密码协议代码执行的安全验证分析综述

张焕国 吴福生 王后珍 王张宜

(武汉大学计算机学院 武汉 430072)

(武汉大学空天信息安全与可信计算教育部重点实验室 武汉 430072)

摘 要 密码协议安全验证分析是信息安全重点研究之一.常用的密码协议安全分析(例如,形式化分析、计算模型分析、计算可靠的形式化分析)只能从理论上验证或证明密码协议的安全,无法确保密码协议代码实际执行的安全.只有当密码协议在代码执行时被验证或证明安全,才能保障密码协议在实现中是安全的.因此,代码级的密码协议安全验证分析是值得关注的方向.文中分别从自动模型提取、代码自动生成、操作语义及程序精化 4 个方面,综述代码级的密码协议安全验证分析,并对当前代码级的密码协议安全验证分析领域中的最新成果进行详细比较、分析、总结和评论.文中以常用程序语言(C、Java、F# 等)编写的密码协议为例,重点阐述密码协议代码执行的安全验证分析,并展望代码级的密码协议安全验证分析的研究方向.

关键词 密码协议;模型;代码;执行;安全验证

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2018.00288

A Survey: Security Verification Analysis of Cryptographic Protocols Implementations on Real Code

ZHANG Huan-Guo WU Fu-Sheng WANG Hou-Zhen WANG Zhang-Yi

(*Computer School of Wuhan University, Wuhan* 430072)

(*Key Laboratory of Aerospace Information Security and Trusted Computing of Ministry of Education, Wuhan University, Wuhan* 430072)

Abstract Security verification analysis of cryptographic protocols is one of the most important fields of researching the information security in network. Although common ways to analyze the security of cryptographic protocols (such as formal analysis, computational model analysis, and computational sound formal analysis) can theoretically verify or prove whether cryptographic protocols are secure, they can't guarantee that cryptographic protocols are secure in the process of implementation on real code. However, if cryptographic protocols are verified or proved to be secure during the process of implementation on real code, it can be insured that cryptographic protocols are implemented safely. Therefore, it is worthwhile to focus on the field of researching security verification analysis of cryptographic protocols implemented on real code. In this paper, first of all, we summarize the research status of security verification analysis of cryptographic protocols at home and abroad. Furthermore, we summarize the research status of security verification analysis of cryptographic protocols implemented on real code, and there are four branches: automated model extraction, automated code generation, operational semantics and program refinement. Meanwhile, we compare, analyze, summarize and comment the latest achievements in the research of security verification analysis of cryptographic protocols implemented on real code. In

收稿日期:2016-07-18;在线出版日期:2017-01-13.本课题得到国家自然科学基金(613030212,61202385)、国家自然科学基金重点项目(61332019)和国家"九七三"重点基础研究发展规划项目基金(2014CB340600)资助.**张焕国**,男,1945 年生,教授,主要研究领域为信息安全、密码学、可信计算.E-mail: liss@whu.edu.cn. **吴福生**,男,1974 年生,博士研究生,研究方向为信息安全、协议代码实现的安全验证分析. **王后珍**,男,1981 年生,博士,讲师,研究方向为信息安全、密码学. **王张宜**,男,1981 年生,博士,讲师,研究方向为信息安全协议.

this paper, taking common programming languages as examples (such as C, Java, F♯, and so on), we carry out the discussion and focus on four kinds of security analysis. Based on classical abstract theories, automated model extraction analyzes the security of a cryptographic protocol by applying an abstract mapping, which maps the properties of a cryptographic protocol from concrete program state space onto a corresponding abstract model. Automated code generation, based on the specifications of a cryptographic protocol, generates the codes automatically and analyzes the security of a cryptographic protocol with these codes. The security analysis of cryptographic protocols implementation on real code which is based on operational semantics analyzes the security of the properties of a cryptographic protocol, such as the sequence of the traces, the match of messages during the process of the implementation of a cryptographic protocol and so on. The program refinement security analysis analyzes the security of a cryptographic protocol by applying the relation of the program refinement. At the end of this paper, we prospect the research direction of security verification analysis of cryptographic protocols implemented on real code in six parts: Model checking techniques (such as program refinement, program verification and so on) are applied to analyze and verify cryptographic protocol implementation on real code; Based on Dolev-Yao model, the security of cryptographic protocols is automatically analyzed and verified on real code; Programming languages (C/C++, Java, F♯ and so on) are applied to build models or frames (F7, CoSP, Spi2Java, etc.) in the process of security verification analysis of cryptographic protocols; Security verification analysis of cryptographic protocol implementation on real code is based on semantic security; Computational sound formal analysis is applied to analyze and verify the security of cryptographic protocols implementation on real code; Concurrency security verification analysis of cryptographic protocols implemented on real code is developed.

Keywords　cryptographic protocols; model; codes; implementation; security verification

1　引　言

安全的密码协议是开放网络空间信息安全传输[1-2]，严防用户隐私泄露的关键技术. 一直以来，密码协议安全验证分析是衡量协议安全可靠的重要手段. 随着大数据、云计算等新型网络的兴起，传统单一的端到端网络通信模式已转换为多对一、多对多的新型网络通信模式. 在这种新型的网络环境下，通信信息更易受到攻击，用户的隐私更易泄露[3-4]. 因此，常用密码协议的安全验证分析(例如，形式化分析、计算模型分析、计算可靠(computational soundness)的形式化分析)已无法满足新型网络通信模式的信息安全需求. 因此需对密码协议的安全验证分析提出更高要求[5].

常用的密码协议安全验证分析只能验证或证明协议的理论安全，无法确保密码协议在代码实际执行中的安全性. 究其原因，是其无法考虑下述几种情况：(1)程序设计语言的语法和语义结构(例如，C语言内存泄露、指针越界、缓冲区溢出等)；(2)程序代码执行时产生很多临时的异常情况(例如，函数调用的不确定返回值、程序中断、程序断言等)；(3)程序代码实际执行的运行环境(例如，不同网络环境、不同语言环境等). 由此可见，代码级的密码协议安全验证分析比形式化模型或计算模型下的密码协议安全验证分析复杂. 故代码级的密码协议安全验证分析是新的研究方向[6].

代码级的密码协议安全验证分析作为一个新的研究方向，在国内外处于起步阶段. 在国外，代码级的密码协议安全验证分析的研究有：代码模型提取[7]；代码自动生成[8]；操作语义分析[9]；精化(refinement)验证分析[10]. 在国内，仍以常用的安全验证分析为主，例如，基于网关口令的认证[11]、基于关键字域可变协议的验证[12]等. 这些安全验证分析基于计算语义安全来分析密码协议的安全性，或用现有工具和软件检测模型技术来验证密码协议是否安全[13-14]. 在代码级上分析密码协议实际执行是否安全的相关研究则少之又少.

本文对密码协议代码执行的安全验证分析进行综述. 以常用程序设计语言(例如，C语言、Java语

言和 F# 语言等)编写的密码协议为实例,综述密码协议代码实际执行的安全验证分析.

本文综述结构(见图 1),第 2 节简介常用密码协议安全验证分析;第 3 节讨论代码级的密码协议安全验证分析(重点综述),包括 4 个方面:(1)密码协议代码模型提取的安全验证分析.代码模型提取以经典的抽象理论作为基础,引入一个抽象映射,把密码协议属性从具体的程序状态空间映射到抽象的模型中,从而分析协议的安全性.本质是避免状态空间爆炸问题;(2)代码自动生成的安全验证分析.代码自动生成是通过密码协议规范自动生成协议代码,从而分析协议的安全性.作用是为了清楚了解密码协议代码实际执行的过程细节,避免协议执行时产生漏洞.原理是引入一个抽象映射,把密码协议属性从抽象模型映射到具体的模型中;(3)操作语义的安全验证分析.基于操作语义的密码协议代码执行安全验证分析是对密码协议执行轨迹,简称迹(traces)的顺序、消息的匹配等属性进行安全验证分析.本质是分析密码协议参与者之间的交互式通信行为安全;(4)代码执行精化的安全验证分析.精化的安全验证分析是基于密码协议代码执行程序的精化关系,从而分析协议的安全性.作用是精化后的程序可以替代精化前的程序.原理是精化后的程序产生的行为不会多于精化前的程序.本质是验证密码协议代码实际执行的安全属性归结为精化关系的验证;第 4 节总结与展望未来研究方向.

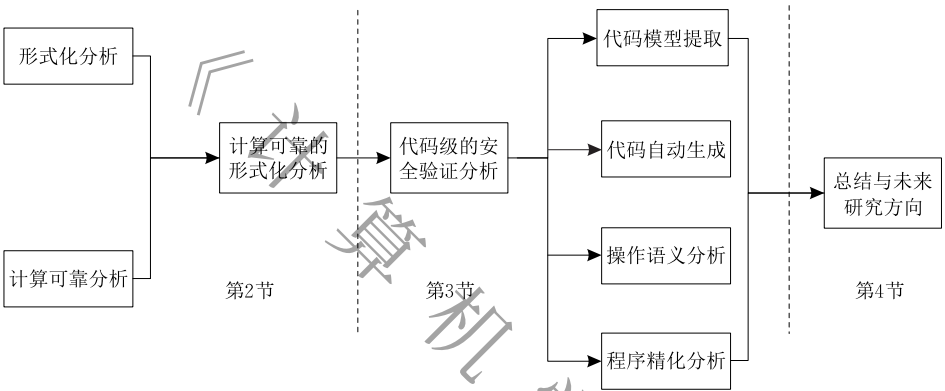

图 1 综述结构

2 常用密码协议安全验证分析

密码协议安全验证分析是验证或证明协议安全属性是否成立,确保密码协议执行时没有缺陷.常用的密码协议安全验证分析有三类:形式化分析、计算模型分析、计算可靠的形式化分析.

(1)密码协议形式化分析.形式化分析对协议的各要素进行抽象符号化,然后基于数理逻辑的推导,从而发现密码协议中的缺陷.文献[15]正式提出密码协议安全验证的形式化分析,形成以 Dolev-Yao 模型假设[16]为基础的形式化安全分析方法.文献[17]首次在密码协议的形式化分析中应用数理逻辑推导,称之为 BAN 逻辑.BAN 逻辑是密码协议形式化安全验证分析的一个里程碑.

(2)密码协议计算模型分析.计算模型分析采用概率大小衡量密码协议的安全性,其理论基础为计算理论(例如,可忽略函数、多项式时间、计算不可区分性等),并涉及难解的数学问题(例如,大整数因式分解、离散对数、椭圆曲线等).相关研究见文献[18-20].

(3)密码协议计算可靠的形式化分析.计算可靠的形式化分析是形式化模型与计算模型有机结合的一种密码协议安全验证分析方法.它证明协议属性如果在形式化模型下安全,则在计算模型下也必然安全,即密码协议安全属性的形式化分析与计算模型分析等价,具有代表的是 Abadi Rogway 模型[21](简称 AR 模型).基于 AR 模型分析方法,文献[22]进一步提出密码协议计算语义与形式化模型结合的分析方法,从协议的逻辑语义分析其安全性.文献[23]证明协议属性在计算语义与形式化两种不同的模型下是安全的,即密码协议安全属性的可靠性(soundness:表示在交互式证明系统中两个状态或模型变换时,某一属性变化的概率可忽略不计,即协议属性可靠).文献[24]提出自动的密码协议安全可靠性证明,提升了密码协议安全属性的可靠性,并得到实际应用[25].基于计算语义与形式化模型,文献[14]提出刚性与相似性概念的安全验证分析方法.这种分析方法以密码协议语法和语义赋值为理论构建有攻击能力模型的非自由消息代数理论

框架，同时证明协议在该框架内满足安全的可靠性.

常用的密码协议安全验证分析已取得丰硕的成果[6]，但无法考虑密码协议代码实际执行时存在的问题：(1) 程序语言结构困难(例如，C 语言的内存泄露、Java 封装、F♯语义等)；(2) 从密码协议规范到代码执行产生漏洞(例如，程序中断、变量的不正确使用等)；(3) 程序代码执行迹的困难(例如，代码的消息匹配、函数调用的不正确返回值等)；(4) 程序代码执行的行为困难(例如：在具体状态空间下，密码协议的属性验证). 要解决这一系列问题，需拓展更为复杂的密码协议代码实际执行安全验证分析.

3　代码级的密码协议安全验证分析

密码协议设计的最终目的是转换成实际执行的程序代码. 在进行安全验证分析时，常用的密码协议即使在理论上被严格证明是安全的，但实际执行中由于引入新的不安全因素，也会导致协议实际执行不安全，因此，代码级的密码协议安全分析重点研究密码协议代码实际执行安全.

