公式被引入最新的 SAT 求解器中，任何被发现的违背模型都将翻译成一个模型的攻击。

TA4SP[19]：其后端可以通过使用正则表达式语言和重写规则近似表达入侵者的知识。对于安全属性，TA4SP 可以显示是否协议存在缺陷的，或者在任意数目的会话下是否是安全的。

Maude-NPA[20]：密码学协议的分析工具，引入其他工具所不包含的代数性质的密码系统，其中包括加解密、异或、指数和同态加密。与 NRL 协议分析器类似，其基于 unification 策略在有限状态上执行向后搜索来决定是否可达。与原始的 NPA 不同，Mauda-NPA 理论基于重写逻辑和收缩，为验证安全协议提供了一系列公理规则。

ProVerif[21]：由 Bruno Blanchet 于 2001 年开发的自动密码学协议验证器。基于 Dolev-Yao 形式化模型[22]，协议通过 Horn 从句进行表示，可以重写规则或者等式解决许多不同的密码基元。另外它可以解决协议的无上界会话以及无上界消息空间，这意味着验证器可以给出错误的攻击。ProVerif 基于 Prolog[23] 以及消解算法，通过所谓的推理来判断某些事实是否能够发生。ProVerif 可以验证私密性、认证性和更常见对应属性、强私密性和进程之间的等价性。

Tamarin[24]：由 ETH 大学 David Basin 教授所提出的安全协议验证工具。它接受一个安全协议模型作为输入，指定以不同角色运行协议的代理所采取的操作、对手的规范和协议所需属性的规范。Tamarin 基于重写规则简化构造一个证明，即使在协议角色的任意多个实例与对手的动作并行交叉时，协议也满足其指定的属性。在 2015 年后，扩展支持了等价属性，支持子项收敛方程，Diffie-Hellman 求幂[25] 等密码原语。

中国知网 https://www.cnki.net

1.2.3 安全协议分析技术应用

1978 年，Needham 和 Scheroder 提出通过形式化分析技术对安全协议进行分析，并提出了 Needham-Sceroder 公钥和共享密钥认证的安全协议。然而在 1996 年，Gavin Lowe[4] 等人通过模型检测器 FDR 发现该协议存在中间人攻击的设计漏洞，随后 Mitchell 等人[26] 受该思想启发，采用 Murphi 模型验证器同样发现了其缺陷，提出了 Murphi 中的安全协议设计思路，对 TMN 协议[27] 进行了验证。该领域的作出重大贡献者 Dolev-Yao 成功地将密码协议的使用规范与密码体系分离。尽管 Dolev-Yao 模型是有限制的，许多大型的协议不能很好地描述，但是作为第一个形式化安全协议模型，其极大限度地约束了针对诚实协议的参与者以及期望的安全目标。Dolev-Yao 模型定义了一个攻击者，遵循密码学的所有基本性质，该模型在安全协议形式化分析领域中受到广泛的关注，不少协议验证工具都是基于此模型开发的。

1989 年，Burrow 等人提出了 BAN 逻辑，与模型检测思想不同，该逻辑从主体拥有的知识和信仰出发，定义了一系列推理规则用以从已有的信仰推出新的信仰。BAN 逻辑成功地分析了 Needham-Schroeder 和 Kerberos 等协议。贾国梁等人提出了改进的 BAN 逻辑结合串空间的形式化分析方法。邓媛勐等人使用 Prolog 语言设计开发了基于 BAN 类逻辑的协议自动分析验证工具。随后 Paulson 提出了一种基于协议消息和事件的归纳证明方法，其称之为 Paulson 归纳法，结合了状态探测方法和信仰逻辑两种方法的观点，将 A 发送 X 给 B 作为对事件的描述方式，并生成每个消息的保证理念。1998 年，Guttman 提出了 Strand Space[28–30] 的理论，是一种最为广泛而有效的安全协议形式化分析方法。它采用简洁的协议模型，协议的正确性证明也容易开展：通过插图式启发手段进行协议描述与推理，使得协议设计者可以清晰地分析协议正确性，协议成立时所需满足的条件等。姚萌萌等人[31] 对串空间理论进行了形式化的分析，并基于该理论对区块链的公平多方不可否认协议进行了验证。张等人[32] 提出了以串空间模型为理论基础的自动化验证算法 IVAP；董学文等人[33] 则设计了一种基于串空间的路由协议攻击分析模型来对安全协议进行分析。1983 年，John Millen 实现了协议验证工具 Interrogator；2000 年 Clarke[34] 实现了 Brutus 工具使用规范语言对安全协议进行建模，支持对多个会话数量的协议描述，但是存在协议的状态空间爆炸问题。

Ahena 工具结合了定理证明和模型检测技术，在工具中实现了扩展的 strand

中国知网 https://www.cnki.net

空间模型，有效地解决了状态空间爆炸问题。学者通过 AVISPA[35–37] 工具分析验证了 ISO-PK 协议、IKEv2 协议以及 H.530 协议的漏洞。由 Blanchet 设计实现的协议验证工具 Proverif 目前仍有大量科研人员使用，JFK、Plutus 文件系统、TPM 命令[38]、安全消息传递、TLS1.3[39]、E-Health[40] 等都通过 ProVerif 进行分析。Proverif 基于 Horn 逻辑，通过计算 invariant 来验证网络协议的安全属性，它不仅支持证明可达性和观察等效性，还支持无限制的会话数和无限制的消息空间。此外，现有其他版本的 Proverif 对其进行了相应的扩展。基于定理证明方法的 Tamarin Prover 的有效性与 ProVerif 相媲美，其使用向后搜索和引理来解决模型验证中状态空间爆炸的问题，基于 Rewrite Rule 逻辑[41]、无限制会话数和观察等价性和异或操作，不仅方便了研究人员对协议进行建模抽象，也加速了其正确性的证明。Tamarin 验证器分析验证了 TESLA、YubiKey 等密钥安全协议[42]。

近来，熊焰[43] 等人提出一个新的通用框架 SmartVerif，提出了有效解决针对协议状态空间爆炸的自动分析验证方法，其突破了现有验证方法自动化能力弱的缺陷，采用启发式智能地搜索证明路径，通过深度强化学习算法以及约束树形式对协议进行分析，无需人工干预。杨锦熊[44] 则在 SmartVerifi 验证工具的基础上进一步优化，设计了一个通用性更强的循环路径判定算法，自动调整参数以应对不同的网络安全协议。苏城[45] 结合了强化学习与形式化自动验证技术，解决了状态空间爆炸问题，所提出的系统扩展了 pi 演算，可将协议自动转换至 ProVerif，并对 5G-AKA 协议进行了形式化分析研究。

1.3 技术路线与主要研究内容

1.3.1 研究目标

本文结合了模型检测和定理证明两种技术对网络安全协议进行形式化分析，其中利用模型检测可以验证给定安全协议中设计是否满足其期望的规范。如图1.1所示，从 Alice&Bob 高层语言规范出发对协议进行描述，基于 ML 的 Menhir/YACC 编译转换到中间语义的扩展串空间框架，随后根据提出的基于 Alice&Bob 语言规范的安全协议自动验证方法生成对应的 Murphi 协议模型，交由 Murphi 模型验证器对生成的 Murphi 模型进行状态空间遍历验证。在验证过程中如没有反例路径生成则表明协议满足所声称的安全规约，反之表明存在设计缺陷，需要根据反例路径对其进行修正。

中国知网 https://www.cnki.net

1.3.2 研究方法与技术路线

基于模型检测的安全协议形式化分析

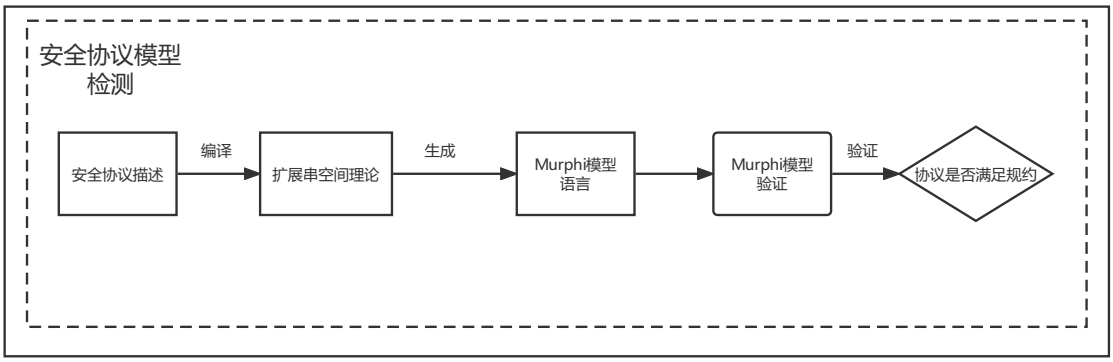


图 1.1 本文技术路线图

基于模型检测的安全协议形式化方法采用基于攻击的检测手段，将协议刻画到符号化的状态上，每一个状态转换则等价于一个状态路径生成过程。验证协议是否满足设计规约首先从协议的初始状态出发，随后对协议参与的正规主体和入侵者的所有可能行为生成的路径进行穷尽遍历，根据 Dolev-Yao 攻击模型找到协议可能存在的设计问题。该分析方法将安全协议的正规主体行为和入侵者行为作为转移规则进行路径生成刻画，所生成的模型状态包括所有正规主体的变量状态，还有全局入侵者的秘密知识库集合，通过模型检测器的并发执行，如若验证器返回不满足安全规约的反例路径，则说明原协议存在某种攻击序列。反例路径刻画出表明恶意攻击者对协议攻击手段的攻击序列。本文采用 Murphi 模型检测器作为自动验证工具，从 Alice&Bob 协议规范出发，自动将安全协议转换生成到 Murphi 模型中，交由 Murphi 用于分析模拟协议的运作过程，从而实现基于模型检测的安全协议形式化自动分析。

1.4 本文组织结构

本文共分为五个章节，文章结构和组织内容如下：

第一章是绪论，首先介绍了安全协议形式化分析的相关背景，阐述了安全协议形式化分析的重要性及其研究价值，从国内外安全协议的研究现状和现有的协议分析工具出发，分析本文技术路径和主要研究内容的可行性，总结了本文的主要贡献和结构框架。

中国知网 https://www.cnki.net

第二章是预备知识与概念，首先介绍了安全协议形式化描述形式，接着对本文工作所涉及的形式化方法模型检测进行了概述，其中包括模型检测器 Murphi 的工作原理和方法。

第三章是安全协议的模型检测，即基于 Alice&Bob 语言规范的安全协议自动研究方法。本章从 Alice&Bob 语言、串空间理论和 Murphi 模型检测器概念介绍出发，提出了扩展的 Alice&Bob 语言规范框架与基于串空间操作语义，在保证形式化语义一致性下将高层次的安全协议描述转换到低层次的 Murphi 模型检测模型。此外还给出了转换算法以及安全协议在 Murphi 中的自动分析验证机制，详细阐述了 AnB2Murphi 的工作机制和工作原理。

第四章是案例研究与分析。本章给出第四章安全协议实例研究，对复杂的 5G EAP-TLS 认证协议进行协议分析并在 Alice&Bob 规约框架中进行建模，随后通过本文方法转换生成到 Murphi 模型验证器中，通过对生成模型的模块分析介绍安全协议自动分析验证机制的研究，最后通过 Murphi 模型验证器验证生成模型，实现 5G EAP-TLS 认证协议的验证分析。

第五章是工作总结与工作展望。本章总结分析了本文的工作，并提出了未来研究方向和工作展望。

中国知网 https://www.cnki.net

第二章　相关技术基础

2.1 安全协议的形式化描述

在密码学协议中，最重要的就是代理主体以及代理主体之间通信交换的消息。在本文工作中，代理主体划分为三种类型: $Server$、$Friend$ 和 Spy，其中 $Server$ 扮演安全协议中的服务器，用以分发主体之间的共享(协商)密钥；$Friend$ 刻画协议中的诚实主体，即安全协议规范按部就班地接收和发送消息；Spy 则扮演恶意攻击者的角色，可以采用中间人攻击，重发攻击，窃听拦截等任何攻击方法对协议进行破坏。因此，主体的类型可以形式化地刻画为:

$$\mathsf{Agent} \;::=\; \mathsf{Server}(N) \;\mid\; \mathsf{Friend}(N) \;\mid\; \mathsf{Spy}$$

其中，N 的可取范围为自然数。在非对称密钥协议中，主体 A 拥有对所有主体公开的公钥 $Pk(A)$ 和只有它知道的私钥 $Sk(A)$，其中 $PK(A)$ 为 $Sk(A)$ 的对称密钥，即 $Pk(A)^{-1} = Sk(A)$。在对称密钥协议中，每个主体则拥有一个与其他主体所共享的长期对称密钥 $shrK(A)$，对称密钥的逆转还是其本身，即 $shrK(A)^{-1} = shrK(A)$。为了阐述说明的简洁性，我们用 k^{-1} 表示密钥 k 的逆转，并假设 (1) 对称密钥和非对称密钥是分离的 (2) 所有密钥函数都是单射，即 $shrK(A) = shrK(A')$。该假设可以得出 $A = A'$，(3) 集合 bad 表示恶意攻击者的集合，最少包括一个 Spy 主体。如果主体 A 不包含在 bad 中，则主体 A 可能是 Server 或者 Friend。

接下来定义在协议中交换的消息类型，我们通过 BNF 注解对消息集合进行描述:

$$\begin{array}{rcllll}
\mathsf{M} & ::= & \mathsf{Agent}(A) & \mid \mathsf{Nonce}(N) & \mid \mathsf{Key}(k) \\
& \mid & \mathsf{MPair}(m_1, m_2) & \mid \mathsf{Crypt}(m, k) & \mid \mathsf{Hash}(m) \\
& \mid & \mathsf{Sign}(m, k) & \mid \mathsf{Mod}(m_1, m_2) & \mid \mathsf{Exp}(m_1, m_2) \\
& \mid & \mathsf{Number}(N) & \mid \mathsf{Tmp}(X) & \mid \mathsf{NULL} \\
\\
\mathsf{K} & ::= & \mathsf{Pk}(A) & \mid \mathsf{Sk}(A) & \mid \mathsf{K}(A, B) \\
& \mid & \mathsf{M} & &
\end{array}$$

中国知网 https://www.cnki.net

其中，A 和 B 都是 Agent 集合中的元素；N 和 k 分别来自自然数集和密钥集 K。在密码学协议中，消息可以划分为原子消息和复合消息，其中原子消息包括主体消息 Agent(A)、用以避免重发攻击的随机数消息 Nonce(N)、密钥消息 Key(k)、自然数消息 Number(N)。而复合消息包括连结消息 Mpair(m_1, m_2)、加密消息 Crypt(m, k) 用以表示消息 h 被密钥 k 所加密。为了简洁表明，我们用 $\{|h|\}_k$ 表示 Crypt(k, h)，$\{|h_1, ..., h_{n-1}, h_n|\}$ 表示 Mpair($h1 \ldots Mpair(h_{n-1}, h_n)$)。另外，复合消息还包括哈希消息 Hash($m$) 和签名消息 Sign($m, k$)、模消息 Mod($m_1, m_2$) 和指数消息 Exp($m_1, m_2$) 用以支持 XOR 理论。最特殊的在于匿名消息 Tmp(X)，用以刻画正规主体所不能解开的消息。

上述我们形式化地定义了密码学代理主体和消息形式。这里通过引入 $Event$ 事件可以更好地描述密码学协议的运行过程，即协议初始化时我们如何创建主体实例；协议运行时诚实主体之间的消息交互过程如何刻画等等。

$$\mathsf{Event} \quad ::= \quad \mathsf{Create}(Role) \quad | \quad \mathsf{Send}(Term) \quad | \quad \mathsf{Recv}(Term)$$

Event 集包括三个动作：建立主体实例、发送项和接受项。值得注意的地方是，发送和接收事件都没有显式地指定发送者和接收者。这是由于发送或者接收消息中的子项可以识别发送者和预期的接受者。在 Dolev-Yao 模型中，恶意攻击者接受窃听所有信道上发送的消息，并且与预期的接受者相互解耦独立。

2.2 模型检测方法

在计算机科学中，模型检查或属性检查是一种检查系统的有限状态模型是否满足给定规范（也称为正确性）的方法。这通常与硬件或软件系统相关，其中规范包含活性要求（例如避免活锁）以及安全要求（例如避免表示系统崩溃的状态）。为了从算法上解决这样的问题，系统的模型和它的规范都用某种精确的数学语言来表述。为此，将问题表述为逻辑任务，即检查结构是否满足给定的逻辑公式。这个一般概念适用于多种逻辑和多种结构。一个简单的模型检查问题包括验证命题逻辑中的公式是否满足给定结构。

当两个描述不相等时，使用属性检查进行验证。在细化期间，规范补充了更高级别规范中不必要的细节。无需根据原始规范验证新引入的属性，因为这是不可能的。因此，严格的双向等价检查被放宽为单向属性检查。实现或设计被视为

中国知网 https://www.cnki.net

系统的模型，而规范是模型必须满足的属性。

目前业界已经开发了一类重要的模型检查方法，用于检查硬件和软件设计的模型，其中规范由时序逻辑公式给出。时序逻辑规范方面的开创性工作由 Amir Pnueli 完成，因提出该工作而获得 1996 年图灵奖。模型检查始于 EM Clarke、EA Emerson、JP Queille 和 J. Sifakis 的[46–48] 的开创性工作。Clarke、Emerson 和 Sifakis[49] 分享了 2007 年的图灵奖以表彰他们在模型检查领域的开创性工作。

模型检测通常用于硬件设计中，对于软件来说，由于不可判定性，这种方法不可能完全算法化来适用于所有系统，并且总是给出一个答案; 在一般情况下，模型检测可能无法证明或反驳一个给定的属性。在嵌入式系统硬件中，验证一个交付的规范是可能的，例如，通过 UML 活动图[50] 或控制解释的 Petri 网[51]。

因此，我们可以形式地将问题表述如下形式：通过一个时序逻辑公式 p 表达给定期望的属性，带有初始状态 s 的结构 M，从而决定判定 M 是否成立，即 $s \models p$。如果 M 是一个硬件设备，其状态是有限的，那么模型检测可以将其规约到图搜索问题上。

