基于区块链智能合约的数字身份可验证凭证零知识认证和管理架构

宋智明[1,2,3], 余益民[1,2], 王贵文[1], 陈韬伟[1]

[1]信息学院 云南财经大学 昆明 中国 650221
[2]智能应用研究院 云南财经大学 昆明 中国 650221
[3]云南省智慧城市网络空间安全重点实验室 玉溪师范学院 玉溪 中国 653100

摘要 数字身份认证和管理系统是计算机和互联网应用的安全基础设施, 当前的数字身份认证和管理系统暴露出了许多的问题, 例如, 数字身份的中心化存储和管理、数字身份的自主权特性较差、数字身份的隐私无保障、数字身份的认证过程不公开透明等。伴随着区块链技术的发展, 其去中心、防篡改、可追溯和安全透明等特点受到了越来越多的关注, 各种基于区块链的数字身份认证和管理架构相继出现, 然而这些架构仍然不同程度存在上述问题。对此, 本文将区块链智能合约技术、非交互式零知识证明技术、可验证凭证数字身份等结合, 提出了一个基于区块链智能合约的数字身份可验证凭证零知识认证和管理架构, 并详细描述了架构角色和协议流程等, 其中, 在协议流程中, 服务提供商根据服务授权要求构造零知识证明约束条件验证程序, 用户基于该程序生成零知识证明, 该零知识证明可以在不透露身份和私钥的情况下向服务提供商证明用户是身份的所有者, 且身份满足服务授权要求, 而服务提供商对零知识证明的认证和管理等都是在智能合约中完成的, 实现了认证和授权的公开透明和安全。此外, 本文设计了该架构的原型系统, 并基于原型系统, 评估了架构中智能合约的成本和时间开销, 同时讨论了基于智能合约进行零知识证明认证和管理的安全性和必要性等, 最后还对整个架构的有效性和安全性等进行了评估和比较。上述结果表明, 本文所提出的架构能较好的实现数字身份认证和管理的去中心化、隐私安全、认证和授权公开透明等, 可为相关系统的设计提供有价值的技术参考。

关键词 数字身份认证和管理; 区块链; 智能合约; 零知识身份认证; 可验证凭证
中图法分类号 TP309.2 DOI 号 10.19363/J.cnki.cn10-1380/tn.2023.01.05

Zero-knowledge Authentication and Management Architecture of Verifiable Certificate of Digital Identity Based on Smart Contracts of Blockchain

SONG Zhiming[1, 2, 3], YU Yimin[1, 2], WANG Guiwen[1], CHEN Taowei[1]

[1] School of Information, Yunnan University of Finance and Economics, Kunming 650221, China
[2] Institute of Intelligent Application, Yunnan University of Finance and Economics, Kunming 650221, China
[3]Yunnan Key Laboratory of Smart City and Cyberspace Security, Yuxi Normal University, Yuxi 653100, China

Abstract Digital identity authentication and management system (DIAMS) is the security infrastructure of computer and internet applications. However, current DIAMSs have exposed many problems such as centralized storage and management of digital identities, poor self-sovereignty management on user's own identity, poor privacy protection, non-transparent authentication and authorization and so on. Currently, along with the development of blockchain technology, its features such as decentralization, tamper-proof, traceability, security, and transparency and so on are receiving increasing attentions, resulting in that various DIAMSs based on blockchain are proposed. However, the problems mentioned above still exist in the DIAMSs based on blockchain. Therefore, this paper combines smart contract of blockchain, non-interactive zero knowledge proof with the verifiable certificate of digital identity, and proposes an architecture of DIAMS based on blockchain. Firstly, in the proposed architecture, entity roles and protocol flows are described in detail. Secondly, in the protocol flows, the verification program of constraint conditions of zero-knowledge proof (VPCCZKP) is constructed by service provider based on service authorization requirements, and user takes advantage of the VPCCZKP to

通讯作者: 宋智明, 博士, 副教授, Email: 339255245@qq.com。
本课题得到国家自然科学基金项目(No. 71964037), 中央引导地方科技发展专项资金项目(No. 202007AD110001), 云南省教育厅科学研究基金(No. 2022J0473), 云南省基础研究计划青年项目(No. 202101AU070132), 云南省刑事科学技术重点实验室(No. YJXK005), 云南省智慧城市网络空间安全重点实验室(No. 202105AG070010-SG-07)的资助。
收稿日期: 2021-09-30; 修改日期: 2022-02-09; 定稿日期: 2022-11-04

generate the zero-knowledge proof which can prove that he/she is the holder of the legal identity which meets service authorization requirements without disclosing his/her private key and legal identity. In the meantime, the generated zero-knowledge proof is provided to service provider, and service provider uses the on-chain smart contracts to authenticate and manage the zero-knowledge proof in order to implement the transparency and security of authentication and authorization. On the other hand, the prototype system of proposed architecture is designed and based on the prototype system, the cost and time overhead of smart contracts of the proposed architecture is assessed. In the meantime, the security and necessity of on-chain authentication and authorization of zero-knowledge proof based on smart contracts is discussed. Furthermore, evaluations and comparisons of effectiveness and safety of the proposed architecture are conducted at the end of this paper. Finally, the results from the prototype system indicate that the proposed architecture can well realize the decentralization, privacy security, openness and transparency of digital identity authentication and management and provide value technology references for designing corresponding DIAMSs.

Key words digital identity authentication and management system; blockchain; smart contract; zero-knowledge identity authentication; verifiable certificate

1 引言

互联网技术的发展方便了人们的工作、学习和生活, 使得人们可以便捷地利用各种数字身份在不同的应用服务商获得服务、交换数据和信息等。然而, 人们在享受各种互联网应用带来的便利的同时, 人们的数字身份所关联的各种特征信息、行为信息也被暴露和存储到了各种应用服务商的数字身份认证和管理系统中, 并由此引发诸如数据非法使用和泄露、身份伪造、敲诈和勒索等网络安全问题。例如, 2018 年, 英国“剑桥分析”公司非法获取 5000 多万 Facebook 用户的身份信息, 并给他们推送政治广告, 干预美国选举。2018 年 4 月, 美团和饿了么等外卖平台用户信息被卖家、运营公司及外卖骑手泄露, 并非法出售这些信息, 其中的信息包括了用户用餐习惯, 用餐地点等隐私数据。2018 年 8 月, 中国华住集团旗下的 3000 家连锁酒店的信息被泄露, 造成超过一亿条用户个人数据和入住记录被黑客获取, 并在网上公开售卖和实施敲诈[1]。

显然, 数字身份具有“虚拟基因”的特征, 保障数字身份的安全对互联网和网络空间安全而言意义重大。由于数字身份与真实身份的映射关系、数字身份的存储与管理、数字身份的认证与授权等都是由数字身份认证和管理系统负责, 因此, 数字身份认证和管理系统的架构是构建安全数字身份体系的前提。

然而, 当前互联网应用中的数字身份认证和管理系统仍然较多的采用“用户名+密码”组合、公钥基础设施 PKI 的 CA 数字证书、动态口令技术、电子身份标识 eID[2]等以应用服务商或者权威机构为核心的中心化架构, 这种架构存在诸如中心机构的设备宕机、中心机构的不诚实与作恶、用户对自主身份的管理权限差、数字身份的认证过程不公开透明, 授权结果不可信, 以及隐私无保障等问题。尽管, 诸如 OpenID、UMA、OAuth2.0 等统一联邦授权架构[3]被相继提出, 但是, 它们仍然存在上述问题。

近年来, 随着区块链技术的发展, 其去中心化、公开透明、可追溯、不可篡改等特性受到了各方的关注, 许多学者也纷纷将区块链技术应用于数字身份认证和管理系统架构的设计中, 并提出了不同的设计方案, 具体可归纳为 3 种主要架构。

首先是基于区块链分布式账本进行数字身份信息及其哈希摘要存储的架构。由于区块链分布式账本具备公开透明、不可篡改、多边维护的特性, 可以克服诸如公钥基础设施 PKI 的 CA 数字证书等架构中的恶意证书颁发、证书撤销列表管理不方便、异构 CA 证书跨域认证困难等问题, 因此, 学者们提出利用分布式账本来存储和管理 CA 数字证书、数字证书和身份信息的哈希摘要, 并基于这些存储信息进行可信身份认证和授权服务等(如图 1 所示)。例如, 麻省理工大学学者提出的 Certcoin 是首个利用区块链分布式账本来对 PKI 数字证书进行管理的去中心化 PKI 身份认证的系统[4], 其以公开的方式将用户身份与证书公钥相关联, 通过区块链交易的形式发布用户及其公钥来实现证书的注册、更新和撤销,通过区块链不可篡改的属性来保障 PKI 的正常运行, 任何用户都可以查询证书签发过程,解决传统 PKI 系统所面临的证书透明度及 CA 单点故障的问题。Wang 等人[5]利用区块链分布式账本交易来记录证书的相关信息, 以使得数字证书发布、撤销、使用过程公开透明和可监管。区块链供应链公司 Everledger 将钻石高、宽、厚度、色泽等相关信息进行哈希摘要, 并存储到不可篡改的区块链分布式账本中, 以此作为钻石鉴定的身份证书, 达到钻石溯源和认证的目的[6]。Wenton 等人[7]提出了一个名为 BlockCAM 的数字身份系统, 该系统利用区块链的分布式账本

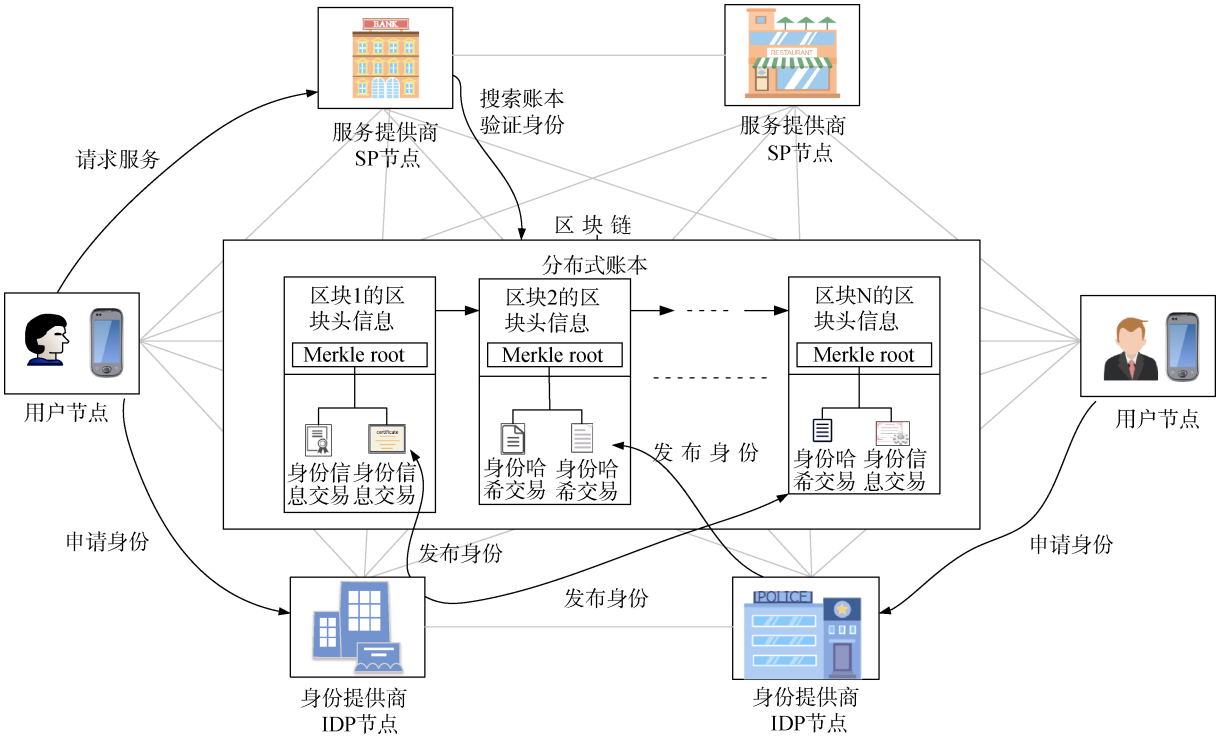


图 1　分布式账本存储数字身份信息及其哈希摘的架构

Figure 1　The distributed ledger architecture for storing digital identities' information and their hash digests

存储跨域实体的数字证书的哈希摘要，在要进行跨域身份认证时，通过将实体所持证书的哈希摘要与区块链中存储的哈希摘要进行比对，实现跨域身份认证。

其次是利用区块链可编程智能合约进行数字身份的声明、发布、认证和授权的架构。由于智能合约具有可编程、不可篡改、状态同步、公开透明的特性，用户与身份提供商(Identity provider, IDP)之间很容易构建起分布式身份声明与身份链上核验与发布的数字身份体系(如图 2 所示)，而且由于智能合约的不可篡改特性，合约地址可以很方便地用来作为统一数字身份标识符，便于数字身份的管理，因此，

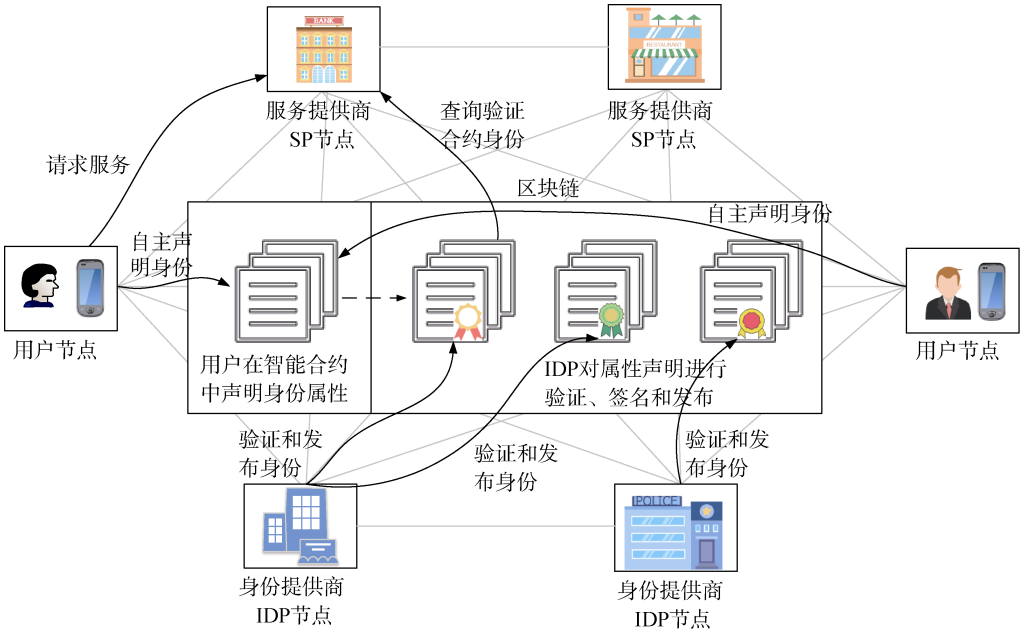


图 2　基于区块链智能合约的数字身份声明与发布的架构

Figure 2　The digital identity architecture of declaration and publication based on smart contracts of blockchain

