计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202330268
60(11): 2555−2566, 2023

基于区块链和可信执行环境的属性签名身份认证方案

冉津豪　蔡栋梁
（复旦大学计算机科学技术学院　上海　200433）
（上海区块链工程技术研究中心（复旦大学）　上海　200433）
（复旦大学义乌研究院　浙江义乌　322000）
（22110240060@m.fudan.edu.cn）

Attribute Signature Identity Authentication Scheme Based on Blockchain and Trusted Execution Environment

Ran Jinhao and Cai Dongliang
（*School of Computer Science*, *Fudan University*, *Shanghai* 200433）
（*Shanghai Engineering Research Center of Blockchain* (*Fudan University*), *Shanghai* 200433）
（*Yiwu Research Institute*, *Fudan University*, *Yiwu*, *Zhejiang* 322000）

Abstract　Identity authentication is a technology widely used in the current digital world. In the era of traffic supremacy, a secure and convenient identity authentication solution is crucial for attracting users to application services. Decentralized identity gives users complete control over their identity by using a fully decentralized technology such as blockchain. In order to further improve the security and convenience of identity authentication, an attribute signature authentication scheme based on blockchain and trusted execution environment is proposed. Existing identity verification methods have problems such as heavy management of user identity certificates and insufficient security. Attribute signatures are used by users to generate persistent credentials pointing to application services, and credentials are extensible. In the process of repeatedly expanding credentials, the user is more likely to be implanted with a Trojan horse than a single generation of credentials. The trusted execution environment can provide hardware-level protection during the signing process to avoid the leakage of intermediate parameters. At the same time, the audit of user identity leakage and fraudulent use is realized with a small additional verification cost, which further improves the security of the scheme.

Key words　blockchain; trusted execution environment(TEE); attribute signature; decentralized identity; zero-knowledge proof

摘　要　身份认证是当前数字化世界中广泛应用的一项技术，对于流量至上的时代，安全而便捷的身份验证方案对于应用服务吸引用户是至关重要的.去中心化身份通过使用完全去中心化的技术如区块链，让用户完全控制自己的身份.为了进一步提高身份认证的安全性与便捷性，提出一种基于区块链和可信执行环境（TEE）的属性签名身份认证方案.现有的身份验证方法存在用户身份凭证管理繁重、安全性不足

收稿日期：2023-04-03；**修回日期**：2023-06-06
基金项目：国家重点研发计划（2019YFB2101703）；国家自然科学基金项目（62272107, U19A2066）；上海市科技创新行动计划（21511102200）；广东省重点领域研发计划（2020B0101090001）
This work was supported by the National Key Research & Development Program of China (2019YFB2101703), the National Natural Science Foundation of China (62272107, U19A2066) , the Innovation Action Plan of Shanghai Science and Technology (21511102200), the Key-Area Research and Development Program of Guangdong Province (2020B0101090001).
通信作者：蔡栋梁（22110240060@m.fudan.edu.cn）

等问题. 用户利用属性签名生成指向应用服务的持久性凭据, 并且凭据是可扩充的. 而用户反复扩充凭据的过程相比单次生成凭据保存更有被攻击者植入木马的风险, 可信执行环境则可以在签名过程中提供硬件级别的保护, 避免中间参数的泄露. 同时还用较小的额外验证代价实现了对用户身份泄露、冒用的审计, 进一步提高了方案的安全性.

关键词 区块链; 可信执行环境; 属性签名; 去中心化身份; 零知识证明

中图法分类号 TP399

2022 年, 中央网络安全和信息化委员会办公室等十六部门联合通知, 公布了 15 个综合性和 164 个特色领域国家区块链创新应用试点名单[1], 区块链应用正在逐步进入常人的生活.

区块链是新一代信息技术的重要组成部分, 在过去十余年里, 部分学者称区块链技术为分布式系统领域革命性的技术[2]. 中本聪[3] 的一种点对点的现金系统首次展现了区块链的应用形式, 他通过将 Haber 和 Stornetta 提出的无信任时间戳的概念[4] 应用到去中心化的环境中, 并结合工作量证明[5-6], 建立了比特币共识协议. 工作量证明协议将计算资源总和超过 50% 的节点共识得到的结果作为正确的结果记录下来, 实现最终的一致性[7]. 同时, 区块链中的区块将通过哈希成链, 保证了链上内容不可篡改的特性[6].

区块链的共识机制让许多问题有了新的解决思路, 其中一个重要的区块链应用就是身份认证. 由于互联网应用及基础设施的发展, 越来越多的人接入互联网生活[8]. 随着对数字身份的需求不断增长, 人们生活的很大一部分将在网络中与服务进行交互, 因此身份认证管理也成为研究的热点. 简单来说, 数字身份是人们通过电子方式向应用服务证明自己身份的一种手段.

分布式的身份认证方案通常可以表述为去中心化身份(decentralized identity, DID), 或者是自我主权身份(self-sovereign identity, SSI)[9], 本文不对这两者区分, 下文统一称为 SSI. 从本质上讲, SSI 是一个允许用户个人完全拥有和管理他们自我数字身份的系统, 主张用户应独立于应用服务存在, 用户的个人数字身份和个人信息完全由个人所拥有和控制, 用户可以自行构建属于自己的可验证凭据(verifiable credentials, VC), 而不需要通过中心化的第三方, 如应用服务、中心机构等.

可信执行环境(trusted execution environment, TEE)是指在计算设备中与操作系统隔离的一块独立、安全、可信的软件执行环境, 从硬件级别保障隐私、敏感数据计算的机密性、完整性. 此方案中, 用户需要在本地环境中计算 VC 及零知识凭据, 而个人用户很难具备较好的安全防护意识, 其专业知识和习惯常常不具有为该计算提供安全防护的能力, 让敌手有机可乘. 因此, 为客户端程序引入 TEE 技术是必要的.

本文的主要贡献包括 3 个方面:

1)提出了一种去中心属性签名[10] 结合零知识证明的 SSI 方案. 区块链公开透明以及不可篡改的特点将约束管制应用机构的行为. 用户可通过身份标识, 注册特定应用服务的认证权限, 自主生成 VC. 同时, 为应对应用服务数量增加的场景, 本文方案优化了用户重复生成 VC 过程中的额外开销, 用户的 VC 可扩充生成, 提高了方案的效率.

2)为用户生成 VC 结合 TEE 的方法. 考虑到普通用户的环境安全性不足的问题, 将用户的 VC 生成过程置于 TEE 中, 降低 VC 生成及多次 VC 扩充过程中硬件级别风险, 提高用户私钥的安全性.

3)设计了一种用户 VC 冒用审计机制. 身份被窃取并冒用的过程是难以察觉的, 这一机制将使得冒用过程有迹可循. 主要依赖将验证过程中对包含验证次数的计数哈希的内容记录到区块链, 用较小的验证代价完成 VC 的冒用审计.

1 相关工作

本文的研究对象是基于区块链和 TEE 的属性签名身份认证方案, 本节将介绍身份认证方案及 TEE 的相关研究现状.