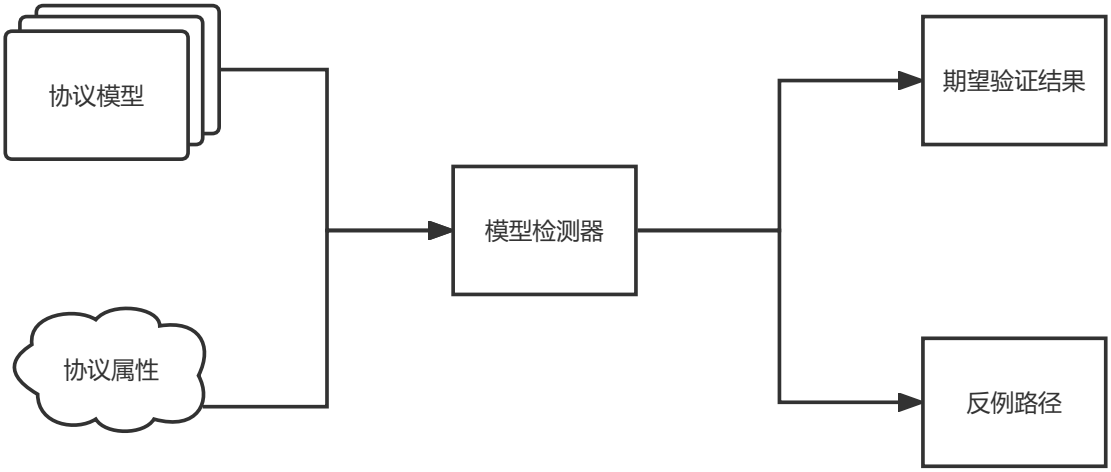


图 2.1 模型检测过程示意图

如图2.1所示为模型检测的整体过程，首先通过建模对软硬件进行形式化的描述，即定义它的状态迁移系统，并通过逻辑公式等对其期望的性质进行描述，将其输入给模型检测器，通过模型检测器对系统模型的状态空间进行既定算法的搜索，以判断每一个状态是否满足设定的逻辑公式（协议属性）。如果验证通过，则返回期望的验证结果，反之给出反例路径供给设计人员参考，设计人员可通过反例路径对系统发生错误的地方进行分析修改并重新进行模型验证，最后

中国知网 https://www.cnki.net

使系统满足设定规约。

模型检测经过三十多年的发展，是学术界以及工业界共同的关注热点之一，其中在计算机硬件、通信协议、安全协议等方面已经趋于成熟。随着计算机硬件的完善提升以及缓解状态空间爆炸的算法设计提出，模型检测技术日渐完善。其中符号模型检测即 Bryant 提出的 BDD 算法，将合取范式以及析取范式压缩为效率更高的布尔范式，显著地提高了模型检测器验证状态数庞大的系统效率，另外 On-the-fly 技术边验证边展开模型的状态，给现在模型检测工具提供了新的思路想法，偏序规约技术则减少验证系统上同一个时间在不同状态下执行的序列，大幅度降低了所需检测的系统迁移状态。抽象技术包括状态合并、数据抽象以及谓词抽象三个常见抽象方法，可以有效去除系统中冗余存在的系统模型。上述技术是被模型检测工具用来缓解状态爆炸问题的有效方法。

另外，由于模型检测验证速度快，即插即用以及自动化程度高，求研制工具也得到了极大的推广，如 Murphi、Spin、NuSMV、UPPAAL 以及 PAT、SLAM、BLAST 等一系列模型检测工具，通过与其他技术结合，将模型检测的优势带到其他领域，如与概率论相结合的概率模型检测，与定理证明结合的演绎推理模型检测等，在不同的领域都得到了不错的效果。

符号化模型检测

与一次枚举一个可到达的状态不同，有时可以通过在单个步骤中考虑大量的状态来更有效地遍历状态空间。当这种状态空间遍历基于一组状态和转换关系的表示形式，如逻辑公式、二进制决策图 (BDD) 或其他相关的数据结构时，模型检查方法是符号的。

历史上，第一个符号方法是 BDD 技术。1996 年人工智能规划问题的命题可满足性成功解决后，同样的方法被推广到线性时间逻辑 (LTL) 的模型检验中: 规划问题对应于安全特性的模型检验。这种方法被称为有界模型检查布尔可满足求解器在有界模型检验中的成功推广了符号模型检验中可满足求解器的广泛应用[52]。

一阶逻辑

模型检验也是计算复杂性理论研究的一部分。具体来说，一阶逻辑公式是固定的，没有自由变量。因此考虑以下决策问题：给定一个有限的解释如描述为关

中国知网 https://www.cnki.net

系数据库的解释，如何决定该解释为给定公式的模型。

这个问题属于电路类 AC0，当对输入结构施加一些限制时，它是可以处理的：例如要求它的竖宽以一个常数为界 (更普遍地意味着单元二阶逻辑的模型检查的可处理性)，为每个域元素的度设界，以及更一般的条件，如有界展开、局部有界展开和无位置密集结构[53]。这些结果已推广到列举一阶自由变量公式的所有解的任务。

Murphi 模型验证器

Murphi 是一个具有自己的输入语言的枚举 (显式状态) 模型检查器，它支持 guard $\longrightarrow$ action 符号。它已经成功地应用于几个工业协议，特别是在缓存一致性协议和多处理器内存模型领域，例如，German 和 Flash 协议。它可以抽象系统的行为，模拟系统的运行规则。Murphi 模型检查器有一个形式化的验证器，它基于显式的状态枚举，其性能与状态的数量更密切相关。在这种模式下遇到的状态保存在一个全局哈希表中。在哈希表中生成的状态不进行扩展。凭借着对称约简协议算法的成功应用，有效地降低了被验证协议的状态。

Murphi 模型的基本结构可以在图中看到，Murphi 输入语言的描述由一系列的常量声明组成，数据类型如子范围、记录、数组、全局变量、转换规则部分、初始化部分和属性部分。在这些部分中，最重要和最复杂的部分是描述从一个状态到另一个状态的转换规则部分。迁移规则主要由卫士条件和动作两部分组成。只有当保护中的谓词是可满足的，操作部分中的语句才能执行。给定一个 Murphi 模型，模型检查器 Murphi 将显式地枚举整个状态空间，直到没有新的可到达的状态可以被探索，或者属性不能保持在协议上。对应的有，所有可能可达状态的集合被认为是可达状态集 (缩写为 $RS(P)$)。Murphi 从初始状态开始，初始化部分已经规定了初始状态。然后随机选择满足保护的转移规则，执行相应的动作。协议所需的属性被指定为 Murphi 中的不变式，它是一种谓词公式，表明在每个可达的状态下必须保持不变。验证策略是 Murphi 验证其在每个新生成的状态中读取状态变量。如果不变式在该状态上满足条件为 false，验证者中断验证并报告作为反例路径的错误消息。在另一种情况下，如果一个状态除了它自己没有后继时，验证器才会停止并反馈错误。

中国知网 https://www.cnki.net

```
声明部分
    --常量声明 Constant declarations
    --类型声明 Type declarations
    --全局变量声明 Global variable declarations
    --过程函数声明 Procedure and function declarations

迁移规则部分
    Rule "ruleName"
      卫士条件部分 guard part -- conjunction of predicate
    ==>
      动作部分 action part -- a set of statements
    endrule

初始部分
    初始状态 Startstates
      --initial the value of variables
    end

属性部分
    不变式 Invariant "inv"
      -- define the security property
    end
```

图 2.2 Murphi 语言的基本组织架构

2.3 本章小结

本章首先形式化地给出了安全协议的表示形式，即对安全协议的参与者、通信交换的消息形式以及收发消息动作进行了刻画。接着分析了两种安全协议形式化验证分析方法：模型检测方法和定理证明方法。最后介绍了本文所用到的两种验证工具 Murphi 模型检测器。

中国知网 https://www.cnki.net

第三章　安全协议的模型检测分析方法

本章我们主要介绍安全协议的模型检测方法，其中包括安全协议的形式化描述、Alice&Bob 语言的形式化规范、扩展的串空间及其操作语义、Murphi 模型验证器、AnB2Murphi 模型转换器和安全协议自动分析验证机制。首先为了捕捉不同协议之间的共通性质，我们引入基础的符号化模型构造推理安全协议。在这种符号化模型中，代理主体之间通讯的消息可以由项代数中的项集刻画，协议本身由主体角色集合表示。每个主体角色关联到事件序列，用以描述实际扮演该角色的代理所执行的动作。Alice&Bob 语言是最常见的用于描述诚实主体之间消息交换的框架，它从高层次上刻画了安全协议的运行过程，但是它与实际运行的协议代码存在距离。串空间是一种高效的用于证明安全协议正确性的框架，帮助我们通过数学的方法来证明协议的安全性，可靠地保证了我们从 Alice&Bob 语言出发生成的 Murphi 模型检测代码的语义一致性，并通过 AnB2Murphi 框架对安全协议的模型进行形式化验证。最后我们给出安全协议在 Murphi 中的自动分析验证机制，其中包括枚举量定义、结构化数据定义和协议运行规则模型。

3.1　安全协议自动验证方法

如图3.1所示，针对安全协议自动验证的方法论主要分为的三个部分:

1. 协议拆解和建模：基于扩展的 Alice&Bob 语言规范对协议模型进行建模，将协议的描述拆解为类型描述、知识表示、主体交互模板、环境描述和验证目标几个部分。其分别刻画了网络协议中所涉及的主体模板、在协议运行前每个主体所具备的前置知识、主体模板之间的交互方式、对主体参量的具体例化方法以及验证的目标性质

2. 向 Murphi 程序的转换：根据步骤 (1) 得到的 Alice&Bob 语言描述的协议模型，根据每个部分的语义将其转换到 Murphi 程序中，以实现在 Murphi 形式化语言中的建模

3. 基于 Murphi 的验证：将协议运行中涉及的所有消息分模式保存下来，在协议运行时刷新全局消息表和消息的模式表，攻击者主体利用新消息扩展知

中国知网　https://www.cnki.net

识集，并尝试发送或接收消息以实现攻击。基于这种攻击模型，可以对协议上所关注的性质进行形式化验证

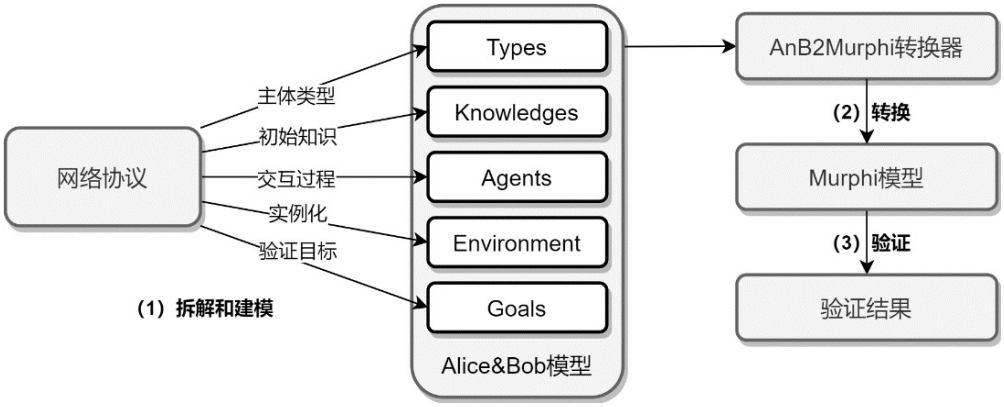


图 3.1 安全协议自动验证方法架构

我们的方法论将安全协议的自动验证拆解为上述三个步骤。对于使用者而言，只需要利用简洁抽象的 Alice&Bob 语言对协议进行建模，只需具备少量的形式化领域知识就能够构建正确且语义明确的模型。然后，通过构造的模型利用工具实现向 Murphi 模型的转换，借助 Murphi 形式化模型检测器来完成最终的性质验证工作。

3.2 Alice&Bob 语言的形式化规范

在安全协议的上下文中，Alice&Bob 注解通常用以描述诚实主体在成功协议运行中的消息交互过程。这种方法简洁明了，但是在描述代理采取动作时具有歧义性，尤其是他们必须检查是否执行了他们的角色以及检查失败时动作是否执行的情况。

首先看一个简单的例子，引出 Alice&Bob 注解存在的问题。考虑协议的一个步骤：

$$A \longrightarrow B : \{|M|\}_K^s$$

表明主体 A 的一个代理发送了由对称密钥 K 所加密的消息 M 给执行主体 B 的一个代理。直观地来看，这个步骤由两个代理 A 和 B 同时参与：A 发送了消息 $\{|M|\}_K^s$ 以及 B 接受了消息 $\{|M|\}_K^s$。此外，为了发送消息，A 必须构造项 $\{|M|\}_K^s$，也就是说 A 必须拥有 $\{|M|\}_K^s$ 项，从而它可以直接发送，或者它拥有组件 M 和 K，随后通过执行加密操作构建该加密项。而对于 B 来说，它的职责

中国知网 https://www.cnki.net

就是从信道上接收消息 $\{|M|\}_K^s$。当它同时拥有了 M 和对应的解密密钥，它可以对该消息进行检查。如果检查失败了，B 应该终止协议的执行。同样，B 也可以对消息 M 用密钥 K 进行加密进行比较。但是，当 B 不拥有项 M 和密钥 K 时，它是无法进行这类检查。

因此，为了推理协议这类检查发生的时刻，通过简单的 Alice&Bob 注解是不够来刻画主体的所有行为。我们在这里对 Alice&Bob 注解进行了扩展，通过引入初始知识块、代理块、目标块和环境块显式地表示这些藏在协议背后的操作，并为这种协议规约语言提供了完备的语义使其可以正确地将 Alice&Bob 协议规约转换到低层次的语言，来执行对安全协议的形式化分析。

接下来，我们通过 Otway-Rees 协议为例对 Alice&Bob 语言规约进行详细讲述。如图3.2所示，该协议使用与密钥服务器所共享的长期对称密钥来分发一个新的会话密钥，从而维护两个客户端之间的会话。Otway-Rees 协议不确定将相同的密钥发送给 A 和 B，只确定如果 A 或 B 中的一个到达了其任务的最后，然后另一个客户端发送对应原始请求的 $\{|Nb, M, A, B|\}_{K(B,S)}$ 或者 $\{|Nb, M, A, B|\}_{K(A,S)}$ 消息，同样会话密钥 K 未被泄漏。

1. $\mathsf{A} \rightarrow \mathsf{B} : \mathsf{M}, \mathsf{A}, \mathsf{B}, \{\mathsf{Na}, \mathsf{M}, \mathsf{A}, \mathsf{B}\}_{\mathsf{K(A,S)}}$
2. $\mathsf{B} \rightarrow \mathsf{S} : \mathsf{M}, \mathsf{A}, \mathsf{B}, \{\mathsf{Na}, \mathsf{M}, \mathsf{A}, \mathsf{B}\}_{\mathsf{K(A,S)}}, \{\mathsf{Na}, \mathsf{M}, \mathsf{A}, \mathsf{B}\}_{\mathsf{K(B,S)}}$
3. $\mathsf{S} \rightarrow \mathsf{B} : \{\mathsf{Na}, \mathsf{Nab}\}_{\mathsf{K(A,S)}}, \{\mathsf{Nb}, \mathsf{Nab}\}_{\mathsf{K(B,S)}}$
4. $\mathsf{B} \rightarrow \mathsf{A} : \mathsf{M}, \{\mathsf{Na}, \mathsf{Nab}\}_{\mathsf{K(A,S)}}$

如图3.3所示，我们给出了 Otway-Rees 协议的 Alice&Bob 语言规约。其中，该规约框架由五部分组成，协议名，初始知识，代理主体，安全目标和运行环境。

- Types 类型块：表达协议中出现的所有类型的标识符，包括通信主体类型 Agent（标识网络协议中的参与者实体）、自然数类型 Number（标识协议运行中涉及的自然数）、函数类型 Function（标识协议中涉及的操作变换或映射关系）和内置的随机数类型 Nonce（可以用于表达密钥）。在该实例中，Types 块定义了标识符为 $Init$、$Resp$ 和 $Sever$ 的三个主体模板，表达协议中的三个正常的通信主体为 $Init$ 发起者，$Resp$ 响应者和 $Server$ 密钥服务器

中国知网 https://www.cnki.net

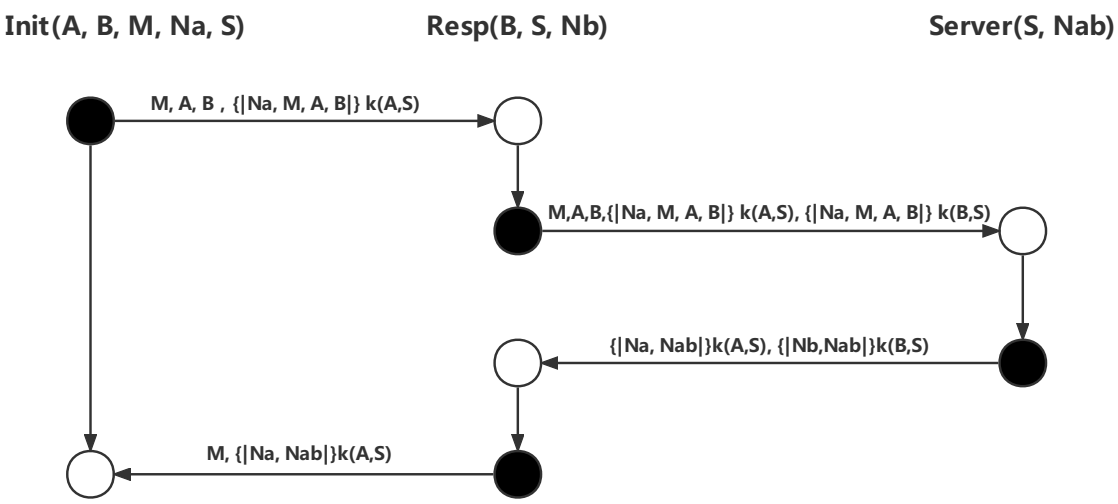


图 3.2 Otway-Rees 协议的正规束形式

- Knowledge 知识块：表达协议中的每个通信实体在初始时所掌握的信息。Knowledge 块定义了主体 $Init$ 知晓协议中 A、B 和 S 的身份标识，并持有一个随机数 Na 和 M；主体 Resp 知道协议中自身和服务器 S 的标识，并持有一个随机数 Nb；密钥服务器 $Server$ 拥有自身的身份标识以及随机数 Nab

- Agents 主体块：是 Alice&Bob 语言规范的核心部分，用于描述在理想情况下协议的正常执行过程，因此只包含协议的正常主体之间的消息交换过程，而不包含攻击者主体。在主体块中，需要依次声明每个主体在信息交换中所需要使用的初始知识以及每条交换策略。一次信息交换需要用一个唯一标号 i 进行标记，并需要两个主体的参与，采用符号 + 表示发送动作（同时需要说明接收者主体和首次使用到的知识），符号-表示接收动作。

- Environment 环境块：表达协议的执行实例，是根据 Agents 块所定义的协议模板确定具体的执行上下文，从而例化为一个实际的执行场景。Environment 块对 $Init$、$Resp$ 和 $Server$ 三个主体各自构造了一个实例。主体模板 $Init$ 的例化结果被记为 $Alice$，持有 Nonce(Na) 和 Nonce(M)，实际的对端通信方为 $Intruder$；实例 $Resp$ 的主体被记为 Bob，持有 Nonce(Nb)。模拟了 Bob 正常提供服务，而 $Alice$ 则实际在与伪装为 Bob 的 $Intruder$ 进行协议通信的场景；实例 $Server$ 则被标记为 $Server$，拥有随机数 Nab 作为会话密钥

- Goals 目标块：指明安全协议应当满足的安全属性，以及后续的分析验证。目前主要关注于证明私密性和非单射一致性。Goals 块描述了一条私密性和

中国知网 https://www.cnki.net

Protocol OTWAY-REES:

Knowledge:(* Intitial Knowledge*)