学者们构建了多种基于区块链的智能合约数字身份系统。例如, 最典型的基于区块链智能合约的身份认证系统就是 UPort, 其利用智能合约的地址作为用户的唯一身份标识符, 并将身份属性的相关索引信息存储在区块链智能合约中(身份属性的真实值存储在链下)[8]。文献[9]提出了一个基于以太坊智能合约的数字身份认证和管理系统, 该系统类似 UPort, 使用以太坊地址作为用户的唯一身份标识符, 身份合法性的验证是通过验证身份上保存的数字签名实现。文献[10]提出了一个名为 EverSSDI 的基于以太坊区块链智能合约的数字身份认证和管理框架, 该框架基于区块链智能合约技术, 能够为用户提供唯一身份标识, 并可实现细粒度身份管理、身份丢失可恢等功能。文献[11]利用区块链智能合约开发了一个名为智能合约公钥基础设施 SCPKI 的身份认证系统, 该系统使用实体、属性、签名和撤销四个智能合约管理用户的身份, 可实现细粒度身份管理和可信授权。

最后是基于区块链可验证凭证(Verifiable certificate, VC)的数字身份架构。由于区块链分布式账本和智能合约的公开透明特性, 前两种架构存在身份隐私泄露风险, 尽管其中部分架构在区块链上存储的只是数字身份的哈希摘要或者加密信息[6,7,10], 但是认证授权身份时仍然需要提交身份的明文信息(明文求哈希后与链上哈希进行对比或者解密明文后查看信息是否正确)。此外, 由于分布式账本和智能合约存储的数据不可篡改, 即只能增加数据, 无法删除数据, 因此, 随着用户量的增加, 系统的存储容量和经济投入是需要考虑的问题。因此, 一些学者提出将数字身份以可验证凭证的形式颁发给用户, 让用户自主管理身份数据, 同时不利用区块链直接存储数字身份相关信息, 仅仅利用其锚定数字身份标识符与真实身份的映射关系, 存储身份的状态信息(激活和撤销)等, 从而增加身份的自主控制权, 并且在不损失身份可信性的同时, 提高区块链的效率和可伸缩性(如图 3 所示)。例如, 著名的万维网联盟(World wide web consortium, W3C)提出了基于可验证凭证的数字身份架构[12], 并逐渐被广大去中心化自主权身份方案采纳[13,14]。文献[15]提出了一个面向区块链可验证凭证数字身份认证和管理架构 TBDID, 以探索和实现自主权数字身份认证和管理。创立于美国旧金山的非盈利组织 Kiva 利用区块链构建了一个称为 Kiva 身份协议的基于属性凭证(可验证凭证 VC)的身份认证系统[16], 并将该系统应用于贷款信用记录的查询中, 以解决非洲塞拉利昂等发展中国家缺乏国家信用机构导致的公民信贷困难的问题。我国互联网巨头百度在认识到基于可验证凭证的去中心数字身份的价值后, 也基于 W3C 的架构, 设计了一个名为 CloudDID 的去中心化可验证凭证数字身份认证和管理架构[17]。

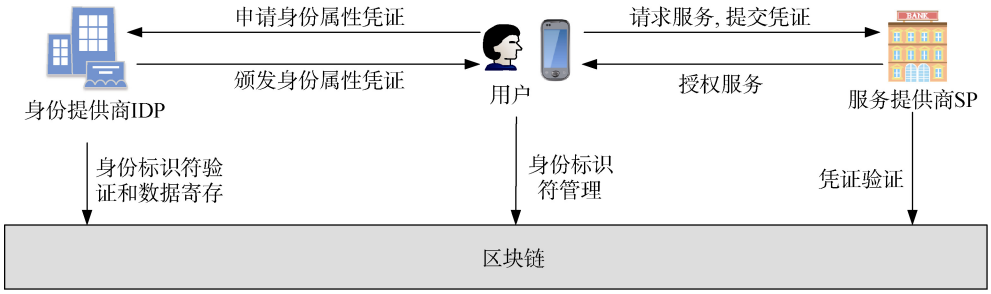


图 3 基于区块链的可验证凭证数字身份架构

Figure 3 The digital identity architecture of verifiable certificate based on blockchain

虽然, 基于区块链的可验证凭证数字身份架构具备诸多优点, 但是可验证凭证上通常记录着用户身份的明文信息, 因此, 在利用可验证凭证进行身份认证与授权时, 仍然存在隐私泄露风险, 即使部分研究提出可以利用选择性披露来减小隐私泄露的程度[17], 或者使用零知识证明来隐藏身份属性[18], 但是这些研究并没有给出一个具体的隐私保护的实现架构, 更没有共享架构的实现代码。此外, 虽然在 Li Q 和 Xue Z 的研究中[19], 将区块链与零知识证明算法结合, 给出了一个具备零知识证明隐私保护的可验证凭证系统框架, 但是该框架仅仅利用零知识证明验证凭证中的身份属性是否满足服务提供商的要求, 并没有在合约中较好验证凭证的颁发机构(身份提供商 IDP)是否是合法的, 同时也没有给出一个有效的方法来验证提交零知识证明的用户是否是可验证凭证的持有者, 即没有同时解决隐私和安全问题。

表 1 总结了上述 3 种主要的区块链数字身份认证和管理架构及相关架构的具体系统, 同时给出了

它们的特征和缺点。显然, 从表 1 中可以更清楚的看到可验证凭证架构在自主权和安全方面的优势, 同时可以看到, 上述 3 种架构的具体系统几乎都在隐私保护、可信验证方面存在问题。因此, 本文将把区块链智能合约与零知识证明结合, 给出了一个完整的具有隐私保护、可信验证的基于区块链的可验证凭证数字身份架构, 并详细介绍架构的实现, 同时共享其中的代码。

表 1　3 种基于区块链的数字身份认证和管理架构及系统的特征和缺点

Table 1　Characteristics and disadvantages of three blockchain-based digital identity management architectures and systems

架构	架构特征	不同架构系统	系统主要特征	系统主要缺点
分布式账本存储数字身份信息及其哈希摘的架构	数字身份信息和身份的哈希摘要通过去中心化、不可篡改、多边维护的分布式账本存储和管理, 避免恶意身份证书的颁发、有利于证书的追溯和跨域验证	Certcoin[4]	利用分布式账本存储用户身份、公钥和它们的关系	用户身份信息和身份所有权关系公开在分布式账本中, 隐私无保障
		Wang 等人提出的系统[5]	利用分布式账本存储证书信息和证书撤销状态, 并对私钥进行分布式管理	用户身份信息和身份所有权关系公开在分布式账本中, 隐私无保障
		Everledger 公司的钻石身份鉴定系统[6]	利用分布式账本公开存储钻石特征信息的哈希摘要	分布式账本存储钻石特征的哈希摘要, 验证时, 需向验证者提交钻石特征信息, 验证者求其哈希摘要后与链上哈希比较, 隐私无保障
		BlockCAM[7]	利用分布式账本存储跨域数字证书的哈希摘要	分布式账本存储证书的哈希摘要, 跨域验证时, 向验证者提交证书明文, 验证者求其哈希摘要后与链上哈希比较, 隐私无保障
基于区块链智能合约的数字身份声明与发布的架构	数字身份通过可编程、不可篡改、状态同步、公开透明的智能合约构建、存储和管理, 合约地址可用作身份的全局唯一标识符	UPort[8]	利用合约构建数字身份, 合约地址作全局唯一身份标识符	智能合约只用于管理和控制身份, 系统不具备基于合约的身份可信验证
		文献[9]的系统	利用合约构建数字身份, 合约地址为全局唯一身份标识符, 具备身份验证功能	身份验证由验签完成, 但验签过程没用可信透明智能合约完成, 且身份明文直接由合约存储, 隐私无保障
		EverSSDI[10]	利用合约构建用户数字身份, 并利用分层确定性密钥技术加密身份的细粒度属性, 保证合约中的细粒度身份的隐私	合约中的身份属性被加密保护, 但身份验证时, 仍然需解密身份属性, 并提供给验证者, 隐私保障仍存在问题
		SCPKI[11]	利用智能合约构建用户数字身份, 身份拥有者之间可以相互数字签名, 构建可信网络	身份的明文信息直接记录在合约中, 隐私无保障, 且验签过程没有用可信透明智能合约完成
基于区块链的可验证凭证数字身份架构	数字身份以可验证凭证的方式颁发给用户, 用户自主管理身份, 区块链(分布式账本和智能合约)只用于管理身份而不存储身份信息, 用户自主权特性和区块链的伸缩性高	TBDID[15]	利用可验证凭证构建自主身份	可验证凭证中的身份信息以明文形式存在, 验证时隐私无保障, 且验证过程不公开透明, 不可信
		Kiva[16]	利用可验证凭证构建自主身份, 实现身份标识的隐私保护	没有具体描述实现隐私保护的架构, 且验证过程不公开透明, 不可信
		CloudDID[17]	利用可验证凭证构建自主身份, 提出可利用选择性披露保障隐私	没有具体实现选择性披露的隐私的方案, 身份隐私仍然无保障
		文献[18]的系统	利用可验证凭证构建自主数字身份, 提出用零知识证明保障隐私	没有给出具体的零知识保障身份隐私的方案, 身份隐私仍然无保障
		文献[19]的系统	利用可验证凭证构建自主身份, 用零知识证明保障隐私	在保障身份隐私时没证明用户是凭证的持有者, 没同时兼顾隐私和安全

本文的贡献如下:

1) 利用非交互式零知识证明(zero-knowledge succint non-interactive arguments of knowledge, zkSNARKs)改善了基于区块链的可验证凭证数字身份架构的身份隐私特性, 改善的架构中, 零知识证明的约束条件验证程序由服务提供商(Service pro-

vider, SP)根据服务的身份属性需求构造, 用户要获得服务, 仅需要根据服务提供商 SP 构造的零知识证明约束条件验证程序, 利用用户的属性凭证 VC 生成零知识证明, 并向服务提供商 SP 提交证明, 以验证证明中的属性满足服务的身份属性需求, 同时证明用户是生成证明的 VC 的持有者, 从而实现了隐私性和安全性的双重保障。

2) 在证明所提交的证明的有效性以及用户是可验证凭证 VC 的持有者的同时, 本文架构也验证了凭证颁发机构(身份提供商 IDP)的有效性, 而这些证明都是通过链上代码智能合约实现的, 即零知识证明的有效性验证、凭证中的身份提供商 IDP 的验证、凭证的状态管理等都将全部由公开透明、安全可信的区块链智能合约实现, 保证了认证的透明可信。进一步地, 上述零知识证明约束条件验证程序和合约代码都被公开分享在了 GitHub 中(https://github.com/20031225/zero-knowledge-VC), 以为需要进行相关研究的读者提供技术参考。

3) 将本文的身份属性可验证凭证定义为仅包含单一身份属性的格式, 以保障隐私最小化披露。同时给出了从凭证申请到完成验证的完整系统框架流程。

2 预备知识

2.1 可验证凭证 VC

凭证广泛应用于人们的日常生活中, 其是个人真实身份属性所生成的用于在某个应用领域证明个人身份属性特征的证明, 例如驾驶员的执照、学生的学位证书, 护照等都是凭证。而可验证凭证是物理凭证的数字化形式, 其是机器可读的, 通过数字签名等算法保证其中的属性真实性的数字化证明。如图 3 所示, 可验证凭证系统通常包含三种角色, 身份提供商 IDP、用户、服务提供商 SP, 其中, 身份提供商是可验证凭证的颁发者, 其在核验用户真实身份后, 为用户颁发包含用户属性、用户全局唯一身份标识符、身份提供商全局唯一身份标识符和数字签名的可验证凭证, 用户是可验证凭证的拥有者, 其将把可验证凭证提交给服务提供商 SP, 并由服务提供商 SP 证明其是凭证中的全局唯一身份标识符的拥有者后获得服务, 服务提供商是服务提供者, 其在验证凭证中的身份提供商 IDP 的数字签有效性和用户对凭证的持有权后为用户提供服务。W3C 提出的基于可验证凭证的数字身份架构中定义了一些可验证凭证的格式规范[12], 在本文中, 为了保证身份隐私最小化披露, 在凭证中仅包含用户单一身份属性, 具体格式如表 2 所示, 其中, User DID=keccak256(PK_{user})是用户公钥 PK_{user} 求哈希摘要后的结果, IDP DID 是身份提供商 IDP 的以太坊区块链账号地址, IDP Signature 是 IDP 的以太坊区块链账号地址的私钥对凭证的数字签名。

表 2 本文可验证凭证的内容和描述

Table 2 The content and description of VC of this paper

凭证项	描述
Attribute	用户的单一身份属性的值
User DID	用户的身份标识符
IDP DID	身份供商 IDP 的身份标识符
IDP Signature	身份提供商 IDP 对凭证的数字签名

2.2 区块链智能合约

智能合约是区块链的参与者在达成共识后部署在区块链上的, 可根据外部触发条件自动执行的链上代码, 其调用者需要向区块链发送交易, 并在交易达成共识后才能实现调用合约的链上操作, 因此, 其可以防止单边篡改合约规则, 具有公开透明、安全可信的特点。通过智能合约, 用户可以在区块链上实现一些可信的复杂操作, 因此, 智能合约又被看作是区块链 2.0 时代的标志。

当前, 支持智能合约的区块链平台主要包括以太坊 Ethereum[20]和超级账本 HyperLedger[21]等。由于本文接下来要介绍的零知识证明工具支持以太坊零知识证明验证智能合约的生成, 因此, 本文将采用以太坊区块链平台完成所有与用户数字身份认证相关的智能合约设计, 从而保证认证过程的公开可信以及可追溯。

2.3 零知识证明

零知识证明是一种加密技术, 通过该技术, 证明者能够使得验证者相信, 其知道某种声明的内容是正确的, 而不透露任何与声明相关的有用信息, 例如, 证明者可以在不透露加密密钥的情况下向验证者证明其知道被加密数据的明文内容[22]。又如, 证明者可以在不透露哈希算法输入结果的情况下证明其知道输入哈希算法的哈希原像[23]。再如, 可以证明两个离散对数相等而不泄露离散对数具体值, 从而实现不可否认签名[24]。

零知识证明算法包括交互式零知识证明[25]和非交互式零知识证明[26], 相比于交互式零知识证明, 非交互式零知识证明在验证过程中不需要进行多次交互通信(至多有一次认证交互), 可以避免串谋攻击, 安全性更高, 特别是在区块链应用中, 区块链节点

