智能合约的合约安全和隐私安全研究综述

胡甜媛[1)] 李泽成[2)] 李必信[1)] 包骐豪[1)]

[1)] 东南大学计算机科学与工程学院，南京，210096

[2)] 香港理工大学计算系，香港，999077

摘　要　区块链作为对等网络中的一种分布式账本技术，集成了密码学、共识机制、智能合约等多种技术，提供一种新型信任体系构建方法。智能合约具有公开透明、实时更新、准确执行等显著特点，在区块链中为信息存储、交易执行和资产管理等功能的实现提供了更安全、高效、可信的方式。但是，智能合约本身仍然存在安全问题，影响了区块链技术的进一步推广使用。所以，近年来围绕智能合约安全问题的相关研究比较多，为了帮助相关人员更好地理解和掌握其中的研究思路，本文采用 Mapping study 方法，通过收集 2015 年以来公开发表的关于智能合约安全问题的各类文献，并进一步通过文献筛查、问题设置、信息提取、结果获取和分析等步骤，总结智能合约安全相关研究的现状和未来发展趋势如下：1）目前智能合约自身面临的安全问题和挑战主要体现在合约安全和隐私安全两方面（问题和挑战）。在调查的 45 篇文献中，有 29 篇主要文献针对合约安全，16 篇文献针对隐私安全。2）智能合约安全保障目前采用的方法主要包括形式化验证、模糊测试、零知识证明、可信执行环境等（保障方法）。3）针对合约安全的研究目前主要集中在合约实现、测试阶段，而针对智能合约设计、部署及运维阶段的研究比较少；针对隐私安全的研究主要集中在合约数据隐私保护，而针对合约代码隐私安全的比较少（覆盖范围）。4）智能合约安全保障研究目前主要从合约实现人员、合约测试人员的角度进行，而从合约维护人员和合约用户角度展开的研究较少（研究角度）。5）未来研究应该围绕智能合约的全生命周期的每个阶段安全问题进一步推进，先验方法和后验方法、定性方法和定量方法、静态方法和动态方法的结合是大势所趋（发展趋势）。综上，本文通过调研发现了现有研究的不足，并建议了进一步的研究工作。

关键词　区块链；智能合约；合约安全；隐私安全

中图法分类号　TP309

Contractual Security and Privacy Security of Smart Contract: A System Mapping Study *

HU Tian-Yuan[1)]　LIZe-Cheng [2)]　LI Bi-Xin[1)]　BAO Qi-Hao[1)]

[1)](School of Computer Science and Engineering, Southeast University, Nanjing 211189, China)

[2)](Department of Computing, Hong Kong Polytech University, Hong Kong 999077)

Abstract　As a distributed ledger technology in a peer-to-peer network, blockchain integrates cryptography, consensus mechanisms, smart contracts and other technologies to provide a new trust system construction method. The smart contract, which is transparent, real-time, deterministic, provides a safer, more efficient, and credible way for the realization of functions such as storage, transaction execution, and asset management. However, the smart contract itself still suffers from some vulnerabilities, which hinders its further promotion and adoption. Therefore, there has been plenty of related research on smart contract security issues recently. In this

本课题得到国家重点研发计划项目(No.2019YFE0105500)、国家自然科学基金(No.61872078)资助.**胡甜媛**，博士研究生，计算机学会(CCF)会员(A1495G)，主要研究领域为软件工程、区块链安全.E-mail: tianyuan.hu@foxmail.com. 李泽成，博士研究生，计算机学会(CCF)会员(F8380G)，主要研究领域为区块链安全. Email：cszcli@comp.polyu.edu.hk.李必信(**通信作者**)，博士教授，计算机学会(CCF)会员(06247S)，主要研究领域为软件工程、区块链安全.E-mail: bx.li@seu.edu.cn.**包骐豪**，博士研究生，计算机学会(CCF)会员(D7650G)，主要研究领域为软件工程、区块链安全. E-mail: qihao_bao@seu.edu.cn

paper, we adopt the mapping study method and collect published papers on smart contract security since 2015. Through literature screening, problem sets, information extraction, result acquisition, and analysis, we summarize the smart contract security research status and future trends as follows: 1) Current security problems and challenges are mainly reflected in contract security and privacy security. Among the 45 documents surveyed, 29 focused on contract security, and 16 focused on privacy security. 2) The main contract security enhancement techniques include formal verification, fuzzing, zero-knowledge proof, and trusted execution environment. 3) Existing contract security researches mainly focused on implementing and testing smart contracts. By contrast, little research explored contract security problems in design, implementation, and runtime stages; Current privacy security researches mainly focused on contract data privacy rather than contract code privacy. 4) The research on smart contract security is mainly carried out from the perspective of contract implementation and testing staff, while there is relatively little research from the perspective of maintainersand users. 5) Future researches should focus on the security issues at each stage of the smart contract lifecycle. The combination of priori and posterior methods, qualitative and quantitative methods, static and dynamic methods is an irresistible trend. To sum up, this paper has found the deficiencies of existing researches through investigation and suggested further research directions.

Key words blockchain; smart contract; contract security; privacy security; mapping study

1 引言

2009 年，比特币开启区块链时代，区块链技术是比特币的基础，它支持去中心化计算模式的转变。区块链系统应用传统的密码学原理结合区块链特有的数据结构实现了安全的数据存储和共享。在 P2P 网络中，共识机制为区块链系统提供了一种剔除可信第三方的可信数据共享机制，为上层应用提供安全的账本支持，区块链在不可信环境下建立起信任关系，实现价值传递和信任传递。并且由于智能合约的引入，区块链应用已经远远超出加密货币的范畴，以智能合约为代表的第二代区块链平台及应用呈现爆发性增长[1]。智能合约作为一种可自动执行的数字化协议，为区块链增添可编程属性，为实现可编程社会提供可能[2]。智能合约与区块链的结合被认为是区块链世界中一次里程碑式升级，随着第一个结合区块链与智能合约的平台以太坊的诞生，迎来“区块链 2.0”时代。

目前，区块链技术在各个领域的应用范围越来越广，区块链的安全性和隐私保护特性是推动区块链长远发展的重要因素。然而，由于这一新兴技术发展时间较短、尚不成熟，各类安全事故频频发生，引起社会各界对区块链安全问题的广泛关注。

针对以太坊去中心化应用（Ethereum Decentralized Application， ETH DApp），本文统计了 2016 年 1 月至 2020 年 6 月发生的 16 起重大攻击事件如表 1 所示①。通过表 1 看出，16 起事件中有 13 起事件是由于智能合约被攻击导致的，智能合约自身存在的缺陷、合约之间的调用等都是影响合约安全的重要因素。

表 1 ETH DApp 攻击事件

序号	攻击时间	攻击目标	损失金额	攻击原因
1	2016-06-17	The DAO	$60,000,000	**重入攻击**
2	2017-07-19	Parity	$31,000,000	**DelegatedCall 调用**
3	2017-11-06	Parity	$1,500,000,000	**无保护自杀**
4	2018-04-22	BEC	$6,000,000,000	**整数溢出**
5	2018-07-10	Bancor	$12,000,000	疑似私钥被盗
6	2018-07-31	Fomo 3D	-	**合约伪随机数生成**
7	2018-08-01	Fomo 3D	$3,000,000	贪婪挖矿激励机制
8	2018-10-09	SpankChain	$40,000	**重入攻击**
9	2020-02-15	bZx	$350,000	**多个智能合约调用**
10	2020-02-18	bZx	$645,000	**多个智能合约调用**
11	2020-04-18	Uniswap	$220,000	**合约重入攻击**
12	2020-04-19	Lendf.Me	$25,000,000	**合约重入攻击**
13	2020-04-25	Hegic	$28,537	**合约漏洞**
14	2020-50-18	tBTC	-	**合约漏洞**
15	2020-06-18	Bancor	$135,229	**合约函数漏洞**
16	2020-06-23	DDM	$40,000	恶意劫持

目前对于区块链技术的安全和隐私保护，业界处于初级探索阶段。频发的智能合约攻击事件严重威胁区块链生态安全，亟需智能合约安全保障技术支持。于是，很多人开始重视并研究智能合约安全，相关研究成果也越来越多。然而，由于研究者来自不同学科领域，大家关注角度不同，采用的技术方法不同，解决的安全问题和预期目标也不同，所以现有研究成果缺乏一定系统性，研究思路也不够清晰，无法为解决智能合约安全问题形成有力指导。

基于当前智能合约安全的研究工作，本文旨在

①ETH DApp attacks,https://hacked.slowmist.io/?c=ETH%20DApp

通过深度调研并分析已有智能合约涉及的合约安全和隐私安全方面的相关研究，发现当前研究中存在的不足，提出需要解决的问题并建议有潜力的研究方向。为此，首先根据 Mapping Study（映射研究）方法确定 3 类感兴趣的问题，然后在选定的电子数据库中广泛检索 2015 年至 2020 年发表的智能合约安全方面的论文，选择有代表性的 45 篇论文，并根据需求信息模板从每篇论文中提取关注的信息，最后通过对信息深度分析、归纳和总结，回答上述 3 类研究问题，发现现有研究不足并进一步建议研究方向。具体来说，本文主要贡献如下：

（1）分析了智能合约安全问题和挑战。从智能合约的生命周期出发，分析智能合约在设计、实现、测试、部署、运维不同阶段面临的安全问题和挑战，以及隐私安全涉及的合约代码安全和合约数据安全面临的安全问题和挑战，明确了研究目标。

（2）总结了智能合约安全保障方法。针对智能合约安全面临的问题和挑战，从智能合约生命周期的角度归纳整理不同阶段的安全保障方法，主要包括合约设计、合约实现、合约测试和合约部署及运维安全保障方法；从合约代码隐私和合约数据隐私角度归纳整理智能合约隐私安全保障方法。

（3）发现了现有方法存在的不足。针对不同的智能合约安全问题和保障方法，分析现有研究的优势和不足。大量研究集中在合约实现、测试阶段，其他阶段的相关研究较少；已有的智能合约测试方法未形成全面、体系的智能合约检测框架；对运行中的智能合约，未实现灵活、有效的合约维护、升级机制；已有的智能合约隐私保护方法存在性能不高、可扩展性较低、未实现完全去中心化等不足。

（4）建议了进一步研究方向和目标。建议未来研究要做到先验方法和后验方法融合、定性方法和定量方法融合、静态方法和动态方法融合。

2 基本术语

2.1 智能合约

智能合约是被部署在区块链上可自动执行的数字化协议，也是可按照预设合约条款自动执行的计算机程序，主要包含相关代码和数据集。智能合约作为区块链的激活器，其自动化和可编程特性使其可以封装分布式区块链系统中各节点的复杂行为，成为区块链构成的虚拟世界中的智能软件代理机器人。本质上，智能合约既不是合约也不智能，具有较低的法律效应，目前智能合约规则也无法自动化实现。但区块链上智能合约具有众多特性，如可编程、不可篡改、去信任等，通过智能合约可以灵活嵌入各种数据和数字资产，安全高效地交换信息、管理资产。区块链中数字资产可以分为数字货币与非数字货币，由关于该资产的数字文件实现，区块链上交易的数字资产类型日渐丰富，有能源、实体资产和知识产权等，由现实中的资产数字化而来。智能合约为传统商业模式变革带来可能，为构建可编程资产、系统和社会奠定基础。

2.2 智能合约应用平台

目前，具有代表性的智能合约应用平台主要是以太坊（Ethereum）和超级账本（HyperledgerFabric）。其中，以太坊作为公有链的代表，其智能合约主要采用 Solidity 语言编写；超级账本作为联盟链（私有链）的代表，其智能合约又称为链码（Chaincode），由 Java 或 Golang 语言编写。以太坊作为最早的智能合约平台，使用较为广泛，因此本文智能合约安全研究主要围绕以太坊平台展开。

以太坊作为去中心化应用平台，其智能合约负责存储业务逻辑（程序代码）和应用程序的相关状态。与比特币、莱特币等搭建自己的区块链运行平台的加密数字货币不同，通过以太坊发布的加密数字货币一般称为代币（Token），代币没有自己的区块链，而是以智能合约形式运行在以太坊平台上。以太坊不仅仅支持数字货币，还通过智能合约支持更多应用场景，例如金融、游戏、公证等。

在以太坊中，一个智能合约实际上是一串代码的合集，包括代码中各种函数以及代码运行过程产生的各种状态。以太坊支持多种高级编程语言，专门的编译器将合约编译成字节码后部署到区块链。发布者在以太坊平台发布合约后，智能合约作为一段程序被运行在网络中所有节点上，合约发布者无法修改已发布的合约。由于智能合约一般运行在隔离的沙箱执行环境中，智能合约会被所有节点执行，并且任何人都可以发布执行合约，为了防止以太坊网络发生蓄意攻击或滥用的现象，以太坊协议规定交易或合约调用的每个运算步骤都需要收取费用，gas 就是该费用的计数单位。

由于智能合约代码涉及数字资产，一旦合约代码缺陷被利用可能造成巨大损失，所以如何保证智能合约安全至关重要。

2.3 智能合约安全

目前，智能合约发展仍然处于初级阶段，传统编程模式和软件生命周期需要进行有效的改造来适应智能合约的安全需求。区块链系统应用智能合约来实现复杂的交易，区块链系统要求合约的数据及其在数据上运行的代码都要公开，每个矿工都要模拟执行合约，该过程不仅涉及智能合约本身的安全问题，还存在隐私泄露的风险。本文认为智能合约安全主要体现为合约安全和隐私安全。

合约安全主要针对合约设计文档、合约代码、合约运行状态等，要求智能合约不存在设计缺陷、代码漏洞等不足，涉及智能合约的设计、实现、测试、运维及部署等多个阶段。

隐私安全主要针对智能合约代码隐私及合约执行过程中涉及的相关数据隐私，要求保护程序代码以及程序执行过程中的交易数据不会被非授权节点获取，主要体现在区块链系统的合约代码和合约数据两方面。

3 综述方法

Mapping Study（映射研究）是系统性文献综述（Systematic Literature Review，SLR）的一种，其目的是对某一研究领域进行广泛的概述[3]。本文之所以选择 Mapping Study 作为调查方法，是因为本文的目标是分析智能合约安全的研究现状，总结现有研究取得的成果，探索智能合约安全的未来研究方向。本文遵循系统性文件综述的方法，参考 Kitchenham 和 Charters[3]提出的指南，根据 Mapping Study 方法的思想，按照以下顺序进行调研分析：

（1）设置研究问题：针对智能合约安全定义相关研究问题，包括智能合约利益相关者、安全问题及挑战、安全保障方法、保障方法验证等；

（2）确定文献筛选策略并选择主要文献：针对智能合约安全主题，定制检索关键词，在选定数据库中检索相关研究，并筛选出主要文献；

（3）评估主要文献质量：定制质量评估标准，评估主要文献是否满足质量标准；

（4）定义信息抽取模板，抽取主要文献的关键信息：定制信息抽取模版，根据信息抽取模版中的规则抽取与研究问题相关的主要内容。

（5）抽取结果及分析：报告信息抽取的结果，并进行关联性、趋势性分析等（参见第 4 节）。

（6）讨论现有方法的不足，归纳总结未来可能的研究方向（参见第 5 节）。

下面，介绍综述研究的各个步骤。

3.1 研究问题

在任何系统性综述中，明确研究问题最重要[3]。本文从智能合约研究人员的角度，总结提炼出如下三类研究问题（如表 2 所示）：通识问题（General Question，GQ）、聚焦问题（Focused Question，FQ）和统计问题（Statistical Question，SQ）。

表 2 研究问题列表

问题类别	具体问题
通识问题	
GQ1	智能合约安全涉及哪些利益相关者？
GQ2	智能合约面临哪些安全问题和挑战？
GQ3	智能合约安全主要有哪些保障方法？
GQ4	智能合约安全保障方法如何验证？
聚焦问题	
FQ1	引起智能合约安全威胁的原因有哪些？
FQ2	现有智能合约的合约安全保障方法有效性如何？
FQ3	现有智能合约的隐私安全保障方法有效性如何？
FQ4	现有研究方法有哪些不足？
统计问题	
SQ1	智能合约安全的研究热度如何？
SQ2	智能合约安全的研究报告或研究论文发表在哪些载体上？

通识问题（GQs）涉及到区块链系统中智能合约安全相关的通识问题。GQ1 关注谁参与智能合约安全保障过程？智能合约从设计到使用涉及多个阶段，智能合约安全保障包含多个方面，完整安全保障过程有哪些人参与，都是直接影响智能合约安全的主导因素。GQ2 关注智能合约目前存在哪些安全问题和挑战？真实存在的安全问题是引起研究者对智能合约安全产生关注的主要原因；鉴于已有的智能合约安全问题，智能合约安全保障技术发展面临哪些挑战？GQ3 关注如何通过具体的方法保障智能合约的合约安全和隐私安全？针对已有的不同类型的问题和挑战，现有研究通过哪些安全保障技术实现了合约安全保障和隐私安全保障？GQ4 关注智能合约安全保障方法如何进行验证？不同的安全保障方法实现不同目的，其安全保障对象和安全保障形式各有不同。

聚焦问题（FQs）在回答了 GQs 问题的基础上，对智能合约安全做出进一步的分析，主要分析现有的智能合约安全保障技术相关的具体问题以及现有研究不足。其中，FQ1 聚焦于智能合约安全威胁产生的原因；FQ2 关注智能合约的合约安全保障方法的有效性；FQ3 关注智能合约的隐私安全保障方法的有效性；最后，FQ4 关注现有智能合约的合约安全和隐私安全保障方法的优势和不足。

从文献发表的角度来看，统计问题（SQs）提供了另一种观察当前研究现状的方式。SQ1 关注文

献的发表时间和数量；SQ2 关注文献发表的载体，可以看出相关研究的趋势和成熟度。

3.2 研究策略与选择

在确定了研究问题的基础上，本小节主要介绍如何选择研究问题相关的文献，主要包括全面文献检索和主要文献筛选。

3.2.1 全面研究检索

首先采用全面的检索策略，找到与研究问题相关的文献，包括检索关键词构建和检索范围界定。为了获得更好的搜索结果，对关键词组进行细化和优化。Kitchenham 等人[3]建议将搜索问题分解为各个搜索单元，包括其同义词、缩略语和替代拼写等，然后通过布尔运算进行组合[3]。PIO（Population, Intervention, and Outcome）标准是定义这种搜索单元的合适方式[4]，Population 涉及到与技术和标准相关的术语，Intervention 解决的是软件方法论（或技术、工具、程序）中的一个具体问题。本文的检索关键词公式如式（1）所示：