$Init : A, B, S, Na, M$

$Resp : B, S, Nb$

$Server : S, Nab$

Agents: (* Play the Role*)

Init(A,B,M,Na,S)

$[1]+ : B, (Na, M, A, B) : M, A, B, \{|Na, M, A, B|\}K(A, S)$

$[4]- : \{|Na, Nab|\}_{K(A,S)}$

Resp(B,S,Nb)

$[1]- : M, A, B, H$

$[2]+ : S, (Nb) : M, A, B, H, \{|Nb, M, A, B|\}_{K(B,S)}$

$[3]- : H', \{|Nb, Nab|\}_{K(B,S)}$

$[4]+ : A, (\ ) : H'$

Server(S,Nab)

$[2]- : M, A, B, \{|Na, M, A, B|\}_{K(A,S)}, \{|Nb, M, A, B|\}_{K(B,S)}$

$[3]+ : B, (Nab) : \{|Na, Nab|\}K(A, S), \{|Nb, Nab|\}K(B, S)$

Goals: (*Security Goals*)

[secrecy] Nab secret of $< A, B, S >$

[weakB] B non-injectively agrees with A on Na

Environments: (*Protocol instance*)

$[\mathrm{str}_1]Init[1] :< Alice, Intruder, Server, Na, M >$

$[\mathrm{str}_2]Resp[1] :< Intruder, Server, Nb >$

$[\mathrm{str}_3]Server[1] :< Server, Nab >$

end

图 3.3 Otway-Rees 协议在串空间规约中的具体实现

一条非单射一致性。其中私密性说明 Nonce(Nab) 在协议运行过程中是保密的，不会被正常主体之外的攻击者窃取；非单射一致性说明如果 B 完成了一次协议交换，那么至少有一个 A 的实例使用 Nonce(Na) 与其通信，且该实例未与其它 B 实例进行协议通信。

中国知网 https://www.cnki.net

3.3 扩展串空间及操作语义

3.3.1 串空间理论

安全协议可以表示为两个或多个主体之间的消息序列，其中这些消息加密用于为新的对话提供身份验证或分发加密密钥。在安全协议设计时，无论专家多么仔细以及设计后被其他专家多次评审，都将发现其设计或多或少存在缺陷，最终导致协议不可用。分析安全协议主要包含两个互补的活动，第一个旨在找到协议中缺陷，第二个旨在确定协议的正确性。这两个活动是相互交织的，因为发现漏洞可以帮助修改协议，从而证明其正确性。串空间理论则对应于第二个活动，当协议实际正确时，用以证明其正确性。

在串空间中，串是一序列的事件：他们表示正规主体在一个安全协议中的执行过程，或者恶意攻击者的一系列动作。而串空间则是一个伴随着由因果关系生成的图架构串集合。协议正确性宣言可以被表示为不同种类之间的串的相互联系。

在章节2.1中，我们给出了安全协议中的形式化描述，包括主体、消息以及事件。在串空间中，给定集合 A，其元素来自于协议中主体之间交换的所有可能消息项。集合 A 中的所有元素称之为项。关系 Subterm 是其中最为重要的一种关系，$t_1 \subset t$ 表明项 t_1 是项 t 的子项。在协议运行中，主体可能发送或者接受项，我们将发送项与 + 绑定，接受一个项与 - 绑定。

定义 3.1 一个符号项是一个二元组 $<\sigma, a>$，其中，$a \in A, \sigma$ 可取为符号 +、-。一个符号项可以表示为 $+t$ 或者 $-t$。$(\pm A)^*$ 是符号项的一个有限序列集合，可以表示为 $\{<\sigma_1, a_1> \ldots <\sigma_n, a_n>\}$

定义 3.2 串空间是一个路径映射集合 Σ，其中 $tr : \Sigma \longrightarrow (\pm A)^*$

定义 3.3 设定 C 为边集，N_C 为与边集 C 中有关联的结点集合，如果 C 是一个束，则需要满足以下条件：

1. C 是有限的
2. 如果 $n_1 \in N_C$，并且项 (n_1) 为负，那么存在唯一结点 n_2 使得 $n_2 \longrightarrow n_1 \in C$
3. 如果 $n_1 \in N_C$ 并且 $n_2 \Longrightarrow n_1$，那么 $n_2 \Longrightarrow n_1 \in C$

中国知网 https://www.cnki.net

4. C 是非循环的

如图3.2所示，我们可以看到两种因果关系:

- 因果关系 $n \Longrightarrow n'$ 表示串内通讯，当且仅当 $n = (s, i)$ 并且 $n' = (s, i+1)$ 时满足成立
- 因果关系 $n \longrightarrow n'$ 表示串间通讯，如 $(S_A, 1) \longrightarrow (S_B, 1)$ 表示主体 A 发送出消息，主体 B 从信道上接受消息。

与 Alice&Bob 语言规约不同，串形式可以很好的视为一种状态机，与底层实现更接近，它包括消息通讯动作的状态以及状态迁移，同时它也表示了一种网络代理之间的异步通讯方式。此外，串空间理论与 Paulson 的归纳方法极为接近。Paulson 将协议建模为一个规则集来扩展事件的序列，认证目标和私密性目标被视作为序列上的属性，可以通过归纳的方式不断生成。然而串空间理论使用偏序结构和束的定义，束中的结点组织成串, 自然而然地所有束可以线性成一个事件序列。

串空间理论与其他理论相比具有两大优势。1）束包含准确的因果相关性 2）串可以捕捉大量的信息，一个特定的串可能拥有束中的结点，从这角度出发，我们可以识别参与者的所有相关动作序列，来辅助隔离协议需要满足的一致性属性。串空间框架除了简单的证明协议的正确性，它还可以用来提供信念逻辑的可选语义，用于密码学协议或者分布式系统，很好地帮助细化和分割模型。因此本文使用串空间理论作为理论基石，实现从 Alice&Bob 规范出发到 Murphi 模型验证器之间保证语义一致性的支撑。

3.3.2 串空间操作语义

操作语义是通过抽象状态机或抽象函数来定义语言的语义，即它更加注重系统执行中数据该如何进行操作加工。它可以是确定的也可以是离散的，其中 Plokin 于 1981 年提出的结构化操作语义被当作为程序语言设计的基础方法。如最常见的标记迁移系统，将程序的执行描述成一个标记系统的迁移过程，状态表示为程序执行期间所观察到的变量取值，迁移关系表示状态之间的变迁。其中的迁移关系可用操作语义进行描述。操作语义的数学基础就是有限状态机，具有易补充、证明弱的特点。

中国知网 https://www.cnki.net

安全协议的运行是扮演正规主体不同正规串 $(i_1, str_1) \mid \ldots \mid (i_n, str_n)$ 与扮演恶意攻击者 Spy 的隐氏入侵者串并行组成。值得注意的是入侵者串不会在协议会话中显式地列出来。基于 Dolev-Yao 模型，入侵者串 spy 控制着整个网络通讯，所有由正规串生成的协议消息 m 都将被恶意攻击者所窃听，并加之其知识库中。Spy 可以根据其知识执行消息推演，将新组建的消息加之知识库中。为了简洁起见，我们用 kn 表示恶意攻击者所获得的所有消息集，Init 和 Run 来表示协议的运行状态。

Init 初始化

串空间框架初始化，我们需要生成所有的串实例，对于类型为 $\mathcal{T}_v$ 的所有变量 v，通过一个状态映射函数 σ 将串 i 与变量 v 映射到某个值上。

$$\sigma(i)(v) = \begin{cases} \mathsf{subVar}(i)(v) & \text{if } v \text{ is in } \mathsf{initKnRole}(\mathsf{ofAgent}(\mathrm{i})) \\ \mathsf{NULL} & \text{otherwise} \end{cases}$$

这里替换函数 subVar 将变量 v 替换到串 i 上，NULL 意味着什么也不做; ofAgent 实例化主体 i，initKnRole 获得主体 i 上的初始知识，从而完成所有变量的初始化。

Run 运行

在介绍运行状态之前，我们需要一个重要的函数 $\mathsf{update}(s, i, m, M)$，在每次执行 $Recv$ 操作中，都需要进行模式匹配以判定协议执行是否该终止。zhi x 函数 $\mathsf{update}(s, i, m, M)$ 递归定义了如何通过模式匹配来更新串 i 上的本地变量副本，最终返回一个新的状态，这里 m 是状态 s 上的本地变量值，M 是状态 s 上的刚获得的值。当 m 和 M 不属于同一个模式集时，匹配则失败，返回状态 $\perp$，意味着陷入一个死锁的状态。

当局部变量 m 和收到的消息 M 都属于 Agent 类型，如果串 i 上的变量值为未初始化 NULL，则更新局部变量 a 的值; 如果 a 的值等同于 A，则返回当前状态，不然返回 $\perp$。

$$\mathsf{update}(s, i, a, A) = \begin{cases} \mathsf{actSub}(i)[a \mapsto A] & \text{if } \sigma_a^i = \mathsf{X} \\ s & \text{if } \sigma_a^i = A \\ \perp & \text{otherwise} \end{cases} \tag{3.1}$$

中国知网 https://www.cnki.net

当局部变量 m 和收到的消息 M 都属于 Nonce 类型，如果串 i 上的变量值为未初始化 NULL，则更新局部变量 n 的值; 如果 na 的值等同于 Na，则返回当前状态，不然返回 $\perp$。

$$\mathsf{update}(s,i,na,Na)=\begin{cases}\mathsf{actSub}(i)[na\mapsto Na] & \text{if } \sigma^i_{na}=\mathsf{X}\\ s & \text{if } \sigma^i_{na}=Na\\ \perp & \text{otherwise}\end{cases} \tag{3.2}$$

当局部变量 m 和收到的消息 M 都属于 Number 类型，如果串 i 上的变量值为未初始化 NULL，则更新局部变量 n 的值; 如果 n 的值等同于 N，则返回当前状态，不然返回 $\perp$。

$$\mathsf{update}(s,i,n,N)=\begin{cases}\mathsf{actSub}(i)[n\mapsto N] & \text{if } \sigma^i_{n}=\mathsf{X}\\ s & \text{if } \sigma^i_{n}=N\\ \perp & \text{otherwise}\end{cases} \tag{3.3}$$

当局部变量 m 和收到的消息 M 都属于匿名消息类型，如果串 i 上的变量值为未初始化 NULL，则更新局部变量 h 的值; 如果 h 的值等同于 H，则返回当前状态，不然返回 $\perp$。

$$\mathsf{update}(s,i,T(h),H)=\begin{cases}\mathsf{actSub}(i)[T(h)\mapsto H] & \text{if } \sigma^i_{h}=\mathsf{X}\\ s & \text{if } \sigma^i_{h}=H\\ \perp & \text{otherwise}\end{cases} \tag{3.4}$$

当局部变量 m 和收到的消息 M 都属于连结消息类型，则需要对该消息进行连续两次的模式匹配，最后返回新的状态 s'。

$$\mathsf{update}(s,i,\{|m_1,m_2|\},\{|M_1,M_2|\})=\mathsf{update}(\mathsf{update}(s,i,m_1,M_1),i,m_2,M_2) \tag{3.5}$$

当局部变量 m 和收到的消息 M 都属于加密消息类型，则需要对本地消息的密钥和收到消息的密钥进行比较，如果无法解开收到的消息，则陷入死锁状态。

$$\mathsf{update}(s,i,\{|m_1|\}_k,\{|M_1|\}_K)=\begin{cases}\mathsf{update}(s,i,m_1,M_1) & \text{if } \sigma^i_{k}=\mathrm{K}\\ \perp & \text{otherwise}\end{cases} \tag{3.6}$$

中国知网 https://www.cnki.net

当局部变量 m 和收到的消息 M 都属于模消息类型，则需要对该消息进行连续两次的模式匹配，最后返回新的状态 s'。

$$\mathsf{update}(s, i, \{|m_1, m_2|\}_m, \{|M_1, M_2|\}_m) = \mathsf{update}(\mathsf{update}(s, i, m_1, M_1), i, m_2, M_2) \tag{3.7}$$

当局部变量 m 和收到的消息 M 都属于指数消息类型，则需要对该消息进行连续两次的模式匹配，最后返回新的状态 s'。

$$\mathsf{update}(s, i, \{|m_1, m_2|\}_e, \{|M_1, M_2|\}_e) = \mathsf{update}(\mathsf{update}(s, i, m_1, M_1), i, m_2, M_2) \tag{3.8}$$

另外，当两者不属于同一种类型时，将返回状态 $\perp$ 表示匹配失败，这里列举出集中情况。

$$\mathsf{update}(s, i, na, \_) = \perp \tag{3.9}$$

$$\mathsf{update}(s, i, a, \_) = \perp \tag{3.10}$$

$$\mathsf{update}(s, i, n, \_) = \perp \tag{3.11}$$

$$\mathsf{update}(s, i, \{|m_1, m_2|\}, \_) = \perp \tag{3.12}$$

$$\mathsf{update}(s, i, \{|m_1, m_2|\}_m, \_) = \perp \tag{3.13}$$

$$\mathsf{update}(s, i, \{|m_1, m_2|\}_e, \_) = \perp \tag{3.14}$$

$$\mathsf{update}(s, i, \{|m_1|\}_k, \_) = \perp \tag{3.15}$$

当串空间框架处于状态 s 时，$[\![v]\!]_s^i$ 表示串 i 上本地变量 v 的值，这里引入当前接受函数 $\mathsf{curRecvs}(s, i, act)$ 来构造主体 i 在状态 s 上等待被接受的消息。

$$\mathsf{curRecvs}(s, i, act) = \begin{cases} \mathsf{NULL} & \text{if } act = (+, m) \\ [\![m]\!]_s^i & \text{if } act = (-, m) \end{cases} \tag{3.16}$$

其中，act 表示发送消息或者接受消息；$\mathsf{curRecvs}(s, i, act)$ 要么返回 NULL，否则返回 $[\![m]\!]_s^i$。

有了匹配函数的帮助，接下去展开介绍本文中的扩展串空间理论的操作语义。

在我们的理论中，主体与主体之间通过一个公共信道相互通讯，而这些信道上的通讯是透明的，被恶意攻击者 Spy 所监听。通过一个四元组 <p, s, m, L> 来

中国知网 https://www.cnki.net

构造会话运行状态，并行组合 p 是一个二元组对 (i, str)，其中 i 是串标识符，str 是束中的一个串。状态 s 从所有主体的视角出发，将每一个变量映射到具体的某个值上。其中，m 要么是公共信道上的某个消息，要么 NULL 表示信道为空。最后 L 是一条消息列表，表示正规主体正在等待接受的消息序列。

如表3.1所示，串空间理论的操作语义由两部分组成，第一部分服务于正规主体，第二部分服务于恶意攻击者。

表 3.1 迁移规则表

$$\textbf{(Send)}\ \frac{s \neq \bot; \mathsf{hd}(str) = (+, m)}{< (i, str), s, \mathsf{NULL}, L > \rightarrow < (i, \mathsf{tl}(str)), s, [\![m]\!]_s^i, add(L, \mathsf{cur}(s, i, \mathsf{hd}(\mathsf{tl}(str)))) >}$$

$$\textbf{(Recv)}\ \frac{s \neq \bot; hd(str) = (-, m)}{< (i, str), s, M, L > \rightarrow < (i, \mathsf{tl}(str)), \mathsf{update}(s, i, m, M), \mathsf{NULL}, L' >}$$

$$\textbf{(Parl)}\ \frac{(p_1, str_1), s, M_1, L \rightarrow (p_1, str_1'), s', M_1', L'}{< (p_1 \| p_2, str_1 \| str_2), s, M_1, L > \rightarrow < (p_1 \| p_2, str_1' \| str_2), s', M_1', L' >}$$

$$\textbf{(Emit)}\ \frac{s \neq \bot; \exists m. M \in [\![\mathsf{Kn}]\!]_s^{spy} \wedge M \in L \wedge match(m, M)}{< (i', str), s, \mathsf{NULL}, L > \rightarrow < (i', str), s, m, L >}$$

$$\textbf{(Flush)}\ \frac{s \neq \bot}{< (i', str), s, M, L > \rightarrow < (i', str), s(i)[\mathsf{Kn} \rightsquigarrow [\![\mathsf{Kn}]\!]_s^{spy} \cup \{M\}], \mathsf{NULL}, L >}$$

$$\textbf{(Sep)}\ \frac{s \neq \bot, \{|m_1; m_2|\} \in [\![\mathsf{Kn}]\!]_s^{spy}, m_1 \notin [\![\mathsf{Kn}]\!]_s^{spy} \vee m_2 \notin [\![\mathsf{Kn}]\!]_s^{spy}}{< (i', str), st, \mathsf{NULL}, L > \rightarrow < (i', str), s(i)[\mathsf{Kn} \rightsquigarrow [\![\mathsf{Kn}]\!]_s^{spy} \cup \{m_1, m_2\}], \mathsf{NULL}, L >}$$

$$\textbf{(Dec)}\ \frac{s \neq \bot, \{|m|\}_k \in [\![\mathsf{Kn}]\!]_s^{spy}, m \notin [\![\mathsf{Kn}]\!]_s^{spy}, k^{-1} \in [\![\mathsf{Kn}]\!]_s^{spy}}{< (i', str), s, \mathsf{NULL}, L > \rightarrow < (i', str), s(i)[\mathsf{Kn} \rightsquigarrow [\![\mathsf{Kn}]\!]_s^{spy} \cup \{m\}], \mathsf{NULL}, L} >$$

$$\textbf{(Cat)}\ \frac{s \neq \bot, m_1, m_2 \in [\![\mathsf{Kn}]\!]_s^{spy}, \{|m_1; m_2|\} \notin [\![\mathsf{Kn}]\!]_s^{spy}, \{|m_1; m_2|\} \in \mathsf{sub}(\mathsf{cur}(L))}{< (i', str), s, \mathsf{NULL}, L > \rightarrow < (i', str), s(i)[\mathsf{Kn} \rightsquigarrow [\![\mathsf{Kn}]\!]_s^{spy} \cup \{|\}m_1; m_2|\}\}], \mathsf{NULL}, L >}$$

$$\textbf{(Enc)}\ \frac{s \neq \bot, m \in s(spy)(\mathsf{Kn}), k \in [\![\mathsf{Kn}]\!]_s^{spy}, \{|m|\}_k \notin [\![\mathsf{Kn}]\!]_s^{spy}, \{|m|\}_k \in \mathsf{sub}(\mathsf{cur}(L))}{< (i', str), s, \mathsf{NULL}, L > \rightarrow < (i', str), s(spy)[\mathsf{Kn} \rightsquigarrow [\![\mathsf{Kn}]\!]_s^{spy} \cup \{|\}m|\_k\}], \mathsf{NULL}, L >}$$

$$\textbf{(Mod)}\ \frac{s \neq \bot, m_1, m_2 \in [\![\mathsf{Kn}]\!]_s^{spy}, \{|m1, m2|\}_m \notin [\![\mathsf{Kn}]\!]_s^{spy}, \{|m1, m2|\_m \in \mathsf{sub}(\mathsf{cur}(L))}{< (i', str), s, \mathsf{NULL}, L > \rightarrow < (i', str), s(spy)[\mathsf{Kn} \rightsquigarrow [\![\mathsf{Kn}]\!]_s^{spy} \cup \{|m1, m2|\}_m], \mathsf{NULL}, L >}$$

$$\textbf{(Exp)}\ \frac{s \neq \bot, m_1, m_2 \in [\![\mathsf{Kn}]\!]_s^{spy}, \{|m1, m2|\}_e \notin [\![\mathsf{Kn}]\!]_s^{spy}, \{|m1, m2|\}_e \in \mathsf{sub}(\mathsf{cur}(L))}{< (i', str), s, \mathsf{NULL}, L > \rightarrow < (i', str), s(spy)[\mathsf{Kn} \rightsquigarrow [\![\mathsf{Kn}]\!]_s^{spy} \cup \{|m1, m2|\}_e], \mathsf{NULL}, L >}$$