之间可以采用非交互式零知识证明减少隐私保护过程中因为多次交互通信引起的系统开销, 能够在不影响系统吞吐量的同时改善系统的隐私, 例如, 著名的零币区块链系统 Zerocash[27]就是使用简洁的非交互式零知识证明算法 zkSNARKs[28]实现了交易信息的隐藏。

zkSNARKs 是一种经过优化的零知识证明方案, 其减少了证明的大小与验证时间,实现了计算和通讯开销的优化。当前, 可实现 zkSNARKs 的非交互式零知识证明的算法主要包括 PGHR13[29]、GM17[30]、G16[31]等, 相比于其他算法, G16 算法拥有最小的验证时间和简洁的零知识证明[32], 很适合在区块链上进行高效的零知识证明应用, 例如, 相比于先前的 Zerocash 系统采用的是 PGHR13 算法, 最新的具有隐私保护的区块链系统, 如 BlockMaze[32]和 Filecoin[33]等几乎都采用性能更好的 G16 算法。因此, 本文在使用 zkSNARKs 改善所提出的可验证凭证系统框架的隐私时, 也将采用 G16 算法。进一步地, 本文将借助 zkSNARKs 算法的集成化工具集 ZoKrates[34]来构建本文的基于 G16 的零知识证明问题。

Zokrates 是一个包含特定域语言(domain-specific language, DSL)处理器、编译器(包括语言解析器和进行算数电路转换的扁平器)、zk-SNARKs 零知识证明生成器(包含将算数电路转换为秩为 1 的约束系统 R1CS 和二次算术程序 QAP、再进一步转换为公共参考字串 CRS 和零知识证明等的处理器)、以太坊零知识证明验证智能合约生成器(将最终证明问题转换为椭圆曲线双线性配对问题, 并利用以太坊预编译的椭圆曲线库 EIP196 和 EIP197 实现双线性配对证明验证的转换器)的零知识证明工具集。利用 Zokrates 可以将 zkSNARKs 的具体算法细节屏蔽, 而只关注于要进行隐私保护的具体问题, 从而实现以黑盒方式在链下描述具体的零知识证明问题, 并最终生成可在链上执行的零知识证明以太坊区块链智能合约。

对于 G16 算法, 要完成一次非交互零知识证明, 通常需要完成构建 Setup、证明生成 Prove 和验证 Verify 三个操作步骤, 其中 Setup 操作主要基于要证明的问题生成公共参考字串 CRS, Prove 操作主要是证明者基于公共参考字串 CRS 生成零知识证明, Verify 操作主要是验证者基于公共参考字串 CRS 验证证明者生成的证明是否有效。在本文所述的架构中, 服务提供商 SP 通常需要基于不同的身份属性进行属性认证和服务授权, 而用户需要基于不同的属性要求生成零知识证明, 因此, 由服务提供商 SP 描述要证明的零知识证明约束条件验证程序, 并完成 Setup 和 Verify 两个操作步骤(作为验证者), 由用户完成 Prove 操作(作为证明者)。图 4 展示了利用 Zokrates 工具集完成上述三个步骤的过程, 其中虚线部分主要是验证者的操作, 实线部分主要是证明者的操作, 下面详细描述该过程:

1) Setup (F) →(proving key, verification key)。验证者描述要用零知识证明实现的约束条件验证程序 Z(利用 Zokrates 的 DLS 语言描述), 并通过编译生成扁平代码 F(算数电路), 并转换为 R1CS 和 QAP, 最后通过构建 Setup 生成公共参考字串 CRS, 并输出 CRS 的证明密钥 proving key 和验证密钥 verification key。证明者可以在需要时获得 Z 和 proving key, 而验证者自己保留 verification key, 以用于在最终生成的验证智能合约中完成基于椭圆曲线配对的零知识证明, 其中, verification key 由 5 个变量组成, verification key={ V_α , V_β , V_γ , V_δ , V_{γ_abc} }, 式中的 $V_\alpha=\alpha G$ 、 $V_\beta=\beta H$ 、 $V_\gamma=\gamma H$ 、 $V_\delta=\delta H$, $V_{\gamma\_\mathrm{abc}}=\{\frac{1}{\gamma}(\beta a_i(x)+\alpha b_i(x)+c_i(x))G\}_{i=0}^{n}$, 且上式中包含了验证者从域 F 中选取的随机变量: α、β、γ、δ、x, 此外, G 为群 G_1 的生成元, H 为群 G_2 的生成元, $\{a_i(x), b_i(x), c_i(x)\}_{i=0}^{n}$ 是零知识证明约束条件验证程序 Z 通过算数电路变换、R1CS 变换、二次算术程序 QAP 变换等操作所生成的与零知识证明约束条件相关的多项式。

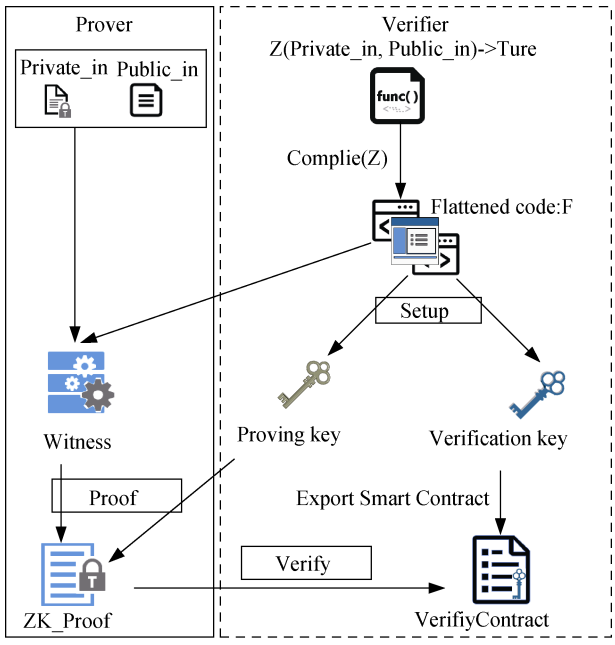


图 4 利用 ZoKrates 实现零知识证明的流程

Figure 4 Process of implementing zero knowledge proof using ZoKrates

2) Prove (Z, proving key, Privacy input, Public input)→zk_proof。证明者获得 Z 和 proving key 后 (proving key 作为 CRS 的组成部分, 其中包含变量α、β、γ、δ、x, $\{x^i\}_{i=0}^{k-1}$, $\{\frac{1}{\gamma}(\beta a_i(x)+\alpha b_i(x)+c_i(x))\}_{i=0}^{n}$, $\{\frac{1}{\delta}(\beta a_i(x)+\alpha b_i(x)+c_i(x))\}_{i=n+1}^{m}$, $\{\frac{1}{\delta}(x^i t(x))\}_{i=0}^{k-2}$ 和相关多项式等), 将要证明的问题转换为 Z 的隐私输入 Privacy input 和公共输入 Public input, 如果 Z(Privacy input, Public input)=True, 那么, 证明者执行 Prove 操作, 生成零知识证明 zk_proof, 其中, zk_proof={ A , B , C , Public input}, 而 $A=\alpha+\sum_{i=0}^{m}z_i a_i(x)+r\delta$, $B=\beta+\sum_{i=0}^{m}z_i b_i(x)+s\delta$, $C=\frac{\sum_{i=n+1}^{m}z_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))+h(x)t(x)}{\delta}+(As+Br-rs\delta)$ 上式中 $\{z_i\}_{i=0}^{n}$ 为 Public input, $\{z_i\}_{i=n+1}^{m}$ 为 Privacy input, r 和 s 为证明者从域 F 选取的随机变量, 上式中的 $\{a_i(x), b_i(x), c_i(x)\}_{i=0}^{m}$, $t(x)$ 是已知的零知识证明约束条件验证程序 Z 生成的与零知识证明约束条件相关的多项式, $h(x)$ 满足: $\sum_{i=0}^{m}z_i a_i(x)\sum_{i=0}^{m}z_i b_i(x)=\sum_{i=0}^{m}z_i c_i(x)+h(x)t(x)$。证明者要证明其知道隐私输入 Privac input, 但又不透露 Privac input 的具体值时, 将 zk_proof 提交给验证者, 以完成具体的证明过程。

3) Verify(verification key, zk_proof)→result。验证者基于 verification key 生成零知识证明以太坊智能合约 Verify Contract, 并以证明者提交的 zk_proof 作为合约的输入参数, 执行零知识证明验证算法, 得到结果 result={Ture or False}, 如果 zk_proof 为满足 Z 的隐私输入 Privac input 和公共输入 Public input 生成, 则结果 result 为 True, 否则结果为 False。当结果 result 为 True 时, 验证者就认为, 证明者在不透露 Privac input 的情况下, 证明其知道满足零知识问题的 Privac input。进一步地, 上述以太坊验证智能合约 VerifiyContract 是以 zk_proof 为输入参数, 与验证者持有的 verification key 一起进行基于椭圆曲线的双线性配对, 具体是判断以下公式(1)的等式是否成立, 成立则 result 为 True, 即用户提交的证明 zk_proof 是有效的:

$$\mathrm{e}(A_1,B_1)==\mathrm{e}(V_\alpha,V_\beta)\cdot\mathrm{e}(V_x,V_\gamma)\cdot\mathrm{e}(C_1,V_\delta) \quad (1)$$

其中, $\mathrm{e}:(G_1\times G_2)\to G_T$, $A_1=AG$, $B_1=BH$, $C_1=CG$, verification key={ V_α , V_β , V_γ , V_δ , V_{γ_abc} }, zk_proof=

{ A , B , C ,Public input}, Public input 为 $\{z_i\}_{i=0}^{n}$, $V_x=\frac{\sum_{i=0}^{n}z_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))G}{\gamma}$ 其中的累加式中 n 是公共输入 Public input 的参数个数。

2.4 零知识证明的完备性和可靠性

一个健壮的零知识证明算法应该满足完备性和可靠性, 其中, 完备性是指如果证明者遵循证明过程的每一个计算步骤, 则其计算得到的零知识证明一定能够通过验证者的验证, 可靠性是指除了证明者本身, 没有人能够假冒证明者生成正确的零知识证明, 并通过验证者的验证。

对于本文所采用的G16算法, 其是健壮的, 因为其满足零知识证明的完备性和可靠性。下面将基于 2.3 节所述的 Setup、Prove、Verify 三个步骤中的相关参数和变量描述 G16 算法的完备性和可靠性.

1) 完备性

从 2.3 节可知, 证明者遵循证明过程, 在 Prove 操作步骤中, 利用获取的 proving key 中的α、β、γ、δ、x 等参数, 以及自身的公共输入 Public input 和隐私输入 Private input, 正确计算出了 zk_proof={ A , B , C ,Public input}, 则证明者将 zk_proof 提交给验证者, 验证者将 zk_proof 中的 A 和 B 相乘得到如下结果: $(\alpha+\sum_{i=0}^{m}z_i a_i(x)+r\delta)\cdot(\beta+\sum_{i=0}^{m}z_i b_i(x)+s\delta)$, 同时, 验证者利用其在 Setup 操作步骤得到的α、β、γ、δ、x, $a_i(x)$, $b_i(x)$, $c_i(x)$ 以及 zk_proof 中的 C 和 Public input= $\{z_i\}_{i=0}^{n}$ 进行乘加运算得到如下结果: $\alpha\beta+\frac{\sum_{i=0}^{n}z_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\gamma}\gamma+C\delta$, 此时, 验证者判断以上所得到的结果是否满足如下等式

$$(\alpha+\sum_{i=0}^{m}z_i a_i(x)+r\delta)\cdot(\beta+\sum_{i=0}^{m}z_i b_i(x)+s\delta)=\alpha\beta+\frac{\sum_{i=0}^{n}z_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\gamma}\gamma+C\delta \quad (2)$$

上述等式的判断可以转换为椭圆曲线双线性配

对, 即通过如下转换

$$\mathrm{e}(G,H)^{(\alpha+\sum_{i=0}^{m}z_i a_i(x)+r\delta)\cdot(\beta+\sum_{i=0}^{m}z_i b_i(x)+s\delta)}= \mathrm{e}(G,H)^{\alpha\beta}\,\mathrm{e}(G,H)^{\frac{\sum_{i=0}^{n}z_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\gamma}}\,\mathrm{e}(G,H)^{C\delta} \tag{3}$$

其中, $\mathrm{e}(G,H)$ 为椭圆曲线双线性配对公式, G 为群 G1 的生成元, H 为群 G2 的生成元, $\mathrm{e}:(G_1\times G_2)\to G_T$, 而根据双线性配对的性质, 可得公式(3)的左右两边与公式(1)的验证等式一致。因此, 当证明者遵循证明过程, 正确计算出 zk_proof={ A , B , C ,Public input}, 一定能够通过验证者的验证。

2) 可靠性

G16 中, 证明者通过 Prove 操作构造的零知识证明与公共参考字串 CRS 中的 proving key 成线性关系(如 2.3 节中的 A, B 和 C 是 proving key 中的α、β、γ、δ、x 等参数的线性函数), 假设某个证明者构造与 proving key 中所有参数成线性关系的零知识证明 A , B , C :

$$A=A_\alpha\alpha+A_\beta\beta+A_\gamma\gamma+A_\delta\delta+A(x)+ \frac{\sum_{i=0}^{n}A_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\gamma}+ \frac{\sum_{i=n+1}^{m}A_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\delta}+A_h(x)\frac{t(x)}{\delta} \tag{4}$$

$$B=B_\alpha\alpha+B_\beta\beta+B_\gamma\gamma+B_\delta\delta+B(x)+ \frac{\sum_{i=0}^{n}B_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\gamma}+ \frac{\sum_{i=n+1}^{m}B_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\delta}+B_h(x)\frac{t(x)}{\delta} \tag{5}$$

$$C=C_\alpha\alpha+C_\beta\beta+C_\gamma\gamma+C_\delta\delta+C(x)+ \frac{\sum_{i=0}^{n}C_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\gamma}+ \frac{\sum_{i=n+1}^{m}C_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\delta}+C_h(x)\frac{t(x)}{\delta} \tag{6}$$

要使得上式通过验证者的验证, 上式中的(4)和(5)的乘积必须与公式(2)的右半部分匹配。将(4)和(5)相乘, 并与公式(2)的右半部分对比, 由于公式(2)的右半部分没有 α^2, β^2, $\frac{1}{\delta^2}$, $\frac{1}{\gamma^2}$, $\frac{\alpha}{\delta}$, $\frac{\alpha}{\gamma}$, $\beta\gamma$, $\alpha\gamma$, 且$\alpha\beta$ 的系数为 1, 则可以得到公式(4)和(5)中的多个系数的值: A_α =1, B_α =0, A_β =0, B_β =1, $A_\gamma=0$, $B_\gamma=0$, $\sum_{i=0}^{n}A_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))=0$ $\sum_{i=0}^{n}B_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))=0$, $\sum_{i=n+1}^{m}A_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))+A_h(x)t(x)=0$, $\sum_{i=n+1}^{m}B_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))+B_h(x)t(x)=0$, 则公式(4)和(5)可简化为:

$$A=\alpha+A_\delta\delta+A(x) \tag{7}$$

$$B=\beta+B_\delta\delta+B(x) \tag{8}$$

进一步地, 将公式(6)与 δ 相乘, 并与式(2)的 $C\delta$ 比较, 其中式(2)的

$$C=\frac{\sum_{i=n+1}^{m}z_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))+h(x)t(x)}{\delta}+(As+Br-rs\delta)。$$