检索关键词 = population *AND* intervention（1）

在本文的智能合约安全场景下，population 和 intervention 的定义如表 3 所示。

表 3 主要研究检索词

类型		检索词
Population	中文	形式化验证、模型检测、符号执行、抽象解释、模糊测试、抽象语法树、安全多方计算、可信执行环境、零知识证明{非交互零知识证明，简洁非交互零知识证明}
	英文	Formal Verification, Model Checking, Symbolic Execution, Fuzzing, Abstract Semantic Tree (AST), Zero-Knowledge Proof (ZKP), Non-Interactive Zero-Knowledge Proof (NIZK), Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge Proof (zk-SNARKs), Secure Multiple-party Computation (SMPC), Trusted Execution Environment (TEE)
Intervation	中文	智能合约安全，合约缺陷{合约漏洞}，代码安全，代码缺陷{代码漏洞}，合约隐私
	英文	Smart Contract Security{Safety}, Contract Defect{Vulnerability}, Code Security{safety}, Contract Privacy Code Defect{Vulnerability},

通过表 3 构建的检索关键词进行检索，本文选取了表 4 中 10 个电子数据库作为本文的检索范围。这些数据库涵盖了计算机科学与工程领域内最相关的期刊和会议（包括研讨会）。在研究选择时，通过人工过滤，排除不同数据库产生的重复结果。本文将开始年份设置为 2015 年、结束年份设置为 2020 年。此外，由于智能合约隐私保护方面的研究相对较少，因此文献搜索过程中可以关注部分具有权威性的技术文档。

表 4 选择的电子数据库

序号	电子数据库	网址
1	知网	https://www.cnki.net/
2	万方	http://www.wanfangdata.com.cn/
3	ACM Digital Library	http://portal.acm.org/
4	EI Compendex	http://www.engineeringvillage.org/
5	Google Scholar	http://scholar.google.com/
6	IEEE Xplore	http://ieeexplore.ieee.org/
7	DBLP Bibliography	http://dblp.uni-trier.de/
8	Springer Link	http://www.springerlink.com/
9	Science Direct	http://www.sciencedirect.com/
10	Web of Science	http://www.isiknowledge.com/

3.2.2 主要文献筛选

由于初步检索后得到的文献数量较多，需要对其进行筛选，从最初的检索结果中排除不相关的文献，选择最具代表性的文献作为主要文献。首先，由于不同电子数据库的特点，搜索结果中包含杂质，如与搜索关键词直接相关的会议名称，将其人工删除以保证结果的准确性；其次，检查文献的标题和摘要，排除不涉及智能合约安全和安全保障技术的研究；还有部分研究在技术上相似，因此选择最具代表性的作为主要文献；最后，本文共选择 45 篇文献作为研究的主要参考资料，其中包含 3 篇技术文档。

3.3 质量评估

针对 45 篇文献，论文的质量评估至关重要，本文给出如下 8 个论文质量评估标准：

QA1：是否有关于研究目的的明确说明？

QA2: 是否有关于研究背景的充分描述？

QA3: 是否有关于研究问题相关工作的回顾？

QA4: 是否有关于智能合约安全保障方法描述？

QA5：方法是否经过验证？

QA6：是否明确与研究目的和宗旨相关的结论？

QA7：是否明确的研究结果陈述？

QA8：是否建议进一步研究方向？

表 5 显示了对每篇主要文献应用质量评估标准的结果。其中，√表示“是”，×表示“否”。除了 Bartoletti 等[5]等 9 篇文献不包含未来工作展望（Q8），大部分研究满足本文设计的标准。由于缺失 Q8 不影响研究结果，本文不将其删除。

3.4 信息抽取

最后，本文设计一个信息抽取模版表来收集解决研究问题的信息，主要针对 45 篇主要文献进行信息抽取。表 6 显示了主要文献的每个信息抽取细节，该表格帮助本文从主要文献中提取相关信息，并了解这些研究如何解决本文设置的智能合约安全相关的研究问题。

表 5 主要文献质量评估

序号	主要文献	QA1	QA2	QA3	QA4	QA5	QA6	QA7	QA8
1	Bartoletti et al [5]	√	√	√	√	√	√	√	×
2	Wohrer et al [6]	√	√	√	√	√	√	√	√
3	Wohrer et al [7]	√	√	√	√	√	√	√	√
4	Destefanis et al [8]	√	√	√	×	√	√	√	√
5	Dingman et al [9]	√	√	√	√	√	√	√	√
6	Chen et al [10]	√	√	√	√	√	√	√	√
7	Chen et al [11]	√	√	√	√	√	√	√	√
8	Krupp et al [12]	√	√	√	√	√	√	√	√
9	Jiang et al [13]	√	√	√	√	√	√	√	√
10	Nikolic et al [14]	√	√	√	√	√	√	√	×
11	Kalra et al [15]	√	√	√	√	√	√	√	√
12	Tsankov et al [16]	√	√	√	√	√	√	√	√
13	Liu et al [17]	√	√	√	√	√	√	√	×
14	Amani et al [18]	√	√	√	√	√	√	√	√
15	Feist et al [19]	√	√	√	√	√	√	√	√
16	Wang et al [20]	√	√	√	√	√	√	√	√
17	Torres et al [21]	√	√	√	√	√	√	√	√
18	Nguyen et al [22]	√	√	√	√	√	√	√	×
19	Gao et al [23]	√	√	√	√	√	√	√	√
20	Gogineni et al [24]	√	√	√	√	√	√	√	√
21	Fu et al [25]	√	√	√	√	√	√	√	√
22	Kosba et al [26]	√	√	√	√	√	√	√	√
23	Zhang et al [27]	√	√	√	√	√	√	√	√
24	Sánchez et al [28]	√	√	√	√	√	√	√	√
25	Cheng et al [29]	√	√	√	√	√	√	√	√
26	Bünz et al [30]	√	√	√	√	√	√	√	√
27	Mavridou et al [31]	√	√	√	√	√	√	√	√
28	Liu et al [32]	√	√	√	√	√	√	√	√
29	Bhargavan et al [33]	√	√	√	√	√	√	√	√
30	Tikhomirov et al [34]	√	√	√	√	√	√	√	√
31	Zhu et al [35]	√	√	√	√	√	√	√	×
32	Zhao et al [36]	√	√	√	√	√	√	√	√
33	Cook et al [37]	√	√	√	√	√	√	√	√
34	Kang et al [38]	√	√	√	√	√	√	√	√
35	Li et al [39]	√	√	√	√	√	√	√	√
36	Steffen et al [40]	√	√	√	√	√	√	√	√
37	Baumann et al [41]	√	√	√	√	√	√	√	√
38	Li et al [42]	√	√	√	√	√	√	√	×
39	Russinovich et al [43]	√	√	√	√	√	√	√	√
40	Origo [44]	√	√	√	√	√	√	√	√
41	Kalodner et al [45]	√	√	√	√	√	√	√	×
42	Dolev et al [46]	√	√	√	√	√	√	√	×
43	Yan et al [47]	√	√	√	√	√	√	√	×
44	Rodler et al [48]	√	√	√	√	√	√	√	√
45	So et al [49]	√	√	√	√	√	√	√	√

表 6 信息抽取模版

序号	位置	描述	研究问题
		文章主要内容	
1	题目	主要文献的标题	
2	摘要	主要文献的摘要	G2
3	目标	主要文献的目标是什么？	G2
4	人员	谁与智能合约安全保障相关？	G1
5	目标	智能合约面临哪些安全问题及挑战？	G2
6	方法	通过什么方法保障合约的设计和实现？	G3
7	方法	通过什么方法实现智能合约的安全测试？	G3
8	方法	通过什么方法监测智能合约的部署运维安全？	G3
9	方法	通过什么方法保护智能合约代码隐私安全？	G3
10	方法	通过什么方法保护智能合约数据隐私安全？	G3
11	验证	通过什么方式验证智能合约安全保障方法？	G4
12	总结	研究总结	G2
		文章参考信息	
13	作者	主要文献的作者	
14	年份	主要文献的发表年份	SQ1
15	类型	主要文献的出版类型	SQ2
16	来源	发表主要文献的出版物名称	SQ2

4 调查结果

在本节中，本文介绍了 45 篇与本文研究主题相关的主要文献的评估结果。表 7 显示了所有主要文献的概述，其中包括主要文献的 id、作者、年份和基本描述。本文对 45 篇主要文献进行深度分析，提炼出智能合约安全理论研究框架如图 1 所示。

针对智能合约的合约安全和隐私安全，从问题及挑战、原因分析、保障方法和方法验证这 4 个角度进行总结，涵盖智能合约的设计、实现、测试、部署及运维多个阶段，涉及智能合约的代码隐私和数据隐私。

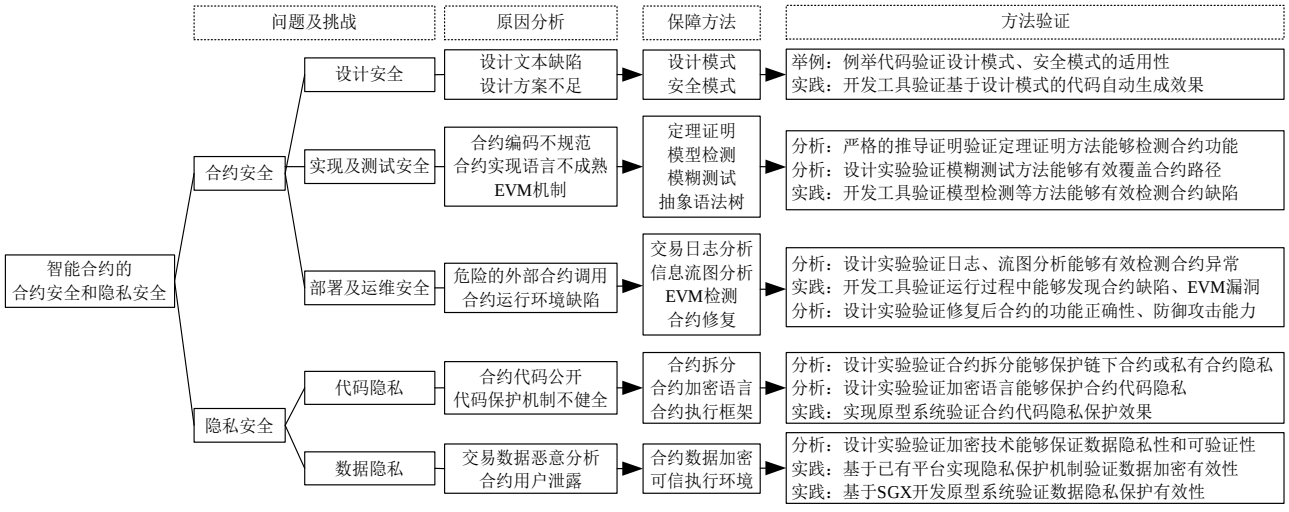

图 1 智能合约安全理论研究框架

表 7 主要文献汇总

序号	主要文献	描述
S1	Bartoletti et al [5]	以太坊智能合约的设计模式
S2	Wohrer et al [6]	安全模式描述了典型安全问题的解决方案，并可由 Solidity 实现人员应用以减轻典型的攻击
S3	Wohrer et al [7]	设计模式为实现人员提供设计指南
S4	Destefanis et al [8]	主张建立区块链软件工程学科
S5	Dingman et al [9]	第一个正式的针对 Solidity 智能合约的缺陷分类
S6	Chen et al [10]	第一个针对智能合约缺陷的实证研究
S7	Chen et al [11]	GASPER，通过分析智能合约的字节码来自动定位消耗 gas 量高的模式
S8	Krupp et al [12]	Teether，自动识别智能合约漏洞的工具
S9	Jiang et al [13]	ContractFuzzer，检测 Ethereum 智能合约安全漏洞的模糊器
S10	Nikolic et al [14]	MAIAN，动态分析工具，通过跟踪每个调用路径检测定义的 3 种类型的错误合约
S11	Kalra et al [15]	ZEUS，验证智能合约的正确性和公平性的框架
S12	Tsankov et al [16]	Securify，用于以太坊智能合约的安全分析器，证明合约行为对于给定资产是否安全
S13	Liu et al [17]	ReGuard，基于模糊测试的分析器，自动检测以太坊智能合约中的重入缺陷
S14	Amani et al [18]	基于扩展 Isabelle/HOL 中现有的 EVM 形式化，对 EVM 智能合约进行形式化验证
S15	Feist et al [19]	Slither，提供漏洞检测、优化检测、代码理解和代码审查的静态分析框架
S16	Wang et al [20]	Vultron，检测由于各种类型的对抗性攻击而导致的不规则交易的技术
S17	Torres et al [21]	ÆGIS，通过攻击模式进行攻击检测并还原攻击的系统
S18	Nguyen et al [22]	sFuzz，在以太坊上运行的智能合约的全自动测试引擎
S19	Gao et al [23]	SmartEmbed，在 Solidity 中学习智能合约特性，用于智能合约的克隆检测、错误检测和合约验证
S20	Gogineni et al [24]	进行多类分类，将智能合约分为自杀、挥霍、贪婪或正常类别
S21	Fu et al [25]	EVMFuzzer，第一个检测 EVM 实现漏洞的差分模糊测试工具
S22	Kosba et al [26]	Hawk，保护用户隐私的智能合约编程框架
S23	Zhang et al [27]	Town Crier，经过认证的数据馈送系统，为依托的智能合约提供源头认证的数据
S24	Sánchez et al [28]	Raziel 将安全多方计算和携带证明的代码结合起来，保证智能合约隐私性、正确性和可验证性
S25	Cheng et al [29]	Ekiden，实现高效的 TEE 支持的保密性智能合约和高扩展性的系统
S26	Bünz et al [30]	Zether，完全去中心化的保密支付机制，兼容 Ethereum 和智能合约平台
S27	Mavridou et al [31]	FSolidM，通过严谨的语义学设计合约框架，可以将一套设计模式作为插件
S28	Liu et al [32]	S-gram，新颖的语义感知安全审计技术
S29	Bhargavan et al [33]	分析验证以太坊合约运行时安全性和功能正确性的框架
S30	Tikhomirov et al [34]	SmartCheck，检测智能合约代码问题的可扩展静态分析工具，提供代码问题的全面分类
S31	Zhu et al [35]	基于安全多方计算的智能合约框架，保障智能合约执行中的输入隐私性和计算正确性
S32	Zhao et al [36]	ContractGuard，以太坊智能合约的入侵检测系统，检测智能合约的潜在攻击行为
S33	Cook et al [37]	针对以太坊智能合约的动态检测和防御工具
S34	Kang et al [38]	FabZK 扩展 Fabric，实现可审计的隐私保护交易
S35	Li et al [39]	链上/链下混合执行模型，增强智能合约的可扩展性和隐私性
S36	Steffen et al [40]	Zkay v1.0，使用隐私类型来指定私有值的所有者的类型化语言
S37	Baumann et al [41]	Zkay v2.0，改进 Zkay v1.0，引入更多保护智能合约隐私的语言特性
S38	Li et al [42]	Phantom，使用基于智能合约的 zk-SNARKs 的高效隐私协议
S39	Russinovich et al [43]	COCO，实现任意区块链系统的隐私保护框架
S40	Origo [44]	隐私保护应用平台 Origo，对交易进行保密同时可以验证合约交易的正确性
S41	Kalodner et al [45]	Arbitrum，设计虚拟机行为的离线验证激励机制，提高智能合约的可扩展性和保密性
S42	Dolev et al [46]	SodsMPC，量子安全的智能合约系统，通过多方计算保证合约执行的正确性，同时也保护数据隐私
S43	Yan et al [47]	CONFIDE，通过 TEE 支持链上机密性的系统设计方案（Antchain）
S44	Rodler et al [48]	EVMPatch，集成许多静态分析工具检测漏洞的框架，提供自动的方法来修补检测到的漏洞
S45	So et al [49]	VERISMART，确保算术操作安全的智能合约分析器

4.1 利益相关者

对于研究问题 GQ1，智能合约在它的设计、实现、测试、使用到销毁的过程中，合约的利益相关者都不同程度影响智能合约安全保障过程。但是，由于不同阶段涉及的利益相关者的任务不同、权限不同，其安全保护的侧重点也有所不同。明确不同利益相关者与智能合约安全的关系，对解决智能合约安全问题、明确智能合约安全挑战有指导意义。本文结合智能合约生命周期，提炼出以下 5 个与智能合约安全相关的利益相关者。

- 合约设计人员。规范的智能合约开发需要完善的合约设计文本，设计文本缺陷可能给合约实现人员带来误导。根据不同智能合约应用场景，可能存在复杂的、重复出现的问题，合约设计人员需要对此进行抽象并提出概念性的设计方案。但是，智能合约实现平台、开发语言和安全环境发展快速，随着新指令增加、安全风险发现，设计方案不足可能给合约维护升级带来更大的困难[6]。
- 合约实现人员。智能合约的实现人员直接影响智能合约的质量，合约实现人员需要根据合约设计方案编写合约，如果实现人员没有严格遵循智能合约设计方案，可能导致智能合约在使用过程中出现功能缺失、功能错误等重大安全问题。此外，每种编程语言都有编码规范，如果实现人员没有遵循官方编程风格指南和安全编程指南，代码可能不利于阅读、理解和维护，还可能导致合约缺陷从而带来安全风险。
- 合约测试人员。测试是智能合约正式发布之前至关重要的一环，对智能合约的质量和开发效

率均有影响。有效的智能合约测试能检验合约的实际结果是否符合设计预期，如果测试人员没有明确合约功能需求、合约设计方案，则合约测试人员无法有效编写测试用例、无法确认合约代码是否实现预期功能。其次，测试人员如果不熟悉合约编码规范，在使用不同的合约测试工具过程中，无法有效分析缺陷报告，导致智能合约缺陷的误报和漏报，从而影响合约整体开发效率、忽略合约中的潜在威胁。相反，高质量的合约测试可以帮助合约实现人员提升整体开发效率、保证合约质量。