给定正规主体 i，与串 str，可以执行以下操作:

- **Send** 规则: 如果当前状态 s 不为死锁，并且当前串的状态符号为正以及公

中国知网 https://www.cnki.net

共信道上为空，表明等待发送消息状态，正规主体可以将消息 m 发送到公共信道上。随后，串 str_i 的头部结点则被弹出，公共信道上的消息存放着消息 $[\![m]\!]_s^i$。

- **Recv** 规则: 如果当前状态 s 不为死锁，并且当前串的状态符号为负，表明主体 i 等待接受消息。首先串 str_i 的头部结点弹出，进入模式匹配状态，公共信道上的消息被取出，状态设置为 NULL。全局等待接受消息加入消息，更正状态为 L'，其中 $L' = \mathsf{add}(\mathsf{sub}(L, \mathsf{curRecvs}(s, i, \mathsf{hd}(str))), \mathsf{curRecvs}(\mathsf{update}(s, i, m, M), i, \mathsf{hd}(\mathsf{tl}(str)))$。

- **Parl** 规则: 如果协议存在数条串实例 $str_1, \ldots, str_n$，那么某条串上的状态迁移会并行地改变所有串实例上的状态。

对于恶意攻击者，给定一个特殊的串实例 i'，其可以执行以下操作:

- **Emit** 规则：恶意攻击者可以发送知识库 kn 中的消息 m 到公共通讯信道上;

- **Flush** 规则：恶意攻击者窃听公共信道上的消息，并截取加入到自身的知识库中;

- **Sep** 规则：恶意攻击者可以分割知识库 kn 中的连结消息 $\{|m_1; m_2|\}$，将其子消息 m_1 和 m_2 加入到知识库 Kn 中;

- **Dec** 规则：恶意攻击者可以解密知识库 kn 中的加密消息 $\{|m|\}_K$，并将解密后的明文消息 m 加入到知识库 Kn 中;

- **Cat** 规则：对于知识库 kn 中的两条消息 m_1 和 m_2，恶意攻击者可以执行连结操作，并将构造后的消息 $\{|m_1; m_2|\}$ 加入到知识库 Kn 中。为了尽可能减少无用伪造消息，我们要求 $\{|m_1; m_2|\}$ 是能够被正规主体所接受的消息集模式;

- **Encrypt** 规则：对于知识库 kn 中的消息 m_1 和密钥 k，恶意攻击者可以执行加密操作，并将构造后的消息 $\{|m_1|\}_k$ 加入到知识库 Kn 中。为了尽可能减少无用伪造消息，我们要求 $\{|m_1|\}_k$ 是能够被正规主体所接受的消息集模式;

中国知网 https://www.cnki.net

- **Mod** 规则：对于知识库 kn 中的两条消息 m_1 和 m_2，恶意攻击者可以执行模操作，并将构造后的消息 $\{|m_1; m_2|\}_m$ 加入到知识库 Kn 中。为了尽可能减少无用伪造消息，我们要求 $\{|m_1; m_2|\}_m$ 是能够被正规主体所接受的消息集模式;

- **Exp** 规则：对于知识库 kn 中的两条消息 m_1 和 m_2，恶意攻击者可以执行指数操作，并将构造后的消息 $\{|m_1; m_2|\}_e$ 加入到知识库 Kn 中。为了尽可能减少无用伪造消息，我们要求 $\{|m_1; m_2|\}_e$ 是能够被正规主体所接受的消息集模式。

对于加密和连接操作，考虑到模型状态空间的限制，我们添加了一个条件来限制入侵者的行为，即伪造的消息必须匹配一个可接收的消息的子项。否则，仿真所探索的状态空间将是无限的。

3.4 AnB2Murphi 模型转换器

我们基于 Alice&Bob 语言规范和 Murphi 模型检测器开发了一个模型转换工具——AnB2Murphi，其总体架构主要由转换和验证两个模块组成如图3.4所示。转换：转换模块的目标是将抽象的 Alice&Bob 模型迁移到具体的 Murphi 模型。它包括两个重要的过程：语法分析和代码生成。在语法分析过程，采用 Ocaml/Menhir 工具来解析 Alice&Bob 规范的结构，将其转换到扩展了操作语义的串空间模型中，然后在代码生成过程将其转换成为 Murphi 模型。验证：验证模块的目标是检查转换阶段生成的 Murphi 模型，并基于其所描述的协议模型，对所查询的性质进行形式化验证。Murphi 编译器首先将模型编译为 C++ 程序。然后将 C++ 程序编译成可执行文件来完成具体的验证过程，这种方式也加速了模型检测的速度。

在展开介绍 AnB2Murphi 架构图之前，我们分析下该方法的可行性，即为什么我们可以将高层次的 Alice&Bob 规约转换到低层次的 Murphi 模型验证器，而不失语义完整性。该部分我们从两个角度出发，1）Alice&Bob 语言规约与 Murphi 的关系 2）扩展串空间与 Murphi 的关系

中国知网 https://www.cnki.net

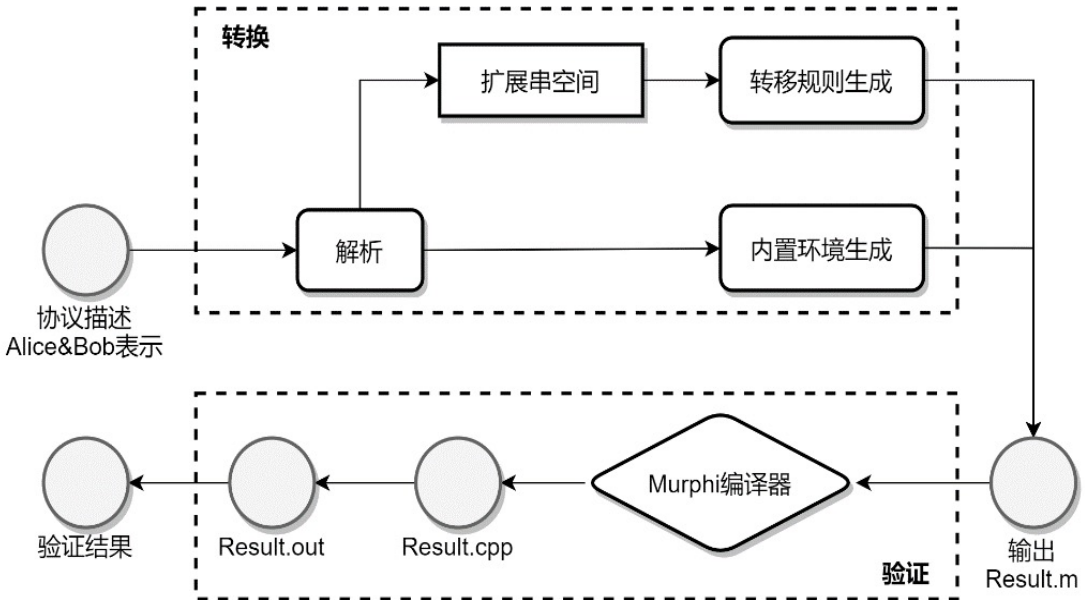


图 3.4 AnB2Murphi 总体架构图

3.4.1 Alice&Bob 规约与 Murphi 模型检测器

在上述章节中，我们详细介绍了 Alice&Bob 语言规约的形式化描述，其中一个密码学协议在 Alice&Bob 语言规约中被划分为 6 个部分: 协议名、Agents 块、Types 块、Knowledges 块、Environment 块和 Goals 块。同时 Murphi 语言的语义也可以大致划分为四个部分：声明部分、迁移规则部分、初始化部分和性质部分。

现在详细介绍各个部分之间的对应关系。

- Agents 块与迁移规则部分：在 Alice&Bob 模型中，Agents 块包含了正规主体名、主体拥有的知识、主体在某信道上的动作事件 (发送或接受消息)。因此在 Murphi 模型中，非常自然地对应到迁移规则部分。迁移规则的规则名则为主体名 + 序号；迁移规则的 guard 为执行动作的条件，如发送消息的前提在于公共信道为空，并且主体处于发送消息状态，接受消息的前提在于公共信道不能为空，等待接受消息序列头为预期想要接受的消息格式等等；迁移规则的 action 部分则为发送或者接受消息，其中调用声明部分的过程或者函数，以帮助构造或者解构消息

- Types&Knowledges 块对应声明部分: Alice&Bob 模型中的 Types 块和 Knowledges 块主要涉及到全局变量、函数（过程）和主体初始化知识的声明。其中 Types 中会声明公钥加密函数、主体名、自然数等，在 Murphi 中，对应

中国知网 https://www.cnki.net

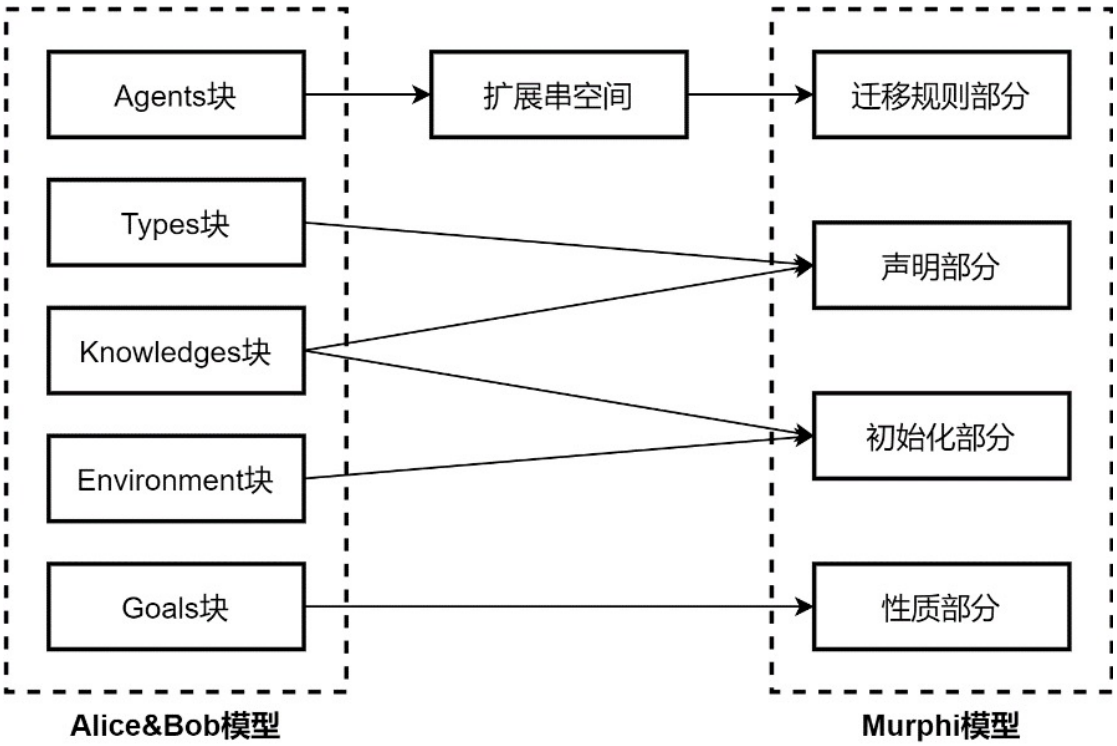


图 3.5 Alice&Bob 语言规范与 Murphi 模型映射

于声明部分中的 var 部分，我们构造相应的 record 以记录主体所拥有的数据类型。如图3.3中，$Init$ 主体拥有 A、B、S 的身份表示，Na 和 M 的随机数类型，则在 $Init$ 的 Record 中，可以访问到这些数据类型，对其进行赋值更改。另外在 Murphi 的声明部分，我们自定义了构造解构消息的函数过程，模式匹配的函数过程等其他调用函数。

- Knowledges&Environment 块对应初始化部分：Alice&Bob 模型在 knowledges 和 Environment 块分别对主体的变量进行了声明和实例化，因此在 Murphi 的初始化部分，我们也做了同样的事。对在 Environment 块中主体的所有知识进行赋值，构造消息集用以接受消息时判断消息的模式是否为想要的，初始化 Spy 恶意攻击者的知识库 Kn 以及所有主体的知识库等。

- Goals 块对应性质部分：在 Alice&Bob 模型中，Goals 块仅简要地介绍了安全协议应该满足什么性质。在 Murphi 模型中，我们对安全协议的认证性和私密性进行了刻画，如随机数的私密性表示在协议运行结束后，该随机数不会出现在恶意攻击者中的知识库；两个主体之间的认证性则要求某个消息在双方的知识库中达到一致。

中国知网 https://www.cnki.net

3.4.2 扩展串空间与 Murphi 模型检测器

在前面，我们介绍了什么是串空间理论，其中串和束的概念最为重要。另外我们给出了串空间的操作语义，这很好地指导我们如何将一个协议运行的束转换到 Murphi 的模型中去。

首先，束是由数条正规主体的串与恶意攻击者的串相互交织组成的，串之间存在这三种关系：串间关系，串内关系和偏序关系。如图3.6所示为束的局部，即 $Alice$ 发送消息给 Bob，随后 Bob 从信道上接受消息，返回确认消息给 $Alice$。可以看到在串 $Alice$ 中，$(S_a, 2)$ 是发生在 $(S_a, 1)$ 之后的，因此在 Murphi 的迁移规则中，我们需要对主体的状态有限制，把这种串内关系转移到一阶逻辑中；而结点上的符号表示发送消息或者接受消息，这两种结点类型也对应这两种迁移规则模板：发送消息要求主体在该状态下构造该消息，将消息摆置到公共信道上，最后更改信道状态以及主体状态；接受消息要求主体从信道上将消息取出，进行模式匹配，更改本地的消息集合，最后更新主体的状态。在这里，我们可以看到没有对发送消息以及接受消息的主体进行限制，因为我们假设所有消息都会被恶意攻击者所监听，如果条件满足，则正规主体也可从公共信道上接受到预期的消息。

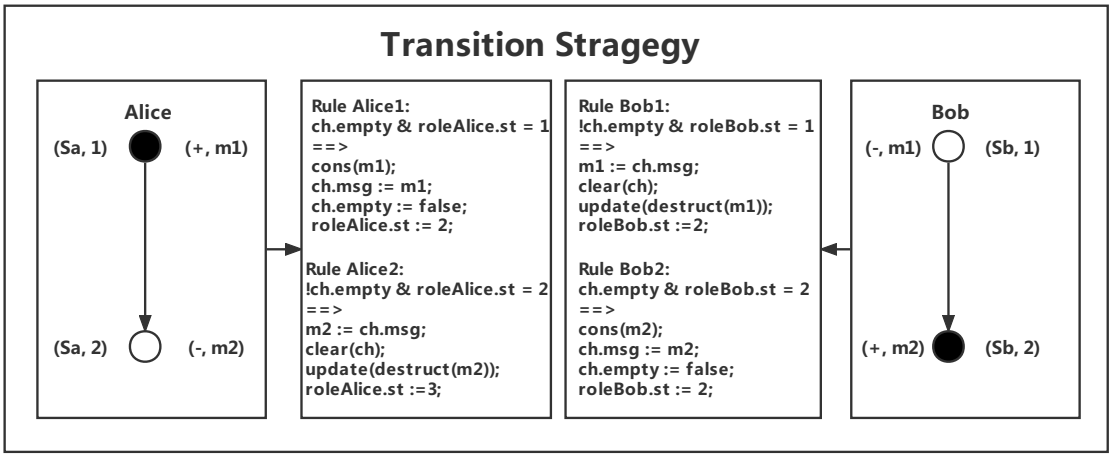


图 3.6 扩展串空间与 Murphi 之间的映射

如图所示3.6所示为正规主体串与 Murphi 的对应关系，下面介绍恶意攻击者串与 Murphi 的对应关系。上面我们已经对串空间框架中恶意攻击者的行为建模，如 Flush、Emit、Sep 等，在 Murphi 中我们对 8 种操作语义共 16 条迁移规则分别进行了刻画。和正规主体不同，恶意攻击者不需要关注协议进行到了哪一步，它

中国知网 https://www.cnki.net

随时监听公共信道上的消息，并对知识库中的消息进行演算，发送伪造的消息给正规主体，达到破坏的目的。

算法 3.1 Emit 操作语义到 Murphi 映射

Input: 恶意攻击者的 Emit 串 $\mathsf{Strand_{Emit}}$

Output: 迁移规则 IntruderEmit $\mathsf{R_{Emit}}$

```
for each Node n in range(len(Strand_E)) do
    if sign(n) == + then
            m = msg(n); //结点消息
            a = n.ag; // 接受者
            pattern = patSet(m); //消息模式
            R_Emit.name = 'IntruderEmit' + n.st;
            R_Emit.guard = a.st == n.st & ch[n.st].empty & !Emit(pattern[n])
            R_Emit.action := { cons(m); ch[n.st] = m; Emit(pattern[n]) = true; }
    end if
end for
```

将恶意攻击者发送消息的操作语义映射到 Murphi 的迁移状态的过程如算法3.1所示。对于输入串为恶意攻击者的 Emit 串，首先遍历串上的所有结点，如果状态 n 的符号为正，表明该恶意结点正准备向公共信道上发送伪造的消息，首先获取结点上消息赋值给 m，获取消息的接受者 ag，随后通过模式匹配判断结点上的消息模式 pattern，随后生成迁移规则 $\mathsf{R_{Emit}}$，首先规则名标识为"IntruderEmit+结点的序号"，在规则的 guard 部分首先判断接收主体的状态是否为发送结点状态，其次判断公共信道是否为空，最后查看该恶意攻击者是否发送过该模式的消息; 当 guard 满足时，执行 action 动作，即构造消息 m，将消息 m 放置在公共信道上，最后将发送该模式的消息设为 True，标识后续恶意攻击者无需再往公共信道上传输该类型的消息。

将恶意攻击者的 Flush 串映射到 Murphy 的迁移状态如算法3.2所示，通过输入恶意攻击者的 Flush 串，得到 Murphi 中的对应迁移规则 IntruderGet 规则。首先遍历 Flash 串上的所有结点，如果当前结点的符号为-，表示该恶意攻击者在该状态窃听并窃取公共信道上的消息，变量 m 表示 Flush 结点上的等待被接收的消息，迁移规则 R_{Flash} 的标识为'IntruderGet'+ 结点的序号；规则的 guard 为信道不为空以及信道上的消息发送者不为恶意攻击者，这样做的好处是可以大量减

中国知网 https://www.cnki.net

算法 3.2 Flush 操作语义到 Murphi 映射.

