软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007624] [CSTR: 32375.14.jos.007624]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向轻量级节点的数据跨域安全传输方案*

芦静文[1], 宋雅晴[1], 张 源[1], 何欣雨[1], 龚 竞[2], 李洪伟[1]

[1](电子科技大学计算机科学与工程学院 (网络空间安全学院), 四川 成都 611731)

[2](四川省大数据技术服务中心, 四川 成都 610041)

通信作者: 宋雅晴, E-mail: yaqing.song@uestc.edu.cn; 张源, E-mail: zhangyuan@uestc.edu.cn

摘 要: 针对数据跨域传输中面临的用户隐私安全与传输数据安全问题, 以及跨域通信系统要求的用户便携性与节点轻量化等实用性问题, 分析现有方案, 提出强对抗环境下面向轻量级节点的数据跨域安全传输方案, 实现: 1) 去中心化公钥与身份认证; 2) 用户便携式系统访问; 3) 通信节点可信密钥协商; 4) 抗密钥泄露的数据跨域传输. 形式化证明所提方案的安全性, 并进行原型系统实现与性能评估, 实验结果表明系统计算开销与通信开销是高效的.

关键词: 跨信任域; 双向身份认证; 数据传输; 抗密钥泄露; 区块链

中图法分类号: TP393

中文引用格式: 芦静文, 宋雅晴, 张源, 何欣雨, 龚竞, 李洪伟. 面向轻量级节点的数据跨域安全传输方案. 软件学报. http://www.jos.org.cn/1000-9825/7624.htm

英文引用格式: Lu JW, Song YQ, Zhang Y, He XY, Gong J, Li HW. Secure Cross-domain Data Transmission Scheme for Lightweight Nodes. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7624.htm

Secure Cross-domain Data Transmission Scheme for Lightweight Nodes

LU Jing-Wen[1], SONG Ya-Qing[1], ZHANG Yuan[1], HE Xin-Yu[1], GONG Jing[2], LI Hong-Wei[1]

[1](School of Computer Science and Engineering (School of Cyber Security), University of Electronic Science and Technology of China, Chengdu 611731, China)

[2](Sichuan Provincial Big Data Technology Services Center, Chengdu 610041, China)

Abstract: To address critical challenges in user privacy and transmission data security in cross-domain environments, as well as practical requirements for user portability and lightweight nodes in cross-domain communication system, this study proposes a blockchain-based secure cross-domain data transmission scheme under highly adversarial conditions. The scheme achieves: 1) decentralized public key management and identity authentication; 2) portable system access for users; 3) trusted key negotiation among communication nodes; 4) key-leakage resistant cross-domain data transmission. The security of the proposed scheme is formally analyzed and proven. Furthermore, a system prototype is implemented to evaluate performance. Experimental results indicate that computation, storage, and communication overheads are all efficiently managed, demonstrating the scheme's practicality and scalability.

Key words: cross trust domain; mutual identity authentication; data transmission; key-leakage resistance; blockchain

人工智能、云计算等新兴技术的快速发展与演进催生了多信任域协同、智能一体化等新兴服务范式, 为用户带来前所未有的服务体验, 对数据在不同信任域间的安全流转提出了新的技术要求[1]. 相应地, 跨信任域、跨平台的数据传输技术能够为以上场景提供基础技术支撑, 促进新兴服务向着更成熟、更便携、更安全的方向稳步迈进[2,3], 进一步促进数字经济发展、数据要素流通与数据价值释放. 典型地, 在智能驾驶领域, 促进车端、路端、云端多信任域协同能够精准捕捉交通数据、路测设备状态等, 为自动驾驶提供更高的执行效率与更强的安全保障. 例如, 北京-雄安-保定启动车路云一体化多元应用试点, 覆盖货运干线物流、机场接驳和工作通勤这 3 大场景, 极

* 基金项目: 国家重点研发计划 (2023YFB3106503); 国家自然科学基金面上项目 (62472074); 第八届中国科协青年人才托举工程 (2022QNRC001); 四川省重大科技专项 (2022ZDZX0038); 四川省自然科学基金 (24NSFSC2271)
收稿时间: 2025-07-17; 修改时间: 2025-09-23, 2025-11-10; 采用时间: 2025-12-26; jos 在线出版时间: 2026-04-22

大提升用户出行体验及通勤效率, 精准响应道路安全威胁以提供更高安全行车保障. 类似地, 数据跨域传输技术在政务协同、跨境支付、工业物联网、智能家居等业务场景中也展现出极大优势[4].