- 合约维护人员。监测智能合约的实际运行状态并且对异常行为采取相应的应对策略，能够降低智能合约的安全风险，因此合约维护人员十分必要。但是，智能合约难维护、难升级，一旦部署到区块链上无法随意对其进行修改，所以智能合约安全对合约维护人员有更高的要求。对于智能合约的运行状态监测，合约维护人员需要建立完善的沟通机制、缺陷收集渠道和合约运行监测机制，否则无法及时发现使用中的合约存在的异常行为，可能导致重大的安全事件。对于智能合约修复和升级，合约维护人员需要熟悉合约缺陷并制定相应修改方案以及合约异常情况的应急方案，否则无法迅速、有效地进行智能合约维护，提高合约被攻击风险。
- 合约用户。合约用户需要根据特定的区块链系统设计原则，通过智能合约实现特定的功能，用户需要遵循基本的私有信息保护规则，否则可能导致私人信息泄漏。但是，合约用户更倾向于合约实现特定功能所花费的时间、资源、效率等，繁琐的使用操作可能增加用户违背使用规则、造成经济损失的风险。

4.2 智能合约安全问题和挑战

关于通识问题 GQ2，通过分析每一篇主要文献面临的安全问题，本文试图确定智能合约主要面临的安全问题和挑战。首先，本文详细介绍智能合约面临的安全问题，然后分析安全问题带来的挑战，明确安全保障目标。如图 1 所示，本文认为智能合约安全问题和挑战主要包含合约安全和隐私安全两个方面。

4.2.1 合约安全问题和挑战

本文从智能合约设计、实现、测试、部署及运维几个阶段，讨论智能合约安全面临的问题和挑战。

4.2.1.1 智能合约设计安全问题和挑战

- 安全问题：智能合约设计安全主要体现在设计文本安全和设计模式安全两个方面。

关于聚焦问题 FQ1，在智能合约设计阶段，智能合约设计缺陷产生的原因主要包括设计文本缺陷、设计方案不足。

智能合约设计文本主要针对合约参与方进行编写，结合智能合约的实际功能在合约文本中明确合约中涉及的各方权利和义务、合约触发的条件、合约执行的情况、合约终止结果等。

智能合约的设计安全是智能合约稳定运行的第一步，智能合约实现人员在编写智能合约之前，需要根据实际功能需求对智能合约进行设计，避免由设计文本错误导致智能合约功能异常。但是，由于智能合约设计人员在设计文本编写过程中，对智能合约功能需求理解不全面，可能导致智能合约设计文本出现安全缺陷。例如，具体的智能合约设计文本安全存在以下几种表现形式：①设计文本遗漏：设计文本遗漏部分智能合约实际需求；②设计文本多余：合约文本出现部分与智能合约实际需求无关的方案；③设计文本不一致：设计文本中方案与实际需求不一致；④设计文本模糊：设计文本中的方案超过一种，存在方案二义性；⑤设计文本错误：设计文本中的方案与实际需求产生冲突。

针对常见的、复杂的智能合约功能，可以通过定义设计模式实现智能合约代码的自动生成，降低在编写复杂功能过程中引入错误的风险。但是，由于智能合约发展时间较短，设计模式的概念不成熟，已有的设计模式针对特定的应用场景不一定具有较好的适用性，可能导致自动生成的智能合约功能缺失或者与目标功能不一致。

- 安全挑战：如何保障智能合约设计安全？（Ch1）

为了确保智能合约符合用户需求、合约实现过程中不会因为合约设计出现错误，需要保障智能合约设计安全。如何避免在智能合约设计过程中合约文本出现上述安全问题，保障智能合约设计过程中不存在缺陷，需要研究如何定制完善的智能合约设计文本编写规范，并通过自动化的方法对合约设计文本进行分析。

如何保障设计模式适用场景的丰富性、设计模式使用的灵活性、应用设计模式自动生成代码的正确性，需要研究如何构建更全面的智能合约设计模式库以及设计模式使用机制，总结并抽象出不同场景适用的智能合约设计模型、安全可靠的智能合约模板，实现更安全的智能合约设计。

4.2.1.2 智能合约实现及测试安全和挑战

- 安全问题：智能合约实现及测试安全主要体现在智能合约代码存在缺陷。

结合智能合约实现、测试原理，关于聚焦问题FQ1，在智能合约实现、测试阶段，智能合约代码缺陷产生的原因主要包括合约编码不规范、合约实现语言不成熟、EVM 机制。

智能合约实现主要是指在完成智能合约设计的基础上，根据合约设计文本进行智能合约实现，合约处理逻辑的正确性、完备性是智能合约的基本要求。但是，智能合约的实现人员能力、编码水平良莠不齐，实现过程中可能导致智能合约存在代码缺陷，实现人员也可能出于利益驱动，在智能合约中留下后门；此外，智能合约的实现语言、编译器、执行机制等都可能与智能合约安全缺陷相关。

智能合约实现完成后、正式部署上线前，采用人工或自动化的方式对智能合约代码进行缺陷检测，合约测试人员对智能合约进行全有效、全面的测试，可以防止合约受到恶意攻击。

近年来，针对智能合约代码缺陷展开的攻击数见不鲜。区块链开源特性使得智能合约代码可以被便捷获取，攻击者可以通过分析智能合约的代码缺陷寻找攻击突破口。由此可见，智能合约代码缺陷是智能合约实现、测试过程中的主要安全问题。

a) 智能合约编码相关的缺陷

软件实现人员编写智能合约过程中存在的问题可能导致智能合约缺陷，例如合约编码缺少提示、程序逻辑限制和身份验证或授权不足等原因。

首先，编码缺少提示可能导致合约缺陷。程序中缺少输入的合规性检查，输入参数可能不合理或者无效，影响程序的正确执行；缺少输出结果的检查提示，如果输出结果异常且不提示，导致不能立即发现并及时处理异常，例如缺乏执行结果提示（Missing Reminding Execution Results，D1）和未检查返回值（Unchecked Return Value，D2）缺陷[50]。缺乏执行结果提示指通过合约的应用程序二进制接口（Application Binary Interface，ABI）调用智能合约，但是 ABI 不会提示函数调用是否成功，可能增加不必要的错误和 gas 浪费。未检查返回值缺陷有两种变体，称为无 gas 发送（Gasless Send）和未检查 send（Unchecked Send）[15, 51]。Solidity 语言提供的部分函数出现计算错误时产生的结果仅返回结果布尔值 false，不会抛出异常并且会继续执行后续的代码，此时如果程序未立即终止将会导致意外交易、造成潜在威胁。

其次，合约编码过程中程序逻辑控制可能导致合约缺陷。理论上合约的程序逻辑设计是正确的，能够实现智能合约的需求功能，但是攻击者可能故意利用程序逻辑，通过特定的输入实现恶意的程序执行，导致智能合约执行出现问题。例如重入（Reentrancy，D3）、余额恒等操控（Manipulated Balance Equality，D4）和循环调用导致的 DoS（DoS with Nest Call，D5）缺陷等。

攻击者利用代码重入缺陷，让程序反复执行攻击者的恶意代码，从中获利[52]。表 1 的攻击事件 TheDAO 中，攻击者利用了重入缺陷，从智能合约中非法转出大量资金。图 2 显示了含有重入漏洞的 2 个智能合约 Reetrancy1 和 Reetrancy2，2 个合约代码不同，其重入的本质都是在通过无 gas 限制的转账操作之后（Reetrancy1-Line10，Reentrancy2-Line11），才进行相应的余额变更操作（Reetrancy1-Line11，Reentrancy2-Line14），攻击者利用在转账操作和账户操作间隙进行重入，在一次账户操作之前进行多次转账操作，造成合约实际转账金额远远大于实际账户余额。

```
1  contract Reentrancy1 {
2      mapping (address => uint) public credit;
3
4      function donate(address to) payable public{
5          credit[to] += msg.value;
6      }
7
8      function withdraw(uint amount) public{
9        if (credit[msg.sender]>= amount) {
10         require(msg.sender.call.value(amount)());
11         credit[msg.sender]-=amount;
12       }
13     }
14 }
```

```
1  contract Reentrancy2 {
2      mapping (address => uint) userBalance;
3
4      function addToBalance() payable public{
5          userBalance[msg.sender] += msg.value;
6      }
7
8      function withdrawBalance() public{
9          // send userBalance[msg.sender] ethers to msg.sender
10         // if mgs.sender is a contract, it will call its fallback function
11         if( ! (msg.sender.call.value(userBalance[msg.sender])() ) ){
12             revert();
13         }
14         userBalance[msg.sender] = 0;
15     }
```

图 2 含有重入缺陷的智能合约

余额恒等操控也被称为“强行发送 Ether 给合约”。当合约的控制流决策依赖于余额属性的特定值时就会出现此缺陷。攻击者可以利用自毁 API（Self-destructure 函数）强行发送 Ehter 给任何合约，利用此漏洞操控程序的执行逻辑①。循环调用导致的 DoS 缺陷指发动拒绝服务的攻击者会通过控制

① Comprehensive list of known attack vectors and common antipatterns, https://github.com/sigp/solidity-security-blog.

程序执行逻辑、持续消耗调用合约需要的 gas，从而让某些代码执行失败。

为了保证智能合约的安全执行，智能合约在设计编写过程中，应该严格设定调用权限，判定调用者的身份，如果合约中缺乏调用者的身份验证或授权，无法检查调用者的身份或特权，可导致受保护的数据项或功能被恶意调用执行。例如，Tx.Origin 身份验证（Authentication through Tx.Origin，D6）、越权访问（Exceed Authority Access，D7）、无保护自杀（Unprotected Suicide，D8）、Ether 泄漏到任意地址（Leaking Ether to Arbitrary Address，D9）和签名验证不足（Insufficient Signature Verification，D10）。Tx.Origin 身份验证缺陷产生的主要原因是智能合约经常需要检查访问者的权限，如果使用 tx.origin 来判断访问者权限可以帮助攻击者轻松绕过权限判断。权限控制问题主要是越权访问，在智能合约中权限控制问题主要由于函数可见性不明确或者缺乏充分的权限检查，导致攻击者能够访问或修改不该访问的函数或变量[53]。无保护自杀①是智能合约为了应对出现故障等一些紧急情况，在合约创建时规定具有特权的合约的“所有者”可以使用自毁方法杀死合约，其关联合约将永久删除，但是由于合身份验证不足可能导致合约被合约“所有者”以外的账户杀死。Ether 泄漏到任意地址是指合约可以给任意地址发送 Ether，但是该地址可能不具备收取 Ether 的资格，该类合约也称为浪子合约，此缺陷产生的直接原因是将 Ether 发送到任意地址时未能检查调用方的身份[14]。签名验证不足由于实现人员没有全面考虑签名验证信息产生，可能导致同一个数字签名对多个交易有效，针对区块链上的交易信息进行重放，造成财产损失。

Solidity 是一种年轻且不断发展的编程语言，由于编译器发展不成熟，编程过程中可能会导致 API 使用不当（Improper Using of API，D11）、编译器版本过时（Outdated Compiler Version，D12）和编译器版本不确定（Unspecified Compiler Version，D13）等缺陷。API 使用不当主要是因为编译器可能仍然支持部分不推荐使用的 API。当合约使用过时的编译器时会导致编译器版本过时缺陷，因为该编译器包含错误使已编译的合约易受攻击。编译器版本不确定是指如果合约没有指定编译器版本，实现人员在以后的代码重用中可能会因为版本差异而遇到编译错误。

编码过程中还可能由于智能合约实现人员不够细心等原因导致构造函数名称错误（Erroneous Constructor Name，D14），导致预期的构造函数成为可由任何人调用的公共函数②；或者实现人员直接把地址写入到智能合约中可能会出现地址不符合规定的情况，即地址硬编码（Hard Coded Address，D15）缺陷，由于智能合约一经部署就无法修改的特性，导致出现非法地址。

最后，区块链设计和应用原理形成某些特定的规则，例如编码过程中使用时间戳等信息作为参数。合约执行结果随着时间戳的不同而改变，恶意攻击者可以提前尝试不同的时间戳，控制智能合约的运行结果，从而产生时间戳依赖（Timestamp Dependency，D16）[50]；使用时间戳等区块信息生成的随机数不是真的随机数，攻击者可以根据随机数生成算法预测随机数达到攻击目的（Generating Bad Randomness，D17）[54]；此外，灵活的区块创建规则意味着矿工创建区块没有任何限制，可以创建有利于自己的区块。矿工根据执行该交易所获的 gas 大小选择交易进行打包，因此产生交易顺序依赖（Transaction Order Dependency，D18 ），攻击者通过支付较高 gas 导致某些交易无法正常执行。

b) 实现语言和 EVM 机制相关的缺陷

智能合约的实现语言及其编译器发展时间较短，并且合约代码执行机制仍然不够完善，存在较多的不足，可能导致出现很多实际与预期不一致的情况，主要原因包括实现语言、EVM 机制等原因。由于特殊的 EVM 机制导致的缺陷有缺少返回值（Missing Return Statement，D19）、整数溢出（Integer Overflow and Underflow，D20）、调用栈深度限制（Call-Stack Depth Limitation，D21）、短地址补全（Making up Short Address，D22）缺陷。

缺少返回值是指某些智能合约定义了返回值，却在函数结尾没有返回。对于这种函数，EVM 虚拟机会自动添加一个默认的返回值，这可能会造成外部程序逻辑判断错误。整数溢出是由于计算机中整数具有上限和下限，当一个整数超过其上限或者小于其下限，该整数就会发生溢出/下溢，可以使用通用基础合约 SafeMath.sol 防止该问题③。EVM 的调用堆栈硬限制为 1024，即调用栈深度限制，当恶

① Smart contract weakness classification and test cases, https://smartcontractsecurity.github.io/SWC-registry

②Constructor modifier issue #3196 Ethereum/solidity github,https://github.com/Ethereum/solidity/issues/3196

③Safemath,https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/utils/math

意攻击者对堆栈调用进行深度攻击时，攻击者递归调用合约 1023 次，导致被攻击合约达到堆栈深度限制，无法对该合约进行任何外部调用。短地址补全 EVM 本身接受不正确填充参数的副作用，攻击者利用这一点可以通过使用专门制作的地址来进行攻击。

- 安全挑战：如何在实现及测试过程中保障智能合约代码不存在缺陷？（Ch2）

首先，严谨的技术规范是保证技术健康有序应用的重要前提，实施规范的实现流程，使用规范的实现和编译工具，能够降低编码过程中引入安全风险的几率；优化实现、编译及部署运行等问题，实现更安全、高效的智能合约，降低计算消耗，节省合约资源开销。

其次，有效的智能合约测试是发现合约代码缺陷的主要方式。与传统软件相似，智能合约的测试包括功能测试、非功能测试、安全性测试等。功能测试包括单元测试、集成测试、用户验收测试等，非功能测试主要包括性能测试、容量测试、可用性测试等。此外，由于智能合约涉及大量数字资产，因此智能合约上线前需要进行严格的安全性测试。本文将智能合约测试面临的主要挑战概括为以下两点：

①如何实现基于合约语法和语义分析的测试

智能合约缺陷产生的主要原因是智能合约实现人员对智能合约系统的底层执行语义和实际语义之间的假设存在差异[50]。因此，如何将智能合约的语法和语义形式化是智能合约测试面临的重要挑战，在对智能合约语法和语义进行抽象表示的基础上，验证智能合约目标功能、检测智能合约缺陷。

②如何探索更多有效的合约执行状态进行测试

探索合约执行状态的测试旨在探索更多、更具有代表性的智能合约状态空间，结合输入数据触发可执行交易，根据合约执行返回的测试结果分析是否存在功能问题或安全问题。例如模糊测试方法可以构造无效输入数据并监视目标软件在运行过程中的异常结果来发现软件故障[55]。该合约检测思路主要面临的挑战是如何探索更多有效的合约执行状态，通过覆盖最具有代表性的执行路径来检测智能合约是否功能正确或存在缺陷。

4.2.1.3 智能合约部署及运维安全和挑战

- 安全问题：智能合约部署及运维安全主要体现在合约运行监测和合约异常修复及防御。

结合智能合约实现、测试原理，关于聚焦问题FQ1，在智能合约部署和运维阶段，安全威胁产生的原因主要包括外部合约调用和运行环境。

对于外部合约调用，由于智能合约编程具有面向对象的思想，本质上一个智能合约就是一个类。在与区块链交互的功能操作非常复杂的情况下，通常需要进行合约之间的调用。外部依赖性意味着当前合约的执行依赖于外部合约的行为，合约的正确执行不仅仅依赖于当前合约，还与其他合约紧密关联，调用智能合约时调用逻辑也是决定安全调用的关键因素之一。通过外部调用可能会导致的缺陷主要包括 DelegateCall 调用（DelegateCall，D23）、冻结的 Ether（Frozen Ether，D24）和调用类型不匹配（Type Cast，D25），具体说明如下。

为促进代码重用，Solidity 中提供 DelegateCall 函数，用于实现合约之间的相互调用以及交互，DelegateCall 调用缺陷带来的威胁是恶意的被调用方合约可以直接修改（或操纵）调用方合约的状态变量[12]。冻结的 Ether 是由于合约不提供任何支出功能而是依靠其他合约实现支出，当被调用的指出合约被意外或故意终止时，用户无法从这些账户中取出资金，从而冻结资金[13]。智能合约使用的 Solidity 语言是强类型语言，函数调用过程不会进行类型检测[50]，面对一些特殊情况，代码在执行过程中有其特有的处理机制。例如，由于 Solidity 语言不会进行类型检测，导致出现调用类型不匹配的情况，即 Solidity 语言编写的合约中可能存在两个不同合约中包含相同函数名的情况，通过被调用方合约的实例可以调用另一个合约的同名函数，并且调用类型不匹配时不会抛出异常，该缺陷的合约模板如图 3 所示。由于调用 add 函数（Line9）时不会检查其合约类型是否为 CounterLibrary，攻击者可以利用这个特点创建攻击合约 CounterLib，诱导 EVM 调用攻击者合约。

```
1  contract CounterLibrary {
2     function add(uint) public returns (uint)
3   }
4  contract CounterLib{
5     function add(uint) public returns (uint)
6   }
7  contract Game{
8     function play(CounterLibrary c) public {
9         c.add(1);
10     }
11  }
```