Input: 恶意攻击者的 Flush 串 $\mathsf{Strand}_{\mathsf{Flush}}$

Output: 迁移规则 IntruderFlash $\mathsf{R}_{\mathsf{Flush}}$

```
1: for each Node n in range(len(Strand_Flush)) do
2:     if sign(n) == - then
3:         m = msg(ch[n.st]); //结点消息
4:         R_Flush.name = 'IntruderFlash' + n.st;
5:         R_Flush= !ch[n.st].empty & !ch[n.st].sender != Intruder
6:         R_Flush.action := {Kn{m} = true; clear ch[n.st]; }
7:     end if
8: end for
```

少窃听自己所发伪造消息的状态数；规则的 action 为将消息 m 加入到恶意攻击者知识库 kn，并清空信道上的所有消息。

算法 3.3 Sep 操作语义到 Murphi 映射.

Input: 恶意攻击者的 Sep 串 $\mathsf{Strand}_{\mathsf{Sep}}$

Output: 迁移规则 Sep $\mathsf{R}_{\mathsf{Sep}}$

```
1: for each Node n in range(len(Strand_Sep)) do
2:     if type(n) == 'Mpair' then
3:         m1= msg(n)[0]; //结点消息一
4:         m2 = msg(n)[1]; // 结点消息二
5:         pattern = patSet(msg(n)); //消息模式
6:         R_Sep.name = 'IntruderSep' + pattern;
7:         R_Sep.guard = !Kn{m1} & !Kn{m2} & Kn{msg(n)}
8:         R_Sep.action = {Kn{m1} = True; Kn{m2} = True;}
9:     end if
10: end for
```

将恶意攻击者的 Sep 串映射到 Murphy 的迁移状态如算法3.3所示，通过输入恶意攻击者的 Sep 串，得到 Murphi 中的对应迁移规则 IntruderSep 规则。首先遍历 Sep 串上的所有结点，如果当前结点 n 的类型为 $Mpair$，则进行 Sep 的状态迁移代码生成。变量 $m1$ 为结点上的第一个索引位置消息，$m2$ 为结点上的第二个索引位置的消息，$pattern$ 为消息 $msg(n)$ 的消息模式。迁移规则 R_{Sep} 的标识为 Intruder+ 消息模式; 规则的 guard 为恶意攻击者掌握了消息 $msg(n)$，并且不知道

中国知网 https://www.cnki.net

变量 $m1$ 和 $m2$ 的值；在规则的 action 中将变量 $m1$ 与 $m2$ 加入到恶意攻击者的知识库中。

算法 3.4 Cat 操作语义到 Murphi 映射.

Input: 恶意攻击者的 Cat 串 $\mathsf{Strand}_{\mathsf{Cat}}$

Output: 迁移规则 IntruderCat $\mathsf{R}_{\mathsf{Cat}}$

```
for each Node n in range(len(Strand_Cat)) do
    if sign(n) == - and n.st == 1 then
        m1 = msg(n)
    else if sign(n)==- then
        m2 = msg(n)
    end if
        m = Mpair(m1,m2); //结点消息
        R_Cat.name = 'IntruderCat' + n.st;
        R_Cat.guard = Kn[m1] & Kn[m2] & !Kn[m]
        R_Cat.action = {cons(m); Kn{m} = true; }
end for
```

与 Sep 操作语义相反，操作语义 Cat 到 Murphi 的映射如算法3.4所示，输入恶意攻击者的 Cat 串，输出 Murphi 中的迁移规则 IntruderCat。首先遍历 Cat 串中的所有结点，如果当前结点的符号为-，并且位于串的第一个位置，表明恶意攻击者在当前状态接受了第一条消息赋值给变量 $m1$，随后变量 $m2$ 标识第二个消息，接着构造变量 m 表示 $m1$ 和 $m2$ 之间的连结消息。迁移规则 R_{Cat} 的标识为 IntruderCat 与结点的序号，规则的 guard 为恶意攻击者掌握了 $m1$ 和 $m2$，并且对两消息的连结并不知悉；规则的 action 为构造该连结消息，并将该消息加入到自身的知识库中。

对于迁移规则 Enc 到 Murphi 的映射如算法3.5所示，输入恶意攻击者的 Env 串，输出 Murphi 中的 IntruderEnc 迁移规则。与上述算法相同，首先遍历串中的所有结点，如果当前结点的类型为 crypt，取出后续两个结点中的消息，分别为加密明文以及密钥，随后将其进行相应的加密算法构造成密文，判别密文的消息模式。迁移规则的标识为 IntruderEnc+ 结点的序号；规则的 guard 为恶意攻击者掌握了加密明文和加密密钥，但是不知道加密密文；规则的 action 为构造其加密消息，并将其加入到知识库中。

中国知网 https://www.cnki.net

算法 3.5 Enc 操作语义到 Murphi 映射.

Input: 恶意攻击者的 Enc 串 $\mathsf{Strand}_{\mathsf{Enc}}$

Output: 迁移规则 IntruderEnc $\mathsf{R}_{\mathsf{Enc}}$

```
for each Node n in range(len(Strand_Enc)) do
    if type(n)==Crypt then
        m = msg(n); //结点消息加密明文
        k = msg(n + 1)]; //结点消息加密密钥
        encmsg = Crypt(m,k) //加密消息
        pattern = patSet(envmsg); //消息模式
        R_Enc.name = 'IntruderEnc' + n.st;
        R_Enc.guard = Kn{m} & Kn{k} & !Kn{encmsg}
        R_Enc.action := {cons(encmsg);Kn{encmsg} := true;}
    end if
end for
```

算法 3.6 Dec 操作语义到 Murphi 映射.

Input: 恶意攻击者的 Dec 串 $\mathsf{Strand}_{\mathsf{Dec}}$

Output: 迁移规则 IntruderDec $\mathsf{R}_{\mathsf{Dec}}$

```
for each Node n in range(len(Strand_Dec)) do
    if type(n) == Dec then
        encmsg = msg(n)[1]; //结点消息
        enckey= msg(n)[2]; //结点消息
        R_Dec.name = 'IntruderDec' + n.st;
        R_Dec.guard = Kn{encmsg} & Kn{enckey}
        R_Dec.action := {plaintext := deconstruct(encmsg, enckey)); Kn{plaintext} := true; }
    end if
end for
```

对于迁移规则 Dec 到 Murphi 的映射如算法3.6所示，输入恶意攻击者的 Dec 串，输出 Murphi 中的 IntruderDec 迁移规则。首先遍历串中的所有结点，如果当前结点的类型为 Dec，取出该结点消息赋值给变量 $encmsg$，结点上的加密密钥 $enckey$。迁移规则 $\mathsf{R}_{\mathsf{Dec}}$ 的标识为 intruderDec+ 结点的序号；规则的 guard 为恶意攻击者掌握加密密文与密文所对应的密钥；规则的 action 为用密钥解构密文，并将解开后的明文赋值给变量 plaintext，最后将该明文加入到恶意攻击者的知识库中。

中国知网 https://www.cnki.net

算法 3.7 Mod 操作语义到 Murphi 映射.

Input: 恶意攻击者的 Mod 串 $\mathsf{Strand}_{\mathsf{Mod}}$

Output: 迁移规则 IntruderMod $\mathsf{R}_{\mathsf{Mod}}$

```
1:  for each Node n in range(len(Strand_Mod)) do
2:      if type(n) == Mod then
3:              m1 = msg(n)[1]; //结点消息
4:              m2= msg(n)[2]; //结点消息
5:              modmsg= Mod(m1,m2); //结点消息
6:              R_Mod.name = 'IntruderMod' + n.st;
7:              R_Mod.guard = Kn{m1} & Kn{m2} & !Kn{modMsg}
8:              R_Mod.action := {cons(modmsg); Kn{modmsg}:=true}
9:      end if
10: end for
```

对于迁移规则 Mod 到 Murphi 的映射如算法3.7所示，输入恶意攻击者的 Mod 串，输出 Murphi 中的 IntruderMod 迁移规则。首先遍历串中的所有结点，如果当前结点的类型为 Mod，取出该结点消息赋值给变量 $m1$ 和 $m2$。迁移规则 $\mathsf{R}_{\mathsf{Mod}}$ 的标识为 intruderMod+ 结点的序号；规则的 guard 为恶意攻击者掌握两个模消息；规则的 action 为构造该 modmsg，并将该消息加入到恶意攻击者的知识库中。

算法 3.8 Exp 操作语义到 Murphi 映射.

Input: 恶意攻击者的 Exp 串 $\mathsf{Strand}_{\mathsf{Exp}}$

Output: 迁移规则 IntruderExp $\mathsf{R}_{\mathsf{Exp}}$

```
1:  for each Node n in range(len(Strand_Exp)) do
2:      if type(n) == Exp then
3:              m1 = msg(n)[1]; //结点消息
4:              m2= msg(n)[2]; //结点消息
5:              expdmsg= Exp(m1,m2); //结点消息
6:              R_Exp.name = 'IntruderExp' + n.st;
7:              R_Exp.guard = Kn{m1} & Kn{m2} & !Kn{expMsg}
8:              R_Exp.action := {cons(expmsg); Kn{expmsg}:=true}
9:      end if
10: end for
```

对于迁移规则 Exp 到 Murphi 的映射如算法3.8所示，输入恶意攻击者的 Mod 串，输出 Murphi 中的 IntruderExp 迁移规则。首先遍历串中的所有结点，如果当

中国知网 https://www.cnki.net

前结点的类型为 Mod，取出该结点消息赋值给变量 $m1$ 和 $m2$。迁移规则 $\mathsf{R_{Exp}}$ 的标识为 intruderExp+ 结点的序号；规则的 guard 为恶意攻击者掌握两个指数消息；规则的 action 为构造该 $expmsg$，并将该消息加入到恶意攻击者的知识库中。

3.4.3 模型转换过程

转换主要由解析和生成过程构成。解析过程将 A&B 规范转换为扩展的链空间，以确保转换过程中语义的一致性。并利用生成过程生成 Murphi 模型中的转换规则部分和内置环境。

1、解析过程

这个过程的主要任务是用 Ocaml/Menhir 工具分析 A&B 规范的结构。这个过程比较简单，我们就不赘述了。解析 A&B 规范的结果直接在下面给出。

```
type ProtocolContext = [
  Protocol of type * knowledge * agent * environment * goal |
  Null
]
```

我们将安全学习翻译到类型为 ProtocolContext 中，其中共包含五个部分：type 类型、knowledge 知识、agent 代理、environment 环境和 goal 目标。Protocol 是协议的标记。通过这种操作，最终 Alice&Bob 规约被转换到 $\mathsf{Protocol(t, k, a, e, g)}$，其中的各个部分分别对应 ProtocolContext 的相应地方。

此外，安全协议中代理之间可能交换的消息定义如下:

```
type message = [
  | Nonce of identifier (*Nonce*)
  | Agent of roleName (*Agent Identifier*)
  | Const of identisirt (*Const Number*)
  | Pk of roleName (*Public Key*)
  | Sk of roleName (*Secret Key*)
  | K of roleName * roleName (*Symmetry Key*)
```


中国知网 https://www.cnki.net

```
  | Mod of message * message (*Mod*)
  | Exp of message * message (*Exp*)
  | Tmp of messageName (*Temporary*)
  | Sign of message * message (*Signature*)
  | Aenc of message * message (*Aencrypt*)
  | Senc of message * message (*Sencrypt*)
  | Concat of message list (*Concatenation*)
]
```

2、生成过程

该过程基于最后一个过程构造的扩展串空间生成 Murphi 模型的输入语言。生成过程包括两个子过程: 生成转换规则和生成内置环境。

$$
\begin{aligned}
&\mathsf{trans}(\mathsf{act}, \mathsf{M}, \mathsf{i}, \mathsf{role_i}) = \\
&\quad \mathsf{let\ atoms\ =\ getAtoms}(\mathsf{M})\ \mathsf{in} \\
&\quad \mathsf{match\ act\ with} \\
&\quad \mid (+, \mathsf{M}) \longrightarrow \mathsf{genRuleName}(\mathsf{role_i}, \mathsf{i}); \\
&\qquad\qquad\quad \mathsf{genSendGuard}(\mathsf{role_i}, \mathsf{i}); \\
&\qquad\qquad\quad \mathsf{genSendAct}(\mathsf{role_i}, \mathsf{i}, \mathsf{atoms}) \\
&\quad \mid (-, \mathsf{M}) \longrightarrow \mathsf{genRuleName}(\mathsf{role_i}, \mathsf{i}); \\
&\qquad\qquad\quad \mathsf{genRecvGuard}(\mathsf{role_i}, \mathsf{i}); \\
&\qquad\qquad\quad \mathsf{genRecvAct}(\mathsf{role_i}, \mathsf{i}, \mathsf{atoms})
\end{aligned}
$$

对于生成转换规则的过程，我们首先为规则主体生成转换规则。Agent 串可以被看作是一个状态机。串中的每个结点都可以被视为状态机的一个状态。当代理接收或发送消息时，其状态将发生变化。对于每个正规主体，我们使用一个变量 $commit$ 来记录它是否完成了协议的执行。在 Murphi 中，规则表达了状态转换。在生成每个代理串后，我们使用 $trans())$ 方法将串中的第 i 个结点转化为 Murphi 规则。规则原则的行为相对简单。它只包括两个部分：构造发送的信息和解构接收的信息。函数 genSendAct() 和 genRecvAct() 生成发送或接收消息的相应规则的动作。

然后我们为恶意入侵者生成推理规则。基于 Dolev-Yao 模型，我们描述了入侵者的行为:

中国知网 https://www.cnki.net

- 窃听和截取网络中的任何消息;
- 以正规主体或假冒法人主体参与协议运作;
- 从已有的知识集推导出新的知识;
- 根据他所获得的知识伪造信息，并将其发送给可能接受的正规主体

3.4.4 模型形式化验证

在转换阶段，我们生成了在 Alice&Bob 规范中说明的 Murphi 安全协议模型，并且我们得到了生成代码文件 result.m。

Murphi 模型检查器将首先检查 Murphi 模型中是否存在错误。如果不是，Murphi 编译器转换结果, 执行编译得到 result.cpp。然后，我们使用 GNU 编译器编译 c++ 程序，这给了我们结果可执行文件，一个计算特定问题可达性的验证器。最后，执行这个可执行文件，对安全协议进行验证，最终得到验证结果。

Alice&Bob 语言规范和 Murphi 模型之间的对应关系如图3.5所示。首先，我们借助 Ocaml 实现的语法分析器提取 Alice&Bob 规范中的 Agents 块，以获取协议中的主体模板信息，并将其行为转换到扩展串空间中，以保证模型转换前后语义的一致性。接着将 Types 块和 Knowledges 块转换到 Murphi 模型的全局声明中，由 Knowledges 块和 Environment 块共同构成模型的初始化过程，而 Goals 块刻画关心的安全性质，将被转换到安全属性描述上。

3.5 安全协议自动分析验证机制

3.5.1 枚举量定义

借助 Murphi 的枚举 (enum) 来定义安全协议中涉及的枚举量，用于描述主体、随机数、常量、消息、密钥的种类和主体所处在的状态等。

主体枚举：AgentEnum 给出了协议运行中的所有主体实例。其中 NullAgent 表示主体未被设定值，此时该参量可以作为初始值；而 OtherAgent 是由 Alice&Bob 语言中涉及的主体生成而来的若干唯一标识。例如在 NSPK 协议中，额外生成的主体枚举有 Alice、Bob 和 Intruder。

$$\mathsf{AgentEnum} ::= \mathsf{NullAgent} \mid \mathsf{OtherAgent}$$

中国知网 https://www.cnki.net

随机数枚举：NonceEnum 给出了协议运行中的所有随机数取值。类似于主体枚举的定义，NullNonce 表示随机数未被设定值，此时该参量可以作为初始值，而 OtherEnum 是由 Alice&Bob 语言中涉及的随机数生成而来的若干唯一标识。例如在 NSPK 协议中，额外生成的随机数枚举有 Na 和 Nb。

$$\mathsf{NonceEnum} ::= \mathsf{NullNonce} \mid \mathsf{OtherNonce}$$

消息枚举：msgEnum 给出了所有能够支持的消息基本模式，能够对应于 Alice&Bob 规范中的消息定义。具体地，NullMsg 为未设定的消息类型；AgentMsg、NonceMsg、KeyMsg 和 NumMsg 分别表示通过主体、随机数、密钥和自然数构造的基本消息；ConcatMsg 表示通过打包操作构造的复合消息；AEncMsg、SEncMsg、SignMsg 和 HashMsg 分别表示通过非对称加密、对称加密、数字签名和哈希等密码学操作构造的复合消息；ModMsg 和 ExpMsg 表示通过取模操作和幂操作构造的复合消息，用于支持密码学中常见的数学运算；TmpMsg 表示临时消息，一般用于消息的转发，而不必关心能否和消息的模式匹配上。

$$\begin{array}{rcllll} \mathsf{MsgEnum} & ::= & \mathsf{NullMsg} & \mid \mathsf{AgentMsg} & \mid \mathsf{NonceMsg} \\ & \mid & \mathsf{KeyMsg} & \mid \mathsf{AEncMsg} & \mid \mathsf{SignMsg} \\ & \mid & \mathsf{ConcatMsg} & \mid \mathsf{HashMsg} & \mid \mathsf{TmpMsg} \\ & \mid & \mathsf{ModMsg} & \mid \mathsf{ExpMsg} & \mid \mathsf{NumMsg} \end{array}$$

密钥枚举：KeyEnum 给出了所有能够支持的密钥类型。具体地，包括公钥、私钥、对称密钥和以消息作为密钥四种枚举。

$$\mathsf{KeyEnum} ::= \mathsf{PK} \mid \mathsf{SK} \mid \mathsf{SymK} \mid \mathsf{MsgK}$$

主体状态枚举：为了记录每个主体在协议执行声明周期中所处在的状态，需要根据 Alice&Bob 语言中所定义的每个主体的状态标号来生成主体状态枚举，下式中 i 是主体模板的标号。例如，在 NSPK 协议中，主体模板 A 的状态取值包括 A_1、A_2 和 A_3，主体模板 B 的状态取值包括 B_1、B_2 和 B_3。

$$\mathsf{StatusEnum[i]} ::= \mathsf{Status\ of\ Agent\ i}$$

中国知网 https://www.cnki.net

3.5.2 结构化数据定义

借助 Murphi 的记录 (record) 机制来定义协议中的结构化数据，以详细刻画密钥、消息、通信信道、主体和消息集等。

密钥结构：KeyStruct 用于定义 Murphi 程序描述的密钥结构。其中，ktype : KeyEnum 为密钥的类型；ag 等；AgentEnum 分别描述非对称密钥的生成主体和对称密钥的会话双方; m : MsgIndex 表示作为消息密钥时的消息在全局消息表中的索引。

$$\mathsf{KeyStruct} ::= (\mathsf{ktype}, \mathsf{ag}, \mathsf{ag}_1, \mathsf{ag}_2, \mathsf{m})$$