3.1　代码模型提取安全验证分析

模型提取方法来源于程序属性的抽象解释论[26]和经典的抽象理论[27]. 代码模型提取是把密码协议的某一具体属性映射成它的某一抽象属性，并证明如果抽象属性成立，那么具体属性也成立. 不妨设 P 是协议，E 是环境，M 为建模，具体协议模型为 $M_c=M(P,E)$，$\alpha(\cdot)$ 是抽象映射，则抽象模型为 $\alpha(M_c)$. 不妨设 $\varnothing$ 为密码协议的具体属性，它对应的抽象属性为 $\alpha(\varnothing)$. 代码模型提取的数学表示为 $\alpha(M_c)\vdash\alpha(\varnothing)\Rightarrow M_c\vdash\varnothing$. “$\vdash$”表示推导“$\Rightarrow$”表蕴含. 由此可见，代码模型提取考虑两个因素：(1) 映射函数 $\alpha(\cdot)$ 的可靠性. 文献[28]把密码协议代码(用 C 语言编写程序代码)的注释假设为可信任项，并把它作为模型提取研究一部分；(2) 协议环境(例如，语言特征、程序运行的环境等). 由于 Java 语言、F♯语言等是具有良好封装性的高级语言，所以在代码模型提取中侧重于考虑程序运行的环境问题.

不同程序设计语言的语义、语法、句法不尽相同. 例如，C 语言考虑指针，Java 语言进行封装，F♯语言具有相对完备的语义且封装程度更高，导致具体的分析方法也不尽相同. 下面分别综述常用 3 种语言 C、Java、F♯的密码协议代码模型提取安全验证分析.

3.1.1　C 语言模型提取分析

C 语言的主要特点是指针运算和函数，但密码协议的规范无法表达 C 语言的内部函数(例如，allocation，pointer 等)与外部函数库(例如，memcpy，strcpy 等)的链接，这使密码协议代码执行安全验证分析变得复杂和困难. 为解决这一矛盾，C 语言代码模型提取把具体模型的协议代码映射为抽象模型的协议规范，然后根据抽象模型的协议规范进行安全验证分析. 本小节从 3 个方面综述 C 语言的密码协议代码模型提取：(1) 基于“信任断言”分析；(2) 基于自动化分析；(3) 基于中间语言分析.

(1) 基于信任断言分析. 信任断言分析是一种 C 语言模型提取的协议代码执行安全验证分析方法[28]，由内部和外部两个模型组成. 它将密码协议代码执行安全验证的复杂性与困难性作为信任假设，构建密码协议代码实际执行时的内部信任(Internal Trust)和外部信任(External Trust). 文献[28]首次提出信任断言模型的 C 语言代码模型提取分析方法. 该方法使用霍恩子句逻辑(Horn clauses logic)来表示密码协议规范和攻击者的能力. 它假设密码协议代码执行的 C 语言指针运算为内部信任，建立信任断言模型(Trust Assertions Model)对协议进行安全验证分析. 具体分析如下：

① 外部信任断言模型(External Trust Assertions Model，ETAM)假设攻击者拥有信任断言的能力(*knows* 作为谓词，*knows*(*X*)表示攻击者具有获得 *X* 知识的能力). 如表 1 所示.

表 1　外部信任断言

霍恩子句模型化	入侵者能力
knows(*nil*) *knows*(*cons*(*X*,*Y*))<—*knows*(*X*),*knows*(*Y*)	入侵者建立列表
knows(*X*)<—*knows*(*cons*(*X*,*Y*)) *knows*(*Y*)<—*knows*(*cons*(*X*,*Y*))	入侵者读取列表所有基本信息
knows(*crypt*(*X*,*Y*))<—*knows*(*X*),*knows*(*Y*)	入侵者能够加密
knows(*pub*(*X*))	入侵者知道公钥
knows(*X*)<—*knows*(*crypt*(*X*,*pub*(*Y*))),*knows*(*prv*(*Y*))	入侵者用他知道的解密密钥能够解密密文
knows(*X*)<—*knows*(*crypt*(*X*,*prv*(*Y*))),*knows*(*pub*(*Y*))	
knows(*X*)<—*knows*(*crypt*(*X*,*sk*(*Y*,*Z*))),*knows*(*sk*(*Y*,*Z*))	

② 内部信任断言模型(Internal Trust Assertions Model，ITAM)假设 C 语言内部的编译序列为信任断言.

设 $x,t\in Atom$，x rec t 表示 x 信任 t. 根据霍恩子句逻辑表示($P\leftarrow Q_1,Q_2,\cdots,Q_n,n\neq 0$，$P$ 表示过

程名,Q_i表示过程调用,且 $i\in n$)可知,逻辑表达式成立:

$$\begin{aligned} &x \text{ rec } t \leftarrow x_1 \text{ rec } t_1,\\ &x_2 \text{ rec } t_2,\cdots,x_n \text{ rec } t_n \end{aligned} \tag{1}$$

这里 x_i rec t_i,表示 x_i信任 t_i. 例如,一条简单的 C 语言代码:$msg\rightarrow id\_1[0]=id[0]$,根据 C 语言内部编译可得到编译序列如下:

$$\begin{aligned} &z=0, x_1=\&id[z], x_2=*x_1,\\ &x_3=msg\rightarrow id\_1,\\ &x_4=x_3[z], *x_4=x_2 \end{aligned} \tag{2}$$

由式(1)和(2)得到表达式(3)成立:

$$\begin{aligned} &z \text{ rec } 0, x_1 \text{ rec } id[z],\\ &x_2 \text{ rec } *x_1, x_3 \text{ rec } msg\rightarrow id\_1,\\ &x_4 \text{ rec } x_3[z], *x_4 \text{ rec } x_2 \end{aligned} \tag{3}$$

由霍恩子句逻辑可知式(4)成立:

$$\begin{aligned} &msg\rightarrow id\_1[0] \text{ rec } id[0]\leftarrow z \text{ rec } 0,\\ &x_1 \text{ rec } id[z], x_2 \text{ rec } *x_1, x_3 \text{ rec } msg\rightarrow id\_1,\\ &x_4 \text{ rec } x_3[z], *x_4 \text{ rec } x_2 \end{aligned} \tag{4}$$

由此可见,基于信任断言的具体分析过程是密码协议源代码经过 C 语言内部编译后,得到一些有序的编译序列,通过霍恩子句逻辑推导得到 ITAM 编译序列. 根据 ITAM 和 ETAM 对密码协议代码执行的安全验证分析.

文献[28]提出信任断言方法考虑到密码协议代码实际执行的情况,但无法解决密码协议代码执行的具体行为(例如,初始化加密、解密和 Hash 函数[29]等),也没有考虑到 C 语言数组越界[30]和缓冲区溢出[31]等情况,更没有考虑到密码协议安全认证. 基于信任断言模型的密码协议安全验证分析仍有待进一步完善,是未来代码级的密码协议安全验证分析研究方向. 有进一步需求的读者,请参考文献[28].

(2)基于自动化分析. 自动模型提取采用自动模型框架对密码协议代码执行进行自动安全验证分析. 文献[32]提出基于 C 语言的 ASPIER(Automated Security Protocol Implementation VerifiER)自动模型框架的密码协议代码执行安全验证分析方法,该方法使用 ASPIER 自动模型框架对 OpenSSL 进行安全验证分析. 文献[33]提出 VCC(Verifying Concurrent C)并发框架和多线程的安全验证分析,适用于通用自动模型提取的密码协议代码执行安全验证分析. 具体分析如下:

① ASPIER 自动分析. 这种分析是基于软件模型检测标准的安全验证分析方法. 在具体的密码协议代码执行自动模型提取分析中,用状态规范机制 SSMs(SSMs=Specification Stats Machines)取代 C 语言库函数,对密码协议安全属性进行验证分析. ASPIER 有类似于 C 语言的语法结构. 因此,ASPIER 自动分析适用于代码级的密码协议安全验证分析. 与 C 语言不同,ASPIER 有自己的语言结构和验证方法,即把密码协议的 C 语言源代码转换为类似于 C 语言的抽象伪代码. ASPIER 自动分析表现在 3 个方面:(a)自动验证;(b)作为抽象工具;(c)OpenSSL 实例化自动验证. ASPIER 自动模型框架的密码协议安全验证分析分成两个阶段:第 1 阶段是密码协议的编译;第 2 阶段通过 COPPERT[34]工具验证. ASPIER 具体工作原理参考文献[32].

② VCC 自动分析. VCC 是基于并发系统和多线程执行环境的密码协议安全验证分析工具,应用于系统的密码协议安全验证分析. VCC 扩展 C 语言的注释,并引入函数的前置与后置条件. VCC 框架有类似于 C 语言的数据结构. 所以,基于 VCC 工具的协议自动分析是通过数据类型与变量结构对密码协议进行并发的安全验证分析. 详细过程参考文献[33]. 为扩展密码协议代码执行的 VCC 自动安全验证分析,文献[35]提出基于 VCC 工具的通用密码协议执行的自动安全验证分析方法. 这种分析方法在 VCC 框架上自动运行具体的密码协议代码,使用断言、假设等手段分析密码协议的安全属性(例如,认证、完整以及弱安全等). 例如,一个典型的属性分析:假设 k 是参与者 A,B 的共享密钥,且 k 公开. 如果存在要么 A 不诚实,要么 B 不诚实情况,这时可使用 VCC 自动安全验证来分析 k 的安全性. 具体请参考文献[35].

(3)基于中间语言分析. 基于中间语言的分析方法是使用中间语言 CIL(C Intermediate Language)[36]代码替代 C 语言代码,对密码协议进行安全验证分析. 这种分析方法既考虑协议高层(High-level)的安全验证分析,又要考虑协议底层(Low-level)的安全验证分析. C 语言存在指针运算、内存、数组等语言结构特点,在对密码协议代码执行的安全验证分析时存在一些无法解决的困难问题(例如,缓冲区的溢出、数组负下标的检索、指针运算等). 为更好地接近代码实际执行的安全验证分析,文献[37]提出类似于 C 语言的中间语言安全验证分析. 基本分析流程如图 2 所示.

从图 2 可知,中间语言的分析过程是:① 编写 C 语言的密码协议源代码(图 3 来自文献[37]的密码

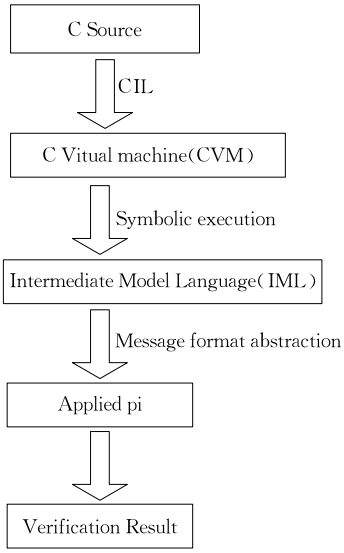

图 2　基本分析流程

协议部分源代码)；② 通过中间语言得到密码协议的中间语言源代码(图 4 来自文献[37]的密码协议中间语言部分源代码)；③ 通过中间语言模型和 pi 演算(也称为 π-演算)建立抽象模型，使用 ProVerif[38] 工具自动验证协议属性的可靠性；④ 验证结果，即完成密码协议代码执行安全验证分析.

```
void *key;
sizet keylen;
readenv ("k", &key, &keylen);
sizetlen;
read (&len, sizeof(len));
if(len>1000)
   exit( );
void *buf=malloc(len+2 *MAC LEN);
read(buf, len);
mac(buf, len, key, keylen, buf+len);
read(buf+len+MACLEN, MAC LEN);
if(memcmp(buf+len,buf+len+MACLEN,MAC LEN)==0)
 event("accept", buf, len);
```

```
in(x1); in(x2); if x2=mac(k,x1) then event accept(x1)
```

图 3　密码协议部分源代码

```
void mac_proxy (void *buf, sizet buflen, void *key,
                sizet keylen, void *mac)
{
  load_buf(buf,buflen);
  load_buf(key,keylen);
  apply("mac",2);
  store buf(mac);
}
int memcmp_proxy(void *a, void *b, sizet len)
{
  int ret;
  load_buf(a,len);
  load_buf(b,len);
  apply("cmp",2);
  store_buf(&ret);
  return ret;
}
```

图 4　类似于 C 语言的密码协议部分源代码

从部分源代码(图 3 所示)到部分中间语言源代码(图 4 所示)相比可知，中间语言的密码协议源代码有类似于 C 语言的语法结构，它的不同点在于引入中间语言函数. 例如，在图 3 中，密码协议源代码中的 mac，memcmpy 函数被中间语言的密码协议源代码的 mac_proxy，memcmp_proxy 函数取代. 从 C 语言源代码到中间语言源代码，用到 C 语言的虚拟机制 CVM(C Virtual Machine)，其优点是 CVM 可避免 C 语言的内存、指针等操作带来不安全的影响. 从图 2、图 3 及图 4 可知，C 到 CVM 用到 CIL 的中间语言函数_proxy(_proxy 表示中间函数一般形式，当表示具体的中间语言函数时，可在下划线前加上具体名称，例如，mac_proxy)，获得类似于 C 语言的中间语言密码协议源代码. 中间语言模型提取用到 CVM 机制、中间语言函数_Proxy 等方法，通过抽象模型语言建模，使用自动分析工具对密码协议代码执行进行安全验证分析. 这种分析方法避免语言结构带来的不安全因素，同时考虑密码协议在高层与底层的安全验证. 这种分析不足在于只能执行单路的安全验证，不能满足并发协议的安全验证分析. 为解决这类问题，文献[39]提出改进方法，在最后的密码协议安全验证阶段采用计算模型工具(CryptoVerif)[40] 而不是符号模型工具(ProVerif)，其优点是使密码协议安全验证更接近代码实际执行的安全验证分析. 对于这方面的研究，Aizatulin 和 Jürjens 等人做了很多工作[37,39].