由于式(2)的 $C\delta$ 中不存在$\frac{\delta}{\gamma}$, 则式(6)的表达式 $\sum_{i=0}^{n}C_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))$ 等于 0, 因此, 式(6)可简化为

$$C=C_\alpha\alpha+C_\beta\beta+C_\gamma\gamma+C_\delta\delta+C(x)+ \frac{\sum_{i=n+1}^{m}C_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\delta}+C_h(x)\frac{t(x)}{\delta} \tag{9}$$

此时, 将公式(7)(9)代入验证公式(2), 可得

$$\begin{aligned}&(\alpha+A_\delta\delta+A(x))(\beta+B_\delta\delta+B(x))\\ &=\alpha\beta+\sum_{i=0}^{n}z_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))+\\ &C_\alpha\alpha\delta+C_\beta\beta\delta+C_\gamma\gamma\delta+C_\delta\delta^2+C(x)\delta+\\ &\sum_{i=n+1}^{m}C_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))+C_h(x)t(x)\end{aligned} \tag{10}$$

上式中, 左右两边包含 α 和 x 以及包含 β 和 x 的项是相等的, 即

$$\alpha B(x)=\sum_{i=0}^{n} z_i \alpha b_i(x)+\sum_{i=n+1}^{m} C_i \alpha b_i(x) \tag{11}$$

$$\beta A(x)=\sum_{i=0}^{n} z_i \beta a_i(x)+\sum_{i=n+1}^{m} C_i \beta a_i(x) \tag{12}$$

上式中, 如果假设 $\{z_i\}_{i=n+1}^{m}=\{C_i\}_{i=n+1}^{m}$, 那么, 公式(11)和(12)变为

$$B(x)=\sum_{i=0}^{m} z_i b_i(x) \tag{13}$$

$$A(x)=\sum_{i=0}^{m} z_i a_i(x) \tag{14}$$

将(13)和(14)代入公式(10), 可取得如下的一个等式

$$\sum_{i=0}^{m} z_i a_i(x)\sum_{i=0}^{m} z_i b_i(x)=\sum_{i=0}^{m} z_i c_i(x)+C_h(x)t(x) \tag{15}$$

由于验证公式(2)将左右两边相同项相消后, 得到的验证等式实质是 2.3 节的 prove 操作步骤中的

$$\sum_{i=0}^{m} z_i a_i(x)\sum_{i=0}^{m} z_i b_i(x)=\sum_{i=0}^{m} z_i c_i(x)+h(x)t(x)$$

, 该式是真实证明者利用其公共输入 $\{z_i\}_{i=0}^{n}$ 及隐私输入 $\{z_i\}_{i=n+1}^{m}$ 与要证明的零知识证明约束条件验证程序 Z 通过算数电路变换、R1CS 变换、二次算数程序 QAP 变换等操作所生成的与零知识证明约束条件相关的多项式 $\{a_i(x), b_i(x), c_i(x)\}_{i=0}^{m}$, $t(x)$ 一起构造的, 而隐私输入 $\{z_i\}_{i=n+1}^{m}$ 只有真正的证明者才知道, 即只有真正的证明者才能使得该式相等, 而公式(15)成立且与上述式子等价的条件是 $\{z_i\}_{i=n+1}^{m}=\{C_i\}_{i=n+1}^{m}$, 其中 $\{z_i\}_{i=n+1}^{m}$ 为证明者的隐私输入, 也就是说只有真正知道隐私输入的证明者才能够使得构造出来的零知识证明 A, B, C (公式(4)~(6))通过验证者的验证。此外, 公式(2)的验证等式可以表示为

$$f(\alpha,\beta,\delta,\gamma,x)=(\alpha+\sum_{i=0}^{m} z_i a_i(x)+r\delta)\cdot(\beta+\sum_{i=0}^{m} z_i b_i(x) +s\delta)-\frac{\alpha\beta-\sum_{i=0}^{n} z_i(\beta a_i(x)+\alpha b_i(x)+c_i(x))}{\gamma} \gamma-C\delta \tag{16}$$

其中, 当 $f(\alpha,\beta,\delta,\gamma,x)=0$ 时, 验证成功, 而根 Schwartz-Zippel 定理可知, 对于一个域 F 中 n 元非零多项式 $f(x_1,x_2,\ldots,x_n)$, $\tau_1,\tau_2,\ldots,\tau_n$ 是从域 F 的有限集合中任意选取的随机变量, 则要使得 $f(\tau_1,\tau_2,\ldots,\tau_n)=0$ 的概率几乎为 0, 也就是说, 如果证明者不是使用验证者生成的包含在 CRS 中的 α、β、γ、δ、x 及其自身的公共输入 public input 及隐私输入 private input 生成零知识证明 zk_proof={ A, B, C, public input}, 而是使用不确定的 α、β、γ、δ、x 等生成的不正确的 zk_proof'来进行验证, 那么要通过公式(2)所示的验证几乎是不可能的。综上, G16 算法是满足可靠性的。

最后, 对于上述G16算法的完备性和可靠性, 读者可以进一步参见该算法的原始文献的第 3 节[31]。

3 架构和合约描述

3.1 系统架构

图 5 展示了本文的数字身份认证和管理系统架构, 相比于图 3 中的传统架构, 本文架构最显著的特征就是增加了基于 ZoKrates 工具集的可验证凭证零知识证明, 同时利用智能合约完成可验证凭证管理和验证的所有操作。此外, 为了方便架构描述, 本文在表 3 中给出了架构描述中用到的一些符合及其含义。

本文框架包含如下 3 种角色:

1) 用户: 用户是可验证凭证 VC 的持有者, 为获得凭证, 其首先需要生成私钥 SK_{user} 和公钥 PK_{user}, 然后将公钥 PK_{user} 和身份真实信息提交给身份提供商 IDP, IDP 完成身份核验后为用户颁发表 2 格式的可验证凭证。用户获得凭证后, 为获得相关服务, 需向相应的服务提供商获取零知识证明约束条件验证程序 Z 和 Proving key, 并利用 ZoKrates 生成可验证凭证的零知识证明 zk_proof。最后, 用户向服务提供商提交 zk_proof 和 IDP 对可验证凭证中的数字签名 Sign(Sign_H), 以请求零知识身份认证和获取服务。

2) 身份提供商 IDP: 身份提供商 IDP 是可验证凭证的颁发机构, 其在收到用户的公钥 PK_{user} 和真实身份信息后, 会核验真实身份的有效性, 然后为用户生成表 2 所示格式的可验证凭证, 其中 IDP Signature= Sign(Sign_H)。此外, 为了管理凭证的撤销状态, 服务提供商会在颁发凭证的同时, 将凭证状态激活, 并存储到凭证状态智能合约 Cert_Status_SC 中。

3) 服务提供商 SP: 服务提供商 SP 是应用服务的提供机构, 其将根据服务所需要的属性, 生成属性的零知识证明约束条件验证程序 Z、proving key、verification key、零知识证明证明验证智能合约, 并在用户请求服务前, 为用户提供 Z 和 proving key, 以

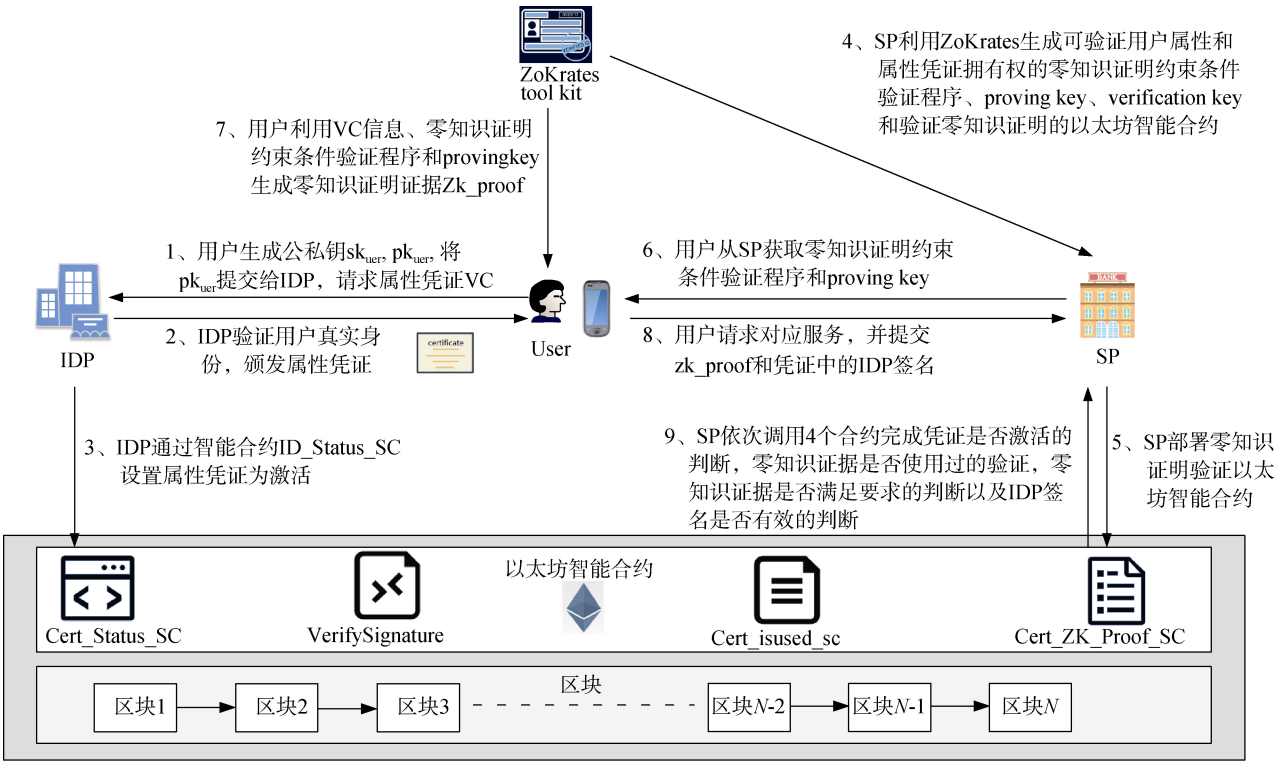


图 5　本文数字身份认证和管理系统架构

Figure 5　The system architecture of this paper

表 3　符号及描述

Table 3　Symbol and description

符号	描述
G	椭圆曲线 G 点
(SK_{user}, PK_{user})	用户的私钥和公钥
H_k(.)	keccak256 哈希函数
Sign_H	H_k (Attribute ‖ User DID ‖ IDP DID)
Sign(.)	IDP 的数字签名
zk_proof	VC 生成的零知识证明
Hash_P	zk_proof 的哈希值
Z	ZoKrates 的零知识证明程序
Private input	Z 的隐私输入部分
Public input	Z 的公共输入部分
Sign_EthAddress	签名者以太坊账号地址

让用户生成零知识证明 zk_proof, 并调用零知识证明验证智能合约完成证明的验证和服务的返回。

3.2　凭证认证合约架构

从图 5 可以看到, 本文系统架构使用了 4 个智能合约对可验证凭证进行管理和零知识认证, 具体合约为:

1) 凭证状态智能合约 Cert_Status_SC: 该合约由身份提供商 IDP 管理, 以在颁发凭证后激活凭证状态或在需要撤销凭证时, 撤销凭证。

2) 零知识证明使用状态智能合约 Cert_isused_SC: 该合约用于防止已经使用过的零知识证明 zk_proof 被非法获取, 并利用其从服务提供商 SP 非法获取服务, 因此, zk_proof 每次使用后都要重新生成, 而使用过的 zk_proof 都将利用该合约进行记录, 以防止重放攻击。

3) 零知识证明有效性验证智能合约 Cert_ZK_Proof_SC: 该合约是服务提供商 SP 基于其所构建的程序 *Z*, 利用 ZoKrates 生成的, 用户提交零知识证明 zk_proof 后, 服务提供商 SP 将 zk_proof 作为参数提交给该合约, 如果 zk_proof 满足条件的, 则该合约返回真值, 否则返回假值。

4) 可验证凭证有效性验证智能合约 VerifySignature: 该合约用于验证可验证凭证中的身份提供商 IDP 的数字签名 IDP Signature=Sign (Sign_H), 以保证生成零知识证明 zk_proof 的可验证凭证是由合法的身份提供商 IDP 所颁发。

图 6 以合约调用的方式展示了服务提供商 SP 在收到用户的请求及 zk_proof 等参数后, 依次调用上述 4 个智能合约完成可验证凭证的零知识证明有效性验证的架构图。

4　架构关键流程设计

4.1　身份属性的可验证凭证颁发

图 7 是用户身份属性的可验证凭证 VC 的颁发流

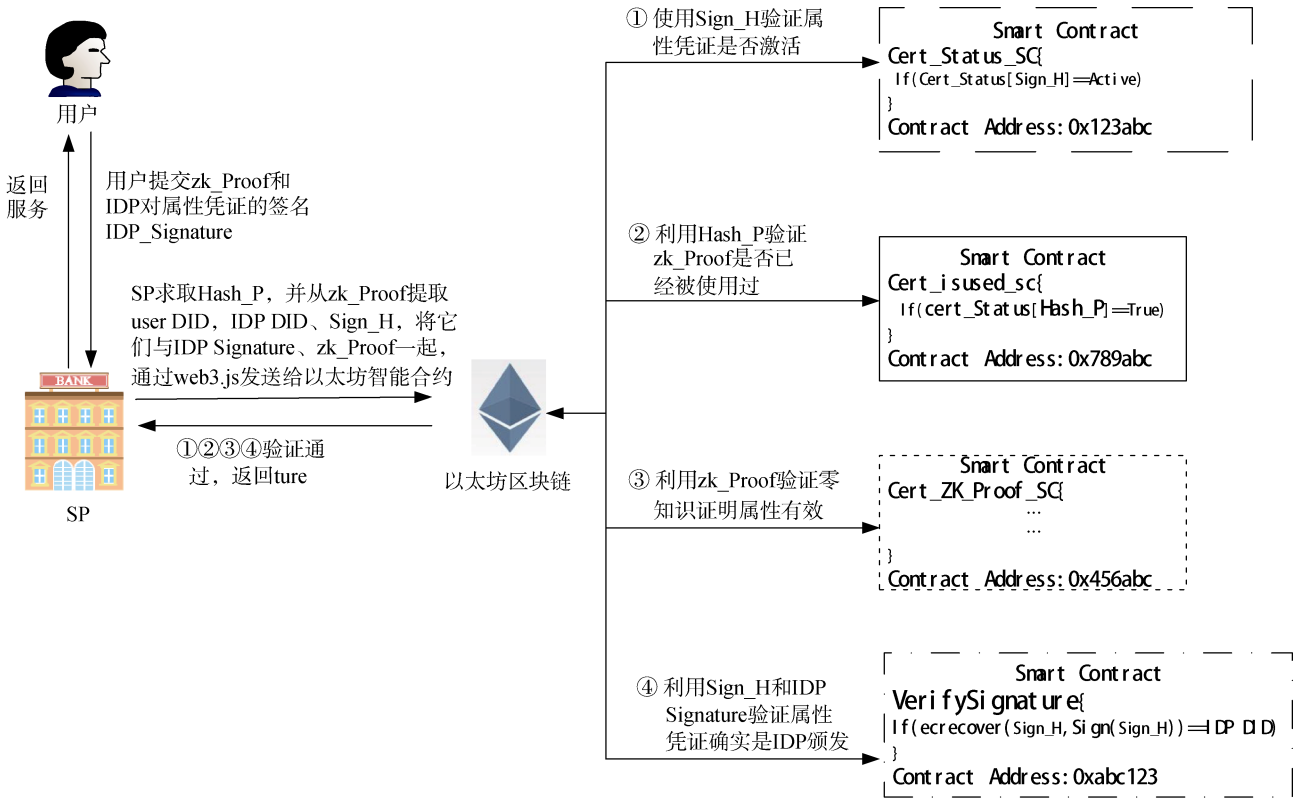


图 6　可验证凭证的零知识证明智能合约验证架构

Figure 6　The architecture of smart contracts used to verify zero knowledge proof of VC