消息结构: MsgStruct 用于定义 Murphi 程序所描述的消息结构。其中，mtype : MsgEnum 为消息的类型；当消息为主体生成的消息时，ag：AgentEnum 记录具体的主体实例；当消息为随机数消息时，nonce：NonceEnum 记录具体的随机数实例；当消息为临时转发用的消息时，tmp：MsgIndex 记录消息位置；当消息为密钥消息时，k：KeyEnum 记录具体的密钥；当消息为模消息时，mod_1、mod_2：MsgIndex 记录两个参量子消息的位置；当消息为幂消息时，e_1、e_2：MsgIndex 记录底数和指数两个子消息的位置；当消息为数字签名所构造的消息时，$sign_m$、$sign_k$：MsgIndex 记录签名内容和签名密钥两个子消息的位置；当消息为非对称加密所构造的消息时，$aenc_m$、$aenc_k$：MsgIndex 记录加密内容和加密公钥两个子消息的位置；当消息为对称加密所构造的消息时，$senc_m$、$senc_k$：MsgIndex 记录加密内容和对称密钥两个子消息的位置；当消息为哈希结果时，hash：MsgIndex 记录哈希前的原文消息的位置；当消息为装包构造的结果时，concat：MsgIndex[] 记录每个子消息的位置。

$$\begin{aligned}\mathsf{MsgStruct} ::= (&\mathsf{mtype},\ \mathsf{ag},\ \mathsf{nonce},\ \mathsf{tmp},\ \mathsf{k},\ \mathsf{mod}_1,\ \mathsf{mod}_2,\ \mathsf{exp}_1,\ \mathsf{exp}_2,\ \mathsf{sign}_\mathsf{m},\\ &\mathsf{sign}_\mathsf{k},\ \mathsf{aenc}_\mathsf{m},\ \mathsf{aenc}_\mathsf{k},\ \mathsf{senc}_\mathsf{m},\ \mathsf{senc}_\mathsf{k},\ \mathsf{hash},\ \mathsf{concat})\end{aligned}$$

通信信道结构：chanStruct 用于定义 Murphi 程序所描述的容量为 1 的通信信道的结构。其中 empty：bool 记录通道是否为空。如果通道不为空，则 msg：MsgStruct 记录通道中存放的消息；sender：AgentEnum 记录该消息的生产者主体实例；receiver：AgentEnum 记录该消息是发给哪个主体实例的。

$$\mathsf{ChanStruct} ::= (\mathsf{msg},\ \mathsf{sender},\ \mathsf{receiver},\ \mathsf{empty})$$

中国知网 https://www.cnki.net

主体结构：RoleStruct 将根据建模阶段采用 Alice&Bob 语言所描述的每个主体模板自动生成，下式中 i 是主体模板的标号。其中 st：StatusEnum[i] 记录主体 i 当前所在的状态，commit：bool 记录该主体是否已经执行完毕，vars 和 bufvars 分别表示实际知识和临时接收的知识两类属性列表。例如，在 NSPK 协议的生成结果中，vars=(N_a, N_b, A, B) 记录实体对这四个基本变量的取值，而 bufvars = ($bufN_a$, $bufN_b$, $bufA$, $bufB$) 记录实体接收消息后解构的临时结果，可以在获取消息并解构后和 vars 中的对端变量比对以获取消息子项的匹配结果。

$$\mathsf{RoleStruct[i]} ::= (\mathsf{vars},\ \mathsf{bufvars},\ \mathsf{st},\ \mathsf{commit})$$

模式集结构：消息的模式可根据其构造方式和位置参量唯一确定，例如 aenc Nonce,Agentpk(Agent) 即为一种消息模式，它表示先将一个取自 NonceEnum 的随机数消息和取自 AgentEnum 的主体消息打包，然后使用根据某个 AgentEnum 生成的公钥进行非对称加密所得的消息。模式集结构 PatSetStruct[j] 则用于记录同一模式 j 的消息集合，逻辑上 PatSetStruct[j] 是一种变长列表；可以借助 Murphi 数组实现。其中 content：MsgIndex[] 记录模式集中的所有消息 (在全局消息表中的索引)；length：int 记录模式集中的元素数目。

$$\mathsf{PatSetStruct[j]} ::= (\mathsf{content},\ \mathsf{length})$$

3.5.3 协议运行规则模型

如果通过前述的结构化数据来描述协议的全局状态，则在 Murphi 中的自动分析和验证可以初始化阶段：初始化阶段的目的是将上述的结构化数据表置为一个正确的初态，并清空用于描述全局状态的各类集合，以准备协议后续的模拟执行。该阶段首先对每个主体模板 i 下的主体实例 j 的角色信息 role[i][j] 按照 Alice&Bob 语言中 Environment 块中的描述进行实例化，然后将描述全局状态的 $chans$、$emits$、$msgs$、$spats$、$rpats$、$knowns$ 和 $spyknowns$ 集合全部清空，此时得到的是一个仅存在主体而不存在消息的协议模型。对于 Alice&Bob 语言中所描述的主体之间直接传递的每条消息，采用递归构造的方式将该消息的所有子消息和模式构造出来，加入到相应的模式集 spats[p] 和知识集 knows[i][j] 中 (对于攻击者主体，这个知识集就是 spyknowns)。另外，对于每个主体，还需要添加其自身的私钥知识。初始化阶段完成后，每个主体的信息被正确例化，全局消息

中国知网 https://www.cnki.net

表 $msgs$ 中保存了协议运行中将会传递的所有消息，模式集中记录了所有的消息模式，每个主体的知识集被设定为初始知识，信道 $chans$、攻击状况 $emits$ 等状态量都被清空。

运行阶段：在运行阶段，对每个正规主体在每个状态位置的操作生成的条件-动作随机执行。合规主体在执行发送动作前需要检查卫条件，即确认相应的信道没有填充消息 (chans 中的对应项为空)，并且该主体的运行未完成 (未处于已提交状态)。发送动作则是先递归地构造要发送的消息项，构造过程中如果出现在全局消息表中不存在的消息，则要加入到消息表中，然后将构造好的消息填充到这次发送动作对应的信道中去，并将发送方主体的状态前进一步，标识发送动作已经执行完成。

正规主体在执行接收动作前也需要检查接收信道不为空，并且该主体的运行未完成，并且该主体在当前状态所期望接收的消息模式和信道中的消息模式能够正确地的匹配上。对于非转发形式的对称加密消息和非对称加密消息的接收还需要做特殊的处理：如果是非对称加密消息，则要求加密公钥的分发主体是自己，这样才能使用相应的配对私钥正确完成解密操作；如果是对称加密，则要求本主体是对称会话双方之一。在上述条件都满足的情况下，将消息的解构，并要求每个子消息能够正确匹配。对于本地曾经没有的消息，这个解构和匹配过程即会扩充主体自身的知识集。在接收操作完成后，从消费的信道中清空消息，然后将接收方主体的状态前进一步，标识接收工作已经执行完成。

攻击者主体的动作可以分为 Get、Emit、DeConcat、EnConcat、ADecrypt、AEncrypt、SDecrypt 和 SEncrypt 八种，分别表示从信道上获取信息、发送伪造信息到信道上、消息解包、消息装包、非对称解密、非对称加密、对称解密和对称加密。

(1) Get 操作：从正规主体交互的信道上窃取消息。由于该操作只关注窃取消息，而不关注解密等消息拆解操作，所以每次窃取到特定模式的消息后，只需要扩充该模式的模式集，以及扩充自己的知识集 (如果不在自己的知识集中) 即可，在窃取完成后将消息从信道上取走。

(2) Emit 操作：向正规主体交互的信道上发送消息。当正规主体处在某个接收状态，且所监听的信道中还没有消息填充时，从自己的知识集中寻找相应模

中国知网 https://www.cnki.net

式的消息填充到该信道中，并记录发送该消息的攻击已经完成 (用于去重)。Emit 操作仅关心将知识集中的消息在特定的场景发送出去以尝试攻击，而不关心消息的构造过程。

(3)DeConcat 操作：对特定模式的消息进行拆解。首先检查是否拆解成功，如果成功，拆解后的消息至少有一个新知识 (否则没有拆解的必要)，然后对消息模式进行匹配，并用拆解得到的消息扩充知识集。

(4) EnConcat 操作：对特定模式的消息进行构造消息。从知识集中抽取构造消息所需要的特定模式的知识，检查如果构造消息成功是否会产生新知识 (否则没有装包的必要)，然后构造消息并获取新消息，最后扩充知识集。

(5) ADecrypt 操作：对特定模式的非对称加密消息进行解密，检查知识集中具备相应的私钥即可解密成功，并扩充自己的知识集。

(6) AEncrypt 操作：对特定模式的消息进行对称加密，检查知识集中具备相应的公钥即可加密成功，并扩充自己的知识集。

(7) SDecrypt 操作：对特定模式的对称加密消息进行解密，检查知识集中具备相应的对称密钥即可解密成功，并扩充自己的知识集。

(8) SEncrypt 操作：对特定模式的消息进行对称加密，检查知识集中具备相应的对称密钥即可，并扩充自己的知识集。

我们根据 Alice&Bob 规范所描述的安全协议生成了前述的 Murphi 模型，通过枚举量、结构化数据、正规主体规则和攻击者主体规则描述了协议的运行以及可能遭受的攻击模式，并将结果输出到文件 result.m。在验证时，Murphi 模型检查器会首先检查模型是否存在语法错误，如果检查通过，Murphi 编译器会将 rusult.m 翻译成一个 C++ 程序 result.cpp。最后，我们使用 GNU g++ 编译器编译这个 C++ 程序，并得到一个可执行文件，即一个为特定问题计算可达性的验证器。执行这个程序即可开始验证安全协议，验证结果将会输出到终端。

3.6 本章小结

本章主要介绍了安全协议的模型检测方法和 Murphi 中的安全协议自动验证机制。首先从安全协议的形式化描述出发，通过扩展的 Alice&Bob 语言规范刻画协议的运行过程和安全性质，接着由扩展的串空间及其操作语义将其协议的

中国知网 https://www.cnki.net

运行过程结构化、模块化，保证从高层次的 Alice&Bob 描述的安全协议能够保持语义的一致性，最后转换到 Murphi 模型验证器中进行形式化验证分析，实现安全协议的模型检测自动化。我们提出了 AnB2Murphi 的模型转换器，将整个流程自动化，极大地方便行业专家和非专业学者对安全协议的理解和设计。

中国知网 https://www.cnki.net

第四章　案例研究与分析

4.1　5G EAP-TLS 协议分析

EAP-TLS 是 5G 协议标准下的一种相互认证协议，通过客户端和服务器双方互相验证数字证书来完成认证操作。如图4.2所示在拆解和建模阶段，我们使用 Alice&Bob 语言描绘了四个正规主体 UE、$SEAF$、$AUSF$ 和 UDM 的理想交互方式。首先，在 Knowledge 块中定义了各个主体的初始知识。例如主体 A 的初始知识有 A、B、C、D 四个主体的身份标识信息，以及 $supi$、ue、$ue1$、$prekey$、$certA$ 和 $eapm$ 分别表示用户身份标识符 (Subscription Permanent Identifier) 实体、用户设备 (User Equipment) 实体、UE 生成的随机数、预先掌握的密钥 (Pre-Master Key)、认证服务器 (Authentication Server Function) 的证书、认证消息。在 Agents 块中定义了各个主体发送和接收消息的通信动作。Goals 块中定义了 EAL-TLS 协议所需满足的安全性质。在 EAP-TLS 协议中我们旨在验证 $prekey$ 的私密性，即 $prekey$ 不会被恶意的攻击者所窃取掌握，还验证正规主体 $AUSF$ 与 UE 在随机数 $ausf$ 上达成一致，即两代理主体之间的弱一致性。最后，在 Enviroment 块中定义了协议实际运行的主体实例，如 A 的实例为 UE，且 UE 与 $SEAF$、$Intruer$、UDM 进行通信。如图4.1所示，为 5G 网络的通信框架图。

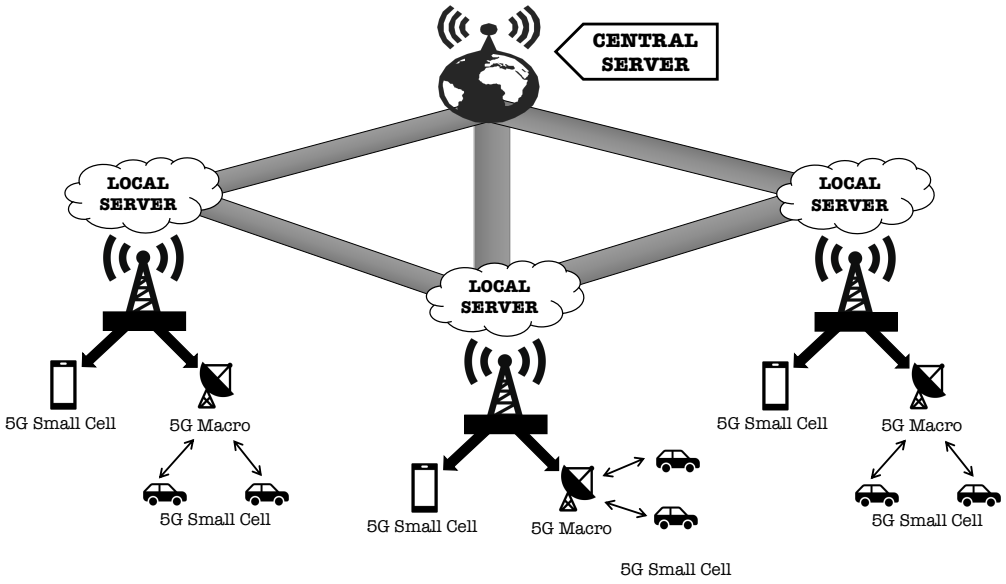


图 4.1 5G 网络框架示意图

中国知网 https://www.cnki.net

Protocol EAP-TLS:

Knowledge:(* Intitial Knowledge*)

$A: A, B, C, D.supi, ue, ue1, prekey, certA, eapm$

$B: A, B, C, seafn$

$C: A, B, C, D, ausf, ausfn, sucm, certC$

$D: A, B, C, D, start$

Agents:

UE(A,B,C,D,ue,ue1,prekey,certA,eapm)

$[1]+, B, (supi, ue) : \{|supi, ue|\}^{a}_{PK(D)}$

$[6]- : start$

$[7]+, B, (ue1) : ue1$

$[10]- : ausf, certC, seafn$

$[11]+, B, (prekey, certA)$:$\{|prekey|\}^{a}_{PK(C)}, certA, \{|start, ue1, ausf, certC|\}^{sign}_{sk(A)}, \{|start,$
$ue1, ausf, certC|\}^{s}_{h(ue1,ausf,prekey)}$

$[14]- : H8$

$[15]+, B, (eapm) : eapm$

$[18]- : sucm$

SEAF(A,B,C,seafn)

$[1]- : H1$

$[2]+, C, (seafn) : H1, seafn$

$[5]- : H2$

$[6]+, A, () : H2$

$[7]- : H3$

$[8]+, C, () : H3, seafn$

$[9]- : ausf, certC.ausfn$

$[10]+, A, () : ausf, certC, seafn$

$[12]+, C, () : H4, H5, H6, H7$

$[11]- : H4, H5, H6, H7$

$[15]- : eapm$

$[13]- : H8$

$[16]+, C, () : eapm$

$[14]+, A, () : H8$

$[17]- : sucm$

$[18]+, A, () : sucm$

AUSF(B,C,D,ausf,ausfn,sucm,certC)

$[2]- : H1, seafn$

$[3]+, D, () : H9$

$[4]- : start$

$[5]+, B, () : start$

$[8]- : ue1, seafn$

$[9]+, B, (ausf, certC) : ausf, certC, ausfn$

$[12]- : H4, H5, H6, H7$

$[13]+, B, ()$:$\{|start|\}_{\{h(start, ue1, prekey)\}^{s}}$

$[16]- : eapm$

$[17]+, B, (sucm) : sucm$

UDM(A,B,C,D,start)

$[3]- : H1$

$[4]+, C, (start) : start$

Goals: (*Security Goals*)

[secrecy] $prekey$ secret of <A, B, C, D>

[weakC] C non − injectively agrees with A on $ausf$

Environments: (*Protocol instance*)

$[str_1]UE[1]$:<$UE, SEAF, Intruder, UDM.supi, ue, ue1, prekey, certA, eapm$>

$[str_2]SEAF[1]$:<$UE, SEAF, Intruder, seafn >$

$[str_3]AUSF[1]$:<$UE, Intruder, AUSF, UDM, ausf, ausfn, sucm, certC >$

$[str_4]UDM[1]$:<$UE, SEAF, AUSF, UDM, start >$

end

图 4.2 5G EAP-TLS 认证协议在串空间规约中的具体实现

中国知网 https://www.cnki.net

4.2 Murphi 协议模型的生成

基于前述的 Alice&Bob 规范所描述的协议模型，我们使用 AnB2Murphi 将其转换生成 Murphi 协议模型。基于我们的方法论转换生成的 Murphi 模型总体结构包含以下几个部分，如表4.1所示：

全局声明	迁移规则	安全属性	初始化	函数/过程
消息定义 主体定义 信道定义	合规主体行为规则 入侵者主体行为规则	私密性 弱一致性 一致性	初始主体 初始模式 初始知识 构造模式	消息匹配 获取索引 模式判断 密钥反转 解构消息

表 4.1 Murphi 模型总体结构

全局声明主要包括消息定义、主体定义和信道定义三个部分。消息定义部分将定义消息类型和消息结构。其中消息类型通过采用枚举量 MsgType 来标识。在 EAP-TLS 协议中，消息类型枚举定义为:

$$\mathsf{MsgType} : \mathsf{enum}\{\mathsf{null}, \mathsf{agent}, \mathsf{nonce}, \mathsf{key}, \mathsf{aenc}, \mathsf{senc}, \mathsf{sign}, \mathsf{concat}, \mathsf{hash}, \mathsf{tmp}, \mathsf{mod}, \mathsf{exp}, \mathsf{number}\}$$

消息结构采用 Murphi 内置的 record 关键字来描述，如图4.3所示。其中，MsgType 为上述消息类型，AgentType 为协议生成的主体类型，NonceType 为随机数类型，KeyType 为密钥类型，而 IndexType 为整数型索引，用于描述消息在全局消息表中的索引位置。

主体定义包括定义主体的类型 AgentType、主体的状态、主体的知识库和主体的结构。5G EAP-TLS 协议中的主体类型定义为

$$\mathsf{AgentType} : \mathsf{enum}\{\mathsf{anyAgent}, \mathsf{UE}, \mathsf{SEAF}, \mathsf{UDM}, \mathsf{AUSF}, \mathsf{Intruder}\}$$