C 语言模型提取是从具体的密码协议代码到抽象模型的密码协议安全验证分析. 虽然这种分析方法比常用的密码协议分析方法(第 2 节提到)更接近密码协议代码实际执行的安全验证分析，但是离具体密码协议代码执行的安全要求还存在一定差距. 为缩小这种差距，文献[41]提出低级语言的安全验证分析. 这种方法是把密码协议原语从源代码中分离，使用已有的密码库来保证密码协议属性的安全，从而保证密码协议的属性安全. 例如，文献[41]在对 TLS 协议安全验证分析时，把密码协议代码分成 Concrete Libraries 和 Symbolic Libraries 两部分，分别引入对应的密码库(Cryptographic Libraries)，自动执行密码协议的安全验证分析. 同样，为缩小这类的差距，文献[42]提出扩展符号模型(VeriFast 模型)分析方法. 这种方法是使用加密协议(加密是为保证协议的安全属性)与已存在密码原语库(PolarSSL)的关联，实现密码协议代码的静态安全验证分析. 这种分析方法的优点是可证明协议底层

的操作安全,例如,协议代码的内存安全.不足是不能提供协议可靠性证明,不能扩展抽象模型.由于篇幅原因,如需要进一步了解请参考上面提到的文献.

C 语言模型提取在代码级上的密码协议安全验证分析已取得一些研究成果,但是还不能满足密码协议代码实际执行的安全需求(例如,身份认证、数据完整性等).文献[37]无法保证协议代码执行的可靠性;文献[43]提出混合情况的 TLS 协议安全验证分析,可在 OpenSSL 环境下进行验证,但并没有完全满足对系统的安分验证分析,原因是 OpenSSL 库并没有完善.综上所述,基于 C 语言编写的密码协议在代码级上的安全验证分析有待进一步改进.

3.1.2 Java 语言模型提取分析

Java 语言是一种面向对象的高级程序语言,有良好的语言封装结构,它的实际代码执行以对象为操作单位.由于 Java 语言的封装性,无需考虑语言底层结构问题(例如,不考虑指针运算),密码协议代码执行行为通过信息流、程序接口(API)等反映来表现,所以 Java 语言模型提取的密码协议代码执行安全验证分析更接近实际应用.下面分别从信息流框架和程序接口概括 Java 语言模型提取的密码协议安全验证分析:

(1)基于信息流框架分析.用密码协议的信息流框架方法来分析协议代码执行的行为安全,这些行为通过协议参与者信息交互执行时产生的信息流来表现.信息流是协议代码执行行为的基本表现.也就是说,Java 模型提取本质是基于信息流框架的安全验证分析.

Jif(Java information flow)[44]是一种常用的密码协议信息流安全分析工具.它的作用是防止信息的机密被不正当地使用,或不受信任的信息被使用.文献[45]提出 Jif 框架的密码协议代码执行安全验证分析.Jif 框架方法是借助信息流类型语言对密码协议代码执行的属性进行安全验证分析.Jif 框架分析由 3 个步骤组成:① Java 程序嵌入到所有的协议参与者中;② 调整应用程序以便适应 Jif 框架;③ 分离协议参与者.例如,Jif 框架在 POKER[46]网络协议的安全验证分析中不需要可信第 3 方参与密码协议代码执行,而是把 POKER 协议中的每个参与者都看作 Java 类代码.因此,在 POKER 协议的安全验证分析中常常把 Jif 框架作为衡量安全的分析标准.为清楚 Jif 框架的密码协议安全验证分析,需了解 Jif 框架规则表达:$type\{Owner\rightarrow Readers\}$,$type$ 是标准的 Java 类型,$\{Owner\rightarrow Readers\}$ 表示变量的规则.例如,$boolean\{Alice\rightarrow Bob\}$ 表示 Alice 拥有与 Bob 相同的可读取信息值.虽然 Jif 框架的密码协议安全验证分析不用考虑语言底层结构问题,但它使用的安全类型语言在真实系统中的安全验证分析面临以下几个方面的挑战:① 如何识别关键安全代码里有不安全因素存在于安全类型里?② 在安全类型的执行过程中,是否对代码实施了不必要的限制?③ 足够透明的安全类型能否保证代码执行安全?④ 如何使用安全类型语言平衡代码执行的优点和缺点?例如,程序片段:$y=1$; if ($x==0$) $y=0$,这里隐含地表明,信息流是从变量 x 到 y,如果 if 语句为假,则表明 $y=0$ 的信息丢失.Jif 是一个比较成熟的信息流安全分析工具,广泛应用于 Java 语言模型提取的密码协议安全验证分析.具体情况参考文献[45].

(2)基于程序接口分析.密码协议的程序接口方法分析协议代码执行的行为安全,这些行为是通过程序代码执行时程序接口行为来表现的.程序接口条件决定程序执行行为的关键,例如,函数的调用顺序、函数调用返回值等决定程序代码执行的轨迹,简称为迹.程序代码执行的迹反映程序代码执行的信息流,迹的部分由程序接口来表现.文献[47]提出 Java 程序语言接口的协议代码执行安全验证分析方法.这种分析方法使用一阶逻辑 FOL(First Order Logic)自动证明工具,构建从协议代码执行到协议规范链接,并采用 JSSE[48](Java Secure Sockets Extension,这是一种标准安全的 Java 套接字扩展技术)技术对 SSL 协议(用 Java 语言编写)进行安全验证分析.JSSE 接口技术的 SSL 安全验证分析使用 RSA 作为加密算法和服务器认证.另外,程序接口分析方法把程序代码中的每个函数接口作为产生行为的点,然后把程序代码执行的行为用 CFG(Control Flow Graph)图描绘出来,在 Dolev-Yao 模型假设下使用一阶逻辑对密码协议进行安全验证分析.具体分析见文献[49].

为提高软件规范与密码协议代码实际执行安全验证分析的效率,文献[50]对 JSSE 接口技术规范进行改进提高.这种改进使得 JSSE 接口技术更适应于密码协议代码上的安全验证分析.规范密码协议代码执行接口技术未来研究在于使用组合的方法,提高密码协议代码执行的安全验证分析.文献[51]提出它的更加详细版本.为更好的分析密码协议接口行为安全属性,文献[52]提出新颖的 SystemM 逻辑认证系统[53].SystemM 接口是基于代码逻辑沙箱安全验证分析,它的作用是限制接口行为,减少恶意代码攻击,并保证系统安全可靠.如果能提供 SystemM

的自动推理证明，会进一步提高密码协议程序代码逻辑的安全验证分析，这是一个值得进一步研究的问题.

Java 程序语言的密码协议模型提取在一定程度上避免 C 语言的结构性问题. 密码协议分析结果更加接近密码协议代码实际执行的安全验证分析. Java 语言具有良好封装性，这导致了密码协议代码在实际执行安全验证分析时，无法细粒度考虑到协议底层的安全验证分析. 如何细粒度分析具有良好封装性语言的密码协议代码执行安全性是一个值得思考的问题.

3.1.3　F♯语言模型提取分析

F♯是一种 SML(Standard Programming Language)的函数程序设计(Functional Programming，FP)语言. F♯语言有完备的语义结构，其密码协议的安全性依赖于语义安全性[54]，可避免 C 和 Java 两种程序语言密码协议代码执行时存在的不足：例如，C 语言密码协议代码执行的安全验证分析无法消除语言结构带来的负面影响；Java 语言的封装性无法保证细粒度分析密码协议代码执行的安全性. 因此，F♯语言代码模型提取的密码协议安全验证分析更接近自然语言安全验证分析. F♯语言语义通过 RCF(Refined Concurrent FPC)语法的 λ-演算递归类型程序语义来表达协议的规范，使用一阶逻辑来证明或验证在具体代码模型下安全的密码协议属性，在抽象模型下也安全. RCF 演算语法见表 2 所示.

表 2　RCF 语法格式

Value expression	Annotation	Value expression	Annotation
a,b,c	name	$A,B::=$	expression
h	constructor	M	value
$M,N::$	value	$M=N$	function application
x,y,z	variable	$M=N$	syntactic equality
$()$	unit	let $x=A$ in B	let
$\lambda x.M$	function	let $(x,y)=M$ in A	pairsplit
(M,N)	pair	match M with hx then A else B	constructor match
hM	Constructor application	$\nu a.A$	restriction
$a?$	Receivemessage off channel	$A\Gamma B$	fork
$assumeF$	Assumption of formula F	$a!M$	transmission of Monchannela
$assertF$	Assertion of formula F		

基于 F♯语言语义安全的密码协议代码执行安全验证分析，文献[55]提出一种新的方法，可在抽象模型工具 ProVerif 和计算模型工具 CryptoVerif 下进行安全验证分析. 这种方法分析密码协议的抽象模型与具体代码执行之间的差距，并在 TLS 协议上比较两种工具验证分析过程，得到不同的结果. 图 5 基于 ProVerif 工具分析结果，图 6 基于 CryptoVerif 工具分析结果(实验结果来自参考文献[55]).

Verified Parts of TLS	Security Goals	F♯ Code	Queries	Time	Memory
Full Handshake	Authentication	1418 lines	2	27 s	60 MB
Full Handshake	Secrecy	1418 lines	2	25 s	80 MB
Full Handshake & Resumption	Authentication	2194 lines	2	8 min	460 MB
Full Handshake & Resumption & Record	Authentication (Record Only)	3344 lines	2	11 min	700 MB
Full Handshake & Resumption & Record	Authentication & Secrecy	3344 lines	10	3.5 h	4.5 GB
Full TLS & Password-based Application	User Authentication	3855 lines	2	1.5 h	1.2 GB

图 5　CryptoVerif 工具分析结果

Verification Result	F# Code	Cypto Assumptions	Games	Time
Record Authentication (Theorem 7)	1967 lines	18 lines	15	1.9 s
Record Secrecy (Theorem 8)	1967 lines	25 lines	14	0.3 s
PMS Random Secrecy (Theorem 9)	2497 lines	33 lines	18	1.1 s
MS Authentication (Theorem 10)	2497 lines	23 lines	8	24.0 s

图 6　ProVerif 工具分析结果

从图 5 和图 6 可知，在 F♯语言模型提取下进行安全验证分析，CryptoVerif 工具比 ProVerif 工具所使用的 F♯语言的代码量更少，所花费的时间更少. 由此可知，在 F♯语言模型提取下进行分析，计算模型的密码协议安全验证分析更能有效地保障密码协议代码实际执行的安全. 以此为依据，文献[56]提出 F♯语言模型提取代码级的密码协议安全验证分析方法. 这种方法是在 F_7 模型[57]下对 F♯语言代码的密码协议进行计算可靠的自动加密与签名的安全验证分析.