1.1 身份认证方案

中心化身份、联邦身份和自我主权身份是系统身份认证发展的 3 个阶段[9, 11]. 中心化身份由应用服务直接控制身份, 用户直接使用用户名及密码登录. 这种方法模式简单、使用方便, 且大量的用户数据由应用服务直接掌控, 可以被应用服务分析优化服务内容, 在今天仍然是大部分应用服务的身份认证手段. 然而, 身份数据的集中保存是一把双刃剑, 首先是用户隐私难以保证, 用户数据泄露问题时有发生[12],

且随着应用服务数量的增多，用户的密码管理负担也会更重. 用户习惯对不同应用设置相同密码，当相同的密码泄露时，大量应用的身份安全将受到威胁. 联邦身份主要由少量的身份提供者为用户提供数字身份，登录到其他应用. 文献 [13] 提出了一种联邦身份架构，允许用户跨安全域动态分发身份信息，从而提高数字身份的可移植性. 文献 [14] 利用安全断言标记语言进行联邦身份管理，降低用户管理成本，还进行了完整的安全性证明. OAuth2.0 协议[15] 帮助第三方应用向存有用户身份信息的服务提供商按需获取用户的授权，整个过程是安全且用户可控的. 国内应用常见的有通过微信、支付宝登录的登录方式，国外应用也有通过谷歌、微软登录的登录方式. 智能手机设备的普及为联邦身份提供了条件，但身份提供者能跟踪用户登录行为，可以拒绝为用户提供身份，甚至冒用用户身份登录到第三方应用服务，用户身份仍然不能自主控制.

自我主权身份方案中，用户将利用密码学的手段，通过管理与个人身份认证有关的私钥信息，实现对自我身份的掌控. 文献 [] 的 uPort 协议使用以太坊智能合约地址作为用户身份标识符，文献 [17] 的 Blockstack 通过使用 virtualchains 以去中心化的方式解决了信任引导的问题，即网络上的新节点可以独立地验证所有数据绑定. Candid[18] 用智能合约存储具有隐私保护的身份数据，用户可自主控制身份属性披露与否，同时还通过连接外部存储来扩大属性容量. 文献 [19] 虽然与分布式公钥基础设施相关，但本质上也具有身份认证功能. 虽然文献 [16−19] 所述的方案都满足了用户身份的分布式特点，但都忽略了用户自主保存私钥过程中的风险. 一旦窃取到用户私钥，其他用户都可以伪造该用户身份登录到应用服务，且这个过程是用户不可察觉的. 文献 [20] 的方案在登录时，如果用户身份被冒用则进行过身份验证，用户可以收到反馈，但该方案缺少对于应用的扩展支持，并且需要在用户环境下执行隐私风险较高的秘钥计算过程.

中心化身份、联邦身份和自我主权身份对比内容如表 1 所示. 其中，考虑到联邦身份方案中，管理用户数字身份的机构通常为互联网巨头公司，具有良好的安全防护能力，且公司的形象驱使它们尽力做好安全防护，所以安全风险较低. 而中心化身份方案中有许多体量较小的公司，没有能力或没有侧重于安全防护，并且由于用户可能倾向于多个应用设置相同密码，所以安全风险较高.

Table 1 Comparison of Centralized Identity, Federated Identity and Self-Sovereign Identity

表 1 中心化身份、联邦身份和自我主权身份对比

特征	中心化身份	联邦身份	自我主权身份
用户管理登录凭证数量	多	较少	唯一
密钥管理主动权归属	应用服务	身份提供者	用户
数字身份保存集中程度	分散	集中	分散
安全风险	高	低	中

1.2 可信执行环境

Intel 软件防护扩展（software guard extensions, SGX）[21] 技术、AMD 内存加密（memory encryption）技术和 ARM 的 TrustZone 技术是各大平台具有代表性的 TEE 技术. 在本文方案中，我们选择普及率更高的 Intel SGX 技术.

Intel SGX 是 x86 架构的扩展，允许用户级代码创建名为飞地（enclaves）[21-22] 的 TEE. SGX 通过 ECalls 和 OCalls 进行飞地的调入和调出，Ecalls 是从飞地外部调用飞地内部的可信函数，而 OCalls 则相反. 飞地的调入和调出将会跨越安全边界，并对安全参数进行检查，因此会产生额外的性能开销. 文献 [23] 测量了飞地的调入和调出取决于缓存命中或错过需要 8 600~14 000 个周期.

TEE 常常被应用于分布式节点计算场景中. TC[24] 使用 SGX 完成 MapReduce 分布式计算，同时保持相应的代码和数据的隐私性、正确性和完整性. T-Counter[25] 基于 SGX 设计了 CPU 资源耗费计数框架，保证云服务中的计算资源耗费准确，防止恶意云服务提供商虚报资源消耗. 在身份认证方向上，Town crier[26] 是在 2015 年便提出的用链下 TEE 为智能合约提供数据的方法. 文献 [27] 用 SGX 及区块链构建了身份认证方案，但方案中仍存在集中了大量用户身份信息的身份提供者，本质上仍是联邦身份方案. 文献 [28] 则用了 ARM 的 Trustzone 保障用户数据的完整性和机密性.

2 预备知识

2.1 ElGamal 加密[29]

G是一个阶为素数 p的乘法循环群，g是群G的生成元，对于一任意群内的元素 $y \in G$，一定存在一个唯一的$x \in \mathbb{Z}_q$，使得等式$g^x = y$成立. ElGamal 加密将分成 3 个步骤：

1）生成密钥. 选择一个随机数$x \in \mathbb{Z}_p$，计算$g^x = y$，

则 ElGama 加密的公钥为(y, g, p)，私钥为x.

2）加密. 对于消息m，随机选择$r \in \mathbb{Z}_p$，计算$c_1 = g^r$，$c_2 = m \times y^r$，用c_1和c_2构建密文对$\langle c_1, c_2 \rangle$.

3）解密. 对于密文对$\langle c_1, c_2 \rangle$，利用私钥x，计算$m = c_2 / c_1^x$.

2.2 Sigma 协议[30]和 Schnorr 协议[31]

设$R \subseteq X \times Y$是一个基于 NP 困难问题的关系，那么一对(P, V)构建在上的一个 Sigma 协议为：

1）P是证明者算法，输入为一对$(x, y) \in R$，V是验证者算法，输入为$y \in Y$；

2）P计算一个承诺C，将其发送给V；

3）V在接收到P发送的承诺C后，随机生成一个挑战值c，并发送给P；

4）在接收到来自V发送的挑战值c后，P根据c计算一个响应值z返回给V；

5）V通过验证输入y、承诺C和交互中的挑战值c、响应值z后，输出接受或拒绝.

Sigma 协议具有 3 个性质：

1）正确性. 在公共输入y上，如果诚实的证明者P得到的输入x使得$(x, y) \in R$，那么验证者V总是接受.

2）可靠性. 给定一个公共输入y，以及V关于y输出接受的 2 个会话$(C, c, z), (C, c', z')$,其中$c \neq c'$，那么任何人都可以计算出x的值，使得$(x, y) \in R$.

3）零知识性. 存在一个高效的概率性算法 Sim（称作模拟器），对于所有的输入$(y, c) \in R$，其中S为挑战值空间，Sim 能够输出一个可接受的会话(y, C, c, z)，并且该会话与诚实的P和V之间的会话分布相同.

Schnorr 协议是 Sigma 协议中的一种. G是一个阶为素数q的循环群，其生成元为$g \in G$，假设一个证明者想要证明它知道某个群元素$h = g^x \in G$的离散对数x. 这里$R = \{(x, h) \in \mathbb{Z}_q \times G : g^x = h\}$. 如图 1 所示，Schnorr 协议的交互过程为：

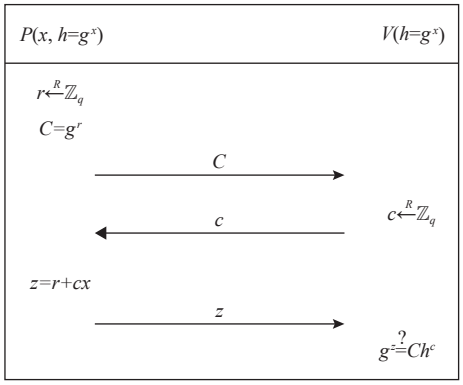

Fig. 1 The interactive flow of Schnorr protocol[31]

图 1 Schnorr 协议交互流程[31]

1）P的输入为$x, h = g^x$，V的输入为h；

2）P选择$r \in_R \mathbb{Z}_q$，计算承诺$C = g^r$发送给V；

3）V选择挑战$c \in_R Z_q$发送给P；

4）P根据挑战c,计算响应值$z = r + cx$返回给V；

5）V通过验证$g^z = Ch^c$是否成立输出结果，若成立则输出接受，否则输出拒绝.