其中 UE 表示为用户设备，$SEAF$ 为服务网络关键模块，UDM 为统一数

中国知网 https://www.cnki.net

```
Message: record
msgType: MsgType;
noncePart: NonceType;
tmpPart: IndexType;
modMsg1: IndexType;
hashMsg: IndexType;
expMsg2: IndexType;
signKey: IndexType;
aencKey: IndexType;
sencKey: IndexType;
concatMsg2: IndexType;
end;
ag: AgentType;
constPart: ConstType;
k: KeyType;
modMsg2: IndexType;
expMsg1: IndexType;
signMsg: IndexType;
aencMsg: IndexType;
sencMsg: IndexType;
concatMsg1: IndexType;
length: IndexType;
```

图 4.3 Murphi 模型消息结构定义

据管理，$AUSF$ 为认证服务功能模块。由于该协议的通信过程较复杂, 因此主体的状态数量比 Needham-Schoreder 协议更多。例如，UE 的状态枚举表示为 $\mathsf{A}_{\mathsf{status}}:\mathsf{enum}\{\mathsf{A}_1,\mathsf{A}_2,\mathsf{A}_3,\mathsf{A}_4,\mathsf{A}_5,\mathsf{A}_6,\mathsf{A}_7,\mathsf{A}_8\}$，枚举数组中的元素表示当前主体所处的状态位置。协议的每个主体都会维护自身的知识集合 $\mathsf{A}_{\mathsf{known}}$: Array[indexType] of boolean，用于保存当前所拥有的知识。最后，主体的结构与消息的结构定义类似，其中的一部分如图4.4所示。

其中，以 loc 开头的部分定义了主体本地的存量知识，而不以 loc 开头的相应属性则用于接收外部知识。在初始状态下，所有的知识都被赋值为未定义值，例如 $locB$ 的初始值为 anyAgent。

信道定义描述信道的结构, 如图4.5所示, 信道 Channel 包括其中的消息 msg、发送者 $sender$、接收接受者 $receiver$ 和空标志 $empty$，正规主体和攻击者主体的发送和接收动作都借助信道来进行。

迁移规则是 Murphi 模型实现状态转移的主要部分，主要由正规主体的行为规则和入侵者的行为规则组成，当全局状态局面满足迁移规则的条件时，规则的动作将会执行。

正规主体的行为规则仅由直观的发送和接收规则组成。其中，发送消息规则对应于协议模型中的符号为 + 的动作。图4.6给出了一个正规主体发送消息的示

中国知网 https://www.cnki.net

```
RoleA: record
supi: NonceType;
ue: NonceType;
prekey: NonceType;
certA: NonceType;
eapm: NonceType;
seafn: NonceType;
ausf: NonceType;
sucm: NonceType;
certC: NonceType;
start: NonceType;
A: AgentType;
B: AgentType;
C: AgentType;
D: AgentType;
x2: Message;
st: AStatus;
locsupi: NonceType;
locue: NonceType;
locprekey: NonceType;
loccertA: NonceType;
loceapm: NonceType;
locseafn: NonceType;
locausf: NonceType;
locsucm: NonceType;
loccertC: NonceType;
locstart: NonceType;
locA: AgentType;
locB: AgentType;
locC: AgentType;
locD: AgentType;
locx2: Message;
commit: boolean;
end;
```

图 4.4 Murphi 模型主体结构定义

```
Channel: record
  msg : Message;
  sender : AgentType;
  receiver : AgentType;
  empty : boolean;
end;
```

图 4.5 Murphi 模型信道结构定义

例，卫士条件被设定。该迁移规则的卫条件是：主体主体 A 所处在处的状态是 A_1 状态，相应信道为的空标志为真，并且主体 A 尚未完成所有动作 (未提交)。在执行动作中，$clear\ msg$ 表示将变量 msg 清空，$cons4$ 表示构造模式为 $pat4$ 的消息并赋值给变量 msg。接着改变信道的状态，由于该示例是发送动作，因此信道的空标志被清除，存放的消息变为刚刚构造的 msg，消息的发送者为主体 A

中国知网 https://www.cnki.net

的实例 A，接收接受者为主体 $Intruder$，最后主体 A 的执行状态更新为 A_2。

```
rule " roleA1 "
roleA[i].st = A1 & ch[1].empty = true & !roleA[i].commit
==>
var msg:Message;
    msgNo:indexType;
begin
  clear msg;
  cons4(roleA[i].supi,roleA[i].ue,roleA[i].D,msg,msgNo);
  ch[1].empty := false;
  ch[1].msg := msg;
  ch[1].sender := roleA[i].A;
  ch[1].receiver := Intruder;
  roleA[i].st := A2;
end;
```

图 4.6 Murphi 模型正规主体发送消息结构定义

图4.7所示为正规主体接受消息。在卫士条件中，函数 judge 用于判断收到的消息的接收方是否为指定主体。接收接受到消息之后，首先进行模式判断，通过函数 isPatX 将匹配编号为 X 的模式的结果返回给变量 $flag_{patX}$。如果模式相匹配，则将收到的消息进行解构，函数 destructX 用于对 msg 拆分，赋值到主体的本地知识库中。随后通过函数 match 进行匹配，如果匹配成功，则清空信道缓冲区，并更新主体的执行状态。

入侵者的行为规则是基于 Dolev-Yao 理论来设计的，入侵者将时刻监视信道上的消息，不断截获信道上传输的消息并加入到自身的知识库中，并能够根据自身所拥有的知识对知识库中的事实进行密码学操作，如分离连结消息、构造哈希消息、解密加密密文等。此外，入侵者可以将与信道相匹配的伪造消息恶意发送给正规主体。

图4.8展示了 EAP-TLS 协议中一个入侵者从信道上截获消息的例子。首先，入侵者判断信道上的消息的发送者是否不是自己，接着获取所截获的消息在全局消息表上的索引值，将所截获消息的 $tmppart$(表示临时值) 赋值索引值, 最终

中国知网 https://www.cnki.net

```
rule " roleA2 "
roleA[i].st = A2 & ch[6].empty = false & !roleA[i].commit &
judge(ch[6].msg,roleA[i].A,msgs[0])
==>
var flag_pat2:boolean;
     msg:Message;
begin
  clear msg;
  msg := ch[6].msg;
  isPat2(msg, flag_pat2);
  if(flag_pat2) then
    destruct2(msg,roleA[i].locstart,roleA[i].locseafn);
    if(matchNonce(roleA[i].locstart, roleA[i].start) &
      matchNonce(roleA[i].locseafn, roleA[i].seafn)) then
      ch[6].empty:=true;
      clear ch[6].msg;
      roleA[i].st := A3;
    endif;
  endif;
end;
```

图 4.7 Murphi 模型正规主体接受消息结构定义

判断该消息是否已经在入侵者的模式集上，如果不存在，则将其加入到模式集中，并将 Spy_known 数组中对应索引位为真，表示该消息已经加入到入侵者知识库中，最后清空信道缓冲区。

入侵者截获到消息以后，可以对消息进行一些密码学操作，进行二次加工，伪造出新消息入侵者截获到消息之后，可以对消息进行密码学操作，将截获的消息进行再次加工，构造出新的伪造的消息，再用伪造消息实现恶意攻击从而达到恶意攻击的目的。图4.9展示了入侵者构造连结消息的例子，首先卫条件中主体的状态表明当前所处的状态条件下要求入侵者拥有 $patSet1$ 的消息，并且可以通过两条相同模式的消息进行连结操作，构造出 $patSet2$ 的新连结消息，同时入侵者的知识库中没有新消息的记录，则通过 constrctIndex 函数首先生成新消息的索

中国知网 https://www.cnki.net

```
rule "intruderGetMsgFromCh[1]"
ch[1].empty = false & ch[1].sender != Intruder ==>
var flag_pat4:boolean;
     msgNo:indexType;
     msg:Message;
begin
  msg := ch[1].msg;
  get_msgNo(msg, msgNo);
  msg.tmpPart := msgNo;
  isPat4(msg,flag_pat4);
  if (flag_pat4) then
    if(!exist(pat4Set,msgNo)) then
      pat4Set.length:=pat4Set.length+1;
      pat4Set.content[pat4Set.length]:=msgNo;
      Spy_known[msgNo] := true;
    endif;
    ch[1].empty := true;
    clear ch[1].msg;
  endif;
end;
```

图 4.8 Murphi 模型恶意攻击者接受消息结构定义

引，如果索引的值大于全局消息数据库的长度，通过构造函数 construct 进行构造消息，并将消息加入到入侵者知识库中。最后，所构造的新的连结消息所在处于的模式集长度长度加一。

入侵者发送消息行为的例子如图4.10所示，因为每个信道对应一种模式的消息，所以入侵者需要搜寻自身知识库中是否存在这种消息，如果存在则试图将该消息发送给可接收的正规主体。示例中卫条件中的 IntruEmit 描述入侵者是否在上一条信道上发送过消息，这是一条用于约简全局状态搜索空间的剪枝策略，以尽量避免状态爆炸问题该策略用以缩减全局的状态空间，防止产生状态爆炸。另外还需要判断 $patSet$ 中消息是否发送过以及入侵者是否拥有该消息；如果为真，则将该消息发送到对应的信道上，并将该消息的发送标志设为真。

中国知网 https://www.cnki.net

```
ruleset i1: msgLen do
ruleset i2: msgLen do
ruleset i: roleBNums do
rule "enconcat 2"
roleB[i].st = B1 & i1<=pat1Set.length & Spy_known[pat1Set.content[i1]] &
i2<=pat1Set.length & Spy_known[pat1Set.content[i2]] & matchPat(construct2
By11(pat1Set.content[i1],pat1Set.content[i2]), sPat2Set)& !Spy_known[con
structIndex2By11(pat1Set.content[i1],pat1Set.content[i2])]
==>
var concatMsgNo:indexType;
    concatMsg:Message;
begin
  concatMsgNo:=constructIndex2By11(pat1Set.content[i1],pat1Set.content[i2]);
  if concatMsgNo = msg_end + 1 then
  msg_end :=msg_end + 1;
  concatMsg:= construct2By11(pat1Set.content[i1],pat1Set.content[i2]);
  msgs[concatMsgNo] := concatMsg;
  endif;
  Spy_known[concatMsgNo]:=true;
  if (!exist(pat2Set,concatMsgNo)) then
    pat2Set.length:=pat2Set.length+1;
    pat2Set.content[pat2Set.length]:=concatMsgNo;
  endif;
end;
endruleset;
endruleset;
endruleset;
```

图 4.9 Murphi 模型恶意攻击者连结消息结构定义

我们的理论和工具能够对弱一致性、一致性和私密性三种安全属性进行验证。在 5G EAP-TLS 协议描述中, 我们定义随机数 $prekey$ 的私密性和主体 $AUSF$ 和 UE 关于随机数 $ausf$ 的弱一致性目标。如图4.11所示, 对于不变式 $weakC$，如果所有主体 A 所处状态 A_6 下，当且仅当存在主体 C 的本地知识 $ausf$ 与主体 A

中国知网 https://www.cnki.net

```
ruleset i: msgLen do
ruleset j: roleANums do
rule "intruderEmitMsgIntoCh[6]"
IntruEmit5=true & roleA[j].st=A2&ch[6].empty=true&i<=pat17Set.length& pat17
Set.content[i] != 0 & Spy_known[pat17Set.content[i]&!emit[pat17Set.content[i]]
==>
begin
  clear ch[6];
  ch[6].msg:=msgs[pat17Set.content[i]];
  ch[6].sender:=Intruder;
  ch[6].receiver:=roleA[j].A;
  ch[6].empty:=false;
  emit[pat17Set.content[i]] := true;
  IntruEmit6 := true;
end;
endruleset;
endruleset;
```

图 4.10 Murphi 模型恶意攻击者发送消息结构定义

的本地知识 $ausf$ 一致，则该不变式成立。对于不变式 $secrecy$ 来说，全局消息数组 $msgs$ 的索引下标所对应的数据类型为随机数，并且随机数部分为 $prekey$，那么入侵者不能拥有获取该索引处记录的所对应的消息。

在初始化模块中，我们根据 Alice&Bob 模型中 Knowledge 块的定义来初始化各个正规主体和入侵者的知识库，根据所有的消息来构造消息的模式种类，并为对每个模式构建模式集，用于后续的消息匹配操作。

主体实例化：根据 Alice&Bob 模型中的 Environment 块来对主体进行参数实例化，例如，主体 A 的实例化过程如图4.12所示。在 EAP-TLS 协议中, 正规主体模板 A 被实例化为 UE、B 被实例化为 $SEAF$、C 被实例化为 $Intruder$。值得注意的是，当消息类型为 tmp 时，表示该消息可以被替代为任何消息，$tmpPart$ 用于表示实际存放的消息索引，因此要被初始化为 0，表示在初始时未存放任何消息。

中国知网 https://www.cnki.net

```
invariant "weakC"
forall i: roleANums do
  roleA[i].st = A6 ->
  (exists j: roleCNums do
  roleC[i].ausf = roleA[j].ausf
  endexists)
endforall;
invariant "secrecy"
  forall i:indexType do
  (msgs[i].msgType=nonce & msgs[i].noncePart=prekey)
  ->
  Spy_known[i] = false
```

图 4.11 Murphi 模型弱认证性结构定义

```
roleA[1].A := UE;                    roleA[1].ausfn := anyNonce;
roleA[1].B := SEAF;                  roleA[1].sucm := anyNonce;
roleA[1].C := Intruder;              roleA[1].certC := anyNonce;
roleA[1].D := UDM;                   roleA[1].start := anyNonce;
roleA[1].supi := supi;               roleA[1].x10.msgType := tmp;
roleA[1].ue := ue;                   roleA[1].x10.tmpPart := 0;
roleA[1].ue1 := ue1;                 roleA[1].m1.msgType := tmp;
roleA[1].prekey := prekey;           roleA[1].m1.tmpPart := 0;
roleA[1].certA := certA;             roleA[1].x2.msgType := tmp;
roleA[1].eapm := eapm;               roleA[1].x2.tmpPart := 0;
roleA[1].st := A1;                   roleA[1].x3.msgType := tmp;
roleA[1].commit := false;            roleA[1].x3.tmpPart := 0;
roleA[1].seafn := anyNonce;          roleA[1].x1.msgType := tmp;
roleA[1].ausf := anyNonce;           roleA[1].x1.tmpPart := 0;
```

图 4.12 Murphi 模型代理初始化结构定义

模式集初始化：模式集有两类。模式集 patSet 用于描述所有消息的模式，而 sPatSet 则用于保存接收接受消息的模式。在该过程中，将模式集的 *content* 和长度 *length* 都设置为 0，如图4.13所示。

中国知网 https://www.cnki.net

```
for i : indexType do
pat1Set.content[i] := 0;
sPat1Set.content[i] := 0;
pat2Set.content[i] := 0;
sPat2Set.content[i] := 0;
pat3Set.content[i] := 0;
sPat3Set.content[i] := 0;
pat4Set.content[i] := 0;
sPat4Set.content[i] := 0;
pat5Set.content[i] := 0;
sPat5Set.content[i] := 0;
pat6Set.content[i] := 0;
sPat6Set.content[i] := 0;
endfor;
pat1Set.length := 0;
sPat1Set.length := 0;
pat2Set.length := 0;
sPat2Set.length := 0;
pat3Set.length := 0;
sPat3Set.length := 0;
pat4Set.length := 0;
sPat4Set.length := 0;
pat5Set.length := 0;
sPat5Set.length := 0;
pat6Set.length := 0;
sPat6Set.length := 0;
```

图 4.13 Murphi 模型模式集初始化结构定义

知识初始化：初始知识的生成。其主要关注正规主体的公私钥知识，以及入侵者在协议初始状态上所拥有的知识。如图4.14所示，对于所有的主体 D 来说，首先扩充扩大 msg_end(全局消息数据数组 msgs 的尾指针 msg_end)。随后，将该索引所对应的消息类型设为 key，表示该消息是一个密钥。密钥的生成方为主体 D 自身，加密类型为公钥加密，消息的长度为 1。另外公钥消息在该模型中所属模式 pat3Set，因此该模式集的长度加一。最后，由于公钥对所有主体都是公开的，所以入侵者也能获取到公钥消息，因此将其加入到入侵者的知识库中最后由于公钥对所有实体都公开，因为入侵者获知该消息，将其加入到入侵者知识库中。

模式构造: 该过程针对于模式 $sPatSet$，对于协议描述中所有接收动作中的消息构造一个对应的消息匹配集合。在入侵者收到消息，可对其进行模式匹配判断是否为信道所需的消息。如图4.15所示，主体 B 在协议运行过程中，等待接收接受模式为 $pat4$ 的消息，则在初始化过程中，通过调用 constructSpat4 事先构造 $sPat4Set$ 模式。

该部分用于生成 Murphi 模型中调用的一系列固有的函数或过程。过程 GetMs-

中国知网 https://www.cnki.net

```
for i : roleDNums do
  msg_end := msg_end+1;
  msgs[msg_end].msgType := key;
  msgs[msg_end].k.ag := roleD[i].D;
  msgs[msg_end].k.encType:=PK;
  msgs[msg_end].length := 1;
  pat3Set.length := pat3Set.length + 1;
  pat3Set.content[pat3Set.length] :=msg_end;
  Spy_known[msg_end] := true;
  D_known[msg_end] := true;
endfor;
```

图 4.14 Murphi 模型代理消息集结构定义

```
for i : roleBNums do
  constructSpat4(roleB[i].supi,roleB[i].ue,roleB[i].D, gnum);
endfor;
```

图 4.15 Murphi 模型构造模式集结构定义

gNo() 用于搜索 msg 在全局 $msgs$ 中的索引位置；PrintMsg() 将消息输出到控制台中；InverseKey() 将消息进行取反操作；Judge() 用于正规主体判断接收接受到的消息是否能解开；ConstructPat() 用于构造消息的模式；LookPat() 用于查找消息的模式是否存在；IsPat() 用于判断消息的模式是否一致; Destruct() 用于将组合的消息分解开; Cons() 用于构造消息; MatchMsg() 用于匹配主体、随机数、自然数或临时消息；Exist() 用于判断消息是否存在模式集合中。

4.3 Murphi 协议的验证分析

通过运行模型，我们发现了对这两个属性的攻击。并且我们在图中4.16列出了弱一致属性的反例。通过协议的运行，将用对手的哈希密钥签名的消息发送到终端。UE 无法分辨最后的消息是来自对手还是来自 $SEAF$。因此，对手在此执行结束时成功拦截了 UE 的机密性。