另外，F♯语言具有完备的语义结构，使密码协议代码执行的安全验证分析更接近于自然语言安全验证分析. 基于这一特征的研究取得一些重要的成果. 例如，文献[58]提出 CoSP 框架的密码协议代码执行的安全证验分析，在 CoSP 框架中嵌入 π-演算，使用公钥加密和数字签名技术保证协议安全属性的可靠性. 但是这种方法没有考虑消息调度，参与者不诚实以及协议内部结构等细节，不能完全保障协议安全属性的可靠性. 文献[59]提出基于 CoSP 框架改进方法，这种改进方法不仅结合密码协议代码执行的静态安全属性和观察等价[60]的安全验证分析，而且把 π-演算嵌入到 CoSP 框架，从而保证协议安全属性的可靠性. 文献[61]提出密码原语的通用组合框架(Universally Composable Framework，简称 UCF 框架)安全验证分析方法. 该方法通过 UCF 框架分析交互式密码协议安全属性的可靠性，并在 CoSP 框架下已证明成立，详细分析过程参考文献[61]. 但是 UCF 框架下的密码协议代码执行的安全验证分析存在抽象与具体代码执行之间的差距. 在基于模型框架分析中，文献[62]提出在 F_7 模型下的密码协议源代码安全验证分析方法. 这种方法在进行安全验证分析时采用 RCF 类型的主谓词技术，使用霍恩子句推导规则，通过 Tarski-Knaster 不动点原理，考虑密码协议代码执行时输入类型的语义安全，从而更好地保障密码协议安全属性的可靠性. F_7 模型安全验证分析可解释为：(1) 霍恩子句是封闭公式，即 $\forall x_1,\cdots,x_k.(C_1\wedge\cdots\wedge C_n\Rightarrow P)$，$C_i(i\in\mathbb{N})$ 表示原子公式；(2) P 表示过程名，即有限的霍恩子句连接集；(3) 如果 P 是逻辑程序，则满足 RCF 解释 I_p. 如需进一步了解，请参考上述文献.

当前，在语言模型提取的密码协议代码执行安全验证分析方法中已提出多种分析模型(例如，断言、信息流、F_7 等)，这些模型在一定程度上保证密码协议的安全属性的可靠性，但是在密码协议代码实际执行的安全验证分析存在差距. 在语言模型提取下进行代码级的密码协议安全验证分析，特别是在计算模型下语义安全的密码协议代码执行安全验证分析，还有很大的研究空间.

3.2 代码自动生成安全验证分析

代码自动生成是密码协议规范的抽象模型与代码实际执行的具体模型之间存在蕴含关系安全验证分析. 不防设，M_a 是抽象的程序模型，$\varnothing_a$ 是抽象模型的安全属性，$\rho(\cdot)$ 是精化映射(把一个程序抽象模型和一些具体的选择映射到一个具体模型)，$\rho(M_a,C)$ 表示具体模型，$\rho(\varnothing_a)$ 表示具体属性，C 代表具体选择(例如，程序语言、运行环境等). 代码生成数学表达为 $M_a\models\varnothing_a\Rightarrow\rho(M_a,C)\models\rho(\varnothing_a)$. 安全协议的代码生成一般具有两个特征：(1) 存在对精化映射可靠性的形式化证明；(2) 用户与第三方交互时协议信息格式的自由选择.

一般高级程序设计语言(例如，C、C++、Java 语言等)没有正式语义，所以这种语言的密码协议代码执行的安全验证分析常常使用编译器或环境解释(C 语言是编译语言，而 Java 语言是解释语言). 具备正式语义[63]的 SML 语言(例如，F♯语言)密码协议代码执行的安全性分析，多使用模型提取[53,56](3.1.3 节中已论述)方法进行分析.

目前，代码自动生成的密码协议安全验证分析已取得一些成果[64-66]. 下面介绍 C 语言和 Java 语言的密码协议代码自动生成的安全验证分析.

3.2.1 基于编译器代码自动生成分析

C 语言密码协议代码执行的安全验证分析无法考虑代码执行过程细节，必须借助编译器这种中间载体来分析代码执行的过程细节. 所以编译器(Complier)在 C 语言中占有重要位置. 编译器在密码协议代码自动生成安全验证分析作用：(1) 编译器可抽象模型；(2) 编译器可转换具体协议执行. 文献[67]提出 XML 语言的密码协议规范自动生成密码协议代码执行的安全验证分析.

编译器作为代码生成工具，被用来分析由抽象协议规范转换成具体协议代码生成的安全验证分析[68]. 为自动分析代码生成的密码协议安全性，文献[69]提出密码协议代码半自动生成分析方法. 这种方法在密码协议安全验证分析时，使用 Prolac[70] 协议语言的静态程序. Prolac 语言编译器把 Prolac 协议规范转换成 C 语言代码程序，一般转换为少量密码协议代码，也称为轻量级代码(不多于 5 行代码)，使用轻量级语法使得小方法更具有可读性. 但

是基于 Prolac 语言编译器的 TCP 协议安全验证分析时间开销不理想，不具有分析各种语言的密码协议安全验证分析的通用性. 对此做出改进，文献[67]提出代码自动生成的安全验证分析. 这种分析方法使用 XML 语言给出协议的规范定义，然后通过编译器自动生成密码协议代码. 基于编译器的代码自动生成分析方法优点是能实现在不同环境下使不同的安全协议快速替换，满足移动终端的需求. 它的不足是不能提供协议的验证功能. 移动终端的密码协议安全属性验证是一个复杂问题，零知识证明协议适用于移动终端的安全属性验证. 文献[71]提出基于零知识证明编译器(ZNCryp)密码协议代码执行的安全验证分析. 这种分析方法验证编译器产生的密码协议代码形式化证明，合并协议代码生成各阶段的安全，保证输出协议代码执行安全. 也就是说，ZNCryp 需要各代码执行阶段合并才能进行验证分析协议的整体安全性. 但实际分析过程中并不能保证各阶段均是安全，因此，ZNCryp 安全验证分析仍有缺陷. 基于编译器的规范语言密码协议安全验证分析，文献[72]提出全面的规范语言和基于 ZK 协议的 ZK-POK(Zero-Knowledge Proofs of Knowledge)编译器分析方法. 这种分析方法克服零知识证明协议安全验证分析的耗时性和易错性，而且用户可以指定任意群同态原像的证明，并通过逻辑 AND 和 OR 运算符将它们结合起来分析协议的安全性. 此外，该方法满足 DAA(Direct Anonymous Attestation)要求，协议安全属性的可靠性在机器辅助定理证明系统(Isabelle/HOL)[73]得到证明.

基于非交互式零知识证明，学者们提出多种协议的安全验证分析方法. 文献[70]提出非交互式代码自动生成分析方法. 该方法分析 C++语言的密码协议代码执的安全性，可有效地降低时间与空间的开销，并且在电子现金上得到实际应用. 但是无法保证协议安全属性的可靠性. 文献[74]提出 ZQL 查询语言编译器的零知识证明协议安全验证分析. 这种方法对客户端数据正确性进行安全验证分析，但缺乏完善的认证功能. 有关编译器代码自动生成的密码协议的安全验证分析请参考上述文献.

基于编译器的密码协议代码执行安全验证分析可以发现协议设计的缺陷，防止协议代码执行时产生漏洞. 但是它有一个前提条件是必须提供协议安全属性的可靠性，而且无法考虑语言结构问题. 因此，该方法并非完美无缺，仍有待完善和改进.

3.2.2　基于 Java 语言代码自动生成分析

Java 是面向对象编程语言，属于一种解释性语言(通过 Java 的虚拟机进行解释)，有良好的语言结构封装性. 所以，Java 虚拟机的安全性可决定程序代码执行的安全性. Java 语言代码自动生成的密码协议安全验证分析存在关系：$M_a \models \varnothing_a \Rightarrow \rho(M_a, J) \models \rho(\varnothing_a)$，在这里 $C=J$，即表示选择 Java 语言. 为解决 ad-hoc 网络通信的易错性，文献[75]提出代码自动生成分析工具 AGVI，用于证明 Java 语言实际代码的执行以及协议的可靠性. 这种密码协议代码执行安全验证分析工具在代码自动生成和执行验证时，可以自动发现并优化密码协议的规范，从而解决 ad-hoc 网络通信的易错性.

基于演算表达的密码协议规范代码自动生成安全验证分析，文献[76]提出抽象符号化演算(例如，CSP(Communication Sequential Processes)，Spi，Hajyle 等)代码自动生成的安全验证分析. 这种分析方法是精化 Mondex[77]协议的抽象状态空间与 Javacar 交互操作，从而分析密码协议代码执行的安全性. 该方法的特点是代码执行的精化和安全属性的可靠性证明，保证协议的安全属性在抽象模型与具体代码执行中成立. 此外，基于常用框架(例如，JML[78]，Spi2Java[79]，JavaSPI[80] 等)密码协安全验证分析，文献[81]提出可利用 Mondex 来进行密码协议安全验证分析. 这种分析方法允许密码协议抽象规范中携带具有证明特征的标识，并对密码协议规范的安全属性进行精化，以便在代码自动生成分析中保证协议安全属性的可靠性. 进一步了解参考上述提到的文献.

密码协议规范经常采用 Spi-演算表示，然后根据这些协议规范自动生成对应的密码协议程序代码. 以此为基础，文献[82]提出通过 Spi-演算的协议规范自动生成协议代码，分析密码协议代码执行的安全性. Spi-演算表达式可参考文献[82]. 基于 Spi-演算，文献[79]提出 Java 语言的密码协议代码自动生成分析方法，称为 Spi2Java 框架分析方法. Spi2Java 框架分析方法用 Spi-演算来表达密码协议规范，自动或半自动地生成密码协议代码，即把 Spi-演算的协议规范转换成 Java 语言可执行的协议代码. Spi2Java 框架分析方法由预处理器、分析器以及安全分析器这 3 个部分组成. Spi2Java 框架的密码协议代码执行安全验证分析的优点是：(1)正确生成密码协议规范对应的 Java 代码；(2)减少编码阶段引入的安全漏洞等. 从规范到代码，本质是一个转换过程. 不妨设，协议规范到生成协议代码之间存在一个转换 $P\sigma$，在这里 P 是 Spi 进程表达式，σ 是绑定 P 的变量替换. 一般的状态转换可以写成：$P_1\sigma \xrightarrow{l}$

$P'\sigma\sigma'_1$，l 表示转换标识符. 由于存在这样的转换，基于 Spi-演算的代码自动生成分析方法不能保证密码协议安全属性的可靠性. 为了保证协议规范到协议代码之间转换的安全属性可靠，文献[83]提出一种基于 Spi2Java 框架代码自动生成的新方法，并在 Spi2Java 框架上验证 SSH-TLP(SSH Transport Layer Protocol)协议安全属性的可靠性. 为保证协议规范转换成代码时，协议的安全属性具有可靠性，文献[84]提出 3 个假设：(1) 对于任意项 M，存在转换假 T，使 $T_s(M)<T(M)$；(2) 对于每个构造函数 C，存在用户数类 $T_s(M)$ 扩展为 $T(M)$；(3) 假设 $Param(M)$ 返回正确值，则该值是用户提供的相互操作参数类型. 文献[85]更进一步提出基于第三方相互操作的可靠性证明方法. 需进一步了解参考文献[84-85].

在上述方法中，代码自动生成分析方法仅限于逻辑核心部分. 根据密码协议的交互性，协议的角色也可以作为规范序列. 如果把角色作为协议规范的一部分，基于 Spi-演算分析方法将无法处理并行输入的情况. 因此，这种方法有局限性：(1) 密码协议规范的序列化和反序列化功能必须经过人工手动编写；(2) 抽象模型输入的限制. 针对这些局限性，文献[81]扩展 Spi2Java 框架，提出新的 Expi2Java[86] 框架分析方法. Expi2Java 框架能满足协议自动规范和协议输入序列化，但是无法提供协议安全属性的可靠性. 文献[80]提出改进的 JavaSPI 框架分析方法，这种改进方法不再将 Spi-演算作为密码协议规范表达，而是以 Java 作为建模和代码实际执语言. 具体的实例化流程如图 7 所示. 更多详细分析见文献[80].

```
Java abstract model
  Message m=new Identifier("Secret message");
  Nonce n=new Nonce();
  SharedKey s=new SharedKey(n);
  SharedKeyCiphered⟨Message⟩mk=
  new SharedKeyCiphered⟨Message⟩(m,s);
Java concrete implementation
  Message m=new IdentifierSR("Secret message");
  Nonce n=new NonceSR("8");
  SharedKey s=new SharedKeySR(n,"DES","64");
  SharedKeyCiphered mk=
  new SharedKeyCipheredSR(m,s,"DES",
  "1234567801g=", "CBC",
  "PKCS5Padding", "SunJCE");
ProVerif model
  new m1;
  new n2;
  let s4=SharedKey(n2) in
  let mk6=SymEncrypt(s4, m1) in
```

图 7 JavaSPI 框架流程

代码自动生成方法借助形式化语义对密码协议进行规范，使用工具自动生成协议代码进行安全验证分析. 代码自动生成安全验证分析是一种行之有效的方法，但这种方法是需在一定假设下才能成立的(不考虑 C 语言的指针越界、缓冲区溢出、Java 语言封装内部程序行为细节等). 这些假设导致密码协议代码实际执行与抽象模型之间存在一定的差距. 如何缩小它们之间的差距？这是值得思考的问题.