使用 Fiat-Shamir 转换[32]，证明者P能够通过$H(h, C)$计算挑战值c，其中H是哈希算法，h是V的输入，C是P的承诺，因此 Schnorr 协议可以变为非交互式的，只需要一个通信轮次. 证明者可以通过承诺值C、挑战值c以及响应值z直接构造$proof = (h, C = g^r, c = H(h, C), z = r + cx)$，将$proof$发送给$V$，$V$可以通过验证$proof$直接输出接受或拒绝.

2.3 离散对数相等（discrete logarithm equality，DLEQ）零知识证明协议

$DLEQ(g, G, t, T)$用于证明离散对数$\log_g G = \log_t T$，当等式成立时验证者输出接受.

证明者P的输入为$s \in Z_q$，计算$G = g^s$，$T = t^s$，向验证者证明$\log_g G = \log_t T$. 验证者的输入为(g, G, t, T). 协议交互流程为：

1）P选择随机数$r \in \mathbb{Z}_p$，计算承诺值$C_1 = g^r$，$C_2 = t^r$并发送给验证者V；

2）V收到(C_1, C_2)后，选择挑战值$c \in \mathbb{Z}_p$发送给P；

3）P根据挑战c计算响应值$z = r + cs$返回给V；

4）V通过验证$g^z = C_1 G^c$和$t^z = C_2 T^c$是否成立输出结果，如果成立输出接受，否则输出拒绝.

2.4 非交互式离散对数相等（non-interactive discrete logarithm equality, NIDLEQ）零知识证明协议

对 DLEQ 协议使用 Fiat-Shamir 转换[32]得到 NIDLEQ 协议，证明者P能够通过$H(C_1, C_2)$计算挑战值c，其中H是哈希算法，C_1和C_2为P的承诺. 证明者可以通过证明内容、承诺值C、挑战值c以及响应值z直接构造$proof = (C_1 = g^r, C_2 = t^r, c = H(C_1, C_2), z = r + cs)$，将$proof$发送给$V$，$V$通过验证$c = H(C_1, C_2)$，$g^z = C_1 G^c$和$t^z = C_2 T^c$是否成立输出结果，如果都成立输出接受，否则输出拒绝.

3 系统模型与安全模型

3.1 系统模型

本文方案由三类通信单位及以太坊区块链组成，通信单位分别为全局控制中心、用户、应用服务. 通信单位间的通信关系如图 2 所示.

1）全局控制中心. 用于全局循环群的素数阶及

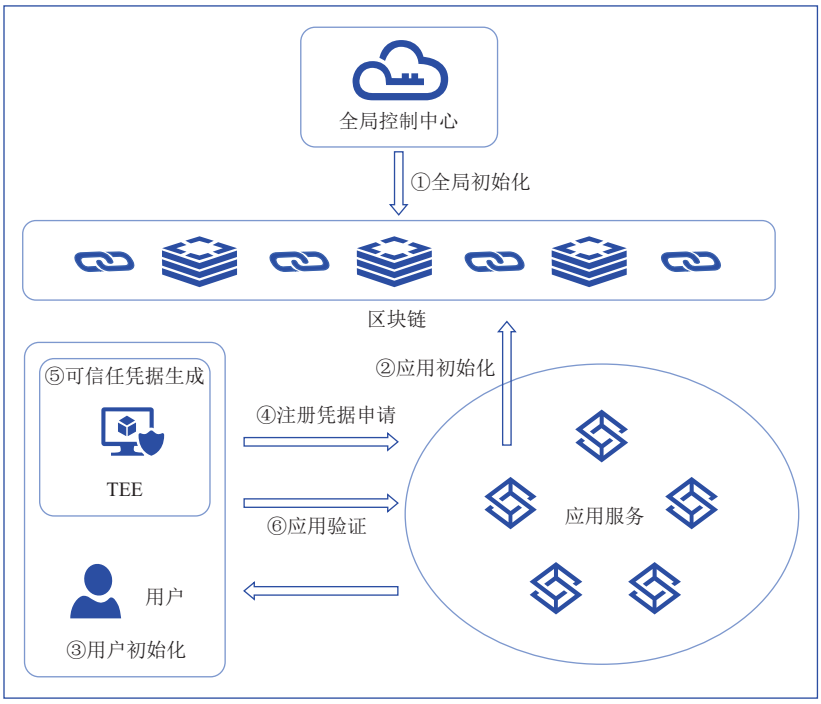

Fig. 2 The system model of our scheme

图 2 本方案系统模型

生成元的生成，并上传到区块链供其他通信单位查询使用. 与已有的身份认证方案的密钥生成中心(key generation center, KGC)不同，常规的密钥生成中心或者权威授权机构可能会承担为用户生成密钥的职责[33-35]. 本文方案的全局控制中心仅用于初始参数生成，参数将全部公开，不存在泄密风险，其工作可以被某一应用服务代替. 本文方案为了更好地展示去中心化的特点，使得各应用程序角色均衡，将该部分内容单独置于全局控制中心.

2) 用户. 应用服务使用者，期望获得自主控制的去中心化身份，能快捷稳定地登录到应用服务中，通常为个人用户. 这类节点单位的特点是操作人员专业性不足、程序执行环境安全性低，因此，面向用户的 VC 生成及验证过程应在保证安全的前提下尽量简洁，用户内部设有 TEE 从硬件级别保护程序执行. 其安全模型将在 3.2 节中详细定义.

3) 应用服务. 应用服务提供者，通过应用初始化加入到身份认证系统中，为用户提供注册凭据申请和应用验证服务.

4) 区块链. 根据区块链去中心化、不可篡改的特点，全局控制中心生成的初始参数、各应用服务公钥、用户登录次数计数哈希都将上传到区块链中供查询和审计，并且在一定程度上可以避免全局控制中心被攻击进而初始参数被修改导致全局故障.

节点行为如图 2 所示. 首先全局控制中心将进行全局初始化生成初始参数. 各应用根据全局参数生成公私钥对，并将初始参数和公钥上传到区块链. 用户首先需要完成初始信息的拟定，根据全局参数生成公私钥对，公钥即为用户标识. 随后，用户向对于需要使用的应用服务提交注册申请，由应用服务提供注册凭证. 用户根据注册凭证即可在 TEE 里生成签名内容，形成 VC. 在用户请求使用某个应用服务，应用服务期望验证用户的身份时，用户需要向应用服务主动出示 VC、非交互零知识证明凭据及登录次数计数哈希 CNT. 验证过程中应用服务将确认以下信息：①确认 VC 有效性；②通过零知识证明凭据确定 VC 使用者与 VC 所有者对应；③验证用户登录次数计数连续性，即用户在上次登录与本次登录的登录次数计数是连续的，并在验证通过后将用户登录记录哈希更新上传到区块链，供用户审计. 对于新增的应用服务，用户可以继续申请注册凭证，并在 TEE 中重新生成签名内容，形成新的可验证凭据.

3.2 安全模型

根据攻击者的执行权限，攻击者可分为第三方用户攻击者 $\mathcal{A}_1$及作恶应用服务攻击者 $\mathcal{A}_2$. 攻击者 $\mathcal{A}_1$ 可以通过执行 5 方面内容获取信息：

1) $\mathcal{A}_1$可以查询所有链上公共可访问信息，包括全局参数、其他应用公钥、验证过程中生成的登录

次数计数哈希.