类似地，对私有属性的攻击如图4.17所示。这个隐私属性要求对手不能解密

中国知网 https://www.cnki.net

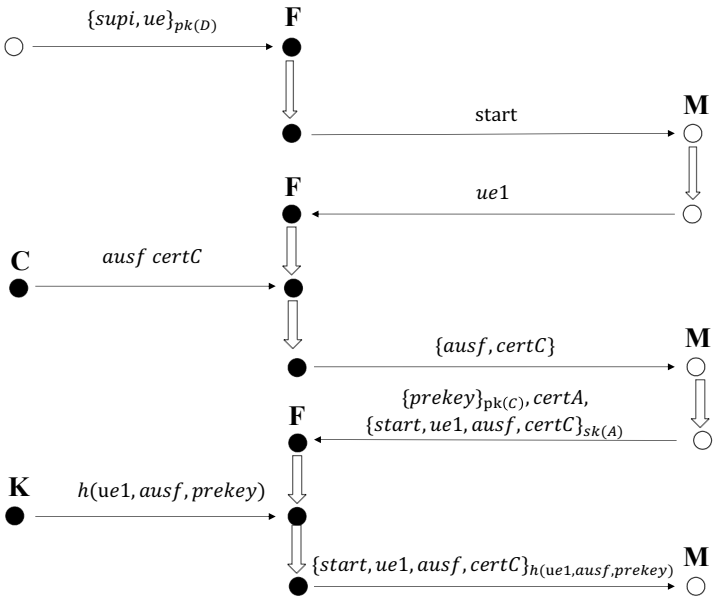


图 4.16 弱一致性反例串分析

预主密钥 $prekey$。通过典型的中间人攻击，属性被侵犯了。一开始，攻击只是通过 Flushing F 和 Text M 截获并转发消息 $\{|supi,ue|\}_{PK(D)}$。在几轮消息交换后，UE 发送一个加密消息 $\{|prekey|\}_{PK(C)}$ 到公共通道，然后对手使用其已知的密钥解密消息，通过密钥 K 获得预主密钥 $prekey$。

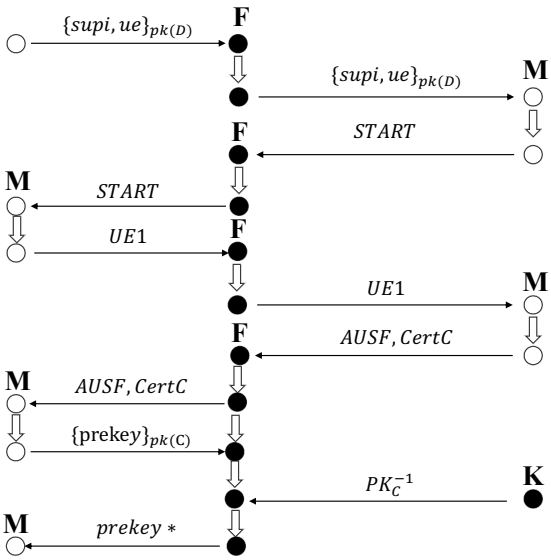


图 4.17 Prekey 的私密性反例串分析

中国知网 https://www.cnki.net

4.4 安全协议验证结果分析

除了 5G EAP-TLS 认证协议以外, 我们还在其它的安全协议上进行了建模验证的尝试。这些实验在一台配备 MacOS Catalina 和 Intel Core i7 (2.6Ghz) 的 PC 上进行，内存为 16GB。这些安全协议的验证结果如所示，关于 AnB2Murphi 的源代码可以在 GitHub 仓库[54] 访问。

表 4.2 实验结果分析

安全协议	安全属性	验证结果	运行时间 (s)
Needham – Schroder	secrecy(Nb)	False	0.10
	weakB(Na)	False	
Lowe's Needham – Schroder	secrecy(Nb)	True	0.10
	weakB(Na)	True	
Diffie–Hellman key exchange	secrecy(Na)	False	0.10
Otway–Rees	secrecy(Kab)	True	2.13
CCITT X.509(1)	secrecy(Ya)	False	0.21
	weakB(Xa)	False	0.84
	weakA(Ya)	False	
CCITT X.509(1c)	weakB(Xa)	True	0.45
	weakA(Ya)	True	
Woo and Lam Pi	secrecy(Nb)	False	0.10
Andrew Secure RPC	secrecy(Kab)	False	2.77
EAP – TLS authentication	secrecy(prekey)	False	1.21
	weakC	False	51.55
EAP – AKA authentication	secrecy(snn)	True	1.50

中国知网 https://www.cnki.net

4.5 本章小节

本章通过模型检测实验出发，对本文所述方法进行了介绍，其中通过 AnB2Murphi 工具验证了 5G EAP-TLS 认证协议，从利用 Alice&Bob 语言规范对协议进行描述，到生成的 Murphi 协议模型，其中在 Murphi 协议模型的每个模块进行阐述，说明其在模型检测其中所起到的作用，对协议的私密性和认证性进行了验证，并给出了反例路径分析，最后给出模型验证对其他安全协议的验证结果。

中国知网 https://www.cnki.net

第五章　总结与展望

5.1　工作总结

在本文中，我们从模型检测分析安全协议，其中两种方法具有共同点，即都从高层的 Alice&Bob 语言规范出发，其中模型检测规范化了 Alice&Bob 安全协议描述语言范式，给出了一套明确的语法和建模规范。然后基于 Alice&Bob 语言规范和 Murphi 模型检测器构建了一套安全协议的建模、转换和验证方法。在这种方法中首先将 Alice&Bob 语言描述的协议模型转换到扩展了操作语义的串空间模型上，以该模型作为中间表示以及转换链路上的跳板，能够很好地描述主体之间的通信关系和它们自身的状态转换，并将该语义对齐到 Murphi 模型上去。我们给出了一种基于 Murphi 模型的安全协议建模和自动分析验证机制，在 Dolev-Yao 模型的基础上，构造了入侵者的推理规则，可以模拟不安全网络中可能发生的攻击，借此将串空间模型中的协议理想运行描述与攻击者行为相结合，从而迁移到基于 Murphi 的形式化描述中。基于上述理论，我们采用 Ocalml/Menhir 实现了一个简单但功能强大的解析器生成工具 AnB2Murphi，它已经成功应用于几种典型的安全协议，验证结果与已证明的结果一致，验证了我们的理论的有效性。

5.2　工作展望

在目前的工作中，协议的 Alice&Bob 语言规范是直观、详尽的。而由于 Murphi 模型检测器能够很好地检查协议的多次运行，并在协议不满足规范时给出反例路径，所以在我们的 Alice&Bob 规范的 Environment 块中指定实际参数会带来不必要的麻烦。在以后的工作中，我们会尝试将机器学习方法结合到我们的工作中，以期望通过自动化手段来分析安全协议可能遭受的威胁。此外，我们计划将我们的框架的功能从检查协议的有界会话扩展到在串空间理论中验证无界会话的正确性。将一个高层次串空间规范编译到 Isabelle 定理证明器中。利用认证测试结果证明了该协议的认证性和保密性，并基于 Isabelle 中串空间的机械形式化工作，或者通过启发式技术以提高我们的框架内入侵者推理的效率。最后，一个良好的、可操作的安全协议语言的语义是 c++/JAVA 中验证的实现的开端，因此

中国知网　https://www.cnki.net

我们打算在 c++ /JAVA 中实现可验证的串空间。

中国知网 https://www.cnki.net

参 考 文 献

[1] 王涛. 浅析计算机网络安全问题及其防范措施[J]. 科技创新与应用, 2013(2): 1.

[2] 冯登国. 国内外密码学研究现状及发展趋势[J]. 通信学报, 2002, 23(5): 9.

[3] Needham R M, Schroeder M D. Using encryption for authentication in large networks of computers[J]. Communications of the ACM, 1978, 21(12): 993-999.

[4] Lowe G. Breaking and fixing the needham-schroeder public-key protocol using fdr[C]//International Workshop on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 1996: 147-166.

[5] 王戟, 詹乃军, 冯新宇, 等. 形式化方法概貌[J]. 软件学报, 2019, 30(1): 29.

[6] Hoare C A R. An axiomatic basis for computer programming[J]. Communications of the ACM, 1969, 12(10): 576-580.

[7] Leino K R M. Dafny: An automatic program verifier for functional correctness [C]//International Conference on Logic for Programming Artificial Intelligence and Reasoning. Springer, 2010: 348-370.

[8] Bobot F, Filliâtre J C, Marché C, et al. Why3: Shepherd your herd of provers [C]//Boogie 2011: First International Workshop on Intermediate Verification Languages. 2011: 53-64.

[9] Jacobs B, Smans J, Philippaerts P, et al. Verifast: A powerful, sound, predictable, fast verifier for c and java[C]//NASA formal methods symposium. Springer, 2011: 41-55.

[10] Moura L d, Bjørner N. Z3: An efficient smt solver[C]//International conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2008: 337-340.

[11] Nipkow T, Wenzel M, Paulson L C. Isabelle/hol: a proof assistant for higher-order logic[M]. Springer, 2002.

[12] Millen J K, Clark S C, Freedman S B. The interrogator: Protocol secuity analysis

中国知网 https://www.cnki.net

[J]. IEEE Transactions on software Engineering, 1987(2): 274-288.

[13] Roscoe B. Model- checking csp[J]. 1994.

[14] Ip C N, Dill D L. Efficient verification of symmetric concurrent systems [C]//Proceedings of 1993 IEEE International Conference on Computer Design ICCD'93. IEEE, 1993: 230-234.

[15] Armando A, Basin D, Boichut Y, et al. The avispa tool for the automated validation of internet security protocols and applications[C]//International conference on computer aided verification. Springer, 2005: 281-285.

[16] Basin D, Mödersheim S, Vigano L. Ofmc: A symbolic model checker for security protocols[J]. International Journal of Information Security, 2005, 4(3): 181-208.

[17] Turuani M. The cl-atse protocol analyser[C]//International conference on rewriting techniques and applications. Springer, 2006: 277-286.

[18] Armando A, Compagna L. Satmc: a sat-based model checker for security protocols [C]//European workshop on logics in artificial intelligence. Springer, 2004: 730-733.

[19] Monniaux D. Abstracting cryptographic protocols with tree automata[C]// International Static Analysis Symposium. Springer, 1999: 149-163.

[20] Escobar S, Meadows C, Meseguer J. Maude-npa: Cryptographic protocol analysis modulo equational properties[M]//Foundations of Security Analysis and Design V. Springer, 2009: 1-50.

[21] Blanchet B, Smyth B, Cheval V, et al. Proverif 2.00: automatic cryptographic protocol verifier, user manual and tutorial[J]. Version from, 2018: 05-16.

[22] Dolev D, Yao A. On the security of public key protocols[J]. IEEE Transactions on information theory, 1983, 29(2): 198-208.

[23] Colmerauer A. An introduction to prolog iii[C]//Computational Logic. Springer, 1990: 37-79.

[24] Meier S, Schmidt B, Cremers C, et al. The tamarin prover for the symbolic analysis of security protocols[C]//International conference on computer aided verification.

中国知网 https://www.cnki.net

Springer, 2013: 696-701.

[25] Rescorla E, et al. Diffie-hellman key agreement method[J]. 1999.

[26] Mitchell J C, Mitchell M, Stern U. Automated analysis of cryptographic protocols using mur/spl phi[C]//Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No. 97CB36097). IEEE, 1997: 141-151.

[27] Tatebayashi M, Matsuzaki N, Newman D B. Key distribution protocol for digital mobile communication systems[C]//Conference on the Theory and Application of Cryptology. Springer, 1989: 324-334.

[28] Fábrega F T, Herzog J C, Guttman J D. Honest ideals on strand spaces[C]// Proceedings. 11th IEEE Computer Security Foundations Workshop (Cat. No. 98TB100238). IEEE, 1998: 66-77.

[29] abrega Jonathan F J T F, Herzog C, Guttman J D. Strand space pictures[J].

[30] Fábrega F J T, Herzog J C, Guttman J D. Strand spaces: Why is a security protocol correct?[C]//Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No. 98CB36186). IEEE, 1998: 160-171.

[31] 姚萌萌, 唐黎, 凌永兴, 等. 基于串空间的安全协议形式化分析研究[J]. 信息网络安全, 2020(2): 7.

[32] 张孝红, 李谢华. 基于串空间的安全协议自动化验证算法[J]. 计算机工程, 2011, 37(5): 3.

[33] 董学文, 马建峰, 牛文生, 等. 基于串空间的 Ad Hoc 安全路由协议攻击分析模型[J]. 软件学报, 2011, 022(007): 1641-1651.

[34] Clarke E M, Jha S, Marrero W. Verifying security protocols with brutus[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2000, 9(4): 443-487.

[35] Vigano L. Automated security protocol analysis with the avispa tool[J]. Electronic Notes in Theoretical Computer Science, 2006, 155: 61-86.

[36] Mirnig A G, Meschtscherjakov A, Wurhofer D, et al. A formal analysis of the iso 9241-210 definition of user experience[C]//Proceedings of the 33rd annual ACM

中国知网 https://www.cnki.net

conference extended abstracts on human factors in computing systems. 2015: 437-450.

[37] Islam S. Security analysis of lmap using avispa[J]. International journal of security and networks, 2014, 9(1): 30-39.

[38] Delaune S, Kremer S, Ryan M D, et al. A formal analysis of authentication in the tpm[C]//International Workshop on Formal Aspects in Security and Trust. Springer, 2010: 111-125.

[39] Bhargavan K, Blanchet B, Kobeissi N. Verified models and reference implementations for the tls 1.3 standard candidate[C]//2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017: 483-502.

[40] Dong N, Jonker H, Pang J. Formal analysis of an e-health protocol[J]. arXiv preprint arXiv:1808.08403, 2018.

[41] Meseguer J. Conditional rewriting logic as a unified model of concurrency[J]. Theoretical computer science, 1992, 96(1): 73-155.

[42] Basin D, Cremers C, Dreier J, et al. Symbolically analyzing security protocols using tamarin[J]. ACM SIGLOG News, 2017, 4(4): 19-30.

[43] Xiong Y, Su C, Huang W, et al. SmartVerif: Push the limit of automation capability of verifying security protocols by dynamic strategies[C/OL]//29th USENIX Security Symposium (USENIX Security 20). USENIX Association, 2020: 253-270. https://www.usenix.org/conference/usenixsecurity20/presentation/xiong.

[44] 杨锦翔. 网络安全协议的形式化自动验证优化研究[J/OL]. 2021. DOI: 10.27517/d.cnki.gzkju.2021.001454.

[45] 苏诚. 基于强化学习的安全协议形式化自动验证技术研究[D]. 中国科学技术大学.

[46] Emerson E A, Clarke E M. Characterizing correctness properties of parallel programs using fixpoints[C]//International Colloquium on Automata, Languages, and Programming. Springer, 1980: 169-181.

[47] Clarke E M, Emerson E A. Design and synthesis of synchronization skeletons

中国知网 https://www.cnki.net

using branching time temporal logic[C]//Workshop on logic of programs. Springer, 1981: 52-71.

[48] Clarke E M, Emerson E A, Sistla A P. Automatic verification of finite-state concurrent systems using temporal logic specifications[J]. ACM Transactions on Programming Languages and Systems (TOPLAS), 1986, 8(2): 244-263.

[49] Queille J P, Sifakis J. Specification and verification of concurrent systems in cesar [C]//International Symposium on programming. Springer, 1982: 337-351.

[50] Grobelna I, Grobelny M, Adamski M. Model checking of uml activity diagrams in logic controllers design[C]//Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30–July 4, 2014, Brunów, Poland. Springer, 2014: 233-242.

[51] Grobelna I. Formal verification of embedded logic controller specification with computer deduction in temporal logic[J]. Przeglad Elektrotechniczny, 2011, 87 (12a): 47-50.

[52] Vizel Y, Weissenbacher G, Malik S. Boolean satisfiability solvers and their applications in model checking[J]. Proceedings of the IEEE, 2015, 103(11): 2021-2035.

[53] Dawar A, Kreutzer S. Parameterized complexity of first-order logic[C]//Electronic Colloquium on Computational Complexity, TR09-131. 2009: 39.

[54] AnB2Murphi. [EB/OL]. https://github.com/AnB2Murphi/AnB2Murphi.

中国知网 https://www.cnki.net

致谢

回想 19 年秋天踏入华东师范大学的校门，满心憧憬着未知的研究生生活。一晃眼之间，宝贵的研究生光景悄然流逝，回想这一期，心中顿时百感交集。是学校和这段时光让一个懵懂无知的大学毕业生找到了人生的努力方向。

首先，感谢我的指导老师赵涌鑫老师，是他带我们打开科研道路的大门，作为引路人，赵老师在科研和工作上始终保持着一丝不苟和不断进取的精神，为我树立了一位合格科研人应该有的榜样；在生活上赵老师则是平易近人的好老师，他时刻关心我们的生活状态，在对科研进度一筹莫展时，给予我们关心和帮助。我们高可信实验室在赵老师的组织下目标一致，学到了许多课堂之外的为人处事的知识。另外感谢中国科学院软件所李勇坚老师的指导，李老师在科研上给予我许多帮助，从每周的学术讨论中解答我的疑惑和难点，将问题一个个突破，从实验的进度把控到论文的撰写，李老师作为我的校外导师尽心尽责，另外肯定了我想从事科研工作的决心，对我的人生道路规划和职业发展起着不可替代的影响。最后感谢辅导员刘国艳，你对学生工作的关心和负责，不厌其烦地填我粗心大意所犯的错误，感激艳姐给予我机会开展研究生工作。

其次，感谢这三年中所认识小伙伴韩文叠、王泽杰、孙衍超和张浩然，感谢最好的你们陪伴了我三年时光，在我最迷惘的时候给予我肯定，一起分享喜悦悲伤以及未知人生的焦虑，在毕业之际希望你们都能够完成心里所想之事；感谢实验室同门勾根旺、蒋家威、吕锦、刘知昊、胡指铭、裘盼锋、董震恒、许伟钰，非常荣幸和优秀的你们在同一个实验室一起科研生活玩耍；感谢中科院软件所认识的何锦龙、胡登杭博士，和你们一起度过的北京时光是我这辈子最难忘的时光，你们身上对科研工作的执着也会一直激励着我。

感谢我的父母，你们无条件的支持和理解，使得我能够坚定不移朝着自己科研梦想前进；感谢沈超然哥哥在我每次焦虑不知所措时给予我帮助和经验，让我直视未来可能遇到的艰难险阻以及其决心。

最后最后感谢这三年中认识的所有人，何其幸运能与你们相识相知。感谢论文评审的各位专家，期待你们给出的宝贵的意见。

中国知网 https://www.cnki.net

攻读硕士学位期间发表论文和科研情况

■ 已公开发表的学术论文

[1] Hongjian Jiang,Yongjian Li,Sijun Tan,Yongxin Zhao. Encoding Induction Proofin Dafny[C]//2021 International Symposium on Theoretical Aspects of Software Engineering (TASE). IEEE, 2021: 95-102.

[2] Yongxin Zhao,Hongjian Jiang,Jin Lv,Sijun Tan, Yongjian Li. AnB2Murphi: ATranslator for ConvertingAlice&Bob Specifications to Murphi [C]. International Conferenceon Software Engineering and Knowledge Engineering 2021

[3] Zhiming Hu,Zheng Wang, Hongjian Jiang, Yuyuan Zhang, Yongxin Zhao.HHML: A Hierarchical Hybrid Modeling Language for Mode based Periodic Controllers[C]. International Conference on Software Engineering and Knowledge Engineering2021

■ 已实质审查的发明专利

[1] 面向协议安全性质的形式化协同规约的方法及图形建模系统

专利申请号:CN202010736660.1

■ 攻读学位期间参加的科研项目

[1] 装备发展部预研项目“内生安全应用构造技术”子课题“应用软件形式化建模与规约技术，2019.09-2020.12

■ 获得荣誉情况

[1]“波克城市”优秀学生奖学金学术科研优秀奖二等奖

中国知网 https://www.cnki.net