3.3 基于操作语义安全验证分析

操作语义密码协议代码执行安全验证分析以密码学原语(例如，离散对数、哈希函数等)为基本的安全单位，来分析协议参与者交互式通信行为的安全性. 一般情况下，这种分析方法只考虑协议通信实体发送信息与接收信息的操作语义安全(例如，发送与接收信息迹的顺序、消息匹配(Message Matching)). 基于操作语义的密码协议代码执行安全验证分析，常使用操作语义工具(例如，Scrypther，Scrypther-proof，Tamarin，Avispa 等)来分析协议的安全性. 下面分别介绍 2 个代表性的操作语义分析工具，对密码协议代码执行安全验证分析.

(1) 基于 Scrypther 工具的密码协议分析

操作语义的密码协议安全验证分析，可以动态分析密码协议交互式通信的行为安全，并可分析并发的多协议代码执行安全. 文献[87]提出新的密码协议验证分析工具，称为 Scrypther. Scrypther 语法类似于 C/Java 语言语法结构. 例如，不妨设一个只有两个参与者的简单协议，用 Scrypther 语法可以表示为

$$\begin{aligned}&Protocol(I,R)\ \{\\&roleI\ \{\ \};\\&roleR\ \{\ \};\\&\};\end{aligned}$$

其中 I，R 分别表示协议的参与者，“{ }”里的内容表示交互动作. 上面简单协议只有两个参与者，无交互内容. Scrypther 工具操作语义密码协议代码安全验证分析过程如下(以 Needham-Schroeder 协议[15] 为例，简称 NS 协议).

① 基本的操作语义语法

BNF 语法[88]：

$Setname::=alt_1\mid alt_2\mid\cdots\mid alt_n$，左边表示推导集合，右边表示推导规则的组成，“|”表示或.

② 密码协议规范：

$RoleEvent_R::=send(\ )\mid recv(\ )$，左边表示 $RoleEvent_R$ 事件，右边的 $send()$、$recv()$ 表示密码协议参与者发送、接收信息事件.

③ 操作语义规则

$$[send]\frac{e=send_l(R_1,R_2,m)\quad \ll AKN,F\gg \xrightarrow{(inst,e)} \ll \bigcup\{\langle inst\rangle(m)\},\ (inst,[e]\cdot s)\in F}{(F\setminus\{inst,[e]\cdot s)\})\bigcup\{(inst,s)\}\gg} \tag{5}$$

$$[recv]\frac{e=recv_l(R_1,R_2,pt)\quad (inst,[e]\cdot s)\in F\quad \ll AKN,F\gg \xrightarrow{(inst',e)} \ll AKN,\ AKN|-m,Match(inst,pt,m,inst')}{(F\setminus\{inst,[e]\cdot s)\})\bigcup\{(inst',s)\}\gg} \tag{6}$$

这里 AKN 表示入侵者拥有的知识，F 表示协议执行的集合，$inst$ 表示实例化，pt 表示匹配模式.

④ NS 协议操作语义表示

NS 协议参与者 I 的操作语义表示.

$$\begin{aligned}NS(I)=(&\{I,R,ni,sk(I),pk(I),pk(R)\}\\&[send1(I,R,\{ni,I|\}pk(R)),\\&recv2(R,I,\{|ni,V|,R\}pk(I)),\\&send3(I,R,\{|V|pk(R)),\\&claim4(I,ni-synch)])\end{aligned} \tag{7}$$

NS 协议参与者 R 的操作语义表示.

$$\begin{aligned}NS(R)=(&\{I,R,nr,sk(R),pk(R),pk(I)\},\\&[recv1(I,R,\{W,I|\}pk(R)),\\&send2(R,I,\{|W,nr|\}pk(I)),\\&recv3(I,R,\{|nr|pk(R)),\\&claim5(R,ni-synch)\end{aligned} \tag{8}$$

I,R 表示协议参与者，ni,nr 表示随机数，V,W 表示存储随机数变量.

⑤ Scrypther 语言(类似于 C/Java)的 NS 协议代码如图 8 所示.

```
1 protocol NS(I,R){
2    role I
3       {
4       const ni: Nonce;
5       var nr: Nonce;
6       send_1(I,R, {I,ni}pk(R));
7       recv_2(R,I, {ni,nr}pk(I));
8       send_3(1,R, {nr}pk(R));
9       claim_i1(I,Secret, ni);
10      claim_i2(I,Secret, nr);
11      claim_i3(I,Niagree);
12      claim_i4(I,Nisynch);
13      }
14   role R{
15   var ni: Nonce;
16   const nr: Nonce;
17   recv_1(I,R,{I,ni}pk(R));
18   send_2(R,I,{ni,nr}pk(I));
19   recv_3(I,R,{nr}pk(R));
20   claim_r1(R,Secret,ni);
21   claim_r2(R,Secret,nr);
22   claim_r3(R,Niagree);
23   claim_r4(R,Nisynch); }
24 }
25
26
```

图 8　NS 协议 Scrypther 语言代码

⑥ Scrypther 语言的 NS 协议代码执行结果如图 9 所示.

Claim				Status		Comments	Patterns
NS	I	NS,i1	Secret ni	OK	Verified	No attacks.	
		NS,i2	Secret nr	OK	Verified	No attacks.	
		NS,i3	Niagree	OK	Verified	No attacks.	
		NS,i4	Nisynch	OK	Verified	No attacks.	
	R	NS,R1	Secret ni	Fail	Falsified	At least 1 attack.	1 attack
		NS,R2	Secret nr	Fail	Falsified	At least 1 attack.	1 attack
		NS,R3	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		NS,R4	Nisynch	Fail	Falsified	At least 1 attack.	1 attack

图 9　NS 协议代码执行结果

⑦ 攻击结果如图 10 所示.

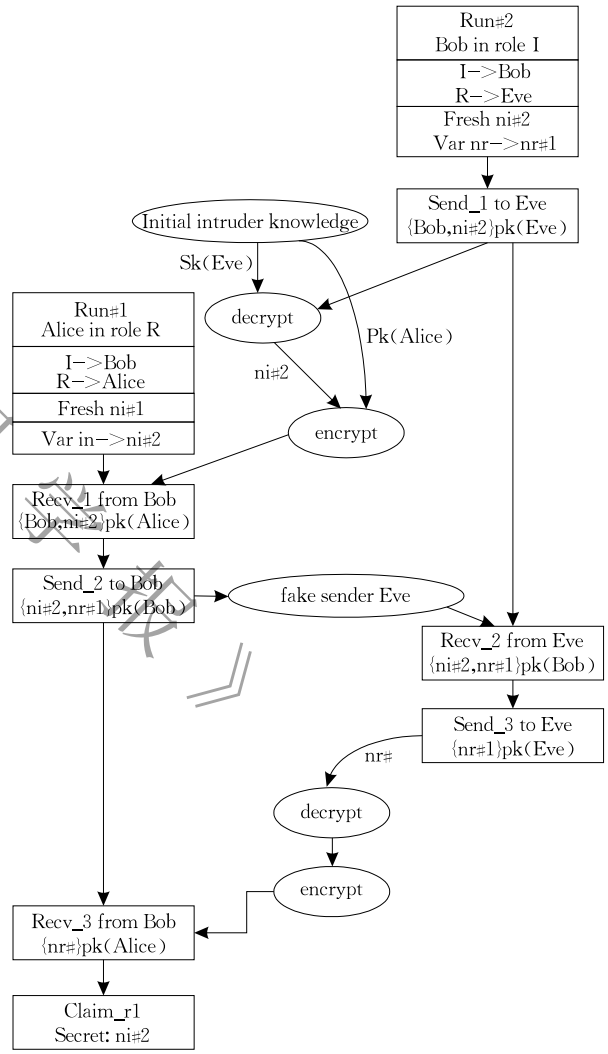

图 10　NS 协议的攻击结果

从密码协议代码执行的分析结果可知，Needham-Schroeder 协议无法抵抗中间人攻击、假冒攻击等，无法确定 n_i 与 n_r 的真实性. 具体分析过程参考文献[87]. Scrypther 工具分析方法的优点是把密码学原语作为基本的安全单位，可自动地分析密码协议通信的行为安全. 不足之处是可能导致密码协议交互

的消息属性传输(例如,同步)失败.具体的密码协议操作语义参见文献[9].

(2)基于 Tamarin 工具的密码协议分析

Tamarin[89]是分析密码协议代码执行安全的操作语义工具,可自动地对密码协议进行安全验证分析.Tamarin 在对密码协议安全验证分析时考虑到协议的并行交互执行情况,并引入敌手攻击.Tamarin 工具分析支持多类密码协议的安全验证分析,例如,Diffie-Hellman 协议[90],TLS 协议[91]以及群组密钥交换协议[92]等.Tamarin 工具的操作语义密码协议代码安全验证分析过程如下(以 Diffie-Hellman 密钥协商协议为例,简称 DH 协议).

① Tamarin 语法

一般安全协议都是由 begin 和 end 来界定.begin 表示协议的开始,end 表示协议的结束.

security_protocol_theory:=

'theory'ident'begin'body'end'.

body:=(signature_spec|rule|axiom|lemma|formal_comment)+.

由于篇幅原因,Tamarin 语法不再详细罗列,请参考 Tamarin 手册①.

② Tamarin 的 DH 协议语义规范.

DH 协议是密钥协商协议,它的安全性是基于离散对数,所以在协议交互过程中考虑指数运算.由 Tamarin 语法可知,DH 协议部分语义表示为

Input

$builtins: diffie\text{-}hellman$

$functions: mac/2, g/0, shk/0\ [private]$ (9)

rulesetp1

$[Fr(tid:fresh), Fr(x:fresh)]-[]\rightarrow$

$[Out(\langle g^x, mac(shk,\langle g^x, a:pub, B:pub\rangle)\rangle),$

$step1(tid:fresh, A:pub, B:pub, x:fresh)]$ (10)

rulestep2

$[step1(tid, A, B, x:fresh),$

$In(\langle Y, mac\langle Y, B, A\rangle\rangle)]$

$-[Accept(tid, Y^x)]\rightarrow[]$ (11)

ruleRevealKey

$[]-[Reveal()]\rightarrow[Out(shk)]$ (12)

properties

$\forall ij\ tid\ key.Accept(tid,key)@i\&k(key)@j$

$\Rightarrow \exists 1.Reavel@l\&l<i$ (13)

output

$Accept\_Secret\ (all-traces)$

$verified\ (9\ steps)$ (14)

③交互模型 Tamarin 语言的 DH 协议部分代码如图 11 所示.

```
Proof scripts
Lemma Accept_Secret_Counter:
  all-traces
  "∀ #i #j tid key.
        ((Accept(tid,key)@#i)  ∧ (K(
key)@#j))  ⇒  (⊥)"
simplilfy
solve(Step1(tid,A,B,~x)  ▷θ #i)
  case Step1
  solve(!KU(mac(shk,<Y,$B,$A>))@
#vk.2)
    case cmac
    solve(!KU(shk)@#vk.5)
      case RevealKey
      solve(splitEqs(θ))
        case split_case_1
        solve(!KU(Y^~x)@#kv.5)
          case Step1
          SOLVED//trace found
          qed
```

图 11 DH 协议 Tamarin 语言部分代码

④交互模型 Tamarin 语言的 DH 协议部分运行结果如图 12 所示.

从密码协议代码执行的结果可知,Diffie-Hellman 协议无法抵抗中间人攻击、假冒攻击等,无法确定 g^x 与 g^y 的安全性.具体分析过程参考文献[89-90].

基于其它操作语义工具的密码协议代码执行安全验证分析,请参考文献[9].操作语义分析工具的演化如表 3 所示.