2) $\mathcal{A}_1$可以根据区块链上的全局参数素数 p、循环群 G 及生成元 g, 选取一个随机的用户私钥, 并计算其公钥.

3) $\mathcal{A}_1$可以提交其全局标识 $GID_{\mathcal{A}_1}$, 向应用服务申请注册凭据. 应用服务为其生成对应应用服务信息的用户属性基密钥及其中间参数, 并传输给 $\mathcal{A}_1$.

4) $\mathcal{A}_1$可以向应用服务发起验证服务请求, 但该过程能收到的信息是有限的, 应用服务仅返回验证通过, 或 VC 可能被冒用.

5) $\mathcal{A}_1$可以获取到已完成应用验证过程的 VC, 本文不讨论该信息的获取过程, 但这是证明重放攻击的前提.

比起攻击者 $\mathcal{A}_1$, 攻击者 $\mathcal{A}_2$作为应用服务可以获取更多的信息, 如可以获得所有曾在该应用服务中进行过应用验证服务的用户 VC.

在我们的属性签名身份认证方案中, 会考虑 3 种类型的攻击:

1) 证明伪造攻击. 指攻击者试图伪造 VC, 被请求应用服务误以为攻击者的身份是有效的, 从而通过用户身份验证请求.

2) 证明重放攻击. 指攻击者获取到了来自用户的一个或多个 VC, 试图重用这些证明以与被请求应用服务进行新的身份验证交互.

3) 信息盗窃攻击. 假设一个应用服务被攻击或者应用服务主动作恶, 试图从接收到的 VC 和区块链上的公共可访问信息中获取经过身份验证的用户身份信息, 进行新的身份验证交互.

Table 2 Symbols and Their Description

表 2 符号及其描述

符号	描述	符号	描述
GP	系统全局参数	m	签名明文信息
Ω	应用服务信息属性合集	σ	属性签名
$AppInfo$	应用服务信息属性	CNT	登录次数计数明文
SK, x	应用服务私钥	$proof_{\mathrm{ZK}}$	零知识证明凭据
PK, y	应用服务公钥	$proof_{CNT}$	身份冒用审计凭据
ASK, k	属性私钥	$r_i, t_i, c_i, d_i, R_i, T_i$	签名中间参数
APK, r	属性公钥	C_{1i}, C_{2i}	零知识证明承诺
USK, s_p	用户私钥	zkc	零知识证明挑战值
GID, pk_p	用户全局标识、用户公钥	z	零知识证明响应值
AK, e	用户属性基密钥	w_{last}	上轮验证中的 ElGamal 加密随机参数
$Param$, $param$	属性基密钥中间参数	w	ElGamal 加密随机参数
H_1, H_2, H_3, H_4	哈希函数	h_{User_p}	登录次数计数哈希

4 方案设计

本节主要提出完整的属性签名身份认证方案. 该方案分为 6 个阶段: 全局初始化、应用初始化、用户注册、VC 生成、VC 扩充及应用验证流程. 该方案的首次生成时流程及扩充生成时流程如图 3 所示. 用户首次生成 VC 时将进行 VC 生成流程, 否则用户将进行 VC 扩充流程. 用户首次执行 VC 生成流程时, 将保留属性签名的中间参数, 在此情况下, 用户可以在执行 VC 扩充流程时提高效率.

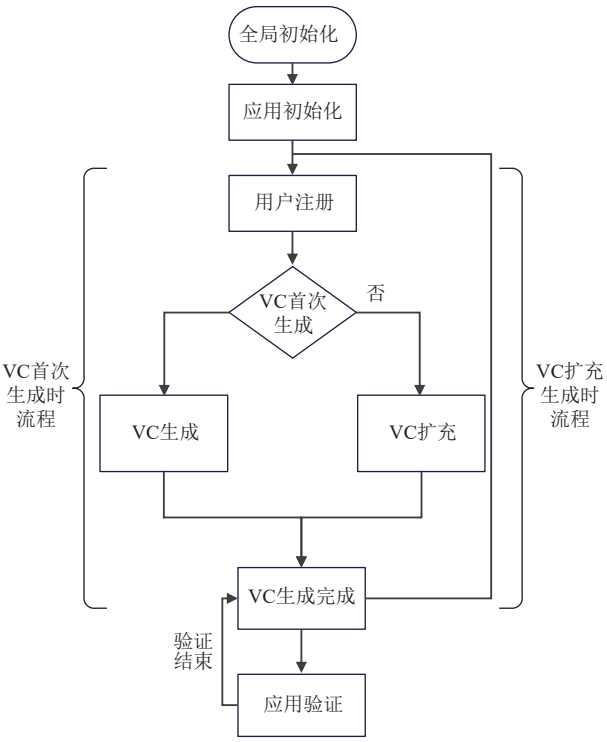

Fig. 3 Flowchart of our scheme

图 3 本方案流程图

4.1 全局初始化阶段

$GlobalSetup(\lambda)\rightarrow GP$: 全局控制中心根据安全参数$\lambda$, 生成全局参数. 采用素数阶双线性映射, 选择一个阶数为素数 p 的循环群 G 及其生成元 g. 记应用服务对应的应用服务信息属性合集为$\Omega=\{AppInfo_1, AppInfo_2, \cdots, AppInfo_n\}$. 选择 4 个哈希函数$H_1:\{0,1\}^*\times G^3\rightarrow\mathbb{Z}_p^*$, $H_2:\{0,1\}^*\times G^{n+1}\rightarrow\mathbb{Z}_p^*$, $H_3:G\times G\rightarrow\mathbb{Z}_p^*$, $H_4:G^3\rightarrow\mathbb{Z}_p^*$. 将安全参数$\lambda$、素数 p 的循环群 G 及其生成元 g、应用服务信息Ω和相关哈希函数上传到区块链, 供所有应用及用户使用.

4.2 应用初始化

$AppSetup(GP)\rightarrow SK_i, PK_i, ASK_i, APK_i$: 每个应用

服务App_i选取自己的私钥$SK_i \leftarrow x_i \in_R \mathbb{Z}_p^*$，公钥$PK_i \leftarrow y_i = g^{x_i}$. 对于属于自己管理的应用服务信息$AppInfo_i$，选择属性私钥$ASK_i \leftarrow k_i \in_R \mathbb{Z}_p^*$和属性公钥$APK_i \leftarrow r_i = g^{k_i}$. 由$App_i$保存私钥$SK_i$和属性私钥$ASK_i$，并将公钥$PK_i$和属性公钥$APK_i$上传到区块链公开.

$$\begin{cases} SK_i \leftarrow x_i \in_R \mathbb{Z}_p^*, \\ PK_i \leftarrow y_i = g^{x_i}, \\ ASK_i \leftarrow k_i \in_R \mathbb{Z}_p^*, \\ APK_i \leftarrow r_i = g^{k_i}. \end{cases}$$

4.3 用户注册

$UserSetup(GP) \rightarrow USK_p, GID_p$：用户$User_p$选取用户私钥$USK_p \leftarrow s_p \in_R \mathbb{Z}_p^*$，计算用户全局标识$GID_p \leftarrow pk_p = g^{s_p}$.

$$\begin{cases} USK_p \leftarrow s_p \in_R \mathbb{Z}_p^*, \\ GID_p \leftarrow pk_p = g^{s_p}. \end{cases}$$