程, 其是图 5 中的步骤 1~3 的操作过程的细化, 下面说明其中的具体内容:

1) GenKey(SK_{user}, PK_{user}): 用户使用 ZoKrates 生成基于 EdDSA 的椭圆曲线私钥 SK_{user} 和公钥 PK_{user};

2) Require(issue_VC, PK_{user}): 用户根据 SP 的服务需求, 请求身份提供商 IDP 颁发基于属性的可验证凭证 VC, 并向 IDP 提交其公钥 PK_{user}, IDP 将用 PK_{user} 为用户生成数字身份标识符 User DID;

3) VerifyRealIdentity(User Identity): 身份提供商 IDP 核验用户真实身份的有效性;

4) H_K(PK_{user})→User DID: 身份提供商 IDP 利用哈希算法 keccak256 求取用户公钥的哈希值, 并将结果作为 VC 中用户的数字身份标识符 User DID;

5) Eth_Addr→IDP DID: 身份提供商利用其以太坊账号地址作为其数字身份标识符 IDP_DID;

6) Sign(Sign_H)→VC: 身份提供商 IDP 利用其以太坊账号私钥对用户申请的身份属性信息进行签名, 同时生成基于身份属性的可验证凭证 VC;

7) Cert_Status→Active: 身份提供商 IDP 以 Sign_H 为参数, 调用 VC 状态智能合约 ID_Status_SC, 将所颁发的 VC 激活, 以说明 VC 可用;

8) IssueVC→User: 身份提供商 IDP 将身份属性的可验证凭证 VC 颁发给用户。

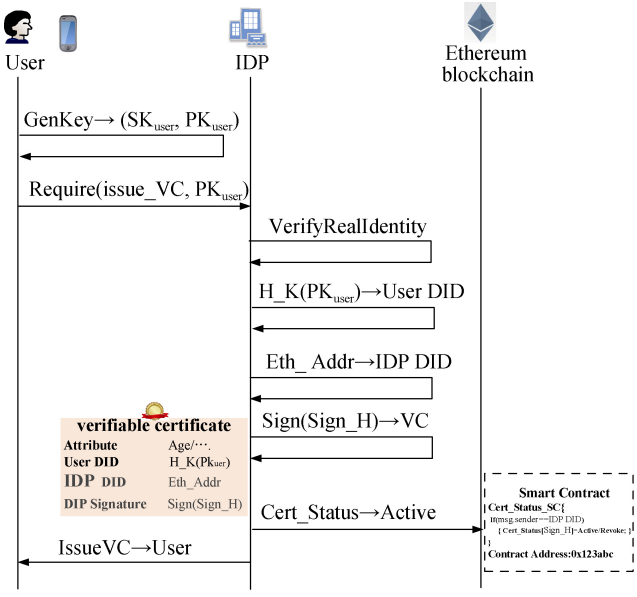


图 7　身份属性的可验证凭证颁发

Figure 7　Issue of VC for identity attribute

4.2　零知识验证智能合约和零知识证明的生成

图 8 是可验证凭证 VC 的零知识验证智能合约和证明 zk_proof 的生成流程, 其是图 5 步骤 4~7 的操作

过程的细化, 下面说明其中的具体内容:

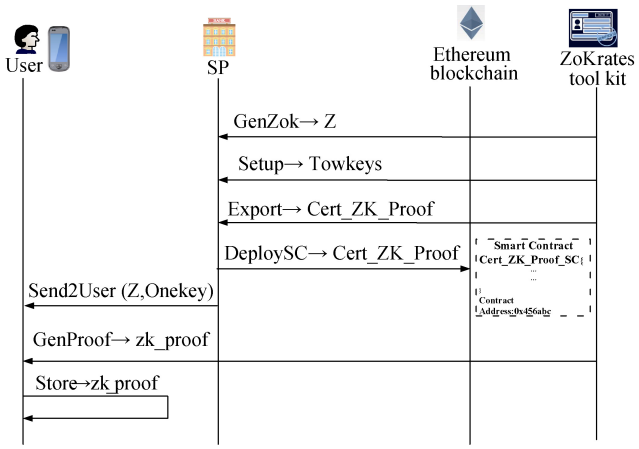


图 8　可验证凭证的零知识验证智能合约和零知识证明的生成

Figure 8　The generation of zero-knowledge verification smart contract and proof of verifiable credential

1) GenZok→Z: 服务提供商 SP 利用 ZoKrates 的 DLS 语言描述零知识证明约束条件验证程序 Z, 该程序的伪代码如算法 1 所示, 其能够使得用户在不透露其 VC 中的属性值的情况下, 证明用户确实是 VC 的持有者(算法 1 中的步骤 1~13, 在不透露私钥 SK_{user} 的情况下证明用户公钥 PK_{user} 由私钥 SK_{user} 生成, 而用户真实的身份标识符 User DID 由用户公钥 PK_{user} 经过哈希运算生成), 证明满足 SP 要求的属性值确实包含在用户的 VC 中(算法 1 中的步骤 14~19, 在不透露隐私属性 Attribute 和用户公钥 PK_{user} 的情况下, 证明 VC 的哈希值 Sign_H 由隐私属性、用户公钥 PK_{user} 和 IDP 公钥共同生成), 证明 VC 中的属性满足 SP 授权服务的要求(算法 1 中的步骤中的步骤 20~24, 在不透露属性值 Attribute 的情况下, 证明 Attribute 包含于 SP 设定的属性条件范围内)。因此, 程序 Z 的隐私输入 Private input 应该包括用户的属性值 Attribute、用户的私钥 SK_{user}、用户的公钥 PK_{user}, 公共输入 Public input 应该包括 VC 中的用户数字身份标识符 User DID、身份提供商 IDP 的数字身份标识符 IDP DID、VC 的哈希值 Sign_H。

2) Setup→Towkeys: 服务提供商 SP 利用图 4 所示的 ZoKrates 的构建 Setup 步骤生成 CRS, 即证明密钥 proving key 和验证密钥 verification key。

3) Export→Cert_ZK_Proof: 服务提供商 SP 利用 ZoKrates 生成零知识证明验证智能合约 Cert_ZK_Proof, 该合约通过调用以太坊区块链的预编译椭圆曲线运算库完成公式(1)所示的双线性配对和判断, 以验证合约的输入参数 zk_proof 是否满足配对要求, 具体算法描述如算法 2 所示, 其中, 算法描述中的 EIP196 表示以太坊区块链的预编译椭圆曲线加法和标量乘法运算库, EIP197 表示以太坊区块链的预编译椭圆曲线配对库, zk_proof 中的 Public_in 对应算法 1 中的公共输入 Public input={User DID, IDP DID, Sign_H}。

4) DeploySC→Cert_ZK_Proof: 服务提供商 SP 将零知识证明验证智能合约 Cert_ZK_Proof 部署到以太坊区块链中。

5) Send2User(Z,Onekey): 服务提供商 SP 在用户请求服务时, 将程序 Z 和 proving key 返回给用户。

6) GenProof→zk_proof: 用户将 VC 中的属性值 Attribute, 其私钥 SKuser 和公钥 PKuser 作为 Z 的隐私输入 Private input, 将 VC 中的 User DID、IDP DID、以及求取的 Sign_H 作为 Z 的公共输入 Public input, 与 proving key 一起, 利用 ZoKrates 生成凭证的零知识证明 zk_proof。

7) Store→zk_proof: 用户保存 zk_proof 到其智能移动终端。

此外, 需要说明的是, 由 2.4 节的关于 G16 算法的完备性和可靠性描述可知, 用户要通过公式(1)的验证, 需要生成正确的 zk_proof={A, B, C, public input}, 即需要将 SP 在执行构建 Setup 操作步骤时, 所生成的 CRS: proving key 和 verification key 中包含的α、β、γ、δ、x 等相关参数, 与用户自身的 Public input 和 Private input 结合, 构造正确的 zk_proof={A, B, C, public input}。而α、β、γ、δ、x 是 SP 在生成 CRS 时随机选取的, 因此, 如果 SP 要更新 CRS, 需要重新执行 Setup 操作步骤, 并随机选取新的α、β、γ、δ、x, 以生成新的 CRS: proving key 和 verification key。此时, 如果用户仍然使用更新前的 zk_proof, 将无法通过公式(1)的验证, 因此, SP 在更新 CRS 后, 用户需要再次从 SP 获取新的 proving key, 同时 SP 也需要部署新的由 verification key 生成的智能合约 Cert_ZK_Proof, 即 SP 如果要更新 CRS, 用户和 SP 需要重新执行本节所述的步骤 2)到步骤 7)的相关操作。

算法 1. ZoKrates 零知识证明算法程序 Z.

输入: 隐私输入 Private input={Attribute, SK_{user}, PK_{user}}, 公共输入 Public input={User DID, IDP DID, Sign_H};

输出: True 或 False.

1　$G = [G_x, G_y]$;

2　PK = $SK_{user} \times G$

3　**IF** PK== PK_{user} **THEN**

4 goto Step 8;
5 **ELSE**
6 **Return** False;
7 **END IF**
8 User_DID'=H_K(PK_{user});
9 **IF** User_DID'== User_DID **THEN**
10 goto Step 14;
11 **ELSE**
12 **Return** False;
13 **END IF**
14 Sign_H'= H_K(Attribute || User DID || IDP DID);
15 **IF** Sign_H'== Sign_H **THEN**
16 goto Step 20;
17 **ELSE**
18 **Return** False;
19 **END IF**
20 **IF** Attribute∈ Attribute_Range **THEN**
21 **Return** True;
22 **ELSE**
23 **Return** False;
24 **END IF**
25 算法 1 结束

算法 2. 验证智能合约 Cert_ZK_Proof 的算法.

输入: zk_proof={ A , B , C , Public_in}

输出: True 或 False

1 **Contract** Cert_ZK_Proof
2 [V_α,V_β,V_γ,V_δ,V_{γ_abc}]= verification key;
3 **For** (i = 0; i < zk_proof.Public_in.length; i++)
4 V_x=**EIP196**.addition(V_x, **EIP196**.scalar_mul (Public_in [i], V_{γ_abc}[i]));
5 **EndFor**
6 **IF**(**EIP197**.Pairing.pairingProd4 (zk_proof.A·G, zk_proof.B·H, V_α, V_β, V_x, V_γ, zk_proof.C·G, V_δ)) ==1)
7 **Return** True;
8 **Else**
9 **Return** False;
10 **EndIF**
11 **End Contract**

4.3 零知识身份验证智能合约的验证与服务的授权

图 9 是服务提供商 SP 调用合约验证用户零知识证明 zk_proof 以及服务授权的流程, 其是图 5 步骤 8~9 的操作过程的细化, 下面说明其中的具体内容。

1) RequireService(zk_proof, sign(sign_H)): 用户向服务提供商 SP 请求相应的服务, 并将服务所需的零知识证明 zk_proof 提交给服务提供商 SP, 同时提交可验证凭证 VC 中记录的数字签名 sign(sign_H)。

2) Extract(zk_proof)→Public input: 服务提供商 SP 从 zk_proof 中提取用户的公共输入 Public input, 其中包括 User DID、IDP DID、Sign_H。

3) CertActive(Sign_H): 服务提供商 SP 以 Sign_H 为输入参数, 调用凭证状态智能合约 ID_Status_SC, 以判断凭证是否处于激活状态。

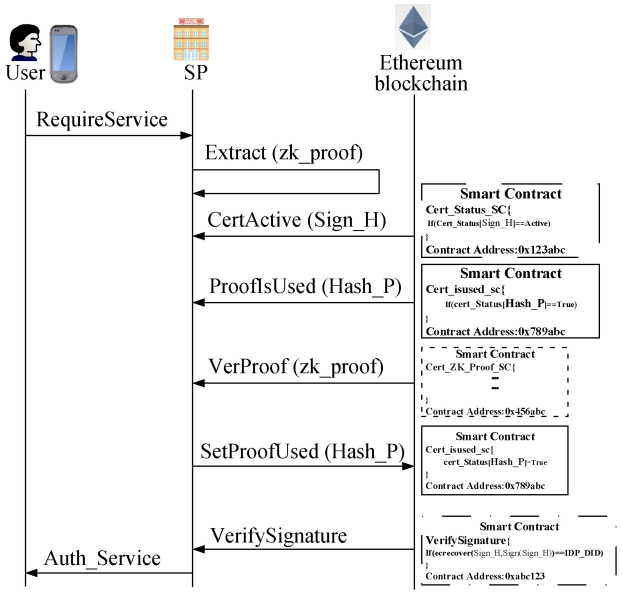


图 9 零知识身份证明的智能合约验证与服务授权
Figure 9 Zero-knowledge authentication of smart contract and service's authorization

4) ProofIsUsed(Hash_P): 如果凭证是激活的, 服务提供商 SP 求取 zk_proof 的哈希值 Hash_P, 并以该哈希值为参数, 调用 Cert_isused_SC 智能合约, 以判断凭证所生成的零知识证明 zk_proof 是否已经被使用过, 以防止 zk_proof 和用户的 VC 相关信息被非法截获后, 从服务提供商 SP 非法获取服务。

5) VerProof(zk_proof): 如果 zk_proof 没有被使用过, 服务提供商 SP 以 zk_proof 为输入参数, 调用 Cert_ZK_Proof_SC 智能合约, 以验证用户确实是 VC 的持有者, 验证 SP 要求的属性值确实包含在用户的 VC 中, 验证 VC 中的属性满足服务授权要求。