表 3 Scrther 工具演化

	Scyther	Scyther-proof	Tamarin
Main reference	CCS'08, CAV'08	CSF 2010	CSF 2012 CAV 2013, S&P 2014
Example applications	Compromising adversaries, protocol security hierarchies, IKE, ISO/IEC 9798, ISO/IEC 11770 (key establishment)	ISO/IEC 9798	Naxos, UM, Signed Diffie-Hellman, group protocols, APIs/protocols with global state, ARPKI

① TamarinManual, http://tamarin-prover.github.io/manual

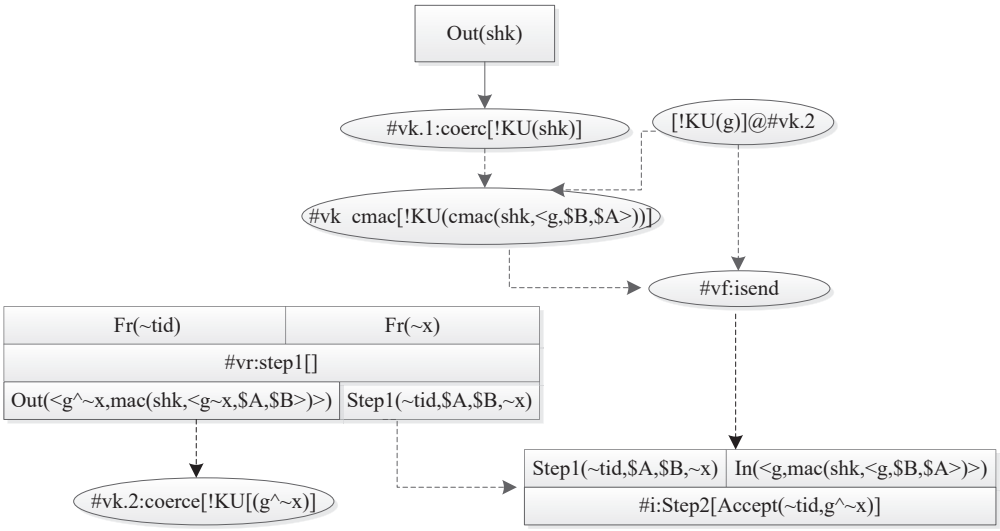

图 12　DH 协议 Tamarin 部分运行结果

基于一般工具的密码协议代码执行安全验证分析可以自动验证或证明协议安全属性是否成立. 常用的协议安全验证分析工具有：Scyther-proof[93]、AVANTSSAR[94]等. 由于篇幅原因，不再一一列举密码协议分析工具，可参考本小节提到相应的文献.

操作语义的密码协议安全验证分析是以密码原语作为基本的安全单位，使用工具分析协议参与者交互通信的行为安全. 虽然可以对密码协议进行自动的安全分析，但是无法保证协议安全属性的可靠性. 如何保证从密码协议的语义安全到代码实际执行安全有待进一步的研究.

3.4　密码协议精化安全验证分析

精化分析通过逐步求精抽象模型与具体模型之间存在的精化关系，使两种模型中的某种属性具有一致性和正确性. 精化分析的理论研究与实践应用在学术界与工业界都得到认可. 常用的精化分析方法有精化检测[95]和程序精化[96]. 基于精化关系的密码协议代码执行安全验证分析研究已取得一些成果[97].

3.4.1　精化检测工具协议分析

精化检测是软件模型属性安全验证分析的常用方法，适用于密码协议代码执行的安全验证分析. 基于高层的精化检测是密码协议安全验证分析的关键. 在分析密码协议安全时，常使用 FDR(Failures Divergence Refinement)[98]、PAT(Process Analysis Toolkit)[99]等精化检测工具来分析密码协议 CSP[100]进程的安全性.

FDR 被应用于并发密码协议的 CSP 进程属性安全验证分析. 文献[101]提出解决并发系统的输入与输出安全性问题，例如，系统的输入输出区别. 但是 FDR 的并发状态空间检测能力有限，对并发协议的 CSP 进程属性的正确性检测效率低. 为提高 FDR 的检测效率，FDR2 对 FDR 进行改进：(1)适应不同的语言；(2)改进多路同步的处理；(3)减少 CSP 规则的限制，严格区分高层与底层结构；(4)丰富的语言表达；(5)引入“缓冲区”概念；(6)提出产生阶段的等价过程. 另外，FDR2 具有验证 Casper[102]工具和 SVA[103]工具后端安全分析的能力. 由于 FDR2 不能提供完善的安全认证机制，FDR3 对 FDR2 作重大的改进：(1)提高系统验证能力；(2)改进它的易用性. 改进的 FDR3 可对并发网络环境下的 CSP 进程和共享内存进行精化分析，并有效检测 CSP 进程的代数模型. 但是 FDR3 无法解决状态空间爆炸问题的安全属性检测，详细参考文献[104].

PAT 精化工具是基于人口协议的可靠性与公平性提出的一种安全验证分析方法. 这种分析方法可处理密码协议不同形式的公平性. 但无法解决网络多结点公平性的安全验证分析. 为解决这一问题，文献[105]提出模型检测与算法分离的密码协议安全分析方法. 这种分析方法使 PAT 精化检测适应于不同语言的密码协议代码执行安全验证分析，详细参考文献[105].

精化工具是分析 CSP 进程属性安全的常用方法. 这种方法在分析分布式系统的密码协议并发执行时具有优势，可对协议的 CSP 进程进行安全验证分析. 但在实际的代码执行中，无法解决程序状态空间协议安全属性验证问题(即在程序状态空间中验证某一属性成立是困难的). 开发优秀的 CSP 进程

分析工具,或完善现有的 CSP 进程分析工具有待进一步研究.

3.4.2 程序精化分析

程序精化分析是验证两个程序之间存在的精化关系.如果程序 C 与程序 C' 存在精化关系,则程序 C 与 C' 之间存在如下性质:(1)精化程序产生的行为是原程序行为的子集;(2)在程序代码执行时,原程序可以被精化程序替换.也就是说,精化的程序不会产生多于原程序的行为.

程序精化的密码协议代码执行安全验证分析有两个方面:(1)基于编译器精化的正确性验证;(2)基于程序与算法精化正确性验证.基于编译器精化的密码协议代码验证分析,文献[106]提出 PILS (Parametric Inter Language Simulations)组合编译器精化安全验证分析方法,称 pilser 编译器精化分析.文献[107]提出基于底层的多方计算精化的密码协议代码执行安全验证分析(密码协代码采用 C++/Java 语言编写).该方法在底层面设计某一领域的特定语言(DSL),并指定多方计算的密码协议编译器精化分析.由于篇幅原因,想要进一步了解可参考上述相应的文献.

为了确保密码协议代码执行过程中的细节安全,文献[108]把密码协议进行分类并执行它的源代码,在代码执行期间逐步精化其过程.文献[109]提出基于严格定义的精化分析方法,这种方法是顺序执行密码协议程序代码,并对它进行逐步精化的安全验证分析.从软件模型精化检测方法可知,密码协议的具体行为不可能完全等价于抽象(即精化前程序)行为,而是必须进行逐步精化,让密码协议具体代码执行的安全更接近抽象理论上证明的安全.

基于精化关系的密码协议源代码执行安全验证分析,文献[110]提出概率 λ-演算验证方法.这种方法验证高阶程序语言系统的密码协议安全性,例如,在对 RF^* 特征进行区别验证时,不再使用 SMT 也可证明精化系统密码协议安全的可靠性.文献[111]提出四级水平逐步精化安全验证分析方法,如表 4 所示.这种分析方法允许开发人员作为构建模型要素逐步纳入系统要求和环境的假设,要求每个模块都有理想函数.在四级水平逐步精化方法的具体实施中,L_1 作为中间协议起到桥梁作用,连接 L_0 协议与消息协议 L_2、L_3.由于篇幅有限,在此只是略为提到逐步精化密码协议安全验证分析,有兴趣的读者不妨参考文献[110,112].

表 4 四级水平逐步数理化

Level	name	features
L_0	security properties	global, protocol-independent
L_1	guard protocols	roles, localstore, nomessages
L_2	channel protocols	security channels, intruder
L_3	crypto protocols	crypto, Dolev-Yao intruder

程序验证是保障软件质量的关键技术,程序精化验证是其中一个重要的研究分支.程序精化应用于密码协议安全验证分析已取得许多成果[112],但还有值得关注的问题.例如,分布式系统网络的密码协议代码并发执行时的安全验证分析,程序语言环境对密码协议代码执行的影响.这些问题在代码级的密码协议安全验证分析研究中都值得思考.

3.4.3 信息流精化分析

程序运行的信息流代表一个程序代码执行轨迹.密码协议代码执行具备信息流的特征(例如,函数调用顺序,程序接口返回信息等).如何利用程序代码信息流特征分析密码协议的安全,或验证密码协议代码执行的安全属性,引起研究者的关注.文献[113]最新提出一种新颖的信息流协议安全验证分析模型,即读写流模(Readers Writers Flow Model).虽然 RWFM 模型在密码协议代码执行的安全验证分析方面具有优势,但未能解决以下 3 个方面的问题:(1)能否适用于协议不同的攻击分析;(2)能否证明协议安全属性的可靠性;(3)能否检测密码协议规范的正确性.为更加准确分析信息流的密码协议代码执行安全,文献[114]提出完善信息流的程序精化安全验证分析.这种分析方法分步精化密码协议程序代码执行时产生的信息流,以便获取特定算法和数据结构等特征.为提高信息流在密码协议安全验证分析的作用,文献[115]提出 C 语言程序和汇编语言程序的信息流在端到端通信方式下密码协议代码执行的安全验证分析.这种方法解决不同地域社区网络上通信的密码协议安全验证分析.

精化的密码协议代码执行安全验证方法,不仅可以验证分析密码协议的安全属性,还可以判断密码协议实际执行的行为安全.这类安全问题有待进一步的研究.

代码级的密码协议安全分析是协议实际执行安全的保障.模型提取、代码自动生成、语义安全以及精化关系为协议实现提供可靠的安全分析,但在缩小协议理论安全与实际执行安全之间的差距仍有待进一步的研究.

4　总结与未来研究方向

代码级的密码协议安全验证分析是协议安全分析的一个新方向.这种新的安全验证分析方法不同于传统的密码协议安全验证分析方法,以密码协议源代码或实际代码执行为研究对象,分析密码协议代码执行的安全性.论文从 4 个方面综述:(1)代码的模型提取:以代码运行的环境与代码的注释作为可信条件,分析密码协议的安全性.优点是避免程序状态空间爆炸问题,不足是忽略程序具体执行的细节,例如 C 语言的指针越界,内存泄露;(2)代码自动生成:采用规范语言(例如,Spi,XML 等)对密码协议进行规范表示,通过具体程序的语言自动生成协议代码,分析密码协议的安全性.优点是可以避免因协议设计考虑不周存在漏洞,不足是存在中间载体(例如,C 语言的编译器),会导致抽象逻辑分析与具体代码执行之间存在差距;(3)操作语义分析:利用具有完善语义的程序语言(例如,F♯语言、Scyther 类 C/Java 语言等)规范密码协议,使用操作语义分析密码协议的安全性.这种方法的优点是接近自然语言,容易发现密码协议存在的攻击,不足是无法分析协议底层安全;(4)程序精化分析:利用密码协议程序代码之间的精化关系,简化程序状态空间,从而分析密码协议的安全性.这种方法的优点是优化程序代码执行的行为,减少不必要的代码冗余和缺陷,节省验证协议安全属性时间开销.它的不足是难以控制精化粒度的大小.除此之外,根据契约规则,契约模块化密码协议的安全验证分析,目前也取得一些成果[116].密码协议代码执行的安全验证分析方法,并不是孤立的,它们之间存在联系,可以有机的组合运用.

代码级的密码协议安全验证分析未来研究工作:

(1)借鉴成熟的模型检测技术(例如,程序精化、程序验证等)对密码协议代码执行安全验证分析.通过证明密码协议的属性在抽象模型与具体模型中具有可靠的一致性和正确性,来分析密码协议代码执行安全验证,有待一步研究.

(2)基于 Dolev-Yao 模型假设的密码协议代码自动化安全验证分析.这类分析常常借助最新的先进工具来实施,如符号化模型自动验证 ProVerift 工具、计算模型验证(或定理证明)CryptoVerif 工具、操作语义验证 Scrything 工具等.改进或开发最新工具来分析代码级的密码协议安全,仍是协议的安全验证分析的研究热点之一.

(3)以程序设计语言(例如,C/C++,Java,F♯等)为基础构建模型或框架(例如,F7,CoSP,Spi2Java 等)的密码协议安全验证分析方法.构建良好模型或框架对密码协议代码执行进行安全验证分析,有待进一步深入.未来这方面的研究趋向于利用组合协议通用模型分析协议的安全.

其次,基于语言结构缺陷(例如,C 语言指针越界,内存泄露,缓冲区以及位的操作等)的代码级密码协议安全验证分析是一个研究难点.