$KenGen\left(APK_i, PK_i, GID_i, AppInfo_i, SK_i, ASK_i\right) \rightarrow AK_{i,GID_p}, Param_{i,GID_p}$：用户$User_p$提交其全局标识$GID_p$，$App_i$生成其对应应用服务信息$AppInfo_i$的用户属性密钥及其中间参数$AK_{i,GID_p}, Param_{i,GID_p}$，并传输给用户，该秘钥即为用户的注册凭据.

$$\begin{cases} AK_{i,GID_p} \leftarrow e_i = H_1\left(AppInfo_i, APK_i, GID_p, PK_i\right), \\ Param_{i,GID_p} \leftarrow param_{i,GID_p} = k_i + e_i x_i. \end{cases}$$

4.4 VC 生成

假设用户$User_p$已注册的应用服务信息集合为$\Omega_p = \left\{AppInfo_1, AppInfo_2, \cdots, AppInfo_t\right\}$，在这一步骤中将进行VC生成，该步骤将在Intel SGX环境中进行：

1）对于$i = 1, 2, \cdots, t$，随机选取$t_i \in \mathbb{Z}_p^*$，计算$R_i = g^{t_i}$；

2）对于$i = t+1, t+2, \cdots, n$，随机选取$c_i, d_i \in \mathbb{Z}_p^*$，计算

$$\begin{cases} e_i = H_1\left(AppInfo_i, r_i, GID_p, y_i\right), \\ Y_i = r_i y_i^{e_i}, \\ R_i = g^{d_i} Y_i^{c_i}. \end{cases}$$

3）对于$i = 1, 2, \cdots, n$，计算$T_i = R_i^{s_p}$，根据结果计算$c = H_2(m, T_1, T_2, \cdots, T_n, GID_p)$；

4）用$n-t+1$个点$(0, c), (t+1, c_{t+1}), \cdots, (n, c_n)$构造$n-t$次拉格朗日插值多项式$P_{n-t}(x)$；

5）计算

$$\begin{cases} c_i = P_{n-t}(i), \\ d_i = t_i - c_i param_{i,GID_p}, \end{cases} \quad i = 1, 2, \cdots, t$$

6）选择随机数$b_i \in \mathbb{Z}_p$，计算承诺值$C_{1i} = g^{b_i}, C_{2i} = R_i^{b_i}$，挑战值$zkc_i = H_3(C_{1i}, C_{2i})$，响应值$z_i = b_i + w_i s_p, i = 1, 2, \cdots, n$；

7）输出签名$\sigma = (c_i, d_i, T_i, r_i, GID_p)(i = 1, 2, \cdots, n)$、多项式$P_{n-t}(x)$和$proof_{\mathrm{ZK}} = (C_{1i}, C_{2i}, zkc_i, z_i)$，并初始化登录次数计数哈希$CNT = 1$，选择随机数$w_{\mathrm{last}} \in \mathbb{Z}_p$，用于完成用户私钥泄露、冒用的审计.

4.5 应用验证

用户$User_p$期望向应用服务APP_v进行身份验证，具体为3个步骤.

1）用户利用ElGamal加密算法选择随机数$v \in \mathbb{Z}_p$，计算密文参数w及密文CNT_m，并组合构建$proof_{CNT}$，该步骤由用户在Intel SGX环境中进行，即

$$\begin{cases} w = g^v, \\ CNT_m = CNT \times y_{APP_v}^v, \\ proof_{CNT} = (w, CNT_m, w_{\mathrm{last}}). \end{cases}$$

2）使用签名及零知识证明凭据$proof_{\mathrm{ZK}}$构建可验证凭据

$$VC = (\sigma, P_{n-t}(x), proof_{\mathrm{ZK}}, proof_{CNT}).$$

用户发送VC到应用服务并更新$CNT = CNT + 1$，$w_{\mathrm{last}} = w$.

3）应用服务APP_v将进行6步验证：

① $P_{n-t}(x)$为$n-t$次多项式；

② 验证$P_{n-t}(i) = c_i, i = 1, 2, \cdots, n$，在此步骤不通过的验证请求，将被返回$P_{n-t}(i) = c_i$等式不成立对应的序号$i$；

③ 验证$P_{n-t}(0) = H_2(m, T_1, T_2, \cdots, T_n, GID_p)$；

④ 验证$\log_{R_i} T_i = \log_g GID_p, i = 1, 2, \cdots, n$，即根据NIDLEQ协议对签名人$User_p$提供的$proof_{\mathrm{ZK}}$进行验证；

⑤ 验证$proof_{CNT}$，首先验证$w \neq w_{\mathrm{last}}$，解密获得CNT明文

$$CNT = CNT_m / w^{x_v}.$$

获取区块链数据中$User_p$对应的登录次数计数哈希h_{User_p}，如果CNT=1，且区块链中数据h_{User_p}不存在则直接通过该项验证，否则验证$h_{User_p} = H(GID_p, CNT - 1, w_{\mathrm{last}})$，在此步骤验证失败将告知用户VC可能被冒用；

⑥ 以上①~⑤全通过，则签名为有效签名，更新$User_p$对应的登录次数计数哈希$h_{User_p} = H(GID_p, CNT, w)$，保存$User_p$对应的链上信息为$(APP_v, h_{User_p})$.

4.6 VC 扩充

当用户在已有VC的情况下获取到新的注册凭据后，可以进行VC扩充. 假设签名人$User_p$已注册的应用服务信息集合为$\Omega_p = \left\{AppInfo_1, AppInfo_2, \cdots, AppInfo_t\right\}$，用户通过$KeyGen$步骤申请到$App_{t+1}$的凭据$AK_{t+1,GID_p}, Param_{t+1,GID_p}$，用户的VC将通过以下5个步骤更新.

1）随机选取$d \in \mathbb{Z}_p^*$，计算$R_{t+1} = g^d$；

2) 计算$T_{t+1}=R_{t+1}^{s_p}$, 并更新哈希结果$c=H_2(T_1, T_2, \cdots, T_n, GID_p)$;

3) 用$n-t$个点$(0,c),(t+2,c_{t+2}),\cdots,(n,c_n)$构造$n-t-1$次拉格朗日插值多项式$P_{n-t-1}(x)$;

4) 根据新的拉格朗日插值多项式计算

$$\begin{cases}c_i=P_{n-t-1}(i),\\ d_i=t_i-c_is_i,\end{cases}\quad i=1,2,\cdots,t+1;$$

5) 更新签名$\sigma=(c_i,d_i,T_i,r_i,GID_p)(i=1,2,\cdots,n)$和多项式$P_{n-t-1}(x)$.

VC扩充只需要更新签名及多项式即可, 此过程不影响零知识证明凭据及链上登录次数计数哈希h_{User_p}的验证.

5 分析与评估

本节将讨论本文方案在正常执行过程中的正确性及在3.2节定义的安全模型下的安全性. 同时实现VC生成、VC扩充及应用验证流程的算法, 并评估其效率.

5.1 正确性证明

定理 1. 签名σ的验证是正确的.

证明. 根据方案的执行流程, 正常运行下$P_{n-t}(x)$由$n-t+1$个点$(0,c),(t+1,c_{t+1}),\cdots,(n,c_n)$构成, 所以$P_{n-t}(x)$为$n-t$次多项式, 且$P_{n-t}(i)=c_i, i=t+1,t+2,\cdots,n$. 同时, 在$i=1,2,\cdots,t$时, c_i由$c_i=P_{n-t}(i)$计算得到, 且$P_{n-t}(0)=H_2(m,T_1,T_2,\cdots,T_n,GID_p)$是显然成立的.