图 3 含有调用类型不匹配缺陷的智能合约模板

如上述介绍的 3 种外部合约调用导致的智能合约缺陷，如果当前合约的执行依赖于外部合约的行为，那外部合约调用的未知性给当前合约执行带来

巨大风险，合约运行监测是重要的合约安全问题。

对于运行环境，智能合约一般运行在隔离的沙箱执行环境中，一旦在运行环境中存在虚拟机自身安全缺陷或验证、控制等机制不完善等问题，攻击者可通过部署恶意智能合约代码，扰乱正常业务秩序，消耗整个系统中的网络、存储和计算资源，进而引发各类安全威胁[25]。如果隔离的沙箱执行环境的实现存在缺陷，必导致严重的后果，例如不同的EVM 执行存在不一致的情况，通过利用 gas 消耗的度量差异等因素来控制合约的生成，导致合约执行输出不一致的情况。因此，应用成熟的测试工具、有效的方法来保证 EVM 安全非常紧迫。

智能合约异常修复及防御主要是指对于部署后的合约在运行过程中发现异常行为以及合约漏洞，需要将已有的智能合约进行升级，否则存在合约漏洞被利用发生恶意攻击的可能。但是，由于以太坊智能合约部署后不可修改，所以原始的合约升级策略需要在新地址部署修复后的合约，并将原合约的数据状态迁移到新合约，该过程需要开发者手动实现。

- 安全挑战：如何保障在智能合约运行过程中能够监测合约中的异常行为并有效修复问题、防御攻击？（Ch3）

为了保障智能合约的运行安全，需要监测智能合约在运行过程中可能出现的异常行为，例如异常的外部合约调用和运行环境导致的合约执行异常。

检测异常的外部合约调用，不仅需要分析合约执行逻辑是否存在缺陷，还需要结合合约的运行状态实现入侵检测，实时监控智能合约的运行状态。首先，智能合约代码调用关系复杂，如何充分结合合约的语法语义信息进行合约执行逻辑分析具有较大难度；其次，智能合约执行日志信息数据量大、内容丰富，如何有效对其进行分析识别异常行为是实现智能合约运行安全保护的一大挑战[37]。

针对合约运行环境安全即虚拟机安全，如何从合约运行的可确定性、资源控制等维度分析和保证智能合约的运行环境安全也至关重要。

此外，传统打补丁或者升级方法不适用于智能合约，因此智能合约在运行时无法及时获得外在的安全监管和防护，这给智能合约运行实时监控和保护研究带来的挑战，如何能够根据运行的异常状态制定相应的防御及应对策略是保障合约运行安全的重要目标。

总的来说，智能合约的安全体现在设计、实现、测试、部署、运维各个阶段。目前，智能合约的 25 种缺陷主要出现在智能合约实现、测试和部署、运维过程。

4.2.2 隐私安全问题和挑战

在区块链系统中，如果使用智能合约实现复杂的交易，要求公开合约数据和源码，并且所有的节点都可以执行该合约。典型公有链不限制节点进出，并向新节点提供无差别服务，在网络通信过程中可能存在恶意攻击者窃取机密信息。公有链上的智能合约对所有节点公开，一般没有传统的加密、审计和访问权限，所有交易数据对所有接入节点公开透明的。新节点可通过有效方式获取完整的交易数据，节点通过独立执行智能合约方式验证交易的有效性，随着各类反匿名技术的发展，例如流量分析和聚类分析等，攻击者利用可获得的数据信息追踪历史交易路径，提取大量用户身份特征相关的信息，导致用户无法实现绝对匿名[56]，这将显著增加隐私泄露的风险。

区块链系统隐私保护的关键在于保证不影响在去中心化区块链系统中进行有秩序交易的同时，确保用户的隐私不会因交易公开而暴露。在许可链（联盟链或私有链）中可以设置访问控制策略，以满足区块链的隐私要求，区块链数据的完全透明仅针对授权节点。但是，在非许可链（公有链）中，所有节点均可自由进出访问区块链数据，区块链隐私问题尤为突出[57]。因此，隐私保护在区块链系统中的重要程度远高于传统的中心化系统，因为区块链系统的不可篡改、去中心化的特性，用户的交易信息一旦被泄露是永久行为，损失很难挽回。

围绕 GQ2，除了在第 4.2.1 小节介绍的合约安全问题和挑战，还有合约隐私安全问题及挑战。本小节将围绕合约隐私安全，结合区块链系统基础架构，分析存在的隐私威胁，主要包括合约代码隐私和合约数据隐私。

4.2.2.1 合约代码隐私

安全问题：合约代码隐私安全主要是指智能合约代码和状态变量等信息泄露问题。

关于聚焦问题 FQ1，在智能合约使用过程中，智能合约代码隐私泄漏产生的原因主要是公有链公开透明的性质，并且区块链系统缺乏合约代码保护机制。

在支持智能合约的主流平台上例如 Ethereum、Hyperledger Fabric 等，智能合约普遍公开透明。智能合约往往需要多个用户共同参与，涉及用户账

户、交易信息、智能合约的状态变量等多种信息。智能合约的代码和状态变量都公开于整个网络中，攻击者可以分析合约代码的具体内容、执行逻辑，若发现该合约的代码缺陷，并以此作为攻击点，则可能造成巨大损失。Kosba 等人[26]指出尽管在设计隐私保护的加密货币（如 Zerocash[58]和其他一些加密货币）方面已经取得了一些进展，但这些系统都放弃了可编程性，而如何在不将交易和数据以明文形式暴露给矿工的前提下实现可编程性，该问题仍未解决。

- 安全挑战：如何在不影响交易功能实现的情况下保障合约代码及状态变量等信息不被泄露？（Ch4）

智能合约代码涉及交易规则、具体交易数据，为了保障参与交易用户、公司的信息隐私安全，保障交易规则不被攻击者获取进行恶意分析，需要保证合约代码不被非授权节点获取。

但是，针对公有链平台的运行机制，智能合约的代码和状态变量都公开于整个网络中，如何在合约代码执行过程中，将函数的调用、参数的传递及其状态变量的变化等信息不以明文的形式公开是保护智能合约代码隐私的一大挑战。如何在不影响交易功能正常实现的情况下，对智能合约代码及相关参数进行加密，从而防止恶意节点获取合约代码并对其进行分析，降低发生恶意攻击的风险，是智能合约代码隐私保护的重要目标。

4.2.2.2 合约数据隐私

安全问题：合约数据隐私安全主要是指智能合约交易数据信息公开不保密。

关于聚焦问题 FQ1，在智能合约执行交易过程中，智能合约数据隐私泄漏产生的原因可能是攻击者恶意分析公开的交易数据或者合约用户泄漏。

区块链中的交易存储在公开的全局账本中，智能合约的整个生命周期均全网广播并被记录，智能合约中的整个行为序列都会在整个网络中传播并记录在区块链上，因此它们是公开可见的。在以太坊中，交易是触发状态变动的唯一途径。交易数据中包含交易发送者所发出的交易数量、执行交易所需要消耗的 gas 价格和 gas 限制、调用者的接收地址、交易金额、账户初始化数据、合约函数调用数据等大量交易信息。

因此，即使合约方可通过创建新的公钥进行交易来提高匿名性，但交易内容和地址等信息仍然是公开的，即每个公钥的所有交易和余额的值都是公开的，这种情况下攻击者可以非常容易得到全网的交易信息。此外，有研究通过分析加密货币的交易图结构，证明了去匿名化攻击的可行性[59,60,61]。如果所有与用户相关的交易数据都可以连接起来，可能通过网络技术、大数据技术等推断交易之间的先后连续性、交易是否属于同一个用户，甚至可以获取包括用户的真实身份以及其他交易类型和频率等真实信息。所以，合约数据隐私与用户身份隐私和用户行为隐私关联紧密，合约数据的泄露可能会提高匿名链接的风险。

- 安全挑战：如何保障智能合约交易数据的安全输入和输出以及合约内交易数据不被无关节点获取？（Ch5）

首先，在智能合约执行过程中，保护智能合约数据隐私的前提是解决具有隐私保护的智能合约数据输入问题，即如何在没有可信赖的服务运营商的情况下，向智能合约提供经过身份验证的数据，保证智能合约的安全执行[27]。

其次，如何解决具有隐私保护的智能合约数据输出问题，即如何有选择性的支持外界数据请求，保证针对合法的用户输出特定数据，而其他无权限用户无法查看请求内容，从而保证智能合约的交易用户隐私。

最后，如何保障合约内交易数据隐私安全，即如何实现基于需求的共享而不是整个区块链的公开。合约内交易数据隐私面临的挑战可以划分为两种不同的场景：对交易无关节点保证隐私和交易内部节点之间保证隐私。如何实现对交易无关节点保证隐私是指智能合约执行交易过程中涉及的用户信息、交易信息、合约的状态变量等不可被与该交易无关的节点获得，保证参与合约交易用户的信息隐私性；同时，如何实现交易内部节点之间保证隐私是指保护同一合约交易执行过程中的多个参与用户之间的隐私，即保证同一交易的不同用户之间的数据隐私性。在合约执行期间，任何参与方不能获得他人的敏感信息、并且每个参与方获得的输出只包含计算后应该获得的信息，防止不公平的经济行为发生[62]。

大部分金融合约交易（例如保险合约或股票交易）的真实数据涉及到机密，因此对外部节点不便公开，而游戏竞猜类合约对参与的所有节点也应当保证数据的独立隐私性。经典例子就是应用智能合约在区块链上进行匿名竞拍，合约在执行过程中需要保证参与竞拍方匿名以及各方出价是私密的。

目前，传统的信息保护措施是通过对信息加密，防止攻击者获得有用的信息。但是在区块链系统进行交易信息加密的过程中，一方面要保证不能让非交易者看到交易信息，另一方面需要验证交易的正确性，对交易内容不能完全加密，这两者本身存在矛盾，这为隐私保护技术带来巨大挑战。

综上所述，智能合约安全问题和挑战主要包括合约安全和隐私安全。合约安全主要关注智能合约生命周期不同阶段存在的安全问题，例如合约设计阶段存在的合约文本和实际功能需求不一致的问题、合约实现及测试阶段存在的多种合约缺陷问题、部署及运维阶段存在的外部合约调用和运行环境缺陷等问题。隐私安全则主要关注合约代码隐私和合约数据隐私，例如合约代码公开可见暴露交易规则、合约数据公开可见泄露交易用户信息。结合不同的合约安全和隐私安全问题，智能合约面临着不同安全挑战。

4.3 智能合约安全保障方法

4.2 节详细介绍了智能合约面临的安全问题和挑战，本文将其划分为合约安全和隐私安全 2 方面。详细了解智能合约面临的安全问题和挑战可以帮助明确安全保障目标并选择合适的安全保障技术。

为了回答通识问题 GQ3，下面分别介绍合约安全保障方法和隐私安全保障方法。

4.3.1 合约安全保障方法

4.3.1.1 合约设计安全保障方法

针对合约安全挑战 Ch1，只有相对较少的研究关注安全的智能合约设计。目前，合约设计安全保障方法主要有智能合约设计模式，设计人员收集和总结 Solidity 安全编程规则，为 Solidity 创建一个结构化的、信息量大的设计模式并动态更新维护，以此作为合约实现人员的指导。Bartoletti 等人[5]对 Solidity 智能合约进行实证分析，总结出智能合约常见的 5 种应用场景，并描述出 9 种常见的设计模式，通过对设计模式和应用领域的相关性分析来推动更安全的智能合约编程实现。但是 Bartoletti 等人[5]的研究不包含安全设计模式，因此 Wohrer 等人[6]阐述了 6 种常见的智能合约安全模式，描述了典型安全问题的解决方案，Solidity 开发者可以应用这些模式来编写更安全的智能合约。进一步的，Wohrer 等人[7]应用多维文献回顾和从基础理论中借鉴的定性研究方法，描述了 18 种合约设计模式，这些模式都基于已有的智能合约安全问题给出相应的解决方案，在未来可以被开发人员用来解决与智能合约编码相关的常见问题。为了帮助开发者创建更安全的合约，Mavridou 等人[31]引入基于有限状态机的智能合约模型，将合约设计成有限状态机的框架，利用设计模式来自动生成代码，方便开发具有复杂功能的正确合约。

4.3.1.2 合约实现及测试安全保障方法

针对合约安全挑战 Ch2，遵循一些概念性、经验性的“最佳实践”有助于提高编码质量，降低引入缺陷风险。除了进行更规范的合约实现，目前大量智能合约安全的研究都集中在智能合约测试，为了便于说明，本小节结合第 4.2.1 节的智能合约测试面临的主要挑战介绍智能合约缺陷检测方法。

（1）基于合约语法和语义的测试

由于智能合约开发者可能对智能合约系统的底层执行语义和实际语义之间的假设存在差异，导致出现大量缺陷[50]，因此大量研究对智能合约的语法和语义进行抽象，在此基础上检测智能合约，具有代表性的技术有定理证明、抽象语法树分析等。

定理证明通过逻辑命题对形式化规范进行表达和描述，通过数学推理对命题进行数学证明，以判断程序实现是否满足形式化规范，即实际程序和形式化规范之间的一致性判定[63]。基于定理证明的缺陷检测研究合约代码[33,64]或者 EVM 语义[65,66,67]对智能合约进行形式化验证。Amani 等人[18]通过字节码级别的程序逻辑扩展 Isabelle/HOL 中现有的 EVM 形式化，将字节码序列构造为代码块，并创建程序逻辑来对此进行推理判断，从而判断智能合约是否具有健全的程序逻辑。Bhargavan 等人[33]提出智能合约分析和验证框架，该框架通过将智能合约源码和字节码转化成函数编程语言 F*，验证智能合约的安全性和功能正确性；Hirai[65]使用 Lem 将 EVM 语义形式化，从中提取交互式定理证明器 Isabelle/HOL 来证明智能合约的安全属性；Hildenbrandt 等人[67]利用 K 框架[68]提出完全可执行的严格形式语义 KEVM，通过用 KEVM 可以验证智能合约的算数操作或转账操作的正确性。

抽象语法树（AST）[69]在计算机科学中的定义是源程序语法结构的一种特殊表现形式，以树状形式展示源程序的语法结构。抽象语法树作为一种重要的程序中间表现形式，能清晰表示软件源代码的语法信息。因此，大量研究抽取智能合约的抽象语法树作为中间表示形式，进一步分析智能合约的数据流、控制流等信息，基于智能合约语法和语义信息检测合约缺陷。

Tikhomirov 等人[34]应用 java 实现一种以太坊智能合约的静态分析工具 SmartCheck，使用 ANTLR① 在 Solidity 源代码上进行词法和语法分析，根据特定模式来检测 Solidity 合约的代码缺陷；Slither[19]是一个静态分析框架，利用 Solidity 抽象语法树生成智能合约的继承图、控制流图和表达式列表，并将合约的整个代码转换为 Slither 的内部表示语言 SlithIR，使用静态单一分析实现各种代码的分析计算；SmartEmbed[23]应用抽象语法树将智能合约代码解析为包含代码结构信息的字节流，基于向量空间使用相似性检查方法，实现克隆合约检测、克隆错误检测、合约漏洞检测。

与基于抽象语法树的检测方法不同，Tsankov 等人[16]实现智能合约安全分析工具 Securify，通过分析合约的依存关系图从代码中推导精确的语义信息，结合语义事实构建的安全模式和违规模式，进行合约缺陷检测和定位；VERISMART[49]旨在通过执行自动、可扩展、精确度高的特定领域算法，分析智能合约算术安全性，即算数操作过程中出现的整数溢出和被零除问题。

为了适用更多缺陷类型，能够检测更全面的已知合约缺陷，并能够预测未知合约缺陷，有研究借鉴当前主流的机器学习等技术保障合约安全。Gogineni 等人[24]提出一种基于 AWD-LSTM 序列学习智能合约多类分类技术，通过使用两个神经网络，学习大量输入数据的语义信息将智能合约分为自杀、挥霍、贪婪或正常这 4 个类别，该方法具有更好的性能。Tann 等人[70]提出一种基于 LSTM 序列学习的智能合约漏洞预测技术，该方法通过将智能合约的字节码指令作为序列学习器的输入，返回合约存在漏洞的概率。Liu 等人[32]提出一种基于语言模型的智能合约漏洞预测技术 S-gram，分析智能合约源码产生智能合约 Token 序列，采用统计语言模型展开分析，预测智能合约的潜在漏洞。

（2）探索合约执行状态的测试

探索合约执行状态的测试通过分析智能合约的执行路径，根据状态变量的变化分析智能合约的执行状态，尽可能覆盖更多程序路径，从而验证智能合约是否正确。

模型检测是对有穷状态系统的一种形式化验证方法，与基于定理证明的形式化方法不同，该方法的主要基本思想是状态搜索，搜索依赖于为合约构建有穷状态的模型[71]。为了适应不同的使用场景，在模型检测的发展过程中衍生出多种逻辑和结构，在智能合约形式化验证过程中大量研究采用抽象解释和符号执行算法。

符号执行的核心思想是使用符号值代替具体执行程序，用符号值代替程序分析过程中任意不确定的变量，解析程序可执行路径上的指令，根据其语义更新程序执行状态[72]。符号执行工具 Oyente[50]是第一个用于分析和检测以太坊智能合约安全问题的工具，该工具从字节码构建控制流图，检查合约是否存在任何易受攻击的模式；同样地，Krupp 等人[12]提出 Teether 工具，该工具通过 EVM 字节码利用后向切片重构控制流图，利用符号执行将关键路径转换为一组约束用于查找合约缺陷。为了检测 gas 消耗不合理的智能合约，Chen 等人[11]实现了一个基于符号执行的静态分析工具 GASPER，该工具通过分析智能合约的字节码自动定位消耗 gas 量高的模式；Mythril②在符号执行的基础上，结合污点分析和控制流检查来检测合约各种安全漏洞；智能合约安全属性分析框架 ZEUS[15]结合抽象解释和符号执行对合约进行建模，减少状态空间探索，快速验证合约安全性。