(4)语义安全的密码协议代码执行安全验证分析.以数理逻辑事件,信息控制流(例如,进程、线程等程序内部活动等)为基础对密码协议进行安全验证分析值得关注,且有实际意义的研究方向.

(5)计算可靠的形式化是常用的密码协议安全验证分析热点之一,这种分析方法已有较成熟的计算理论(概率)与计算模型(AR 模型).如何利用现有成熟的计算理论和计算模型分析代码级的密码协议安全验证分析是值得期待的研究之一.

(6)密码协议并发执行是否安全,决定分布式网络空间(例如,大数据,云计算等)信息交换是否安全.并发协议[117]模型提取的代码级的密码协议安全分析是未来研究分布式网络协议安全的趋势.

参　考　文　献

[1] Zhang Huan-Guo, Han Wen-Bao, Lai Xue-Jia, et al. Survey on cyberspace security. Science China Information Sciences, 2016, 46(2): 125-164(in Chinese)
(张焕国, 韩文报, 来学嘉等. 网络空间安全综述. 中国科学: 信息科学, 2016, 46(2): 125-164)

[2] Zhang Rui, Xie Rui, Lin Dong-Dai. The vast cloud security architecture. Science China Information Sciences, 2015, 45(6): 796-816(in Chinese)
(章睿, 薛锐, 林东岱. 海云安全体系架构. 中国科学: 信息科学, 2015, 45(6): 796-816)

[3] Feng Deng-Guo, Zhang Min, Li Hao. Big data security and privacy protection. Chinese Journal of Computers, 2014, 37(1): 246-258(in Chinese)
(冯登国, 张敏, 李昊. 大数据安全与隐私保护. 计算机学报, 2014, 37(1): 246-258)

[4] Cheng Ke-Fei, Weng Jian. Data security and privacy protection in cloud computing environment. Journal of Hangzhou Normal University (Natural Science), 2014, 13(6): 561-571 (in Chinese)
(陈克非, 翁健. 云计算环境下数据安全与隐私保护. 杭州师范大学学报(自然科学版), 2014, 13(6): 561-571)

[5] Cao Zhen-Fu. New development of cryptography. Journal of Sichuan University (Engineering Science Edition), 2015, 47(1): 1-12(in Chinese)
(曹珍富. 密码学的新发展. 四川大学学报(工程科学版), 2015, 47(1): 1-12)

[6] Lei Xin-Feng, Song Shu-Min, Liu Wei-Bing, Xue Rui. A survey on computationally sound formal analysis of cryptographic protocols. Chinese Journal of Computers, 2014, 37(5): 993-1016(in Chinese)
(雷新锋, 宋书民, 刘伟兵, 薛锐. 计算可靠的密码协议形式化分析综述. 计算机学报, 2014, 37(5): 993-1016)

[7] Bhargavan K, Fournet C, Gordon A D, Tse S. Verified interoperable implementations of security protocols. ACM Transactions on Programming Languages and Systems, 2008, 31(1): 1-61

[8] O'Shea N. Verification and Validation of Security Protocol Implementations[Ph. D. dissertation]. School of Informatics, University of Edinburgh, UK, 2010

[9] Cremers C, Mauw S. Operational Semantics and Verification of Security Protocols. Berlin Heidelberg: Springer-Verlag, 2012

[10] Abadi M, Lamport L. The existence of refinement mapping. Theoretical Computer Science, 1992, 82: 253-284

[11] Wei Fu-Shan, Zhang Zhen-Feng, Ma Chuan-Gui. A framework for gateway oriented password authenticated key exchange in the standard mode. Chinese Journal of Computers, 2012, 35(9): 1833-1843(in Chinese)
(魏福山, 张振峰, 马传贵. 标准模型下网关口令认证密钥交换协议的通用框架. 计算机学报, 2012, 35(9): 1833-1843)

[12] Miao Yin-Bin, Ma Jian-Feng, Liu Zhi-Quan, et al. Verifiable search over encrypted data with variable keyword-field. Journal of Xidian University (Natural Science), 2017, 44(1): 57-63 (in Chinese)
(苗银宾, 马建峰, 刘志全等. 关键字域可变的可验证密文检索. 西安电子科技大学学报(自然科学版), 2017, 44(1): 57-63)

[13] Tang Chao-Jing, Lu Zhi-Yong, Feng Chao. A verification logic for security protocols based on computational semantics. Acta Electronic Sinica, 2014, 42(6): 1179-1185(in Chinese)
(唐朝京, 鲁智勇, 冯超. 基于计算语义的安全协议验证逻辑. 电子学报, 2014, 42(6): 1179-1185)

[14] Tian Yuan, Wang Ying, Jin Feng, Jin Yue. Non-malleability and emulation bases approach to cryptographic protocol analysis. Chinese Journal of Computers, 2009, 32(4): 618-634(in Chinese)
(田园, 王颖, 金锋, 金月. 基于刚性与相似性概念的密码协议分析方法. 计算机学报, 2009, 32(4): 618-634)

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

[16] Dolev D, Yao A C. On the security of public key protocols// Proceedings of the 22nd Symposium on Foundation of Computer Science. Oakland, USA, 1981: 350-357

[17] Burrows M, Abadi M, Needham R. A logic of authentication. Palo Alto, USA: Digital Equipment Corp. (DEC), Systems Research Center: 39, 1989

[18] Blu M, Micali S. How to generate cryptographically strong sequences of pseudo random bits//Proceedings of the 23rd Annual Symposium on Foundations of Computer Science. Los Alamitos, USA, 1982: 112-117

[19] Yao A C. Theory and application of trapdoor functions// Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science. California, USA, 1982: 80-91

[20] Goldwasser S, Micali S. Probabilistic encryption. Journal of Computer and System Sciences, 1984, 28(2): 270-299

[21] Abadi M, Rogaway P. Reconciling two views of cryptography: The computational soundness of formal encryption. Journal of Cryptology, 2002, 15(2): 103-127

[22] Micciancio D, Warinschi B. Completeness theorems for the abadi-rogaway logic of encrypted expressions. Journal of Computer Security, 2004, 12(1): 99-129

[23] Blanchet B. Computationally sound mechanized proofs of correspondence assertions//Proceedings of the 20th IEEE Computer Security Foundations Symposium. Venice, Italy, 2007: 97-111

[24] Cortier V, Warinshi B. Computationally sound, automated proofs for security protocol analysis. Logical Methods in Computer Science, 2007, 3(3): 396-458

[25] Backes M, Pfitzman B, Waidner A. A composable cryptographic library with nested operations//Proceedings of the 10th ACM Conference of Computer and Communications Security. New York, UK, 2003: 122-136

[26] Cousot P, Cousot P. Abstract interpretation frameworks. Journal of Logic and Computation, 1992, 2(4): 511-547

[27] Clarke E M, Grumberg O, Long D E. Model checking and abstraction. ACM Transactions on Programming Languages and System, 1994, 16(5): 1512-1542

[28] Goubault-Larrecq J, Parrennes F. Cryptographic protocol analysis on real C code//Proceedings of the International Conference on Verification, Model Checking, and Abstract Interpretation. Paris, France, 2005: 363-379

[29] Menezes A J, van Oorschot P C, Vanstone S A. Handbook of Applied Cryptography. Florida, USA: CRC Press, 1996

[30] Ganesh V, Dill D L. A decision procedure for bit-vectors and arrays//Proceedings of the 19th International Conference on Computer Aided Verification. Berlin, Germany, 2007: 519-531

[31] Wagner D, Foster J S, Brewer E A, Aiken A. A first step towards automated detection of buffer overrun vulnerabilities// Proceedings of the Network & Distributed Systems Security. San Diego, USA, 2000: 3-17

[32] Chaki S, Datta A. ASPIER: An automated framework for verifying security protocol implementations//Proceedings of the 22nd IEEE Computer Security Foundations Symposium. Los Alamitos, USA, 2009: 3-12

[33] Cohen E, Dahlweid M, Hillebrand M, et al. VCC: A practical system for verifying concurrent C//Proceedings of the Theorem Proving in Higher Order Logics, International Conference. Munich, Germany, 2009: 23-42

[34] Chaki S, Ivers J, Sharygina N, Wallnau K. The ComFoRT reasoning framework//Proceedings of the 17th International Conference on Computer Aided Verification. Scotland, UK, 2005: 164-169

[35] Dupressoir F, Gordon A D, Jürjens J, Naumann D A. Guiding a general-purpose C verifier to prove cryptographic protocols. Journal of Computer Security, 2014, 22(5): 823-866

[36] Necula G C, McPeak S, Rahul S P, Weimer W. CIL: Intermediate language and tools for analysis and transformation of C programs//Proceedings of the 11th International Conference on Compiler Construction. London, UK, 2002: 213-228

[37] Aizatulin M, Gordon A D, Jürjens J. Extracting and verifying cryptographic models from C protocol code by symbolic execution//Proceedings of the ACM Conference on Computer and Communications Security. Chicago, USA, 2011: 17-21

[38] Blanchet B, et al. An efficient cryptographic protocol verifier based on Prolog rules//Proceedings of the 14th IEEE Computer Security Foundations Workshop. Cape Breton, Canada, 2002: 82-96

[39] Aizatulin M, Gordon A D, Jürjens J. Computational verification of C protocol implementations by symbolic execution//Proceedings of the ACM Conference on Computer and Communications Security. Raleigh, North Carolina, USA, 2012: 16-18

[40] Blanchet B. A computationally sound mechanized prover for security protocols. IEEE Transactions on Dependable and Secure Computing, 2007, 5(4): 193-207

[41] Bhargavan K, Corin R. Cryptographically verified implementations for TLS//Proceedings of the ACM Conference on Computer and Communications Security. Alexandria, USA, 2008: 27-31

[42] Vanspauwen G, Jacobs B. Verifying protocol implementations by augmenting existing cryptographic libraries with specifications //Proceedings of the Formal Methods-20th International Symposium. Oslo, Norway, 2015: 53-68

[43] Beurdouche B, Bhargavan K, Delignat-Lavaud A, et al. A messy state of the union: Taming the composite state machines of TLS//Proceedings of the IEEE Symposium on Security and Privacy. Oakland, USA, 2015: 535-552

[44] Pullicino K. Jif: Language-based information-flow security in java. https://arxiv. org/pdf/1412. 8639, 2014

[45] Askarov A, Sabelfeld A. Security-typed languages for implementation of cryptographic protocols: A case study//Proceedings of the European Symposium on Research in Computer Security. Milan, Italy, 2005: 197-221

[46] Fortune S, Merritt M. Poker Protocols//Proceedings of the Advances in Cryptology. California, USA, 1984: 454-464

[47] Jürjens J. Using interface specifications for verifying crypto-protocol implementations//Proceedings of the Foundations of Interface Technologies. Budapest, Hungary, 2008: 1-15

[48] Bush W R, Pincus J D, Sielaff D J. A static analyzer for finding dynamic programming errors. Software-Practice and Experience, 2000, 30(7): 775-802

[49] Scott Oaks. Java Security. 2nd Revised Edition. New York, USA: O'Reilly Media, 2001

[50] Jürjens J. Security analysis of crypto-based java programs using automated theorem provers//Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering. Tokyo, Japan, 2006: 167-176

[51] Jürjens J. Automated security verification for crypto protocol implementations. Verifying the Jessie Project Electronic Notes in Theoretical Computer Science, 2009, 250(1): 123-136

[52] Jia L, Sen S, Garg D, Datta A. A logic of programs with interface-confined code//Proceedings of the 28th Computer Security Foundations Symposium. Verona, Italy, 2015: 512-525

[53] Jia L, Sen S, Garg D, Datta A. System M: A program logic for code sandboxing and identification. arxiv. org/abs/1501. 05673, 2015

[54] Datta A, Derek A, Mitchell J C, Pavlovic D. A derivation system and compositional logic for security protocols. Journal of Computer Security, 2005, 13: 423-482

[55] Bhargavan K, Fournet C, Corin R, Zălinescu E. Verified cryptographic implementations for TLS. ACM Transactions on Information and System Security, Special Issue on Computer and Communications Security, 2012, 15(1): 1-32

[56] Backes M, Maffei M, Unruh D. Computationally sound verification of source code//Proceedings of the ACM Conference on Computer and Communications Security. Chicago, USA, 2010: 387-398

[57] Bengtson J, Bhargavan K, et al. Refinement types for secure implementations//Proceedings of the 21st IEEE Security Foundations Symposium. Pennsylvania, USA, 2008: 17-32