另一方面, 高价值数据在跨域流通过程中面临严峻的安全威胁, 首先, 数据跨域传输中身份冒充攻击频发, 恶意敌手试图通过冒充合法用户身份进行通信以获取用户隐私信息或发布恶意信息等; 进一步地, 数据在公开信道传输易遭受被动敌手窃听攻击, 甚至主动敌手的篡改攻击, 破坏传输数据的机密性与完整性. 例如, 在智能驾驶领域, 车辆传感器能够收集道路环境、周边车辆及行人数据, 然而若恶意敌手能够获取以上数据并冒充合法车辆身份, 并恶意向周边车辆发送恶意信息, 此外, 传感器能够捕捉车内乘客面部及动作数据、车内人员谈话内容等隐私信息, 敌手根据窃取到的隐私数据影响用户行为, 严重威胁行车安全甚至人身安全. 类似针对数据完整性的攻击已成为现实, 美国铁路中涉及的列车首部装置 (head-of-train, HoT) 与列车尾部装置 (end-of-train, EoT) 之间的通信协议缺乏对指令数据的完整性校验机制, 使得攻击者可以伪造无线数据信号, 从而触发列车紧急制动甚至导致铁路系统瘫痪[5].

为同时保证数据跨域传输中的身份认证性、数据机密性与完整性, 现实中众多系统[3,6]通过采取“身份认证机制+认证加密算法”范式来提供相应的安全保障. 然而, 在更高数据敏感性与更强威胁模型下, 会话密钥泄露问题普遍存在, 若其中一个参与方被敌手控制, 敌手能够获取会话密钥, 利用该密钥解密历史通信内容并持续监听后续通信过程, 此时该合法用户在系统内的数据内容将不具有隐私性. 此外, 在实用性方面, 现有方案具有极强的设备依赖性, 即用户为实现安全的身份认证与数据传输, 需在本地安全维护与身份绑定的密钥信息, 该方式高度依赖于初始注册设备, 用户无法通过其他设备终端使用已注册账户进入该信息系统, 极大限制了用户的使用条件. 这种单设备绑定的非便携性已无法满足当前用户对全时域设备独立的访问需求.

本文提出一种面向强对抗环境的数据跨域安全传输方案 SCDT (secure cross-domain data transmission scheme), SCDT 提供身份认证性、数据机密性与完整性保障, 还具备抗密钥泄露安全与便携性等特性. 具体来说, 为抵御跨信任域节点通信过程中面临的身份冒充攻击, 我们使用基于区块链的公钥认证机制[1,7–9], 使得跨域服务器无需共同依赖一个可信第三方机构即可验证公钥合法性. 为实现用户便携性, 实现设备独立的身份认证与服务请求, 我们使用基于口令的加密机制[10–12]加密用户私钥并将密文记录在区块链上. 进一步地, 为抵御口令内在低熵特性导致的字典猜测攻击 (dictionary guess attack, DGA), 我们引入密码服务器为用户提供口令加固服务, 使得在不牺牲口令优势的前提下抵御 DGA. 节点在服务器协助下进行双向身份认证的过程中, 能够通过双方身份信息执行密钥协商来计算获得共享密钥用于消息密钥派生, 无需目标节点参与以实现异步节点间的密钥协商. 针对通信过程中存在的会话密钥泄露问题, 我们基于 Signal 协议[13,14]实现了跨信任域端到端数据加密技术[15,16], 使得消息密钥随通信轮次的增加而更新, 达到前向及后向安全性 (即抗密钥泄露安全性, 此后均称作前向及后向安全性). 为同时保证传输数据机密性与完整性, 我们使用带附加值的认证加密 (authenticated encryption with associated data, AEAD) 算法进行数据处理. 本文主要贡献如下.