模糊测试是一种自动测试技术，该技术覆盖许多边界情况，使用无效数据作为输入，确保不存在可以利用的程序漏洞[73]。应用模糊测试方法对智能合约进行测试时，首先利用随机引擎生成大量的随机数据构成可执行交易；为了探索尽可能多的智能合约状态空间，随机引擎会根据测试结果的反馈，动态调整生成的数据；最后基于有限状态机分析每一笔交易的状态，检测是否存在攻击威胁。

ContractFuzzer[13]是第一个应用模糊测试技术对以太坊智能合约进行检测的框架，该方法基于合约 ABI 规范生成模糊测试输入，通过定义不同缺陷的缺陷特征已检测合约缺陷；Trail of Bits 的产品 Echidna③是一种基于属性的测试工具，采用模糊测试技术在 EVM 字节码的基础上创建被测合约的函数调用序列，检查合约是否违反了用户定义的某些属性；ReGuard[17]扩展了模糊引擎（如 AFL 和 LibFuzzer），记录合约关键的执行跟踪信息，自动识别重入缺陷；sFuzz[22]是针对智能合约进行测试的全自动引擎，通过模糊测试引擎生成合约运行轨迹，利用漏洞模式分析发现合约中的安全漏洞。

①ANTLR, https://www.antlr.org

②Mythril: an open-source security analysis tool for Ethereum smart contracts, https://github.com/ConsenSys/mythril

③Trailofbits/Echidna: Ethereum fuzz testing framework, https://github.com/crytic/echidna

4.3.1.3 合约部署及运维安全保障方法

为了解决安全挑战 Ch3，目前合约部署及运维安全保障方法主要针对智能运行监测和智能合约异常修复及防御。

在智能合约运行维护过程中，跟踪智能合约执行交易中的关键特性对识别导致合约故障的缺陷非常重要。MAIAN[14]是用于智能合约执行跟踪的符号分析器，能够处理合约的字节码，并尝试构建交易跟踪以查找和确认错误，其重点是在调用合约的过程中检测漏洞。Vultron[20]是一种在智能合约运行时检测其漏洞的方法，遵循的原理是大多数漏洞是由于转移的金额与合同内部账本逻辑反映的金额不匹配造成的，该方法能捕获与交易相关的缺陷，不局限于任何特定攻击模式。ÆGIS[21]是一个通过攻击模式检测和还原恶意交易的系统，这些模式通过使用一种新型的特定领域语言描述恶意控制和数据流，通过区块链快速、透明地更新这些攻击模式，防止智能合约在运行时被攻击。

在智能合约运行过程中发现异常行为以及智能合约本身的缺陷至关重要，及时修复错误并部署正确的合约更具有挑战性。现有的升级智能合约的解决方案主要采用手动更新方式并且容易出错，Rodler 等人[48]提出 EVMPatch 框架，用于自动修补有问题的智能合约，使可升级的智能合约的部署和管理自动化，经验证 EVMPatch 可以检测合约中的整数溢出和权限控制缺陷。

此外，针对智能合约攻击防御，有研究在针对智能合约特有的运行机制进行缺陷检查的基础上指定不同攻击的防御策略。为了对智能合约进行实时的安全监管和防护，Zhao 等人[36]提出入侵检测系统架构 ContractGuard，将业务处理逻辑和安全防护逻辑融为一体，保护正在运行的智能合约，为智能合约提供防御，同时能够监测未知缺陷引起的异常攻击。DappGuard[37]是一种针对以太坊智能合约的检测和防御工具，该工具可以通过分析交易日志识别攻击，例如通过交易日志中的 gas 使用量、异常消息值和可疑的 fallback 函数调用发现智能合约漏洞攻击。

上述主要是针对运行过程中智能合约的监测，还有针对智能合约运行环境即虚拟机的安全保障方法。EVMFuzzer[25]是第一个使用差分模糊技术来检测 EVM 漏洞的工具，EVMFuzzer 可以高效地挖掘引发 EVM 性能差异用例以及不一致的执行结果，从而发现虚拟机漏洞。

4.3.2 隐私安全保障方法

智能合约的隐私保护对象主要是合约代码和合约数据，具有隐私保护的智能合约实现旨在智能合约实现过程中，实现合约代码的保护、用户身份匿名、交易数据加密等，避免在区块链上清晰地存储合约代码、合约变量、交易地址、交易金额等，从公众的角度保留交易隐私。

目前，智能合约隐私保护方法的研究主要体现在智能合约的实现过程和智能合约的执行过程，包括零知识证明（ZKP）、安全多方计算（SMPC）和可信执行环境（TEE）等技术。其中，零知识证明是具有强大隐私保护特性的密码技术，其基本思想是证明者可以向验证者证明某些断言是否准确，无需向验证者提供任何有用信息。零知识证明（ZKP）随后依次出现两种变体，即非交互零知识证明（NIZK）[74]和简洁非交互零知识证明（zk-SNARKs）[75]。通过这些关键技术，实现合约隐私保护，保证智能合约隐私安全。接下来主要从合约代码隐私保护方法和合约数据隐私保护方法两方面进行介绍。

4.3.2.1 合约代码隐私保护方法

为了解决隐私安全挑战 Ch4，针对智能合约代码隐私保护的研究集中在合约编写语言或执行机制设计上。

Kosba 等人[26]提出保护用户隐私的智能合约编程框架 Hawk，Hawk 帮助实现人员在不使用任何混淆技术或代码加密的情况下构建私有智能合约。Hawk 将智能合约分为私有和公开两部分：私有部分负责合约中涉及的秘密数据或功能，私有的资金流向和交易金额都以加密方式隐藏；公共部分负责对外部实体透明的公共代码，不涉及隐私数据和金钱。Hawk 框架向区块链发送加密信息，依靠 zk-SNARKs 确保合约正确执行，实现资金保护[76]。

从对合约代码进行加密的角度考虑，微软提出 Coco 框架[43]。Coco 框架规定用户发送的交易内容包括合约代码可以全部用私钥加密，并保证交易在可信计算黑盒内才能解密和执行，还支持在智能合约中指定用户查看合约的权限。Coco 框架利用可信执行环境 TEE 的证明功能和黑箱性质，将经过证明的区块链代码完全放入 TEE 中运行，区块链的运行状态可信且无法被外界获取。Arbitrum[45]是一个支持智能合约的加密货币系统，允许用户将私有智能合约作为一个虚拟机来实现，对合约规则进行编码，Arbitrum 中的验证者可以在不暴露虚拟机的任

何内部状态的情况下高效地验证交易，从而保证了智能合约的代码隐私。

对合约代码进行加密的同时需要验证合约正确性，通过 NIZK 可以保证合约状态更新的正确性，但是 NIZK 语句的表达能力有限，只适用于部分合约功能。针对智能合约和 NIZK 无法有效融合的问题，Samuel Steffen 等人[40]提出 zkay 语言，zkay 语言规定了智能合约中隐私变量的读写权限控制和逻辑，定义私有值所有者的隐私类型，允许实现人员通过将变量注释为特定账户的私有数据来指定数据所有权。为了执行 zkay 合约，自动将 zkay 合约转化为可以在公有链上执行的合约，转换后的合约同时保证隐私性和功能正确性，例如在没有解密情况下，不允许将私有数据写入公共存储。

虽然 Steffen 等人提出的 Zkay v0.1 原型实现展示了该方法的可行性，但其概念验证的实现受到了限制，例如加密不安全、缺乏重要语言功能。Baumann 等人[41]针对 Zkay v1.0 的局限性提出 Zkay v0.2，显著提高了系统的安全性、可用性、模块化和性能。特别地是，Zkay v0.2 支持最先进的非对称加密和混合加密，引入新的语言特性（如函数调用、私有控制流），支持更丰富的智能合约代码隐私保护。为了提高效率，减少合约编译时间和链上成本，允许不同的 zk-SNARKs 后端证明合约正确。

由于智能合约完全公开在链上并由矿工执行，导致智能合约的可扩展性和隐私性较低。与 Hawk 划分隐私和公开两部分的思路相似，Li 等人[39]提出将合约分解为链下合约和链上合约，从而增强智能合约的可扩展性和隐私性。具体地说，涉及高成本计算或敏感信息的合约功能可以拆分为链下合约，只由利益相关方签署和执行。该方法允许参与者在所有参与者都诚实的情况下，在链下达成一致协议，从而节省矿工的计算资源，同时保护涉及敏感信息的链下合约的隐私性。

以上研究针对合约代码隐私提出了相应的合约实现框架或语言，主要可以概括为以下 3 种思路：

（1）拆分合约，将涉及敏感信息的合约设计为私有合约或链下合约，不对外公开；

（2）定义智能合约语言，对合约中涉及敏感信息的变量、函数等元素进行权限控制；

（3）构建智能合约执行框架，对智能合约进行加密，结合 TEE 对合约进行解密和执行。

4.3.2.2 合约数据隐私保护方法

为了解决隐私安全挑战 Ch5，主要保护合约输入输出数据安全和合约内交易数据安全。

（1）合约输入和输出数据安全

首先针对智能合约数据输入输出问题，Zhang 等人提出了一种可信数据输入系统 TC（Town Crier）[27]，TC 将以太坊的智能合约和可信硬件 Intel SGX（Intel Software Guard eXtensions）结合，在没有可信赖服务运营商的情况下，向智能合约提供经过身份验证的数据；此外，支持带有加密参数的私有数据请求，使得区块链中其他用户无法查看请求内容，保证智能合约的用户隐私。

TEE 为智能合约数据提供了区块链不具备的机密性，通过链下可信计算环境也解决了智能合约和区块链无法应对的复杂计算场景问题。为了解决区块链缺乏保密性和性能差的问题，Cheng 等人[29]提出 Ekiden，利用一种新颖的架构将共识与执行分开，实现了 TEE 支持的智能合约隐私保护，并具有较高的可扩展性。Ekiden 将整个合约状态加载到 TEE 中，保证交易执行前的数据完整性，这适用于公有区块链中的简单和小型智能合约，但是不适用金融服务情境下复杂的智能合约。为了适用更复杂的应用场景，Yan 等人[47]提出了一种通过 TEE 支持链上机密性的系统设计方案 CONFIDE，其安全数据传输协议、数据加密协议保证数据的机密性和完整性。目前，CONFIDE 在联盟链上支持数以百万计的商业交易。

然而，隐私的保护依赖于可信硬件的安全性，一旦可信硬件被破坏，这些方案将变得无效。与基于 TEE 实现的智能合约隐私保护方案不同，Zhu 等人[35]提出基于 SMPC 的智能合约框架，系统地阐述基于 SMPC 的智能合约执行流程、语言结构和语法规范，实现了具有输入隐私性和计算正确性的 SMPC 方法，增强区块链中智能合约执行安全性。

（2）合约内交易数据安全

针对如何保障合约内交易数据不被无关节点获取，在智能合约执行过程中，合约数据隐私保护主要体现在参与智能合约交易的用户地址、交易金额、交易内容的隐私加密上。针对以太坊智能合约平台，Bünz 等人[30]提出一种分布式、保密的隐私协议 Zether，能够在多种账户模型上实现匿名支付。Zether 智能合约不仅能够保密用户账户的余额，还能够隐藏智能合约交易金额，除了交易本身的细节之外，还允许混淆交易中各方的身份，隐藏参与者的身份。为了使基于智能合约的隐私保护协议更实用，Li 等人[42]应用 zk-SNARKs 提出一种高效隐私

协议 Phantom，该协议通过选择合适的哈希加密函数平衡了智能合约的gas成本和zk-SNARKs计算复杂性的关系，实现了数字货币等去中心化应用的隐私保护。

为了保证合约交易用户身份等信息，从增强匿名性出发，隐私保护应用平台 Origo[44]在隐私协议中加入 ZKP 为智能合约添加匿名保护，将用户信息、交易金额和合约执行细节进行保密，但其他用户仍可以验证这笔合约交易是否被正确执行。还有研究利用零知识证明技术和安全多方计算实现保护隐私的智能合约编写框架，保证交易和合约参与方身份对合约以外的人匿名。Raziel 编程框架[28]将 SMPC 和 NIZK 结合，使用 NIZK 向第三方证明合约有效性而不泄露任何信息，通过 SMPC 提供一个编程框架，实现方便形式化验证和隐私保护的智能合约。SodsMPC[46]是一个量子安全的智能合约系统，SodsMPC 权限服务器（验证节点）通过安全多方计算协议执行合约，保证合约执行正确的同时保护数据隐私。此外，SodsMPC 保护合约用户匿名身份的同时，也保护合约业务逻辑隐私。

为了保护交易的隐私性和数据的不可篡改性，Kang 等人[38]设计并实现了支持 Hyperledger Fabric 的 FabZK，FabZK 只存储每笔交易的加密数据（如支付金额），并将区块链网络中成员之间的交易关系（如付款人和收款人）匿名化，隐藏共享账本上的交易细节，通过 Pedersen 承诺和零知识证明实现可审计的智能合约隐私保护。

智能合约隐私保护方法主要针对合约代码隐私和合约数据隐私。在合约代码隐私保护方面，主要通过拆分涉密合约、定义具有隐私保护能力的合约语言、构建具有隐私保护能力的合约执行框架等方式进行；在合约数据隐私保护方面，应用零知识证明、安全多方计算等技术进行合约数据加密，或者将合约数据加载到可信执行环境中保证数据的隐私性和可验证性。

安全保障技术类别	比例
合约设计	13.3%
合约实现、测试	37.8%
合约部署、运维	13.3%
合约隐私保护	35.6%

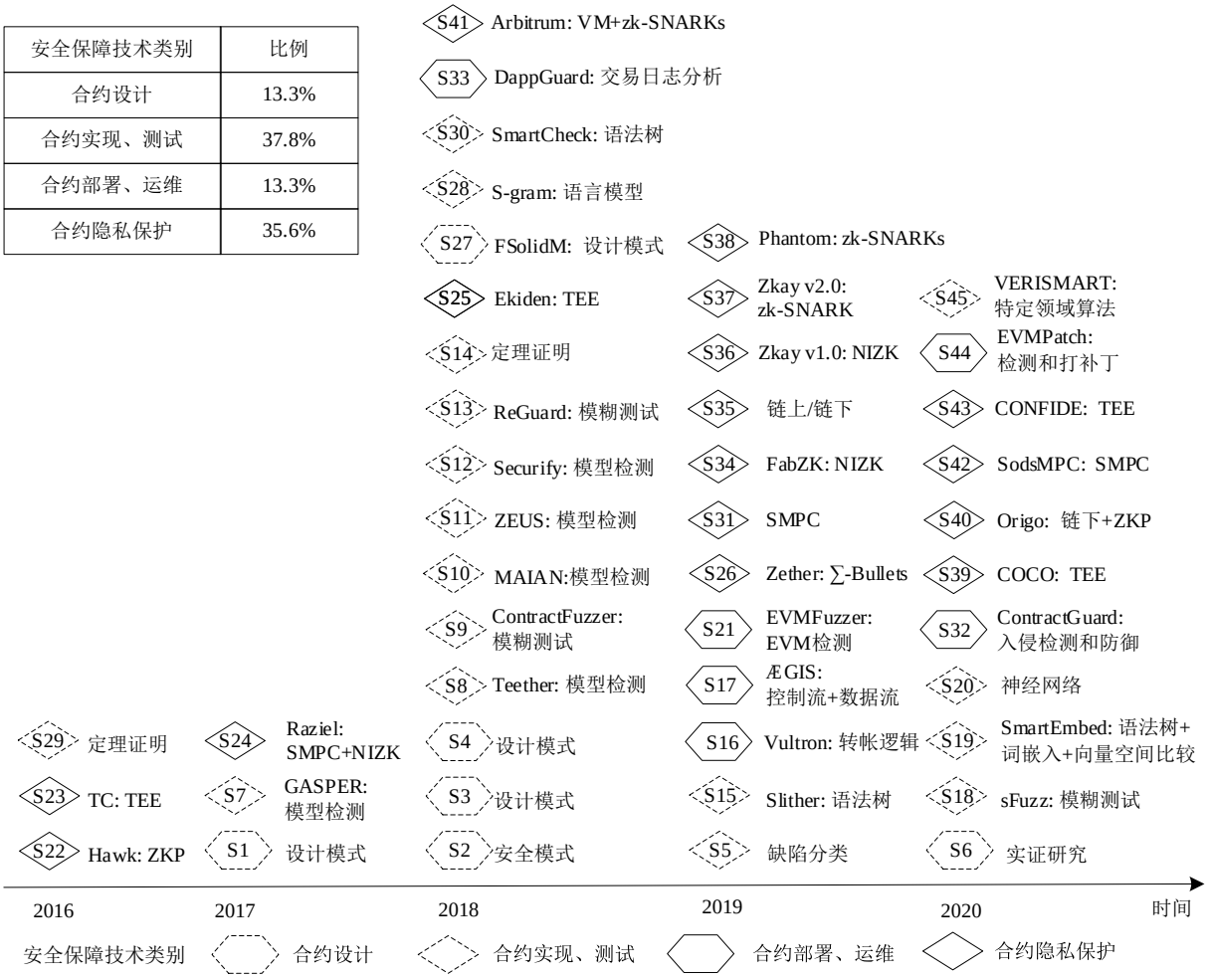

图 4 智能安全保障技术发展图

智能合约安全保障技术研究分布如图 4 所示，通过分析图 4 总结出 2 方面主要结论。在数量方面：

智能合约测试相关技术所占比例最大为 37.8%（17/45），其次是隐私保护相关研究比例为 35.6%（16/45），设计、部署、运维阶段相关研究相对较少，比例均为 13.3%（6/45）。本文分析产生该现象的主要原因有：首先，智能合约缺陷可能直接造成巨大的经济利益损失，因此合约的测试在合约正式投入使用前必不可少，准确、全面、高效的智能合约测试技术在智能合约的发展过程中将持续占据非常重要的地位；其次，智能合约发展时间较短，形成安全的合约设计理论需要一定时间的积累，因此在智能合约设计方面的研究成果并不显著；最后，具有隐私保护的智能合约实现主要依赖于传统的密码学技术，如何对此取得更大的突破具有一定难度。在发展规律方面：初始阶段（2016 年-2017 年），智能合约安全保障技术主要集中在隐私保护方面，主要应用密码学技术对智能合约隐私安全进行研究；随着智能合约的实际应用发展，其缺陷导致的安全问题逐渐显露，2018 年大量研究采用形式化验证的方式对智能合约缺陷检测展开研究并取得显著研究成果；2019 年，在智能合约静态检测的基础上，越来越多的研究结合智能合约的运行状态对合约进行分析，隐私保护方面也涌现出大量对已有密码学技术进行改进的合约隐私安全保障方法；2020 年，随着缺陷检测技术发展，新型研究旨在结合新兴技术突破传统缺陷检测技术限制，例如机器学习、深度学习等，也有更多的公司或社区（例如阿里巴巴、微软、Origo 等）关注智能合约隐私安全。