基础的签名验证协议采用Schnorr协议, 根据$d_i=t_i-c_is_i$及$s_i=k_i+e_ix_i$, 可以推出$g^{d_i}Y_i^{c_i}=g^{t_i-c_is_i}\left(r_iy_i^{e_i}\right)^{c_i}=g^{t_i-c_i(k_i+e_ix_i)}\left(g^{k_i}g^{x_ie_i}\right)^{c_i}=g^{t_i}=R_i(i=1,2,\cdots,t)$.

$$R_i=\begin{cases}g^{t_i}, i=1,2,\cdots,t\\ g^{d_i}Y_i^{c_i}, i=t+1,t+2,\cdots,n\end{cases}=g^{d_i}Y_i^{c_i}, i=1,2,\cdots,n.$$

由Sigma协议的正确性性质, 验证者总是接受. 因此签名σ的验证是正确的. 证毕.

定理 2. 零知识证明凭据的验证是正确的.

证明. 对于验证是否$\log_{R_i}T_i=\log_gGID_p$,其中$R_i=g^{d_i}Y_i^{c_i}, i=1,2,\cdots,n$. 若方案正常运行, $T_i=R_i^{s_p}=\left(g^{d_i}Y_i^{c_i}\right)^{s_p}$, $GID_p=g^{s_p}$, 由NIDLEQ协议的正确性性质, 验证者总是接受, 因此零知识证明凭据的验证是正确的. 证毕.

定理 3. 身份冒用审计凭据的验证是正确的.

证明. 在不存在身份冒用、用户正常逐次进行身份验证的情况下, 登录次数计数$CNT=CNT_{\text{last}}+1$总是成立, 对于$h_{User_p}=H\left(GID_p,CNT_{\text{last}},w_{\text{last}}\right)$总是有以下等式成立:

$$H\left(GID_p,CNT_{\text{last}},w_{\text{last}}\right)=H\left(GID_p,CNT-1,w_{\text{last}}\right).$$

因此身份冒用审计凭据的验证是正确的. 证毕.

5.2 安全性分析

分析 1. 证明伪造攻击分析.

假设第三方用户攻击者$\mathcal{A}_1$试图伪造用户VC, 向应用服务发起应用验证请求. 伪造VC需要伪造4个部分的内容, 分别为签名σ、多项式$P_{n-t}(x)$、零知识证明凭据$proof_{\text{ZK}}$、登录次数计数证明$proof_{CNT}$. 攻击者$\mathcal{A}_1$可以主动申请私钥及注册凭据构建签名及多项式, 但签名内容GID_p是应用服务区分用户身份的关键. $\mathcal{A}_1$无法在不知道用户私钥USK_p的前提下通过NIDLEQ零知识证明协议构建$proof_{\text{ZK}}$, 因此$\mathcal{A}_1$进行伪造攻击的难度不小于求解$\log_gGID_p$, 即离散对数问题的难度.

分析 2. 证明重放攻击分析.

假设第三方用户攻击者$\mathcal{A}_1$获取到来自用户的已完成应用验证过程的VC, 试图重用这些证明以与被请求应用服务进行新的身份验证交互. 在用户未进行新的VC扩充之前, VC除了$proof_{CNT}$以外不会发生改变.

在应用验证的过程中, 每一次验证都会对CNT进行自增的更新操作. $proof_{CNT}$使用了ElGamal加密算法, 保证了CNT不被攻击者$\mathcal{A}_1$得知. 虽然ElGamal加密算法具有同态加密的特性, 但验证过程中将拒绝加密参数相同的$proof_{CNT}$, 即$w\neq w_{\text{last}}$. 因此$\mathcal{A}_1$无法在不知道应用服务私钥SK的前提下解密明文CNT及构建$proof_{CNT}$, 所以证明重放攻击在本方案中是无效的.

分析 3. 信息盗窃攻击分析.

应用服务攻击者$\mathcal{A}_2$作恶, 试图从接收到的VC和区块链上的公共可访问信息中获取经过身份验证的用户身份信息, 进行新的身份验证交互. 该攻击明显强于证明重放攻击, 因为$\mathcal{A}_2$应用服务直接掌握了应用服务私钥SK, 可以直接解密用户$proof_{CNT}$访问到CNT, 通过构建$proof_{CNT+1}$, 但该过程是有前提的.

$proof_{CNT}$包含了3个元素, 即$w, CNT_m, w_{\text{last}}$. 其中$w_{\text{last}}$既可以用于应用服务与用户审计验证$h_{User_p}=H(GID_p, CNT-1, w_{\text{last}})$, 同时也保证了$\mathcal{A}_2$构建$proof_{CNT+1}$需要满足用户在应用服务的$\mathcal{A}_2$处进行应用验证过程后, 未与其他应用服务进行新的身份验证交互的条件. 因此, $\mathcal{A}_2$有能力在一定时间区间内构建$proof_{CNT+1}$, 进一步构建VC, 与其他应用服务进行交互, 但该过程会使$\mathcal{A}_2$有3点风险:

1) 当$\mathcal{A}_2$与其他应用服务验证完成后, 区块链将进行信息更新链上信息, 用户进行登录次数计数哈希审计可得知身份信息被冒用;

2）用户进行常规应用验证流程，由于w_{last}及CNT发生改变，应用验证将返回，可得知身份信息被冒用；

3）应用验证流程需要更新链上信息，由于区块链去中心化、公开透明、可追溯、不可篡改的特点，当用户得知身份信息被冒用后，用户可以直接追溯到用户最后一次正常验证的应用服务的$\mathcal{A}_2$.

综上，本文方案存在应用服务在较短时间区间内信息盗窃攻击的可能性，但该过程是用户可审计、易察觉的，加之区块链可追溯、不可篡改的特性，该攻击的责任人将是明确的.

5.3 同类工作对比

本文方案是用户在 TEE 下计算、管理秘钥的自主主权身份方案，不存在可能泄露秘钥信息的中心信息机构，且无法被重放攻击威胁. 本节将与其他方案对比身份验证方案特性. 文献 [19] 使用了开源框架 ZoKrates 生成零知识证明凭据以认定可验证凭据的归属权. 但该零知识证明凭据将被传输到链上，且在验证时该凭据会发送给验证者，因此恶意用户及恶意应用服务可以直接获取相应的用户证书及零知识证明凭据，登录到其它应用服务，实现证明重放攻击. 文献 [20] 使得用户在身份存在冒用记录时收到反馈，形成对重放攻击的抵御，但该方案缺少对于应用服务的扩展支持，并且需要用户执行隐私风险较高的秘钥计算过程. 文献 [27] 的身份认证方案在秘钥计算流程上用 SGX 技术保证用户环境下的秘钥计算安全，但方案存在集中了大量用户身份信息的身份提供者及信息认证依赖方，本质上不是自我主权身份. 文献 [19, 20, 27] 所述方案的特性比较如表 3 所示.

Table 3 Feature Comparison of Identity Authentication Schemes

表 3 身份认证方案特性对比

方案	抗重放攻击	安全秘钥计算	自我主权身份
文献 [19]			√
文献 [20]	√		√
文献 [27]	√	√	
本文方案	√	√	√

注：“√”表示相关方案具有相应的特性.

5.4 实验环境说明

本文方案的实验环境由本地物理机连接到服务器，并在服务器环境下开发完成. 物理机硬件配置为：物理机型号联想小新 Pro14，芯片 Intel i9-12900H，内存 16 GB，操作系统 windows11. 服务器配置如表 4 所示.

Table 4 Server Experimental Environment Configuration Instructions

表 4 服务器实验环境配置说明

硬件/软件	型号/版本
CPU	Intel® Xeon®Gold 6142 CPU @ 2.60 GHz
内核数	4
内存	8 GB
操作系统	Ubuntu22.04 Linux 5.15.0-43-generic
Go	1.20.2 Linux/AMD64
EGo	v1.2.0
Solidity	v0.6.12+commit.27d51765
Truffle	v5.7.4
Ganache	v7.7.3
Node.js	v19.6.0
Web3.js	v1.8.2

EGo[36] 是一个通过 Go 语言构建 TEE 应用程序的框架. 通过 EGo 可以模拟应用程序运行在加密和可验证的 Intel SGX 飞地中.