6) SetProofUsed(Hash_P): 如果 zk_proof 验证通过, 服务提供商 SP 以 Hash_P 为输入参数, 调用 Cert_isused_SC 智能合约, 以设置 zk_proof 已经被使用, 此时, 如果用户要想再次服务提供商 SP 的服务, 需要重新利用 ZoKrates 生成新的 zk_proof, 由于 ZoKrates 每次所生成的零知识证明 zk_proof 中, 除了用户的公共输入 Public input 不变外, 其余数据具有

随机特性, 因此, 通过智能合约判断 zk_proof 是否使用过可以有效的防止重放攻击。

7) VerifySignature(Sign_H,Sign(Sign_H)): 服务提供商 SP 以 zk_proof 中的用户公共输入 Public input 中的 Sign_H 和用户提交的 VC 中的数字签名项 IDP Signature=Sign(Sign_H) 为输入参数, 调用 VerifySignature 智能合约, 其中, 该合约使用以太坊预编译的 ECDSA 椭圆曲线数字签名验证算法 ecrecover(Sign_H, Sign(Sign_H))来验证 Sign(Sign_H)是否由合法的身份提供商 IDP 生成, 以此判断可验证凭证 VC 是由合法的身份提供商 IDP 颁发。具体的验证算法是判断 ecrecover(Sign_H,Sign(Sign_H))的返回地址 Sign_Eth_address 是否与 zk_proof 中的用户公共输入 Public input 中的 IDP DID 相等, 如果相等, 则验证成功。

8) Auth_Service User: 服务提供商授权用户相应服务。

5 实际用例

为了验证本文所提出的系统框架, 以及评估该框架应用于实际场景时的相关性能, 本文以年龄属性的零知识证明与授权服务为例, 设计了一个原型系统, 对于该原型系统, 本文利用 ZoKrates+Remix+Solidity 实现了系统中的相关智能合约的生成与设计, 并将智能合约部署到了自主搭建的以太坊私有链中, 同时利用 Html+Javascript+Web3.js 设计了原型系统的以太坊去中心化应用(DApp)客户端(图 10 展示了其中的服务提供商 SP 的客户端)。

在该原型系统中, 用户首先按照图 5 的步骤 1~3 从身份服务提供商 IDP 获得一个如表 4 所示的年龄身份属性(Age)的可验证凭证。用户在拿到该凭证后, 按照图 5 的步骤 4~7 将凭证转换为零知识证明 zk_Proof。接着, 用户按照图 5 的步骤 8~9, 将 zk_ Proof、凭证中 IDP 的签名值 IDP Signature 发送给服务提供商 SP, SP 调用相关智能合约完成用户的属性可验证凭证的零知识证明, 并在证明通过后, 向用户授权服务。

表 4 IDP 颁发给用户的身份属性凭证

Table 4 The VC of user issued by IDP

凭证项	值
Age	33
User DID	0x6631c2d300000000025350fc000000009462b13c000000000e865b48e00000000
IDP DID	0x6e819b34c53dc81400d95ab87bfdbe3ae80e2ea2
IDP Signature	0x6c8e7d5d6f13dcc32db36aafaa87b988944f5ba6ab7bed1f2949d21187cf300a45d5ab9ab4f394896f04dfaf20e26dc4cb56c936b313d8db6d9b4182808634331b

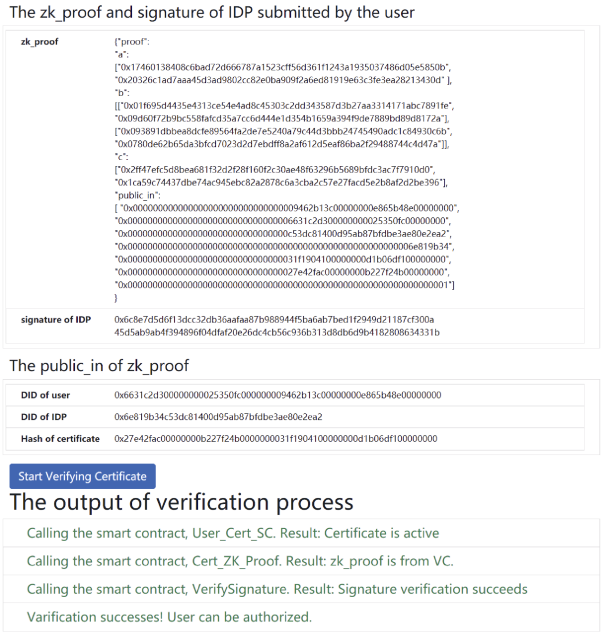


图 10 服务提供商 SP 客户端界面

Figure 10 Interaction interface of service provider

图 10 中, zk_proof 为用户提交的零知识证明, IDP Signature 为 IDP 对用户年龄凭证的签名值, 当服务提供商 SP 收到这两个值后, 图 10 的客户端程序会自动提取 zk_proof 中的公共输入 Public input(图 10 中的 public_in)中的用户的身份标识符 user DID, 身份提供商 IDP 的身份标识符 IDP DID, 用户年龄凭证的哈希值 Sign_H。然后, 客户端将以手动或者自动的方式, 以 user DID、zk_proof、IDP DID、Sign_H、IDP Signature 为参数, 分别调用 Cert_Status_SC、Cert_ZK_Proof、Cert_isused_SC、VerifySignature 智能合约完成年龄凭证的零知识证明, 证明成功后, 即可向用户进行服务授权。显然, 从图 10 可以看到, 用户在不需要提交任何年龄相关的明文信息情况下可以完成年龄、年龄凭证的拥有权以及年龄凭证的有效性的有效证明, 因此, 原型系统表明, 本文提出的系统框架是有效的。

此外, 尽管本文的原型系统是以年龄属性的零知识证明与授权服务为例, 但是, 对于其他需要向服务提供商证明具体属性数值的范围的证明也是适用的(例如银行贷款时证明收入范围, 新冠疫情行程码排查时, 证明到访的位置范围等), 区别仅在于, 需要将零知识证明程序中的属性判断部分(算法 1 中的步骤 20~24)改为对应的属性判断(本文原型系统中, 服务提供商 SP 进行服务授权的条件是年龄大于 18 岁, 小于 60 岁, 即算法 1 中的步骤 20 的属性判断部分为 Attribute∈[18,60], 对于其他应用, 需要根据实际情况修改该部分的属性判断算法)。

进一步地, 尽管不同应用的零知识证明程序 Z 中的属性判断部分不同, 但是最终生成的用于提交给服务提供商 SP 的零知识证明相关参数(User DID、zk_proof、IDP DID、Sign_H、IDP Signature)结构是相同的(仅数值上存在差异), 因为对于不同应用的参数, 仅仅 zk_proof 可能会因为输入属性 Attribute 的不同导致不同, 但是对于 zk_proof 而言, 其生成结构仅仅与公共输入有关(算法 1 中的输入参数 Public input), 而对于本文架构不同的应用, 公共输入是不变的, 仅仅是隐私输入部分的属性项 Attribue 发生变化, 而该变化并不影响最终生成的 zk_proof 的结构。此外, 相关智能合约也是几乎不变的, 因此, 基于本文系统框架的不同应用在调用智能合约完成基于属性凭证的零知识证明与授权服务时的系统指标完全可以利用原型系统来进行评估和分析。

6 用例与架构评估分析

6.1 成本分析

在区块链中, 任何的行为操作都将产生交易, 而交易都需要支付一定的手续费, 在以太坊区块链中, 该手续费主要以气(Gas)的消耗来度量, 因此, 以太坊中的气的消耗可以用来衡量一个行为操作或一组行为操作所需的成本。由于本文框架的性能可以通过本文第 4 部分所述的原型系统来评估, 而该原型系统中, 气的消耗量主要包括部署合约时的交易消耗以及执行合约时的交易消耗两部分组成, 表 5 给出了这两部分的消耗量, 其中表中的 Cert_isused_SC 和 Cert_Status_SC 两个合约的执行消耗包括了合约变量的设置和读取的消耗, Cert_isused_SC 设置合约变量状态的消耗为 43424, 读取合约变量状态的消耗为 26354, User_Cert_SC 设置合约变量状态的消耗为 43467, 读取合约变量状态的消耗为 23324。

表 5 本文架构的合约部署与调用的以太坊气的消耗
Table 5 Gas consumption of smart contracts of the proposed architecture

合约	部署消耗	执行消耗
Cert_Status_SC	425545	66791
Cert_isused_SC	298661	69778
Cert_ZK_Proof	2687360	814385
VerifySignature	487738	48595
总消耗	3899304	999549

通常情况下, 为了估算智能合约的成本, 需要将气的消耗转换为实际的以太币, 具体的转换公式如下:

$$num_{ether} = Gas_{used} \times Gas_{price} \tag{17}$$

其中, num_{ether} 为以太币的数量, Gas_{used} 为调用智能合约所消耗的气的数量, Gas_{price} 为当前气的价格。在此, 气的价格 Gas_{price} 按照撰稿时, 以太坊公有链中调用智能合约的平均价格 75Gwei 来计算, 则可以计算出, 合约部署的总的以太币消耗量为 0.292ether, 执行合约完成一次可验证凭证的零知识证明验证的以太币总的消耗量为 0.075ether。显然, 从该结果可以看出, 完成一次零知识证明验证所消耗的以太币并不高, 而且在此是以公链的气的价格计算得到的结果, 如果该系统架构部署在一个机构或者某些应用中, 此时的区块链多以以太坊联盟链为主, 其中的气的价格要低得多(也可不需要交易费用, 而由系统主节点分配以太币), 对应的消耗量将更低, 因此, 本文架构的成本并不高。

6.2 吞吐量分析

系统架构的吞吐量与其所部署的以太坊区块链环境直接相关, 公有链具备较强的去中心化特性, 但是当前公有链的每秒交易数 TPS 仅为 15 笔左右[35], 很难满足系统框架在多用户场景下的并发服务请求和授权应用的吞吐量要求, 因此, 部署到私有链或者联盟链中是一种提高系统吞吐量的方法, 尽管本文的模型系统是在私有链中部署验证的, 但是正如 5.1 所述, 真实应用场景中多以联盟路为主, 而此处分析的系统吞吐量适用于以太坊联盟链。

以太坊区块链系统的吞吐量 TPS 通常可以通过以下理论公式进行评估:

$$TPS = \frac{Gas_{\text{limit}}}{Tx_{Gas} \times Block_{\text{Time}}} \tag{18}$$

其中 Gas_{limit} 是一个区块可容纳的气的上限, Tx_{Gas} 是执行一次交易的气消耗, $Block_{\text{Time}}$ 是生成区块的时间间隔。对于 Gas_{limit} 和 $Block_{\text{Time}}$ 而言, 可在创世区块中根据区块链具体情况和公式(18)的计算结果来设置相关参数, 而对于 Tx_{Gas}, 需要根据完成具体行为操作所需要的气来评估, 从表 5 可以看到, 本文提出的框架系统执行智能合约完成一次基于属性的可验证凭证颁发和零知识证明所需要的气为 999549, 因此, Tx_{Gas} 设为 999549, 而对于 $Block_{\text{Time}}$ 的值而言, 为了避免影响区块同步速度, 通常可设置 5s, 因此, 可以设置不同的 Gas_{limit} 来获得不同的吞吐量 TPS, 图 11 展示了本文分析的 TPS 和 Gas_{limit} 的关系, 从图中可以看到, 为了获得较高的系统吞吐量 TPS, 可以将 Gas_{limit} 设置得较大(正相关), 但是 Gas_{limit} 太高的

话, 又会导致单个区块的尺寸太大, 进而又会影响区块同步的速度。通常, 对于大多数应用系统的数字身份认证和管理业务而言, 吞吐量*TPS*在100左右完全能够满足实际应用需求(有 100 个用户同时请求服务提供商SP进行身份验证, 服务提供商SP同时发起100笔交易验证用户身份, 对于大型企业和应用系统而言, 如果有更多用户同时请求, 可以采用多链结合跨链等方式满足更高的吞吐量, 在此, 仅说明单链情况下, *TPS* 为 100 的情况), 因此, 对于本文所提出的系统框架而言, 要达到 *TPS*=100, 根据公式(18)的计算, 可以将 Gas_{limit} 设置为 499774500。

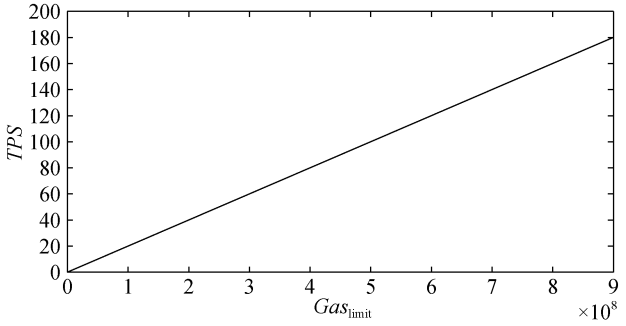


图 11　本文架构的 *TPS* 与 Gas_{limit} 的关系

Figure 11　The relationship between *TPS* and Gas_{limit}.