4.4 安全保障方法验证

本文参考 Shaw[77]提出的验证方法分类，结合本文智能合约安全相关文献的特点，选择出以下常见的验证方法。

- 分析。通过严格的推导证明或精心设计的实验验证研究方法。
- 实践。研究方法有对应的原型工具或者应用于实际场景、项目中，并通过收集真实信息验证该方法。
- 举例。通过小规模举例验证研究方法。

表 8 显示了 45 篇主要文献的验证方法的分布。根据表 8 可以看出，大部分研究都采用了分析（21/45）或实践（18/45）的方式验证智能合约安全保障方法的有效性，6 篇文献究采用举例的方式说明了方法对部分应用场景有效。

4.4.1 合约安全保障方法验证

（1）合约设计安全保障方法验证

在智能合约的设计阶段，安全保障方法主要以经验理论的形式呈现，体现为合约设计模式的定义，其验证方通过举例或实践，主要验证设计模式、安全模式针对不同应用场景的适用性[5,6,7]和基于设计模式生成代码的正确性[31]。

表 8 主要文献的验证方法分布

序号	验证方法		
	分析	实践	举例
S1			√
S2			√
S3			√
S4			√
S5			√
S6			√
S7	√		
S8		√	
S9		√	
S10		√	
S11	√		
S12		√	
S13	√		
S14	√		
S15		√	
S16	√		
S17	√		
S18		√	
S19	√		
S20	√		
S21		√	
S22	√		
S23		√	
S24	√		
S25	√		
S26		√	
S27		√	
S28	√		
S29	√		
S30		√	
S31	√		
S32	√		
S33		√	
S34		√	
S35	√		
S36	√		
S37	√		
S38	√		
S39		√	
S40		√	
S41		√	
S42	√		
S43		√	
S44		√	
S45	√		
总计	21	18	6

（2）合约实现及测试安全保障方法验证

在合约实现、测试阶段，安全保障的主要研究对象是智能合约源码或字节码，其目标是验证合约功能正确、发现合约缺陷，主要以命令行工具、UI 界面工具等形式出现，因此大部分研究采用分析或实践的方式进行验证。通过严格的推导证明验证定理证明方法是否能够检测合约功能正确[18,33]，通过设计实验验证模糊测试方法能够有效生成测试路

径、检测合约缺陷[17,22]，通过开发工具验证模型检测、抽象语法树等方法能够有效检测缺陷[12,16,19,22,34]。合约测试重点关注缺陷检测效果以及效率，应用二分类算法评估检测工具效果，评估指标如下：

●TP：工具检测为缺陷且正确的样本数量；

●FP：工具检测为缺陷但不正确的样本数量；

●FN：工具未检测出但实际为缺陷的样本数量；

●查准率 Precision=TP/(TP+FP)：工具检测出的所有缺陷样本中，被正确地检测出的样本比率，该指标越高越好；

●查全率 Recall= TP/(TP+FN)：人工标注的所有缺陷样本中，工具正确检测出的样本比率，该指标越高越好。

●漏报率 FNR=FN/(TP+FN)：人工标注的所有缺陷样本中，工具未正确检测出的样本比率，该指标越低越好。

（3）合约部署及运维安全保障方法验证

在合约部署及运维阶段，安全保障的主要研究对象是智能合约运行状态，目的是通过分析智能合约运行信息，识别异常行为、检测合约缺陷、修复合约缺陷、定制防御机制，主要采用分析和实践的方式进行验证。通过设计实验验证分析交易日志、控制流、数据流、资金流等方式能够检测合约异常[20,21]，通过开发工具验证运行状态下合约缺陷[14]、EVM 漏洞[25]的检测效果，通过设计实验验证合约修复方案是否正确[48]、是否能够防御攻击[36,37]。

4.4.2 隐私安全保障方法验证

（1）合约代码隐私保障方法验证

智能合约代码隐私保障方法主要针对智能合约执行机制和开发语言，目的是保证智能合约代码的编写安全和运行安全，主要采用分析和实践的方式进行验证。通过设计实验模拟合约执行验证链上链下合约拆分、公有私有合约拆分的方式能否保护合约隐私[26,39]，通过设计实验验证应用合约加密语编写的智能合约能否有效执行并能够保护其中机密代码[40,41]，通过开发原型系统验证合约代码隐私保护框架的隐私保护效果[43,45]。

（2）合约数据隐私保障方法验证

智能合约数据隐私保障方法主要针对合约数据输入、输出和计算过程，目的是保护合约内交易数据不被无关节点获取，主要采用分析和实践的方式进行验证。

对于应用零知识证明、安全多方计算等数据技术进行合约数据隐私保护，可以通过使用加密技术进行实验验证数据的隐私性和可验证性[28,42,46]，也可以在已有区块链平台上实现隐私保护机制验证数据加密的有效性[30,38,44]；对于将数据加载至可信执行环境的安全保障方法，可以基于 Intel 的 SGX 开发原型系统验证可信执行环境可以保证系统中运行的合约数据无法被无关节点获取[27,47]。

4.5 研究论文分布情况

本文还对其他相关信息感兴趣，如主要文献的时间和地点等。本文收集了每篇主要文献的出版年份和来源，并将所有的主要文献分为会议论文、期刊文章、研讨会论文和技术报告 4 种类型。图 5 提供了所有主要文献的发表年表，本文将会议、期刊名称的缩写或技术报告所属单位缩写放在每篇文献的旁边；对于研讨会论文，将 A@B 作为来源的格式，其中 A 是研讨会名称的缩写，B 是举办该研讨会的会议名称。

图 5 中的虚线总结了 2015 年至 2020 年每年发表的主要文献的数量。2015 年，智能合约发展处于早期阶段；2016 年研究主要体现在结合传统的密码学技术展开的合约隐私保护研究；2016 年 6 月发生的 The DAO 事件造成的巨大经济损失，引起各界学者的关注；2017 年，智能合约安全的相关研究开始出现；2018 年，智能合约安全相关的研究迅速增多；2019 年，研究数量有所下降，但是不乏高水平论文。

45 篇主要文献中，大部分研究发表在会议上，其中 6 篇文献[13,17,20,22,25,32]发表在软件工程的顶级会议上、10 篇文献发表在安全领域四大顶级会议上[12,15,16,21,26,27,40,41,44,45]。接近四分之一的研究发表在期刊上，其中 2 篇文献发表在软件工程领域的顶级期刊 TSE 上[10,23]。5 篇文献发表在研讨会上，均为软件工程领域或安全领域顶级会议举办的研讨会。最后，还有两项隐私保护相关的研究是微软和 Origo 发表的技术报告。从这些收集到的数据中，可以看出关于智能合约安全的论文数量在不断增加，但仍然较少。考虑到智能合约安全的研究领域和相对较短的历史，主要文献的数量相当可观。另外，由于还有许多难题需要解决，可能有更多研究工作正在进行并会在未来发表。

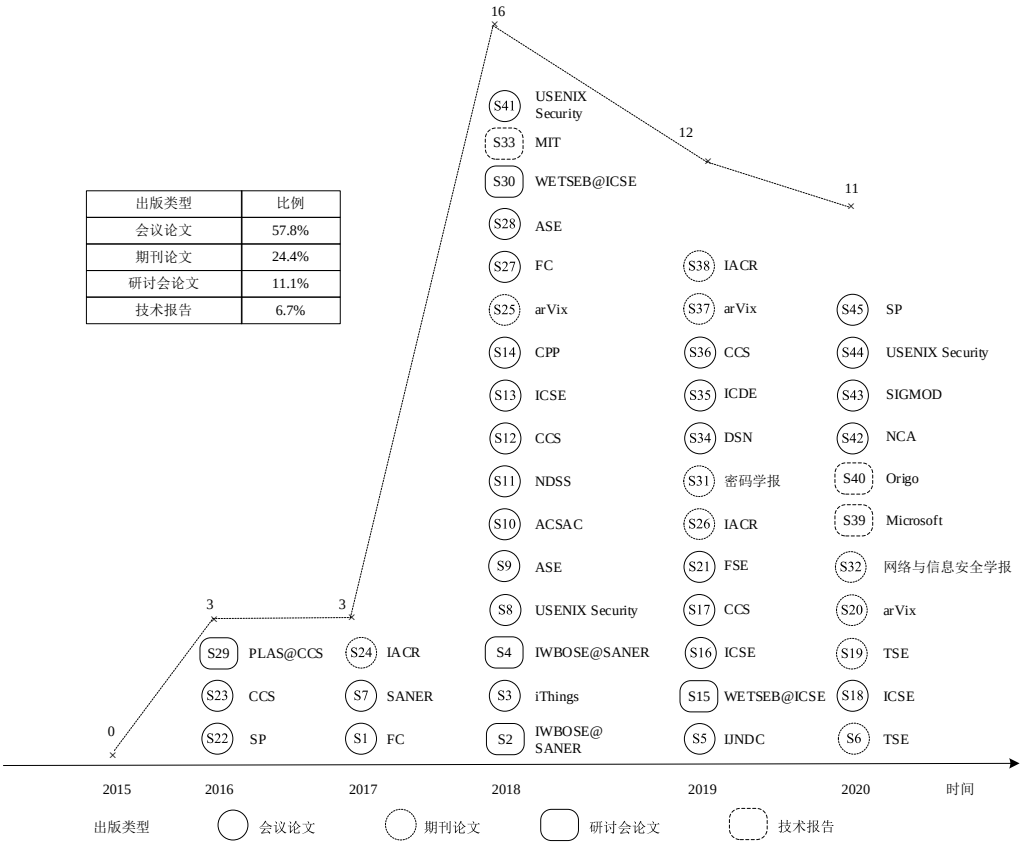

图 5 主要文献出版统计图

5 讨论

5.1 现有研究不足

5.1.1 合约安全研究不足

（1）设计安全保障技术不足

目前，智能合约设计模式的研究仍然处于起步阶段，相关研究数量有限，需要对已经整理好的模式进行扩展，为 Solidity 语言创建一个结构化的、信息量大的设计模式库，涵盖典型的和常见的编码场景，作为合约实现人员的指导；也可以在自动代码生成框架中应用设计模式，用来提取代码构件，这些构件可以被集成到自动代码生成框架中。Solidity 设计模式可以与其他智能合约平台中发展的编码实践进行比较，进一步揭示出更抽象的设计模式，这些模式独立于底层实现框架，对一般的智能合约有效[78]。

（2）实现及测试安全保障技术不足

针对主流的智能合约缺陷检测原理，可以将智能合约缺陷检测技术划分为：基于定理证明的缺陷检测、基于模型检测的缺陷检测、基于模糊测试的缺陷检测、基于语法树的缺陷检测和其他缺陷检测。针对每种技术，其优缺点如下所示：

定理证明的形式化验证方法通过抽象的数学描述得到良好的模型和属性表达，所以该方法的优势是即使针对状态多而复杂的系统，也不会出现状态空间爆炸的问题[79]。定理证明可以以任意属性为目标，提供强有力的形式化验证保证，验证方法非常精确，并且不会产生误报[18,80]。但是，基于定理证明的形式化验证方法存在较多局限性，例如机制复杂，需要测试人员系统学习该证明方法的工作原理和如何读取输出等，因此该方法具有较高的门槛；此外，基于定理证明的形式化验证工具使用通用方法来构造代码模式和定理，证明智能合约的安全性[18,33,68]，整个验证过程很难实现完全自动化，需要大量人工来构建智能合约的证明和分析[18]。人工构建的方式带来的问题首先是该方法扩展性不足，无法适用于大型智能合约，也很难实现大批量智能合约验证，还可能导致验证效率降低和人的主观性对验证结果产生影响。

模型检测方法的优势是自动化程度较高，即便只给出部分合约，也可以提供关于已知部分正确性的有用信息。特别地是，模型检测会输出终止时的反例，即程序缺陷存在的实例。直接输出缺陷实例

的方式可以有效帮助实现人员理解程序缺陷并修改程序。模型检测方法的不足是存在状态爆炸问题，若是基于较大的状态空间，模型检测算法会随着状态数目增多而造成运算时间爆炸式增长和运算空间膨胀。

模糊测试方法的优势是可根据实际代码定制不同的输入，可选择不同的覆盖范围发现更深层次的错误，更加精确的检测出合约的真实漏洞。模糊测试方法的不足是测试用例随机生成，只能覆盖部分系统行为，无法找出所有潜在错误；另外，该方法选择特定的测试用例作为种子，基于种子不断生成新的测试用例，检测性能可能会随着初始种子的选择而发生变化，实际检测性能难以控制。

采用抽象语法树作为智能合约的基础表现形式的优势在于不依赖智能合约具体的语言细节，对程序进行逻辑分析，抽取重要的程序执行逻辑信息；但是，抽象语法树构建的合约缺陷检测规则不适用过于复杂的问题，可能会导致较高误报率；特别地是，基于抽象语法树的方法表示过于高级，无法准确反映低级信息，例如 gas 计算等。

（3）部署及运维安全保障技术不足

目前，针对智能合约运行环境安全检测的研究数量较少，本文仅涉及 1 项相关研究，即 EVM 漏洞检测工具 EVMFuzzer，该工具目前主要检测 EVM 执行结果不一致的漏洞。针对合约运行状态的监测，ContractGuard[36]由于嵌入了额外的代码，一定程度上会增加智能合约的部署开销与运行开销。Vultron[20]检测方法不适用于不使用内部记账逻辑的智能合约，无法检测时间戳依赖性或交易顺序依赖性等缺陷。ÆGIS[21]系统保护已部署的智能合约，实现攻击模式的动态更新，但是该工具对于新攻击模式的引入依赖于预定投票群体达成共识，攻击模式以纯文本的形式公开，可能被潜在攻击者利用。对于智能合约缺陷修复，EVMPatch [48]对存在缺陷的合约进行打补丁，但是可能导致修复后的合约代码产生更高 gas 消耗

本文将智能合约缺陷检测方法划分为静态分析和动态分析。静态分析在非运行时环境中分析程序或编译代码，其主要优势能够识别智能合约中的关键模式。但是，静态分析无法检测执行期间发生的缺陷，针对一些缺陷可能会导致误判。

动态分析是一种在执行或运行时检查程序的方法，其优点是通过动态分析可以检测出静态分析无法检测的执行期间异常，识别出未知合约缺陷导致的攻击，还可以验证静态代码分析的结果。但是，动态分析的不足是合约的动态执行基于用户交互或自动测试，因此不能保证覆盖源代码所有执行路径；同时，需要保证在不影响智能合约正常运行的情况下获取合约的执行状态信息并进行有效的跟踪和分析，这也对智能合约动态运行监测提出更高要求。

最后，总结上述智能合约缺陷检测工具的局限性如下表 9 所示：

表 9 智能合约缺陷检测工具汇总表

分类	工具	检测对象	局限
静态分析	GASPER [11]	字节码	只能探索部分程序路径，依赖于消耗 gas 量高的模式清单
	Teether [12]	字节码	准确率较低，无法检测在特定情况下才会出现的缺陷
	ZEUS [15]	源码	需要针对合约定义特定的策略，且 ZEUS 发现的缺陷的实际可用性不确定
	Securify [16]	字节码	可检测的智能合约版本有严格限制
	Slither [19]	源码	缺乏形式语义无法实现更严格的分析，表示过于高级无法准确反映低级信息，如 gas 计算
	SmartEmbed [23]	源码	需要构建包含缺陷的智能合约代码库，缺陷的检测范围局限于代码库中的缺陷类型
	SmartCheck [34]	源码	无法描述更复杂的规则，导致误报率较高
	VERISMART [49]	源码	目前主要检测整数溢出和被零整除两种算术操作问题
	Oyente [50]	字节码	限制循环次数防止路径爆炸，导致部分缺陷漏报，无法检测逻辑相关问题
动态分析	Echidna①	字节码	无法保证 API 功能的稳定性
	Mythril②	字节码	无法检测出智能合约的业务逻辑问题，误报率高
	ContractFuzzer [13]	字节码+ ABI	测试用例随机生成，路径覆盖率有限
	MAIAN [14]	字节码	只能检测 3 种缺陷，针对其他合约调用引起的问题无法进行检测
	ReGuard [17]	字节码/源码	检测的缺陷局限于重入缺陷
	Vultron [20]	合约运行状态	不适用于不使用内部记账逻辑的智能合约，无法检测时间戳依赖或交易顺序依赖等缺陷
	ÆGIS [21]	合约运行状态	新攻击模式引入依赖于预定投票群体达成共识，攻击模式以文本形式公开可能被攻击利用
	sFuzz [22]	字节码	性能可能会随着初始种群的选择而变化
	ContractGuard [36]	合约运行状态	无法有效地防御复杂的逻辑错误，用于有丰富特性的智能合约时会产生相当高的额外开销
	DappGuard [37]	交易日志	结合 Oyente 检测已有合约缺陷存在一定局限
	EVMPatch [48]	合约运行状态	修复后的合约代码可能产生更高 gas 消耗

①Trailofbits/Echidna: Ethereum fuzz testing framework, https://github.com/crytic/echidna
②Mythril: an open-source security analysis tool for Ethereum smart contracts, https://github.com/ConsenSys/mythril

本文共调查了表9中20种智能合约缺陷的方法适用的合约缺陷，图 6 是智能合约缺陷分类及检测图，第 1 层展示了智能合约缺陷产生的相关原因，第 2 层展示不同缺陷，第 3 层展示了缺陷检测工具。如图 6 所示，编译器版本过时（D12）、硬地址编码（D15）和调用类型不匹配（D25）没有相应检测工具。D12 是编译相关的缺陷，实现人员应在实现过程中遵循最佳实践，例如锁定 Pragma 版本避免一起升级带来的相关风险；针对地址合法验证产生的 D15 需要人工核查地址是否真实，防止资金转入不存在账户而永久丢失；D25 只在特定外部合约调用情况下才能被触发，常规情况下可能没有任何异常。