在公共参数的选择上，使用了阶数p为 1 024 位的素数群. 为了更符合实际场景，我们设定用户注册的应用数量大致占总应用服务数量的 2/3，如在图 4 中，VC 生成时间数据点（30，0.001 47）表示 VC 生成算法执行时应用服务总数为 30 个，用户拥有的注册凭据数量为 20，算法执行时间为 0.001 47 s.

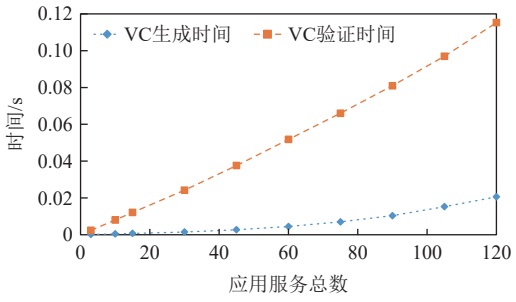

Fig. 4 VC generation time and verification algorithm time in common environment

图 4 普通环境中的 VC 生成时间及验证算法时间

5.5 性能分析

首先考虑算法在常规环境中的效率. 图 4 是普通环境中的 VC 生成及验证算法时间. 由图 4 可见，随着用户数量的增多，算法执行时间将持续上升，但可喜的是，应用程序总数为 120 时，VC 生成时间不超过 0.04 s，VC 验证时间将不超过 0.2 s，这在实际应用场景中是可以接受的.

用户应用注册需要向应用服务申请新的注册凭据，该流程是简洁且基本固定的，在此不做讨论. 但用户申请新的注册凭据，随后扩充 VC 的过程是会频繁发生的. 图 5 是 VC 生成时间及 VC 扩充时间的比较，两者同是根据注册凭据信息计算 VC，但 VC 扩充算法将省去部分重复计算过程，提高运算效率. 实验数据表明，在应用服务总数小于 100 个时，效率的提升在 50% 以上，在 120 个应用服务时效率提升大约为 35%.

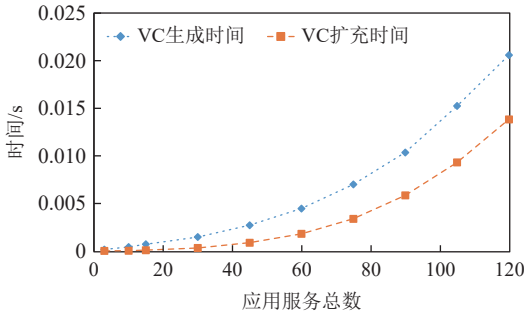

Fig. 5 VC generation time and expansion time

图 5 VC 生成时间及 VC 扩充时间

探究用户算法在可信执行环境中的执行情况将更有意义. 正如我们在 1.2 节提到的，在 SGX 的应用中，飞地的调入和调出将会跨越安全边界，并对安全参数进行检查，因此会产生额外的性能开销，因此我们在实验中不仅计算 VC 算法的时间，而且还要计算调用 VC 算法的环境初始化时间. 图 6 是环境初始化及 VC 生成算法在普通环境及 SGX 环境中的 VC 生成流程时间比较，可见在应用服务总数相同的情况下，SGX 环境中额外需要的时间大致为 1.75 s，且在应用服务总数增长的过程中，该时间不存在明显增长. 如果该计算过程在应用服务中执行，则从 0.25 s 的时间开销提升到 2 s 的时间开销将导致服务器的吞吐量降低 87.5%，但这个计算过程是在用户环境下执行的，用户在本方案仅需要关注自身的 VC 生成，因此该时间差是完全可以接受的.

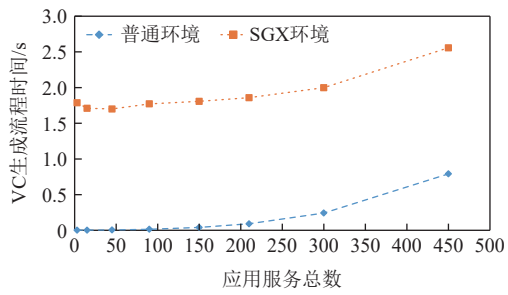

Fig. 6 VC generation process time in different environments

图 6 不同环境中 VC 生成流程时间

为了使用户身份泄露可审计，同时也防止针对 VC 的重放攻击，方案引入了 $proof_{CNT}$. $proof_{CNT}$的组成决定了其大小是固定的，这在实验中得到了验证. 图 7 是不同应用服务数量下各验证凭据占用存储空间大小及 $proof_{CNT}$的生成时间. 可以发现，相比签名主体及零知识证明凭据，$proof_{CNT}$的额外通信开销可以忽略不计且与应用服务总数没有关联的. 加之从时间上分析，$proof_{CNT}$的生成时间总维持在 0.001 4 s 左右，说明了本方案的$proof_{CNT}$设计是低负载且高效的.

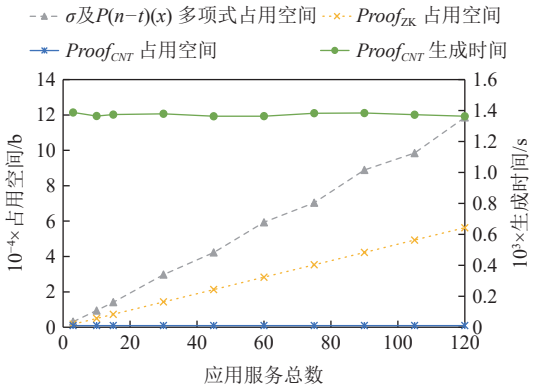

Fig. 7 The storage space occupied by each authentication credential and the generation time of $proof_{CNT}$

图 7 各验证凭据占用存储空间大小及 $proof_{CNT}$ 生成时间

6 总 结

基于区块链和 TEE 技术，本文提出并实现了一种去中心化的属性签名身份认证方案.

本文方案旨在从用户在身份认证流程中的需求出发，解决可验证凭据管理繁重、安全性不足的问题，结合 Schnorr 协议与非交互式离散对数相等零知识证明协议，让用户自主构建身份信息. 当用户需要更新身份时，可通过扩充可验证凭据的方式重新构建身份信息.

为了让用户完全控制自身的身份信息，将可验证凭据的生成与扩充流程置于用户本地的 TEE 中，TEE 的引入保证了用户生成、扩充可验证凭据时的安全性，这对于通常来说安全防护意识不足、防护能力较差的用户环境而言是重要的. 登录次数计数哈希的记录使得用户自身的身份信息以任何可能的方式泄露而被冒用的行为可察觉、可审计，使得用户可及时应对. 通过实验验证分析，本文方案在实际应用中是高效可行的.

本文方案流程仅针对身份验证及身份冒用审计.

在未来，对身份信息泄露后的身份信息归属判断及恢复行为会是我们研究的重点.

作者贡献声明：冉津豪提出了思路并做了源代码实现，撰写论文；蔡栋梁提出了优化思路并协助撰写论文.