进一步地, 本文在自主搭建的以太坊私有链中测试了同时向服务提供商SP发出请求的用户数量从1变化到100时, 服务提供商 SP 以相同数量的以太坊账号并发调用 4 个智能合约完成身份认证的响应时间, 以模拟和评估系统吞吐量从 1 变化到预设的100时, 服务提供商完成所有请求用户的零知识身份验证所需要的时间(每秒处理的交易数量从 1 变化到100 时, 完成零知识身份验证所需要的时间)。图 12展示了测试结果, 其中, 横坐标为用户数, 纵坐标为不同并发用户数量下, 服务提供商完成不同数量用户的零知识身份验证所需要的平均时间。从图中可以看到, 随着同时访问服务提供商 SP 的用户的数量的增加(交易数量逐渐增加), 系统的并发响应时间逐渐变长, 当同时访问的用户数达100时, 完成所有用户的零知识身份验证的时间将达到 2.8 s(将公式(18)中的 $Block_{\text{Time}}$ 控制在 5 以内, 时间基本稳定在 2.8 s附近, 但是当控制其值大于 5 时, 该时间都比 2.8 s大, 而且从图 12 可以看到, 交易数量(*TPS*)越大, 所需时间越长, 而 *TPS* 与 Gas_{limit} 正相关, 因此, 并不是取得越大的 Gas_{limit} 就是越好的, 因为当实际的 *TPS* 到达一定数量时系统响应会变慢)。显然, 在同时访问的用户数量较多时, 系统的响应速度明显减慢, 这也是当前众多的区块链系统面临的主要性能瓶颈, 但是, 相比于非区块链架构的身份认证系统, 区块链架构的系统在牺牲性能的前提下保证了系统的安全和公开透明, 因为所有的并发调用都以交易的形式存储到了区块中。

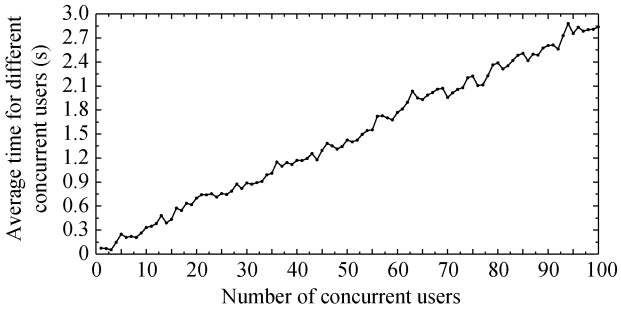


图 12　系统吞吐量测试结果

Figure 12　Test result of system throughput

6.3　零知识证明的生成时间分析

本系统架构中, 为了防止重放攻击, 用户所生成的零知识证明 zk_proof 只能使用一次, 如果用户需要再次访问服务提供商, 以获得服务授权, 需要重新生成新的zk_proof。因此, 需要对zk_proof的生成时间做一个评估, 以确定用户重复访问服务提供商 SP 获得服务授权的时间间隔。

由图 5 和图 8 可以看到, zk_proof 的生成是由用户在本地完成, 因此, 其生成时间在很大程度上可由用户的计算机硬件配置决定, 对于本文的原型系统而言, 采用的计算机硬件配置为: 处理器为 Intel Core i7-8550U @ 1.80GHz 2.00 GHz, 内存为 24G RAM, 基于该配置, 原型系统在本地生成 zk_proof 的时间为 3.1 s, 如果更换性能更优的硬件配置, zk_proof 的生成时间将更快。而且, 从图 12 中可以看到, 重新生成 zk_proof 的时间大于吞吐量为 100时, 服务提供商 SP 进行零知识验证的平均时间, 因此, 系统可以满足用户快速重复的服务访问和获取。

6.4　基于区块链智能合约的链上验证的安全问题、应对措施和必要性分析

本文利用智能合约实现公式(1)的双线性配对等操作的链上验证, 以实现零知识证明的公开可信验证, 从而达到避免单点故障和增加服务授权的可信和透明度的目的。但是, 智能合约的公开透明是否会泄露零知识证明中的相关参数, 在公开透明的智能合约验证情况下, 是否会出现串谋攻击以及如何避免攻击, 同时, 基于智能合约的公开透明验证是否是必要的呢? 对此, 本节对这些问题进行分析和讨论。

首先, 本文采用 G16 算法构造零知识证明系统, 该算法中, 验证者的参数对于证明者是公开的, 当验证者合法的利用智能合约进行零知识证明的链上验证时, 是不存在泄露零知识证明参数的问题的。

G16 算法是一种利用零知识证明约束条件验证程序(本文的 2.3 节的程序 Z 和算法 1)所生成的算术电路参数, 构建零知识证明验证的算法, 其中, 验证者通过 Setup 操作, 利用零知识证明约束条件验证程序生成算数电路约束的相关参数, 并生成公共参考字串 CRS: proving key 和 verification key, 而其中的 verification key 是由验证者自己保留, 用于生成零知识证明验证智能合约, proving key 会被提供给证明者, 而证明者得到的 proving key 中, 是包含了 verification key 中的参数的(从本文的 2.3 节的 Setup 操作和 Prove 操作步骤中的参数可以看到), 证明者得到 proving key 后, 将其与满足零知识证明约束条件验证程序的公共输入(本文的 2.3 和 2.4 节中的 $\{z_i\}_{i=0}^{n}$)和隐私输入(本文的 2.3 和 2.4 节中的 $\{z_i\}_{i=n+1}^{m}$)结合, 构造零知识证明 zk_proof, 并将其提交给验证者, 验证者利用 zk_proof 和 verification key 组合, 判断组合式是否满足事先构造的验证约束方程(本文的 2.4 节的公式(2), 其可以通过公式(1)进行验证), 如果满足, 则通过验证, 否则无法通过验证。因此, 验证者本身并没有什么秘密参数, 验证者的参数都是公开给证明者的, 关键的就是, 证明者的公共输入 $\{z_i\}_{i=0}^{n}$ 和隐私输入 $\{z_i\}_{i=n+1}^{m}$ 要满足约束条件验证程序 Z, 并能生成满足事先构造的验证约束方程(本文的 2.4 节的公式(2))的零知识证明 zk_proof。也就是说, 正常情况下, 验证者的参数都是公开给证明者的, 利用智能合约进行公开验证, 并不存在泄露秘密参数的情况。

其次, 本文采用的 G16 算法利用智能合约公开验证零知识证明, 虽然不存在泄露秘密参数的情况, 但是可能存在基于错误的 Verification key 构建恶意的零知识证明智能合约所引起的串谋攻击, 但是由于区块链的智能合约的不可篡改、地址唯一、部署和调用合约的交易数据会存储到不可篡改的分布式账本中, 该合谋攻击是可以有效避免的。

已知验证者自己保留 verification key, 并用其构造零知识证明验证智能合约, 当验证者收到证明者的零知识证明 zk_proof 时, 验证者利用 zk_proof 和 verification key 组合, 判断组合式是否满足事先构造的验证约束方程, 并根据判断结果决定是否通过验证。因此, 恶意的验证者可以自己构造一个不来自零知识证明约束条件验证程序生成的参数的 verification key, 并生成零知识证明验证智能合约, 同时在公开透明的合约中间接泄露 verification key 参数或者直接通过非公开方式泄露给恶意证明者 verification key 参数, 此时, 恶意的证明者就可以基于 verification key 构造一个对应的可以通过验证的零知识证明 zk_proof, 并提交 zk_proof 给服务提供商 SP, 服务提供商 SP 调用恶意的智能合约, 就可以通过验证。但是, 这种情况成立的条件需要满足: (1)恶意的验证者部署了恶意的验证合约后, 篡改了服务提供商 SP 的合约调用程序, 使得该程序通过该恶意合约的地址调用合约, 并完成恶意验证; (2)恶意的验证者与服务提供商也共谋了, 并合谋篡改服务提供商的合约调用程序, 使得该程序通过恶意合约的地址调用合约, 并完成恶意验证。然而, 对于区块链的智能合约而言, 当完成合约代码设计和编译后, 要部署到区块链中, 而在部署到区块链后, 合约使用方要使用合约, 就需要根据合约地址, 调用合约, 并获取合约参数或者传输合约参数, 从而完成合约的计算任务。而无论部署合约还是调用合约, 都需要向区块链发送交易, 而这些交易在共识和确认后将会被记录到不可篡改的区块链分布式账本中, 其中, 部署合约时, 存入区块链账本的交易数据包括部署者的区块链账号地址、合约地址。调用合约时, 存入区块链账本的交易数据包括调用者的区块链账号地址、合约地址、调用传入和返回的参数。而且, 合约在部署后, 合约的代码和合约地址是不可更改的。因此, 假设存在上述第 1 种作恶情况, 即(1)恶意的验证者部署了恶意的验证合约后, 篡改了服务提供商的合约调用程序, 使得该程序通过恶意合约地址调用恶意合约, 并完成恶意验证。此时, 服务提供商的合约调用程序在调用合约地址时, 会在区块链分布式账本中记录下调用者的区块链账号、合约地址、调用合约传入和输出参数, 而服务提供商在进行审计时, 是可以通过查询区块链中的账本信息, 发现恶意合约的地址与合法的合约地址是不一致的(因为恶意合约是后期部署的, 与最开始部署的合法的验证合约在地址上是不同的), 并对恶意行为进行及时处理。进一步地, 假设存在上述第 2 种作恶情况, 即(2)恶意的验证者与服务提供商也共谋了, 并合谋篡改服务提供商的合约调用程序, 使得该程序通过恶意合约地址调用恶意合约, 并完成恶意验证。那么调用合约时, 同样会在区块链分布式账本中记录下调用者的区块链账号、合约地址、合约参数, 而服务提供商同样可以通过查询区块链中的账本信息, 发现恶意合约地址与最开始部署的合法的合约地址不一致,

并发现恶意合约调用者的区块链账号, 而该账号在服务提供商内部应该与具体的实名操作者关联, 此时, 服务提供商内部在处理恶意行为时, 还可以对内部合谋者进行处理。此外, 对于两种情况, 也可以找到共谋的恶意证明者, 并进行惩罚, 因为两种情况中, 证明者要证明恶意构造的零知识证明 zk_proof 满足验证条件, 需要提交合法凭证中的 User DID 和 IDP DID(如图 10 的 DID of User 和 DID of IDP), 而 IDP DID 是合法权威的身份提供商 IDP 的身份标识符, 因此, 可以找到服务提供商 IDP 查询到 User DID 对应的实名者, 从而对实名信息对应的证明者进行惩罚。

因此, 上述利用错误的 verification key 进行串谋攻击的情况是可以通过查询和检测(审计)区块链不可篡改的分布式账本发现的, 而要尽快发现、解决和避免这种串谋攻击, 需要服务提供商 SP 在进行验证系统设计前, 制定一套内部服务和管理体系, 并在体系中明确, 出现问题时的责任承担者, 其中, 服务和管理体系中, 应该规定零知识证明验证合约的部署者应该使用唯一指定的区块链账号地址、部署后的合法合约的地址应该被唯一确认和指定、合法的合约调用者应该使用唯一指定的区块链账号地址, 这样就可以利用审计机制尽快发现和解决这种攻击, 并利用责任承担者机制减小内部作恶发生。同时, 服务提供商 SP 的服务和管理体系中也应该明确, 合约在部署前, 应该经过严格的合约代码审查和责任认定, 以从源头最小化恶意的 verification key 引起的串谋攻击发生。

最后, 基于智能合约的公开透明验证是有意义的, 是必要的。

第一, 相比于非智能合约的零知识证明验证而言, 基于智能合约的零知识证明的验证是在分布式环境中进行的, 不仅可以克服单点故障问题, 而且验证过程中的行为操作可以记录在不可篡改的区块链中, 安全风险更小, 更有利于避免基于错误的 verification key 进行的串谋攻击。具体就是, 在不基于智能合约的零知识证明的验证中, 验证者对零知识证明的验证大部分是在中心化的服务器中完成的, 这不仅存在单点故障风险, 而且验证者中的恶意共谋者在生成错误的 verification key 后, 首先修改授权验证系统中的验证程序和操作行为记录日志, 接着通过非公开信道透露给共谋的恶意证明者 verification key, 恶意证明者基于获得错误的 verification key 后, 可以构造恶意的零知识证明 zk_proof, 并提交给服务提供商的被修改了的验证程序, 而此时, 验证程序和行为记录日志已经被修改(传统的行为记录日志系统不具备区块链分布式账本的不可篡改特性, 对其进行修改相对容易), 因此, 恶意的零知识证明 zk_proof 就可以通过验证, 而且这个错误验证由于行为日志的修改无法很快被发现和解决。

第二, 基于区块链的链上操作和验证可以保证可信, 并增加服务提供商的服务透明度和公平性, 而且当前的众多基于区块链的应用研究都提出利用基于区块链智能合约的链上操作和验证解决可信和公平问题, 但是其中的部分研究内容由于不熟悉链上验证中的基于椭圆曲线加法和乘法以及双线性配对的可实现性, 而本文的合约代码是公开的, 可以为这些研究提供技术参考。

当前, 在区块链的应用研究中, 利用智能合约完成链上操作的研究越来越多, 比如文献[36]利用智能合约构建了安全可信的分布式物联网访问控制系统, 文献[37]~[39]利用区块链智能合约对农产品上链溯源信息进行链上检测和校对, 以保证检测和校对过程安全可信等。特别地, 文献[40]提出利用智能合约完成可验证加密签名算法中的双线性配对等操作的链上验证, 以取代可信第三方实现双边合同签署的可信和公平, 但是该文的作者们对以太坊区块链的链上双线性配对等操作的实现不了解, 导致最终没有真正意义的实现可验证加密签名的链上验证(仅在链下测试), 而本文的链上验证具备公开可信的特点, 而且合约代码是公开的, 其中就包括了基于椭圆曲线加法和乘法以及双线性配对的链上验证的操作内容的实现, 可以为相关研究提供技术参考。

综上所述, 利用智能合约完成零知识证明的验证虽然需要一定的成本和时间开销(成本在联盟链中可以忽略), 可以解决单点故障问题, 增加验证的可信和透明度, 保证服务授权的公平, 并且能够为相关研究提供技术参考, 因此, 本文的智能合约完成零知识证明的验证从多个方面而言都是有意义和必要的。

6.5 系统安全性分析

系统框架的安全性包括区块链底层的安全以及应用层的安全, 因此将分别对两者进行介绍。

区块链底层的安全:

1) 系统框架的安全: 本文的系统框架基于属性凭证的框架, 该框架的属性凭证和私钥由用户自主持有, 区块链中并不存储任何与属性凭证明文信息相关的内容, 而仅仅使用属性凭证中的哈希值 Sign_H 查看凭证是否激活, 以及使用零知识证明

zk_proof的哈希值proof_H查看零知识证明是否被使用过, 而这些过程都不需要向服务提供商 SP 提交用户凭证的明文信息。

2) 零知识证明的安全: 本文的零知识证明的智能合约和相关参数的生成是基于 ZoKrates 工具集的 G16 算法的, 由本文 2.4 节及文献[31]可知, 该算法是满足零知识证明的完备性、可靠性和零知识性的。

3) 凭证的零知识证明验证过程和结果的安全: 本文的所有验证过程都在区块链上利用智能合约完成, 而智能合约在部署后是不能随意更改的, 合约的执行代码是公开透明的, 而且智能合约的调用和执行过程都将产生交易记录, 方便审计和追溯, 并且 6.4 节对智能合约的链上验证的相关的安全性等进行了深入分析。此外, 本文的零知识验证不仅验证了用户的身份属性是否满足要求、而且验证了用户是否是凭证的持有者, 同时验证了凭证的签名是否合法。

应用层的安全:

1) 凭证属性信息最小化: 本文的系统框架中的可验证凭证仅包含一种属性信息, 用户的多种属性信息可由多个来自不同的身份提供商 IDP 签发的属性凭证证明, 保证属性信息的最小披露。

2) 可防止重放攻击: 尽管零知识证明 zk_proof 可能会因为用户与服务提供商 SP 通信过程中泄露, 或者是被不诚实的服务提供商 SP 透露给其他人, 但是由于每一个 zk_proof 仅可以使用一次, 用户要再次访问服务提供商 SP 可以生成新的 zk_proof。

3) 零知识规则由服务提供商SP制定: 零知识证明验证规则由 SP 根据服务的属性需求制定, 且验证智能合约也由其部署, 保证服务授权自主和安全。

6.6 相关系统对比

最后, 为了从具体指标上说明本文系统框架的优势, 本文将前文提到的部分系统框架与本文系统框架进行了一个比较(从表 1 中抽取不同架构的典型代表, 并增加了最传统的 CA 架构), 比较结果如表 6 所示, 从表中可以看到本文系统框架具备显著的优势特点。需要特别强调的是, 本文架构与文献[19]的架构特征类似, 但是本文架构在进行零知识证明时, 证明了用户是凭证的持有者, 兼具隐私和安全的特性, 然而, 这种特性是以消耗更多的以太币为代价的(文献[19]的架构执行零知识证明的以太币的消耗为 0.00462ether, 而从本文的 6.1 节可以看到, 本文的消耗为 0.075ether), 因此, 在利用区块链设计隐私保护系统时, 安全与成本之间的平衡是一个需要考虑的问题(联盟链可以不考虑成本)。