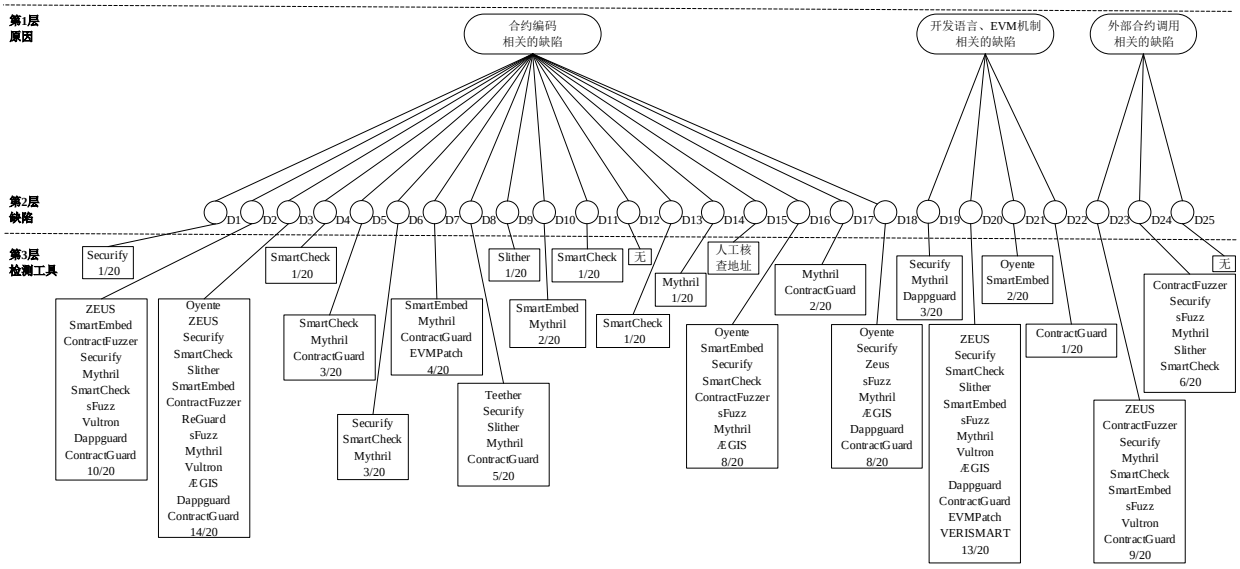

图 6 智能合约缺陷分类及检测方法

5.1.2 隐私安全研究不足

（1）合约代码隐私保护不足

保护用户隐私的智能合约编程框架 Hawk[26]可以为部分智能合约代码提供隐私性，但 Hawk 不是完全去中心化的，存在一个管理者作为可信第三方进行监督。此外，对于简单合约，由于 Hawk 利用 zk-SNAKRs 保证资金转移和合约执行的正确性，导致较高计算开销。目前，Hawk 由于效率较低，不能直接部署在大多数区块链系统上。

对于通过定义智能合约语言 Zkay[40,41]保护智能合约中的关键信息，该方法需要结合语言发展不断引入新的语言特性，以支持更丰富的智能合约代码隐私保护。此外，智能合约代码形式多样，通过对部分私有代码元素进行加密是否能全面保护私有信息不被泄露需要进一步验证。并且，为了执行 Zkay 合约，需要将其转换为具有相同功能且具有隐私保护能力的 Solidity 合约在以太坊上执行，推广使用难度较高。

与 Hawk、Zkay 不同，结合 TEE 的 Coco[43]框架具有更好的通用性，理论上可以实现任意区块链系统的隐私保护。然而，Coco 框架是针对联盟链的优化技术，对现有比特币、以太坊等架构影响有限。

（2）合约数据隐私保护不足

TC[27]、CONFIDE[47]和 Ekiden[29]依赖于 TEE 为智能合约提供数据隐私，TEE 为智能合约数据提供区块链所不具有的机密性，解决智能合约和区块链无法应对的复杂计算场景问题。在应用 TEE 实现高效的区块链隐私保护的同时，最大弊端就是安全依赖于可信硬件 Intel SGX，需要信任硬件厂商及其平台运营方，导致没有实现完全去中心化。本文参考 Abraham 等人[81]采用以下6个指标对本文涉及的隐私保护方法进行评估，评估结果如表 10 所示。

●完整性：在数据存在的整个过程保证其完整且准确；

●可用性：合约数据可以被授权节点访问使用；

●隐私性：私有数据不可被未授权节点访问；

●持久性：合约数据永久可用；

●性能：该技术生成和验证证明的速度；

●可扩展性：能够扩展到区块链网络中的参与节点数量。

如表 10 所示，根据隐私保护技术对智能合约隐私保护方法进行分类，可以分为基于软件的合约数据隐私保护方法和基于硬件的合约数据隐私保护方法。基于软件的隐私保护方法就是使用密码学技术，

主要包括零知识证明、同态加密和安全多方计算；　　基于硬件的隐私保护方法就是使用可信执行环境。

表 10 合约数据隐私保护方案对比

指标	方案	Zether	FabZK	Phantom	SodsMPC	Raziel	Origo	Arbitrum	TC	CONFIDE	Ekiden
	技术	∑-Bullets	NIZK	zk-SNARKs	SMPC	SMPC +NIZK	Off-chain +ZKP	VM+ zk-SNARKs	TEE	TEE	TEE
完整性		No	No	No	No	No	No	No	Yes	Yes	Yes
可用性		Yes	Yes	Yes	No	No	No	Yes	No	No	No
隐私性		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
持久性		Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No
性能		Low	Low	High	Low	Low	High	High	High	High	High
可扩展性		No	No	No	No	No	No	Yes	Yes	Yes	Yes

目前，密码学技术在实际应用中存在不同局限性。零知识证明可以隐藏用户身份和交易内包含的知识，但是零知识证明过程十分繁琐、构造复杂、表达能力有限。同态加密方案构造相对简单，但是效率低、占用区块链存储空间，不适用小成本、时效性要求高的智能合约。例如，Zether[30]应用改进的零知识证明机制∑-Bullets，一定程度上提升了性能，但单次加密转账的成本接近每个区块的最大 gas 消耗量，因此 Zether 可行性非常低；Phantom[42]通过改进默克尔树和哈希函数的方式显著提升效率，一笔加密交易的 gas 成本只有 1M 左右，交易生成时间不到 6 秒；为了弥补零知识证明的不足，Arbitrum[45]和 Orgio[44]通过链下计算的方式提升性能，降低链上计算成本。

安全多方计算模型定义多方协议，允许在不侵犯输入隐私的情况下对私人数据输入进行联合计算，对手除了联合计算的输出之外，对真实方的输入一无所知。但是，安全多方计算协议或方案构造复杂，目前安全多方计算协议多适用于两方，限制了参与方数目，并且协议扩展困难，存在性能瓶颈。例如，SodsMPC 在性能上与 HoneybadgerMPC[82]相比得到一定提升，但是当共享信息数目超过一定数量之后，性能明显下降。

总的来说，使用密码学技术保护智能合约数据安全隐私主要存在的问题就是性能效率较低和可扩展性较低，即使不同的隐私保护方案在性能方面都有一定提升，但是与基于硬件的合约数据隐私保护方案相比效率仍然较低。

可信执行环境提供了一个完全隔离的运行环境，通过限制其他软件应用程序和操作系统的访问，保持了在 TEE 中运行的代码或应用程序的完整性，保证了运行时智能合约的保密性，并且具有较高的性能和可扩展性。但是，如表 10 所示，TEE 既不能保证数据的可用性，也不能提供持久存储，主要因为在可信硬件上进行计算，只有将数据存储到区块链上才可被所有节点获取，一旦可信硬件损坏，所有数据将丢失。此外，TEE 修复升级代价高，很难实现安全强度升级，并且存在管理者泄露用户私人数据的风险。总的来说，不同的隐私保护技术的优势和局限性如下表 11 所示：

表 11 隐私保护技术局限性

隐私保护技术	优势	局限性	性能
零知识证明	重点在于证明，不需要数据细节	构造复杂、门槛高、表达能力有限、不支持多方计算、证明数据透明	较低
同态加密	构造相对简单	性能瓶颈、表达能力有限、数据验证追责难	较低
安全多方计算	重点在于计算	协议/方案构造复杂、基于两方、扩展困难、参与方数目限制、性能瓶颈	较低
可信执行环境	性能高，扩展性好	信任硬件厂商、信任平台运营方、修补升级代价高、安全强度升级困难	较高

目前智能合约隐私保护的方法核心仍然集中在密码学技术，复杂的密码学技术原理和隐私保护过程对交易数量、交易处理效率产生限制，也是制约智能合约隐私保护技术的主要因素。不同的隐私保护技术都有其各自的优势和不足，如何针对区块链技术的不同应用场景，因地制宜定制不同隐私保护策略是未来研究方向之一。

5.1.3 相关综述论文的不足

Atzei 等人[51]分析了 10 种以太坊智能合约的安全漏洞，针对每种不同的漏洞，详细展示了利用这些漏洞展开的真实攻击事件。该研究 2017 年进行调研，覆盖的智能合约安全漏洞类型有限。Alharby 等人[83]使用 Mapping Study 对智能合约进行研究，从技术角度收集与智能合约相关的所有研究，将研究划分为智能合约的编码、安全、隐私和性能这 4 类，该文章的部分内容与智能合约相关，但没有对合约的安全保障技术进行系统性的全面分析。

大部分的综述性研究针对智能合约的漏洞检测展开，Parizi 等人[84]针对 Solidty 语言编写的以太坊智能合约安全漏洞，对智能合约测试工具进行实验评估，包括 Oyente、Mythril、Securitfy 和 SmartCheck。该研究是针对合约漏洞检测工具的实证研究，没有

针对智能合约漏洞和缺陷检测技术进行全面的分析和汇总。Praitheeshan 等人[79]调查了智能合约的 16 个安全漏洞，旨在从内部机制和软件安全漏洞的角度识别以太坊智能合约中的关键漏洞，并从静态分析、动态分析和形式化验证 3 个方面探索了 11 个检测智能合约安全漏洞的工具。

与上述的综述性文献相比，本文的优势在于对智能合约安全进行了全面的分析，主要针对合约安全和隐私安全，结合利益相关者、安全问题及挑战、安全保障技术和技术安全保障方法验证这 4 个方面进行了系统性的分析。对于合约安全，本文讨论了合约设计、实现、测试、部署、运维各个阶段的安全问题和安全保障技术，总结了 25 种智能合约可能存在的缺陷，统计了 20 种主流的缺陷检测方法；对于隐私安全，结合区块链系统原理，从合约代码隐私和合约数据隐私两个方面总结了安全问题以及现有的隐私保护技术，分析了合约隐私保护研究成果及其不足之处。

5.2 未来研究方向

综合分析现有智能合约安全保障研究存在的不足，本文总结出以下未来研究方向：

（1）先验方法和后验方法的有机融合

为了减少智能合约的合约安全问题、提高隐私保护能力，未来研究工作应当首先关注基于先验方法的智能合约安全保障。研究如何在智能合约的设计、实现阶段，针对不同的智能合约功能需求、编程语言、执行环境，定制智能合约开发标准，规范智能合约开发流程；根据实际应用场景，结合不同隐私保护技术的优缺点，扬长避短、因地制宜地设计隐私保护方案。在智能合约正式使用之前保障合约安全，减少实现过程中产生的缺陷、降低合约安全风险以及合约维护的代价。但是，在去中心化的区块链系统中，合约用户丰富、合约之间可以任意组合，这都给智能合约带来未知安全风险。因此，仅应用先验方法进行智能合约安全保障并不健全，无法有效保护运行使用中的智能合约安全。

针对开发完成后的智能合约必须加强基于后验方法的智能合约安全保护，重点研究如何通过智能合约测试的方式进行全方位的安全检测，如何针对部署后的智能合约进行实时的运行监测和分析，如何在系统实际使用过程中监测是否存在隐私泄露。

将先验方法和后验方法有机融合，从智能合约生命周期各个阶段保障智能合约安全，从设计、实现、使用、运维全过程保护合约隐私安全。

（2）定性方法和定量方法的有机融合

在合约的设计、开发过程中，通过定性分析的方法合理分析开发过程中可能存在的不同风险因素，有效的对合约安全威胁、综合影响等进行分类和分级，明确目前智能合约的整体安全状况和存在的问题。但是，定性分析方法通常结合人员经验，分析依据的精确度可能受主观因素的影响。

定量评估方法比较客观，提供的数据分析比较精确。在合约的测试、运维过程中，通过定量分析的方法检测存在风险的事件，确定风险因素在智能合约中的位置，梳理风险因素之间的关系，及时发现智能合约中存在的安全问题。

结合定性分析和定量分析的方法特点，将两者有机融合。针对智能合约生命周期中各个阶段的特点，使用定性分析方法对智能合约进行整体安全的分析，从中找出关键的功能模块、严重的安全威胁和巨大隐患；然后，针对这些重点使用定量的分析方法，收集数据进行深入分析，根据准确的分析结果，选择性价比高的风险应对措施。

（3）静态方法和动态方法的有机融合

针对合约安全，大量研究集中在合约测试，但大多采用单一的静态方法或动态方法进行合约缺陷检测。为了实现全方位的智能合约缺陷检测，可以考虑将静态方法和动态方法有机融合的方式，结合不同合约缺陷特征有针对性的获取合约静态和动态信息。重点研究构建完整的静态信息、动态信息相结合的合约分析框架，充分融合智能合约本身的语法、语义信息以及智能合约运行状态信息，进行全方位的智能合约分析。

目前，智能合约的动态分析研究相对较少，因此更具有难度。针对部署后实际运行的智能合约，不仅针对合约本身的运行状态进行监测，还需要考虑智能合约运行环境可能带来的安全威胁。针对合约运行异常行为分析，不仅需要能够发现已知缺陷，还应关注智能合约未知缺陷识别，以及如何建立健全的容错和异常终止处理机制。

总的来说，智能合约未来的研究应针对不同的业务目标定制相应的验证规范描述，突破成本昂贵、不适应大规模合约等技术限制，从验证一般功能属性和安全属性、检测常见缺陷到逐步实现全方面的区块链系统安全评估。

5.3 本文方法的局限性

本文方法可能存在有两个潜在的局限性。（1）检索范围限制，难免有遗漏；（2）数据提取过程难

以保障 100%的准确性和完整性。

首先，为了保证选题过程的公正性，本文设计了检索策略，通过检索关键词搜索相关文献。但是，关键词的选择具有主观性，可能会导致遗漏一些相关研究。此外，本文选择了一组全面的、重要的以计算机科学为核心的电子数据库作为检索范围，并收集了部分权威机构的技术报告，以尽可能多地覆盖相关研究。尽管如此，由于智能合约发展时间较短、技术较为新颖，与学术性研究论文相比，智能合约相关的技术网站可能具有更强的时效性，因此本文可能会遗漏部分相关研究。

第二，提取结果可能会有一些不准确和不完整的地方，尤其是针对智能合约缺陷部分。不同的研究在对智能合约进行测试的过程中，对存在安全问题的智能合约定义各不相同。本文根据缺陷产生原因，对智能合约存在的 25 种典型缺陷进行了分类汇总，但是这 25 种缺陷并不能覆盖 45 篇主要文献涉及的所有智能合约缺陷，并且在分类汇总过程中，本文统一了部分表达方式不一致但具有相似安全问题的智能合约缺陷或漏洞。

6 结论

本文对智能合约安全进行了全面的调研，主要有以下研究发现。

GQ1：在智能合约安全保障过程中，共涉及合约设计人员、合约实现人员、合约测试人员、合约监管人员和合约用户这 5 类利益相关者。

GQ2：智能合约的安全和挑战主要包括合约安全和隐私安全。针对合约安全，主要体现在合约设计、实现及测试、部署及运维，隐私安全则主要关注合约代码隐私和合约数据隐私，针对 5 种不同类型的合约安全和隐私安全问题，智能合约面临着不同安全挑战。

GQ3：针对智能合约安全保障方法，合约设计相关研究主要通过定义抽取设计模式展开；合约实现及测试相关研究主要应用定理证明、模型检测、模糊测试、抽象语法树等技术检测智能合约缺陷；合约部署及运维主要分析日志或信息流图，监测合约异常行为，发现合约缺陷、EVM 漏洞并制定相关防御策略；代码隐私保护方法主要集中在合约拆分、合约语言设计和 TEE 技术应用；数据隐私主要结合零知识证明、安全多方计算等密码学技术对数据进行加密，或将数据加载至可信执行环境保护隐私。

GQ4：针对智能合约安全保障方法验证，合约安全保障方法主要通过分析、实践、举例的方式进行验证。

FQ1：对于合约安全，设计文本、合约编码规范、实现语言、EVM 机制、外部合约调用、运行环境等因素都可能与合约安全威胁产生相关；对于隐私安全，公有链公开透明的性质和合约隐私保护机制不健全对合约代码隐私产生威胁，此外，公开数据的恶意分析和合约用户泄漏都可能导致数据隐私威胁。

FQ2：已有的合约安全保障研究初步提出了合约设计理念、实现了大部分智能合约缺陷检测、完成了基本的合约运行状态监测。但是，安全可信的智能合约设计仍没有形成健全的体系；合约测试方法多样，每种方法可检测的缺陷类型有限，未形成全面、体系的智能合约缺陷检测框架，通用性不高；对于合约部署及运维安全，尚未实现灵活、有效的合约维护机制。

FQ3：已有的合约代码隐私保护方案能够实现智能合约代码核心内容保护，但存在应用场景不丰富、适用平台不广泛等问题；合约数据隐私保护方案结合密码学技术实现不同程度的合约数据隐私保护，主要存在性能和没有实现完全去中心化的问题。

FQ4：智能合约安全的未来研究方向针对合约安全和隐私安全展开，支持智能合约全方位、全生命周期的安全保障，基本思路是先验方法和后验方法结合、定性方法和定量方法结合、静态方法和动态方法结合。

SQ1：该领域的研究始于 2015 年，2016 年仅有少量相关研究成果发表，2018 年开始研究成果数量增长迅速，并有继续上升的趋势。

SQ2：18 篇文献发表在软件工程或安全领域的顶级期刊和会议，说明智能合约安全是一个备受关注的新兴课题。

总的来说，目前已有较多关于智能合约安全的研究，值得研究者投入更多精力解决这些有趣问题。

参考文献

[1] Buterin V. A next-generation smart contract and decentralized application platform. White paper, 2014

[2] Yuan Y, Wang F Y. Blockchain and cryptocurrencies: model, techniques, and applications. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, 48(9): 1421-1428

[3] Kitchenham B, Charters S. Guidelines for performing systematic

literature reviews in software engineering. Department of Computer Science, Durham: University of Durham, EBSE Technical Report: EBSE-2007-01, 2007

[4] Petticrew M, Roberts H. Systematic reviews in the social sciences: a practical guide. Pondicherry: SPI Publisher Services, 2006

[5] Bartoletti M, Pompianu L. An empirical analysis of smart contracts: platforms, applications, and design patterns//Proceedings of International Conference on Financial Cryptography and Data Security (FC). Sliema, Malta, 2017: 494-509

[6] Wohrer M, Zdun U. Smart contracts: security patterns in the Ethereum ecosystem and solidity//Proceedings of the 2018 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE@ SANER). Campobasso, Italy, 2018: 2-8

[7] Wohrer M , Zdun U . Design patterns for smart contracts in the Ethereum ecosystem//Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings). Halifax, Canada, 2018: 1513-1520

[8] Destefanis G, Marchesi M, Ortu M, et al. Smart contracts vulnerabilities: a call for blockchain software engineering? //Proceedings of the 2018 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE@SANER). Campobasso, Italy, 2018: 19-25

[9] Dingman W, Cohen A, Ferrara N, et al. Defects and vulnerabilities in smart contracts, a classification using the NIST bugs framework. International Journal of Networked and Distributed Computing, 2019, 7(3): 121-132

[10] Chen J, Xia X, Lo D, et al. Defining smart contract defects on Ethereum. IEEE Transactions on Software Engineering (TSE), to appear

[11] Chen T, Li X, Luo X, et al. Under-optimized smart contracts devour your money//Proceedings of the 24th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). Klagenfurt, Austria, 2017: 442-446

[12] Krupp J, Rossow C. Teether: Gnawing at Ethereum to automatically exploit smart contracts//Proceedings of the 27th USENIX Security Symposium. Baltimore, USA, 2018: 1317-1333.