参 考 文 献

[1] Ciberspace Administration of China. Sixteen departments including the central network information office jointly announced the name of the national blockchain innovative application pilot[EB/OL]. 2022[2023-02-28]. http://www.cac.gov.cn/2022-01/29/c_1645059212139691.htm(in Chinese)
(国家互联网信息办公室. 中央网信办等十六部门联合公布国家区块链创新应用试点名[EB/OL]. 2022[2023-02-28]. http://www.cac.gov.cn/2022-01/29/c_1645059212139691.htm)

[2] Gayvoronskaya T, Meinel C. Blockchain: Hype or Innovation[M]. Cham, Switzerland: Springer, 2020

[3] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[EB/OL]. 2008[2023-02-28]. https://bitcoin.org/bitcoin.pdf

[4] Haber S, Stornetta W S. How to Time-stamp a Digital Document[M]. Berlin: Springer, 1991

[5] Dwork C, Naor M. Pricing via processing or combatting junk mail[C]// Proc of the 12th Annual Int Cryptology Conf on Advances in Cryptology. Berlin: Springer, 1993: 139−147

[6] Back A. Hashcash−a denial of service counter-measure[EB/OL]. 2002[2023-05-23]. http://www.hashcash.org/papers/hashcash.pdf

[7] Vukolic M. Eventually returning to strong consistency[J]. IEEE Data Engineering Bulletin, 2016, 39(1): 39−44

[8] National Bureau of Statistics. Statistical Bulletin of the People's Republic of China on National Economic and Social Development in 2022[EB/OL]. 2023[2023-02-28]. http://www.stats.gov.cn/xxgk/sjfb/zxfb2020/202302/t20230228_1919001.html (in Chinese)
(国家统计局. 中华人民共和国2022年国民经济和社会发展统计公报[EB/OL]. 2023[2023-02-28]. http://www.stats.gov.cn/xxgk/sjfb/zxfb2020/202302/t20230228_1919001.html)

[9] Avellaneda O, Bachmann A, Barbir A, et al. Decentralized identity: Where did it come from and where is it going?[J]. IEEE Communications Standards Magazine, 2019, 3(4): 10−13

[10] Wei Liang, Huang Zhenjie, Chen Qunshan. Decentralized attribute-based non-repudiation signature[J]. Computer Engineering and Science, 2020, 42(6): 1003−1011 (in Chinese)
(魏亮, 黄振杰, 陈群山. 去中心基于属性不可否认签名[J]. 计算机工程与科学, 2020, 42(6): 1003−1011)

[11] Dib O, Toumi K. Decentralized identity systems: Architecture, challenges, solutions and future directions[J]. Annals of Emerging Technologies in Computing, 2020, 4(5): 19−40

[12] Dell'Amico M, Michiardi P, Roudier Y. Password strength: An empirical analysis[C]//Proc of IEEE INFOCOM 2010. Piscataway, NJ: IEEE, 2010: 1−9

[13] Maler E, Reed D. The venn of identity: Options and issues in federated identity management[J]. IEEE Security & Privacy, 2008, 6(2): 16−23

[14] Groß T. Security analysis of the SAML single sign-on browser/artifact profile[C]//Proc of 19th Annual Computer Security Applications Conf. Piscataway, NJ: IEEE, 2003: 298−307

[15] Hardt D. The OAuth 2.0 authorization framework[EB/OL]. 2012[2023-05-30]. https://www.rfc-editor.org/rfc/rfc6749

[16] Naik N, Jenkins P. uPort open-source identity management system: An assessment of self-sovereign identity and user-centric data platform built on blockchain[C]//Proc of 2020 IEEE Int Symp on Systems Engineering (ISSE). Piscataway, NJ: IEEE, 2020: 1−7

[17] Ali M, Nelson J, Shea R, et al. Blockstack: A global naming and storage system secured by blockchains[C]//Proc of the 2016 USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2016: 181−194

[18] Maram D, Malvai H, Zhang Fan, et al. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability[C]//Proc of 2021 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2021: 1348−1366

[19] Yuan Hexin, Liu Baixiang, Kan Haibin, et al. Distributed public key infrastructure scheme based on blockchain and decentralized non-repudiation attribute signature[J]. SCIENTIA SINICA Informationis, 2022, 52(6): 1135−1148 (in Chinese)
(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)

[20] Alangot B, Szalachowski P, Dinh T T A, et al. Decentralized identity authentication with auditability and privacy[J]. Algorithms, 2022, 16(1): 4−25

[21] Mckeen F , Alexandrovich I , Berenzon A , et al. Innovative instructions and software model for isolated execution[C/OL]//Proc of the 2nd Int Workshop on Hardware and Architectural Support for Security and Privacy. New York: ACM, 2013.[2023-05-30]. https://doi.org/10.1145/2487726.2488368

[22] Hoekstra M , Lal R , Pappachan P , et al. Using innovative instructions to create trustworthy software solutions[C/OL]//Proc of the 2nd Int Workshop on Hardware and Architectural Support for Security and Privacy. New York: ACM, 2013[2023-05-30]. https://doi.org/10.1145/2487726.2488370

[23] Weisse O, Bertacco V, Austin T. Regaining lost cycles with HotCalls: A fast interface for SGX secure enclaves[J]. ACM SIGARCH Computer Architecture News, 2017, 45(2): 81−93

[24] Zhang F, Cecchetti E, Croman K, et al. Town crier: An authenticated data feed for smart contracts[C]//Proc of the 2016 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 270−282

[25] Dong Chuntao, Shen Qingni, Ding Xuhua, et al. T-Counter: Trustworthy and efficient CPU resource measurement using SGX in the cloud[J]. IEEE Transactions on Dependable and Secure Computing, 2023, 20(1): 867−885

[26] Schuster F, Costa M, Fournet C, et al. VC3: Trustworthy data analytics in the cloud using SGX[C]//Proc of 2015 IEEE Symp on

Security and Privacy. Piscataway, NJ: IEEE, 2015: 38−54

[27] Song Tianlin, Wang Wei, Lang Fan, et al. P2A: Privacy preserving anonymous authentication based on blockchain and SGX[C] //Proc of the 16th Int Conf on Information Security and Cryptology. Berlin: Springer, 2021: 257−276

[28] Zhao Bo, Xiao Yu, Huang Yuqing, et al. A private user data protection mechanism in TrustZone architecture based on identity authentication[J]. Tsinghua Science and Technology, 2017, 22(2): 218−225

[29] ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms[J]. IEEE Transactions on Information Theory, 1985, 31(4): 469−472

[30] Damgård I. On Σ-protocols[EB/OL]. 2002[2023-05-30]. https://www.cs.au.dk/~ivan/Sigma.pdf

[31] Schnorr C P. Efficient signature generation by smart cards[J]. Journal of Cryptology, 1991, 4: 161−174

[32] Fiat A, Shamir A. How to prove yourself: Practical solutions to identification and signature problems[C]//Proc on Advances in Cryptology (CRYPTO'86). Berlin: Springer, 1987: 186−194

[33] Zhang Qiong, Wang Yuke, Jue J P. A key management scheme for hierarchical access control in group communication[J]. International Journal of Network Security, 2008, 7(3): 323−334

[34] Garg N, Wazid M, Das A K, et al. BAKMP-IoMT: Design of blockchain enabled authenticated key management protocol for Internet of medical things deployment[J]. IEEE Access, 2020, 8: 95956−95977

[35] Ahmad S, Mehfuz S, Beg J. Cloud security framework and key management services collectively for implementing DLP and IRM[J]. Materials Today:Proceedings, 2022, 62(7): 4828−4836

[36] Edgeless Systems. EGo [EB/OL]. [2023-02-28]. https://github.com/edgelesssys/ego

Ran Jinhao, born in 1998. Master candidate. His main research interests include blockchain, attribute based encryption, and trusted execution environment.

冉津豪, 1999 年生. 硕士研究生. 主要研究方向为区块链、属性基密码、可信执行环境.

Cai Dongliang, born in 2000. PhD candidate. His main research interests include blockchain, attribute based encryption, and zero-knowledge proof.

蔡栋梁, 2000 年生. 博士研究生. 主要研究方向为区块链、属性基密码、零知识证明.