表 6 本文系统架构与相关架构的比较

Table 6 Comparisons between the proposed architecture and other architectures

名称	架构	去中心化	属性信息的隐私保护(零知识披露)	属性信息最小化	智能合约链上验证	隐私和安全双重保障
CA	传统架构	×	×	×	×	×
Certcoin[4]	基于分布式账本	√	×	×	×	×
BlockCAM[7]	基于分布式账本	√	×	×	×	×
UPort[8]	基于智能合约	√	×	×	×	×
EverSSDI[10]	基于智能合约	√	×	√	×	×
SCPKI[11]	基于智能合约	√	×	×	×	×
TBDID[15]	基于可验证凭证	√	×	×	×	×
CloudDID[17]	基于可验证凭证	√	×	√	×	×
文献[19]	基于可验证凭证	√	√	√	√	×
本文	基于可验证凭证	√	√	√	√	√

7 总结与展望

数字身份的安全关乎着互联网和网络空间的安全, 而传统的数字身份认证和管理系统存在中心化、自主权特性差、隐私无保障、认证过程不公开透明等问题, 基于区块链的数字身份认证和管理系统为解决上述传统系统存在的问题提供了途径。当前, 基于区块链的数字身份认证和管理系统主要有三种架构, 相比于其他两种架构, 基于可验证凭证的架构具备更好的用户身份自主权特性、区块链可伸缩性等, 但是当前大部分的基于可验证凭证的架构都存在隐私安全问题, 对此, 本文提出了一个基于零知识证明的可验证凭证数字身份认证和管理架构, 架构中的零知识证明的约束条件验证程序由服务提供商 SP 按照服务所需的身份属性生成, 用户基于该程序生成零知识证明, 并将证明提交给服务提供商 SP, 由服务提供商SP在区块链智能合约中完成零知识证明的链上可信验证, 因此, 架构兼具隐私和安全等

特性。进一步地, 本文基于架构的实际用例对架构进行了测试、分析和比较, 结果表明, 本文架构优势显著, 能够为未来设计基于区块链的数字身份认证和管理系统提供有价值的技术参考。

近年来, 各国相继出台了相应的法律和法规, 以强化对个人数字身份的安全和隐私保护, 例如欧盟在2018 年 5 月颁布实施了《通用数据保护条例》[41], 以加强欧盟境内数字身份的安全和隐私保护, 我国在2021 年 8 月通过了《中华人民共和国个人信息保护法》[42], 以明确数字身份保护的重要性, 并规范数字身份处理者的行为等。特别地, 在《中华人民共和国个人信息保护法》中, 强调了数字身份处理者不得随意通过自动决策方式分析数字身份所有者的行为习惯、兴趣爱好等。通常情况下, 数字身份处理者是通过对数字身份所有者的身份标识符进行大数据分析和决策来获得数字身份所有者的行为习惯、兴趣爱好等的, 而在本文系统架构中, 尽管使用了零知识证明隐藏了数字身份所有者的身份属性信息, 但是其身份标识符 User DID 是以明文方式提交给服务提供商的, 服务提供商很容易基于该标识符进行用户行为的分析, 因此, 为了尽可能规范服务提供商行为, 本文未来的工作将同时实现数字身份所有者的身份属性信息和身份标识符的零知识证明, 而对于身份标识符的零知识证明, 可以将身份标识符与一个随机数ε进行哈希运算 Hash(User DID || ε), 并将该运算结果作为用户可验证凭证中的身份标识符项, 以隐藏真实 User DID, 之后用户只需要证明其知道哈希运算结果的输入原像即可在不透露真实 User DID 情况下证明其是 User DID 的持有者。但是需要注意, 当用户真实身份标识符 User DID 被隐藏后, 其对应的实名者就很难从身份提供商 IDP 查询到, 因此, 隐私和安全之间的比重对于不同的系统应用应该特殊考虑。

参考文献

[1] Zhou T. Research and application of tokenization method of trusted data based on blockchain technology[D]. Hefei: University of Science and Technology of China, 2019.
(周桐. 基于区块链技术的可信数据通证化方法的研究与应用[D]. 合肥: 中国科学技术大学, 2019.)

[2] Yang K, Wang J S. The Reasearch of Internet Identity System Based on eID and Personal Information Protection Legal System[J]. Journal of Information Security Research, 2019, 5(5): 440-447.
(杨珂, 王俊生. 基于 eID 的网络身份制与个人信息保护法律制度研究[J]. 信息安全研究, 2019, 5(5): 440-447.)

[3] Jaroucheh Z, Álvarez I A, Communication N A B T, et al. Secretation: toward a decentralised identity and verifiable credentials based scalable and decentralised secret management solution[C]. 2021 IEEE International Conference on Blockchain and Cryptocurrency, 2021: 1-9.

[4] Fromknecht C, Velicanu D, and Yakoubov S. CertCoin: A namecoin based decentralized authentication system 6.857 class project. https://courses.csail.mit.edu/6.857/2014/files/19-fromknecht-velicann-yakoubov-certcoin.pdf. May. 2014

[5] Wang Z, Lin J Q, Cai Q W, et al. Blockchain-Based Certificate Transparency and Revocation Transparency[J]. IEEE Transactions on Dependable and Secure Computing, 2022, 19(1): 681-697.

[6] Crosby M, Pattanayak P, Verma S, et al. Blockchain technology: Beyond bitcoin[J]. Applied Innovation, 2016, 2(6-10): 71

[7] Wang W T, Hu N, Liu X, et al. BlockCAM: A blockchain-based cross-domain authentication model[C]. 2018 IEEE Third International Conference on Data Science in Cyberspace, 2018: 896-901.

[8] Lundkvist C, Heck R, Torstensson J, et al. Uport: a platform for self-sovereign identity. https://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf. Oct. 2016.

[9] Diebold Z. Self-Sovereign Identity using Smart Contracts on the Ethereum Blockchain[D]. Dublin: University of Dublin, Trinity College, 2017.

[10] Zhao H, Zhou T, Li X F. EverSSDI: Blockchain-Based Framework for Verification, Authorisation and Recovery of Self-Sovereign Identity Using Smart Contracts[J]. International Journal of Computer Applications in Technology, 2019, 60(3): 281.

[11] Al-Bassam M. SCPKI: A Smart Contract-Based PKI and Identity System[C]. The ACM Workshop on Blockchain, Cryptocurrencies and Contracts, 2017: 35-40.

[12] Manu S, Dave L, and David C. Verifiable Credentials Data Model 1.0 - Expressing verifiable information on the Web. https://www.w3.org/TR/vc-data-model. Nov. 2021.

[13] Fan X X, Chai Q, Xu L, et al. DIAM-IoT: A Decentralized Identity and Access Management Framework for Internet of Things[C]. The 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure, 2020: 186-191.

[14] Kortesniemi Y, Lagutin D, Elo T, et al. Improving the Privacy of IoT with Decentralised Identifiers (DIDs)[J]. Journal of Computer Networks and Communications, 2019, 2019: 1-10.

[15] Aydar M, Ayvaz S, Cetin S C. Towards a Blockchain Based Digital Identity Verification, Record Attestation and Record Sharing System[EB/OL]. 2019: arXiv: 1906.09791. https://arxiv.org/abs/1906.09791

[16] Wang F N, de Filippi P. Self-Sovereign Identity in a Globalized World: Credentials-Based Identity Systems as a Driver for Economic Inclusion[J]. Frontiers in Blockchain, 2020, 2: 28.

[17] Baidu. Baidu Cloud DID Method. http://did.baidu.com/. Feb. 2020.

[18] Belchior R, Putz B, Pernul G, et al. SSIBAC: self-sovereign identity based access control[C]. 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications, 2021: 1935-1943.

[19] Li Q N, Xue Z H. A Privacy-Protecting Authorization System Based on Blockchain and Zk-SNARK[C]. The 2020 International Conference on Cyberspace Innovation of Advanced Technologies,

2020: 439-444.

[20] Wood G. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151(2014): 1-32.

[21] Androulaki E, Barger A, Bortnikov V, et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains[C]. The Thirteenth EuroSys Conference, 2018: 1-15.

[22] Cao Z H, Zhao L. A Design of Key Distribution Mechanism in Decentralized Digital Rights Management Based on Blockchain and Zero-Knowledge Proof[C]. ICBCT '21: 2021 The 3rd International Conference on Blockchain Technology, 2021: 53-59.

[23] Yang X H, Li W J. A Zero-Knowledge-Proof-Based Digital Identity Management Scheme in Blockchain[J]. Computers & Security, 2020, 99: 102050.

[24] Yuan H X, Liu B X, Kan H B, et al. Distributed Public Key Infrastructure Scheme Based on Blockchain and Decentralized Undeniable Attribute-Based Signature[J]. Scientia Sinica, 2022, 52(6): 1135-1148.
(袁和昕, 刘百祥, 阚海斌, 等. 基于区块链和去中心不可否认属性签名的分布式公钥基础设施方案[J]. 中国科学: 信息科学, 2022, 52(6): 1135-1148.)

[25] Feige U, Fiat A, Shamir A. Zero-Knowledge Proofs of Identity[J]. Journal of Cryptology, 1988, 1(2): 77-94.

[26] Lesavre L, Varin P, Mell P, et al. A Taxonomic Approach to Understanding Emerging Blockchain Identity Management Systems[EB/OL]. 2019: arXiv: 1908.00929. https://arxiv.org/abs/1908.00929

[27] Ben Sasson E, Chiesa A, Garman C, et al. Zerocash: decentralized anonymous payments from bitcoin[C]. 2014 IEEE Symposium on Security and Privacy, 2014: 459-474.

[28] Ben-Sasson E, Chiesa A, Tromer E, et al. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture[C]. The 23rd USENIX conference on Security Symposium, 2014: 781-796.

[29] Parno B, Howell J, Gentry C, et al. Pinocchio: nearly practical verifiable computation[C]. 2013 IEEE Symposium on Security and Privacy, 2013: 238-252.

[30] Groth J, Maller M. Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs[M]. Advances in Cryptology - CRYPTO 2017. Cham: Springer International Publishing, 2017: 581-612.

[31] Groth J. On the Size of Pairing-Based Non-Interactive Arguments[M]. Advances in Cryptology - EUROCRYPT 2016. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016: 305-326.

[32] Guan Z S, Wan Z G, Yang Y, et al. BlockMaze: An Efficient Privacy-Preserving Account-Model Blockchain Based on Zk-SNARKs[J]. IEEE Transactions on Dependable and Secure Computing, 2022, 19(3): 1446-1463.

[33] Gailly N, Maller M, Nitulescu A. SnarkPack: Practical SNARK Aggregation[M]. Financial Cryptography and Data Security. Cham: Springer International Publishing, 2022: 203-229.

[34] Eberhardt J, Tai S, Communication N A B T, et al. ZoKrates - scalable privacy-preserving off-chain computations[C]. 2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data, 2019: 1084-1091.

[35] Alsayed Kassem J, Sayeed S, Marco-Gisbert H, et al. DNS-IdM: A Blockchain Identity Management System to Secure Personal Data Sharing in a Network[J]. Applied Sciences, 2019, 9(15): 2953.

[36] Du R Z, Liu Y, Tian J F. An Access Control Method Using Smart Contract for Internet of Things[J]. Journal of Computer Research and Development, 2019, 56(10): 2287-2298.
(杜瑞忠, 刘妍, 田俊峰. 物联网中基于智能合约的访问控制方法[J]. 计算机研究与发展, 2019, 56(10): 2287-2298.)

[37] Xu J P, Wang J, Zhang X, et al. Information Supervision Modeling of Rice Supply Chain Driven by Blockchain[J]. Transactions of the Chinese Society for Agricultural Machinery, 2021, 52(5): 202-211, 101.
(许继平, 王健, 张新, 等. 区块链驱动的稻米供应链信息监管模型研究[J]. 农业机械学报, 2021, 52(5): 202-211, 101.)

[38] Xu J P, Sun P C, Zhang X, et al. Prototype System of Information Security Management of Cereal and Oil Food Whole Supply Chain Based on Blockchain[J]. Transactions of the Chinese Society for Agricultural Machinery, 2020, 51(2): 341-349.
(许继平, 孙鹏程, 张新, 等. 基于区块链的粮油食品全供应链信息安全管理原型系统[J]. 农业机械学报, 2020, 51(2): 341-349.)

[39] Ge Y, Huang C L, Chen M, et al. HACCP Quality Traceability Model and System Implementation Based on Blockchain[J]. Transactions of the Chinese Society for Agricultural Machinery, 2021, 52(6): 369-375.
(葛艳, 黄朝良, 陈明, 等. 基于区块链的 HACCP 质量溯源模型与系统实现[J]. 农业机械学报, 2021, 52(6): 369-375.)

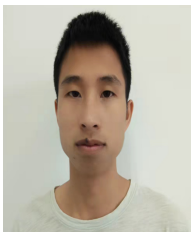
[40] Zhang L, Zhang H L, Yu J, et al. Blockchain-Based Two-Party Fair Contract Signing Scheme[J]. Information Sciences, 2020, 535: 142-155.

[41] Voigt P, Von dem Bussche A. The eu general data protection regulation (gdpr)[J]. A Practical Guide, 1st Ed., Cham: Springer International Publishing, 2017, 10: 3152676.

[42] Personal Information Protection Law of the People's Republic of China[J]. Gazette of the Standing Committee of the National People's Congress of the People's Republic of China, 2021(6): 1117-1125.
(中华人民共和国个人信息保护法[J]. 中华人民共和国全国人民代表大会常务委员会公报, 2021(6): 1117-1125.)

宋智明 于 2020 年在中国科学院大学天文技术与方法专业获博士学位。现任云南财经大学信息学院(智能应用研究院)副教授。研究领域为基于区块链的信息安全应用。研究兴趣包括区块链应用、零知识证明、数字身份等。Email: 339255245@qq.com

余益民 于 2010 年在成都电子科技大学获计算机软件与理论博士学位。现任云南财经大学信息学院(智能应用研究院)教授, 院长。研究领域为跨境信息共享与安全、数字经济。研究兴趣包括区块链跨境身份管理、跨境电子认证等。Email: yym@ynufe.edu.cn



王贵文 于 2015 年在杭州电子科技大学获学士学位。现在云南财经大学信息学院攻读硕士学位。研究领域为计算机安全应用、公钥密码学。研究兴趣包括: 区块链主、计算机安全应用软件设计等。Email: 1850392020@qq.com

陈韬伟 于 2010 年在西南交通大学获计算机应用技术博士学位。现任云南财经大学信息学院教授。研究领域为公钥密码学、密码学与区块链结合的安全系统构建。研究兴趣包括区块链应用、人工智能等。Email: 11919007@qq.com