[13] Jiang B, Liu Y, Chan W K. ContractFuzzer: Fuzzing smart contracts for vulnerability detection//Proceedings of the 33rd IEEE International Conference on Automated Software Engineering. Montpellier, France, 2018: 259-269

[14] Nikolic I, Kolluri A, Sergey I, et al. Finding the greedy, prodigal, and suicidal contracts at scale//Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC). San Juan, USA, 2018: 653-663

[15] Kalra S, Goel S, Dhawan M, Subodh Sharma. ZEUS: Analyzing safety of smart contracts//Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS). San Diego, USA, 2018: 18-21

[16] Tsankov P, Dan A M, Drachsler-Cohen D, et al. Securify: Practical security analysis of smart contracts//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS). Toronto, Canada, 2018: 67-82

[17] Liu C, Liu H, Cao Z, et al. ReGuard: Finding reentrancy bugs in smart contracts//Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings (ICSE). Gothenburg, Sweden, 2018: 65-68

[18] Amani S, Myriam B, Bortin M, et al. Towards verifying Ethereum smart contract bytecode in Isabelle/HOL//Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP). Los Angeles, USA, 2018: 66–77

[19] Feist J, Grieco G, Groce A. Slither: A static analysis framework for smart contracts//Proceedings of the 2nd IEEE International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB@ICSE). Montreal, Canada, 2019: 8-15

[20] Wang H J, Li Y, Lin S W, et al. Vultron: Catching vulnerable smart contracts once and for all//Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE: NIER). Montreal, Canada, 2019: 1–4

[21] Torres C F, Baden M, Norvill R, et al. ÆGIS: Smart shielding of smart contracts//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS). London, UK, 2019: 2589-259

[22] Nguyen T D, Pham L, Sun J, et al. sFuzz: An efficient adaptive fuzzer for Solidity smart contracts//Proceedings of the 43rd International Conference on Software Engineering (ICSE). Madrid, Spain, 2020: 778-788

[23] Gao Z, Jiang L, Xia X, et al. Checking smart contracts with structural code embedding. IEEE Transactions on Software Engineering (TSE), to appear

[24] Gogineni A K, Swayamjyoti S, Sahoo D, et al. Multi-class classification of vulnerabilities in smart contracts using AWD-LSTM, with pre-trained encoder inspired from natural language processing. arXiv:2004.00362, 2020

[25] Fu Y, Ren M, Ma, F C, et al. EVMFuzzer: Detect EVM vulnerabilities via fuzz testing//Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE). New York, USA, 2019: 1110–1114

[26] Kosba A, Miller A , Shi E , et al. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts//Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP). San Jose, USA, 2016:839-858

[27] Zhang F, Cecchetti E, Croman K, et al. Town crier: An authenticated data feed for smart contracts//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS). Vienna, Austria, 2016: 270–282

[28] Sánchez D C. Raziel: Private and verifiable smart contracts on blockchains. IACR Cryptology ePrint Archive, 2017

[29] Cheng R, Zhang F, Jernej K, et al. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. arXiv:804.05141, 2018

[30] Bünz B, Agrawal S, Zamani M, Boneh D. Zether: Towards privacy in a smart contract world. IACR Cryptology ePrint Archive, 2019

[31] Mavridou A, Laszka A. Designing secure Ethereum smart contracts: A finite state machine sased approach//Proceedings of the 22nd International Conference on Financial Cryptography and Data Security (FC). Nieuwpoort, Curaçao, 2018: 523-540

[32] Liu H, Liu C, Zhao W, et al. S-gram: Towards semantic-aware security auditing for Ethereum smart contracts//Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE). Montpellier, France, 2018: 814-819.

[33] Bhargavan K, Delignat-Lavaud A, Fournet C, et al. Formal verification of smart contracts: Short paper//Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security (PLAS@CCS). Vienna, Austria, 2016: 91-96

[34] Tikhomirov S, Voskresenskaya E, Ivanitskiy I, et al. SmartCheck: Static analysis of Ethereum smart contracts//Proceedings of the 1st IEEE/ACM International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB@ICSE). Gothenburg, Sweden, 2018: 9-16

[35] Zhu Y, Song X X, Xue X B, et al. Smart contract execution system over Blockchain based on secure multi-party computation. Journal of Cryptologic Research, 2019, 6(2): 246-257 (in Chinese)
(朱岩, 宋晓旭, 薛显斌, 秦博涵,等. 基于安全多方计算的区块链智能合约执行系统. 密码学报, 2019, 6(2): 246–257)

[36] Zhao G S, Xie z J, Wang X M, et a1. ContractGuard: Defend Ethereum smart contract with embedded intrusion detection. Chinese Journal of Network and Information Security, 2020, 6(2): 35-55 (in Chinese)
(赵淦森，谢智健，王欣明，等. ContractGuard: 面向以太坊区块链智能合约的入侵检测系统.网络与信息安全学报, 2020, 6(2):35-55)

[37] Cook T, Latham A, Lee J H. DappGuard: Active monitoring and defense for solidity smart contracts. Boston,USA: MIT, Technical report: 23, 2018

[38] Kang H, Dai T , Jean-Louis N , et al. FabZK: Supporting privacy-preserving, auditable smart contracts in Hyperledger Fabric// Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Portland, USA, 2019: 543-555

[39] Li C, Palanisamy B , Xu R H. Scalable and privacy-preserving design of on/off-chain smart contracts//Proceedings of the35th IEEE International Conference on Data Engineering Workshops (ICDE), Macao, China, 2019:7-12

[40] Steffen S, Bichsel B, Gersbach M, et al. Vechev. Zkay: Specifying and enforcing data privacy in smart contracts//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS). London, UK, 2019: 1759-1776

[41] Baumann N, Steffen S, Bichsel B, Tsankov P. Vechev: Zkay v0.2: Practical data privacy for smart contracts. arXiv:2009.01020, 2020

[42] Li X, Zheng Y, Xia K X, Sun T C, Beyler J. Phantom: An efficient privacy protocol using zk-SNARKs based on smart contracts. IACR Cryptology ePrint Archive, 2020

[43] Russinovich M, Ashton E, Avanessians C, et al. CCF: A framework for building confidential verifiable replicated services. Washington: Microsoft, Technical report: MSR-TR-2019-16, 2019

[44] Origo. Privacy preserving platform for decentralized appilaction. White paper, 2019

[45] Kalodner H, Goldfeder S, Chen X Q, et al. Felten. Arbitrum: Scalable, private smart contracts//Proceedings of the 27th USENIX Conference on Security Symposium (SEC). Baltimore, USA, 2018:1353–1370.

[46] Dolev S, Wang Z Y. SodsMPC: FSM based anonymous and private quantum-safe smart contracts//Proceedings of the 19th IEEE International Symposium on Network Computing and Applications (NCA). Cambridge, USA, 2020: 1-10

[47] Yan Y, Wei C Z, Guo XP, et al. Confidentiality support over financial grade consortium blockchain//Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020. Portland, USA, 2020: 2227-2240

[48] Rodler M, Li W T, Karame G, er al:EVMPatch: Timely and automated patching of Ethereum smart contracts//Proceedings of the 30th USENIX Conference on Security Symposium. Vancouver, Canada, 2021.

[49] So S, Lee M, Park J, et al. VERISMART: A highly precise safety verifier for Ethereum smart contracts//Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP). San Francisco, USA, 2020: 1678-1694

[50] Luu L, Chu D H, Olickel H, et al. Making smart contracts smarter//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna, Austria, 2016: 254-269

[51] Atzei N, Bartoletti M, Cimoli T. A survey of attacks on Ethereum smart contracts (sok)//Proceeding of the Principles of Security and Trust (POST). Uppsala, Sweden, 2017: 164 – 186

[52] Rodler M, Li W, Karame G O, et al. Sereum: Protecting existing smart contracts against re-entrancy attacks//Proceedings of the 26th Annual Network and Distributed System Security Symposium(NDSS). San Diego, USA, 2019:1-15

[53] Chen H, Pendleton M, Njilla L, et al.A Survey on Ethereum systems security: Vulnerabilities, Attacks, and Defenses. ACM Computing Surveys, 2020, 53(3): 1-43

[54] Watanabe H, Shigeru F, Atsushi N, et al. Blockchain contract securing a blockchain applied to smart contracts//Proceedings of the IEEE International Conference on Consumer Electronics (ICCE). Las Vegas, USA, 2016: 467-468

[55] Bekrar S, Bekrar C, Groz R, et al. Finding software vulnerabilities by

smart fuzzing//Proceedings of the 4th IEEE International Conference on Software Testing, Verification and Validation (ICST). Berlin, Germany, 2011: 427 - 430

[56] Azaria A, Ekblaw A, Vieira T, et al. MedRec: Using blockchain for medical data access and permission management//Proceedings of the 2nd International Conference on Open and Big Data (OBD), Vienna, Austria, 2016:25-30

[57] Feng Q, He D B, Sherali Z, et al. A survey on privacy protection in blockchain system. Network and Computer Applications, 2019, 126(1): 45-58

[58] Sasson E B, Chiesa A, Garman C, et al. Zerocash: Decentralized anonymous payments from Bitcoin//Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP). Berkeley, USA, 2014:459-474

[59] Meiklejohn S, Pomarole M, Jordan G, et al A fistful of bitcoins: Characterizing payments among men with no names. Communications of the ACM, 2013, 59(4): 86-93

[60] Ron D and Shamir A. Quantitative analysis of the full Bitcoin transaction graph//Proceedings of the 17th International Conference of Financial Cryptography and Data Security. Okinawa, Japan, 2013: 6-24

[61] Fleder M, Kester M, Pillai S. Bitcoin transaction graph analysis. arXiv:1502.01657, 2015

[62] Peng L , Feng W , Yan Z , Li Y F, Zhou X K, Shimizu S. Privacy preservation in permissionless blockchain: A survey. Digital Communications and Networks, 2020

[63] Schumann J M, Loveland D. Automated theorem proving in software engineering. Berlin: Springer –Verlag, 2001

[64] Abdellatif T, Brousmiche K L. Formal verification of smart contracts based on users and blockchain behaviors models//Proceedings of the 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS). Paris, France, 2018: 1–5

[65] Hirai Y. Defining the Ethereum virtual machine for interactive theorem provers//Proceedings of the Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA. Sliema, Malta, 2017: 520–535

[66] Grishchenko I, Maffei M, Schneidewind C. A semantic framework for the security analysis of Ethereum smart contracts//Proceedings of the 7th International Conference on Principles of Security and Trust (PST). Thessaloniki, Greece, 2018: 243-269

[67] Hildenbrandt E, Saxena M, Rodrigues N, et al. KEVM: A complete formal semantics of the Ethereum virtual machine//Proceedings of the 31st IEEE Computer Security Foundations Symposium (CSF). Oxford, UK, 2018: 204-217

[68] Rou G, Şerbanut T F. An overview of the K semantic framework. Journal of Logic and Algebraic Programming, 2010, 79(6): 397-434

[69] Li Y K, Wang L Z. Specifying and detecting behavioral changes in source code using abstract syntax tree differencing//Proceedings of the International Standard Conference of Trustworthy Computing and Services (ISCTCS), Beijing, China, 2021:466-473

[70] Tann W J, Han X J, Gupta S S, et al. Towards safer smart contracts: a sequence learning approach to Detecting Vulnerabilities. arXiv:1811.06632, 2018

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

[72] Wu S Z, Guo Tao，Dong G W. Software vulnerability analysis technology. Beijing: Science Press, 2014 (in Chinese)
(吴世忠，郭涛，董国伟. 软件缺陷分析技术. 北京: 科学出版社，2014)

[73] Liang H L, Pei X X, Jia X D, et al. Fuzzing: State of the art. IEEE Transactions on Reliability, 2018, 67(3):1199-1218

[74] Blum M, Feldman P, Micali S. Non-interactive Zero-knowledge and its applications (Extended Abstract)//Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC). Chicago, USA, 1988. 103-112

[75] Bitansky N, Canetti R, Chiesa A, et al. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again//Proceedings of the Innovations in Theoretical Computer Science 2012 (ITCS). Cambridge, USA,2012: 326-349

[76] Sah C P, Jha K, Nepal S. Zero-knowledge proofs technique using integer factorization for analyzing robustness in cryptography//Proceedings of the 3rd International Conference on Computing for Sustainable Global Development (INDIACom). New Delhi, India, 2016: 638-642

[77] Shaw M. Writing good software engineering research paper//Proceedings of the 25th International Conference on Software Engineering (ICSE). Portland, USA, 2003: 726-737.

[78] Bosu A, Iqbal A, Shahriyar R, et al. Understanding the motivations, challenges and needs of Blockchain software developers: a survey. Empirical Software Engineering, 2019, 24(4): 2636-2673

[79] Praitheeshan P, Pan L, Yu J, et al. Security analysis methods on Ethereum smart contract vulnerabilities: a survey. arXiv:1908.08605, 2019

[80] Bai X, Cheng Z, Duan Z, et al. Formal modeling and verification of smart contracts//Proceedings of the 7th International Conference on Software and Computer Applications (ICSCA). Kuantan, Malaysia, 2018: 322–326

[81] Abraham M, Mohan K. Decision framework and detailed Analysis on privacy preserving smart contract frameworks for enterprise blockchain applications//Proceedings of the 2020 International Conference on Omni-layer Intelligent Systems (COINS). Barcelona, Spain, 2020: 1-6

[82] LuD , YurekT, KulshreshthaS, et al. Honeybadgermpc and asynchromix: Practical asynchronous MPC and its application to anonymous communication//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS). London, UK, 2019:887-903

[83] Alharby M, van Moorsel A. Blockchain-based smart contracts: A systematic mapping study. arXiv:710.06372, 2017

[84] Parizi R M, Dehghantanha A, Choo K K R, et al. Empirical vulnerability analysis of automated smart contracts security testing on blockchains//Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering (CASCON). Markham, Canada, 2018: 103-113

HU Tian-Yuan, Ph.D. candidate. Her research interests include software engineering, blockchain security.

Li Ze-Cheng, Ph.D. candidate. His research interest lies in the blockchain security, blockchain scalability, and network security.

Li Bi-Xin, Ph.D., professor, Ph.D. supervisor. His research interests include software engineering, blockchain security.

Bao Qi-Hao, Ph.D. candidate. His research interests include software engineering, blockchain security.

Background

This paper is a system mapping study belonging to the field of blockchain security,which surveys recent works on smart contract security. The combination of smart contracts and blockchain is considered a milestone upgrade in the blockchain world. As an automatically executable digital protocol, smart contracts add programmable properties to the blockchain. The security and privacy protection features of the blockchain are important factors in promoting the long-term development of the blockchain. However, due to the immaturity of this emerging technology, frequent smart contract attacks seriously threaten the security of the blockchain ecosystem. There is an urgent need for technical support related to smart contracts security.

There are plentiful studies focusing on smart contracts security. However, the existing researches lack systematization and clarity, resulting in a lack of strong guidance for solving smart contract security problems. In order to provide references for further solutions to smart contract security issues, this paper concentrates on several high-quality studies related to smart contracts security.

In this paper, the current state of researches are summarized and analyzed by setting several research questions related to smart contract security. Firstly, the existing security problems and security challenges of smart contract security are analyzed according to the characteristics of smart contract in blockchain system. Smart contract security is divided into contract security and privacy security. From the perspective of the lifecycle of smart contract, the contract security is guaranteed in different stages of design, implement, testing, deployment and maintenance. Moreover, the privacy security involved code privacy and data privacy. Furthermore, the existing security assurance techniques are collected aiming at the security issues and challenges of smart contract security. Effectiveness and limitations of different techniques are analyzed by combining the verification of different researches. A multi-dimensional analysis of different techniques assuring smart contract security was conducted. Strengths and weaknesses of security assurance techniques for contract security and privacy security are concluded. Finally, future research directions mainly focused on smart contract detection and privacy protection are suggested.