[58] Backes M, Hofheinz D, Unruh D. CoSP: A general framework for computational soundness proofs//Proceedings of the ACM Conference on Computer and Communications Security. Chicago, USA, 2009: 66-78

[59] Backes M, Mohammadi E, Ruffing T. Computational soundness results for ProVerif bridging the gap from trace properties to uniformity//Proceedings of the 3rd Conference on Principles of Security and Trust. Grenoble, France, 2014: 42-62

[60] Milner R. Communicating and Mobile Systems: The π-Calculus. Cabridge: Cabridge University Press, 1999

[61] Backes M, Mohammadi E, Ruffing T. Computational soundness for interactive primitives//Proceedings of the European Symposium on Research in Computer Security. Vienna, Austria, 2015: 125-145

[62] Bengtson J, Bhargavan K, Fournet C, Gordon A D. Modular verification of security protocol code by typing//Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming. Madrid, Spain, 2010: 17-23

[63] Milner R, Harper R, MacQueen D, Tofte M. The Definition of Standard ML(Revised). Massachusetts, USA: MIT Press, 1997

[64] Bangerter E, Camenisch J, Krenn S, et al. Automatic generation of sound zero-knowledge protocols//Proceedings of the European Cryptology Conference Poster Session. Cologne, Germany, 2009: 471

[65] Meiklejohn S, Erway C C, Küpçü A, et al. ZKPDL: A language-based system for efficient zero-knowledge proofs and electronic cash//Proceedings of the 19th USENIX Conference on Security. Washington, USA, 2010: 193-206

[66] Backes M, Busenius A, Hritcu C. On the Development and Formalization of an Extensible Code Generator for Real Life Security Protocols. Berlin Heidelberg, Germany: Springer, 2012, 7226: 371-387

[67] Kiyomoto S, Ota H, Tanaka T. A security protocol compiler generating C source codes//Proceedings of the International Conference on Information Security and Assurance. Busan, Korea, 2008: 20-25

[68] Dembinski P, Budkowski S. Specification Language Estelle. Springer International, 1998, 124(7): 9-10

[69] Kohler E, Kaashoek M F, Montgomery D R. A readable TCP in the prolac protocol language. ACM SIGCOMM Computer Communication Review, 2000, 29(4): 3-13

[70] Sidhu D, Chung A, Blumer T P. A Formal Description Technique for Protocol Engineering [Ph. D. dissertation]. University of Maryland, College Park, Maryland, USA, 1990

[71] Almeida J B, Barbosa M, Bangerter E. Full proof cryptography: Verifiable compilation of efficient zero-knowledge protocols//Proceedings of the ACM Conference on Computer and Communications Security. Raleigh, USA, 2012: 488-500

[72] Almeida J B, Bangerter E, Barbosa M, et al. A certifying compiler for zero-knowledge proofs of knowledge based on Σ-protocols//Proceedings of the European Symposium on Research in Computer Security. Athens, Greece, 2010: 151-167

[73] Paulson L. Isabelle: A Generic Theorem Prover. Berlin, Germany: Springer-Verlag, 1994

[74] Fournet C, Kohlweiss M, Danezis G, Luo Zhengqin. ZQL: A compiler for privacy-preserving data processing//Proceedings of the 22nd USENIX Security Symposium. Washington, USA, 2013: 14-16

[75] Song D, Perrig A, Phan D. AGVI—Automatic generation, verification, and implementation of security protocols//Proceedings of the Computer Aided Verification. Paris, France, 2102: 241-245

[76] Hubbers E, Oostdijk M, Poll E. Implementing a formally verifiable security protocol in java card//Proceedings of the 1st International Conference on Security in Pervasive Computing. Boppard, Germany, 2013: 213-226

[77] Schellhorn G, Banach R. A concept-driven construction of the Mondex protocol using three refinements//Proceedings of the Abstract State Machines, B and Z, First International Conference. London, UK, 2008: 57-70

[78] Leavens G T, Baker A L, Ruby C. Preliminary design of JML: A behavioral interface specification language for java. ACM SIGSOFT Software Engineering Notes, 2006, 31(3): 1-38

[79] Pozza D, Sisto R, Durante L. Spi2Java: Automatic cryptographic protocol java code generation from spi calculus//Proceedings of the 18th International Conference on Advanced Information Networking and Application. Fukuoka, Japan, 2004: 400-405

[80] Avalle M, Pironti A, Sisto R, Pozza D. The JavaSPI framework for security protocol implementation//Proceedings of the 6th International Conference on Availability, Reliability and Security. Vienna, Austria, 2011: 746-751

[81] Grandy H, Bischof M, Stenzel K, et al. Verification of Mondex electronic purses with KIV: From a security protocol to verified code//Proceedings of the 15th International Symposium on Formal Methods. Turku, Finland, 2008: 165-180

[82] Pironti A, Sisto R. Provably correct java implementations of spi Calculus security protocols specifications. Computers & Security, 2010, 29: 302-314

[83] Pironti A, Pozza D, Sisto R. Formally-based semi-automatic implementation of an open security protocol. Journal of Systems and Software, 2012, 85(1): 835-849

[84] Pironti A. Sound Automatic Implementation Generation and Monitoring of Security Protocol Implementations from Verified Formal Specifications [Ph. D. dissertation]. Politecnico di Torino, Torino, Italy, 2010

[85] Hahnle R, Heisel M, Reif W, Stephan W. An interactive verification system based on dynamic logic//Proceedings of the International Conference on Automated Deduction. Oxford, England, 1986: 306-315

[86] Busenius A. Expi2Java: An Extensible Code Generator for Security Protocols[Ph. D. dissertation]. Saarland University, Saarland, 2008

[87] Cremers C J F. The scyther tool: Verification, falsification and analysis of security protocols//Proceedings of the Computer Aided Verification, 20th International Conference. Princeton, USA, 2008: 414-418

[88] Chomsky N. Three models for the description of language. IRE Transactions on Information Theory, 1956, 2(3): 113-124

[89] Meier S, Schmidt B, Cremers C, Basin D. The TAMARIN prover for the symbolic analysis of security protocols//Proceedings of the Computer Aided Verification. Saint Petersburg, Russia, 2013: 696-701

[90] Schmidt B, Meier S, Cremers C, Basin D. Automated analysis of Diffie-Hellman protocols and advanced security properties//Proceedings of the Computer Security Foundations Symposium. Cambridge, USA, 2012: 78-94

[91] Cremers C, Horva M, Scott S, van der Merwe T. Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication//Proceedings of the 2016 IEEE Symposium on Security and Privacy. San Jose, USA, 2016: 470-485

[92] Schmidt B, Sasse R, Cremers C, Basin D. Automated verification of group key agreement protocols//Proceedings of the 2014 IEEE Symposium on Security and Privacy. California, USA, 2014: 179-194

[93] Meier S, Cremers C J F, Basin D A, Strong invariants for the efficient construction of machine-checked protocol security proofs//Proceedings of the 23rd IEEE Computer Security Foundations Symposium. Edinburgh, UK, 2010: 231-245

[94] Armando A, Arsac W, Avanesov T, et al. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures//Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Tallinn, Estonia, 2012: 267-282

[95] Wang Ting, Chen Tie-Ming, Liu Yang. Refinement checking based on simulation relations. Journal of Software, 2016, 27(3): 580-592(in Chinese)
(王婷，陈铁明，刘杨. 基于模拟关系的精化检测方法. 软件学报，2016，27(3)：580-592)

[96] Liang Hong-Jing. Refinement Verification of Concurrent Programs and Its Applications[Ph. D. dissertation]. University of Science and Technology of China, Hefei, 2014(in Chinese)
(梁红瑾. 并发程序精化验证及其应用[博士学位论文]. 中国科技大学，合肥，2014)

[97] Avalle M, Pironti A, Sisto R. Formal verification of security protocol implementations: A survey. Formal Aspects of Computing, 2014, 26(1): 99-123

[98] Gibson-Robinson T, Armstrong P, Boulgakov A, Roscoe A W. FDR3: A modern refinement checker for CSP//Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Grenoble, France, 2014: 187-201

[99] Sun J, Liu Y, Dong J S, Pang J. PAT: Towards flexible verification under fairness//Proceedings of the 21st International Conference on Computer Aided Verification. Grenoble, France, 2009: 709-714

[100] Hoare C A R. Communicating sequential processes. Communications of the ACM, 1978, 21(8): 666-677

[101] Roscoe A W. CSP and determinism in security modelling//Proceedings of the IEEE Symposium on Security and Privacy. Oakland, USA, 1995: 114-127

[102] Lowe G. Casper: A compiler for the analysis of security protocols. Journal of Computer Security, 1998, 6(2): 18-30

[103] Roscoe A W, Hopkins D. SVA: A tool for analysing shared-variable programs. Proceedings of AVoCS, 2007, 4(1): 14-25

[104] Gibson-Robinson T, Armstrong P, Boulgakov A, Roscoe A W. FDR3: A parallel refinement checker for CSP. International Journal on Software Tools for Technology Transfer, 2016, 18(2): 149-167

[105] Liu Yang, Sun Jun, Dong Jin-Song. PAT 3: An extensible architecture for building multi-domain model checkers//Proceedings of the 22nd IEEE International Symposium on Software Reliability Engineering. Hiroshima, Japan. 2011: 190-199

[106] Neis G, Hur C-K. Pilsner: A compositionally verified compiler for a higher-order imperative language. ACM SIGPLAN Notices, 2015, 50(9): 166-178

[107] Laud P, Randmets J. A domain-specific language for low-level secure multiparty computation protocols//Proceedings of the ACM Conference on Computer and Communications Security. Denver, Colorado, USA, 2015: 12-16

[108] Bieber P, Cuppens N B. Formal development of authentication protocols//Proceedings of the 6th Refinement Workshop. London, UK, 1994: 5-7

[109] Morgan C. The Shadow Knows: Refinement and security in sequential programs. Science of Computer Programming, 2009, 74: 629-653

[110] Barthe G, Fournet C, Grégoire B. Probabilistic relational verification for cryptographic implementations//Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. San Diego, USA, 2014: 22-24

[111] Sprenger C, Basin D. Refining key establishment//Proceedings of the IEEE 25th Computer Security Foundations Symposium. Cambridge, USA, 2012: 230-246

[112] Sprenger C, Basin D. Developing security protocols by refinement//Proceedings of the ACM Conference on Computer and Communications Security. Chicago, USA, 2010: 361-374

[113] Narendra Kumar N V, Shyamasundar R K. POSTER: Dynamic labelling for analyzing security protocols//Proceedings of the ACM Conference on Computer and Communications Security. Denver, USA, 2015: 12-16

[114] McIver A K. Program refinement, perfect secrecy and information flow. Engineering Trustworthy Software Systems, 2016, 9506: 80-102

[115] Costanzo D, Shao Zhong, Gu Ronghui. End-to-end verification of information-flow security for C and assembly programs//Proceedings of the ACM SIGPLAN Symposium on Programming Language Design & Implementation. Santa Barbara, USA, 2016: 13-17

[116] Shmatikov V, Mitchell J C. Finite-state analysis of two contract signing protocols. Theoretical Computer Science, 2002, 283: 419-450

[117] Yao A C-C, Yung Moti, Zhao Yunlei. Concurrent knowledge extraction in public-key models. Journal of Cryptology, 2016, 29(1): 156-219

ZHANG Huan-Guo, born in 1945, professor. His research interests include information security, cryptography, trusted computing, etc.

WU Fu-Sheng, born in 1974, Ph. D. candidate. His research interests include information security, security verification analysis of protocols implement on real code.

WANG Hou-Zhen, born in 1981, Ph. D., lecturer. His research interests include information security, cryptography.

WANG Zhang-Yi, born in 1978, Ph. D., lecturer. His research interest is information security protocols.

Background

This work is supported by the National Natural Science Foundation of China (Nos. 613030212, 61202385), the State Key Program of National Natural Science of China (No. 61332019), and the National Basic Research Program (973 Program) of China (No. 2014CB340600).

Currently, the security verification analysis of cryptographic protocols is one of the most important parts of the secure information in network. Traditional analysis methods can verify the security of the cryptographic protocols theoretically, but it can't guarantee the implementation of cryptographic protocols on real code to be safe. Therefore, it is worth focusing on the study of security verification analysis of cryptographic protocols on real code. This paper summarizes the research status of security verification analysis of cryptographic protocols. Meanwhile we compare, analyze, summarize and comment the latest achievements in the research of security verification analysis of cryptographic protocols on real code. We prospect the study direction of security verification analysis of cryptographic protocols implemented on real code.