1) 构建跨信任域环境下面向轻量级节点的去中心化双向身份认证机制与数据传输方法. 节点与身份认证服务器交互, 构建其身份区块并通过服务器认证, 将定期更新的中期密钥安全存储在服务器端, 通信节点双方借助信任域间身份认证服务器交互获取双方身份区块地址及中期密钥信息以进行密钥协商计算; 基于 Signal 通信协议实现安全的数据传输机制, 使得节点在经过上述可靠身份认证后, 能够建立安全的通信连接获得共享密钥并进行传输数据处理, 实现传输数据的机密性与认证性, 实现前向及后向安全性.

2) 提出基于区块链的高效公钥认证机制与基于口令的密钥管理方法. 信任域内密码服务器与身份认证服务器分布式安全存储一个备份密钥, 并使用该备份密钥对身份密钥信息加密并上传至区块链系统, 使得用户无需本地存储密钥信息, 且任何外部敌手无法恢复用户隐私信息或篡改用户身份信息.

3) 构建面向轻量级节点的数据跨域安全传输方案 SCDT. 通过整合以上所提方法构建 SCDT 实现身份认证性、数据机密性与完整性、前向及后向安全与便携性, 抵御强对抗环境下恶意敌手的冒充、窃听、篡改、字典猜测及密钥泄露攻击, 使得用户能够全时域设备独立地访问信息系统.

4) 形式化安全证明与原型系统实现. 结果表明 SCDT 能够在满足用户节点便携性的基础上实现数据跨域传

输的前向安全与后向安全. 此外我们构建原型系统进行评估实验, 表明对于资源受限节点, 计算开销与存储开销在更高的安全假设与更强的实用性下是高效的.

本文第 1 节阐述相关工作. 第 2 节介绍基础密码学原语. 第 3 节描述 SCDT 的系统模型、变量设置、威胁模型及设计目标. 第 4 节详细介绍 SCDT 具体方案. 第 5 节证明 SCDT 方案的安全性. 第 6 节描述系统实现细节与实验分析结果, 并与现有方案进行性能对比. 第 7 节对全文作出总结并展望未来进一步的研究方向.

1 相关工作

不同信息管理系统相互协作完成复杂任务已成为当前现实应用的重要方式, 多域合作催生了跨域管理的需求, 跨信任域端到端数据传输为以上业务场景提供适配的解决方案及重要的安全保证.

在现实体系中, 基于口令的认证机制[17–19]是目前较为主流的身份认证方式, 客户端选择便于记忆的低熵口令作为认证因子, 服务器安全地维护用户身份及口令映射列表. 为实现用户公钥认证性, 基于公钥基础设施 (public-key infrastructure, PKI) 的证书认证体系[20]已得到广泛应用, 将服务器身份与其公钥绑定, 由中心化证书颁发机构 (certificate authority, CA) 为服务器颁发数字证书. 借鉴基于口令的认证机制的基本思想, 为实现用户密钥的有效管理, 便于用户在任意设备上安全存储及恢复其密钥, 可以构建基于口令的加密机制[10,21,22], 使用口令将用户密钥加密并公开记录在区块链上或存储在服务器端, 而口令通常是低熵的, 且不可避免地与用户自身姓名或生日信息紧密相关, 若用户使用口令作为认证因子向服务器请求注册, 极易遭受 DGA, 例如当文献 [22] 中的硬件模块不可信时, 用户在服务器端存储的数据无法抵御 DGA. 为抵御低熵口令带来的猜测攻击, 研究通过引入额外服务器为用户提供口令强化服务[10], 使用服务器秘密与用户口令计算用户身份凭证, 只要满足服务器秘密是安全的, 则认证方案能够抵御 DGA, 为避免对单个密码服务器的依赖, 可构建基于多密码服务器的口令强化机制[4,11,23].

关于基于区块链的公钥认证机制, 魏欣等人[2]通过引入边缘网关以屏蔽物联网的底层异构性问题, 部署联盟链构建不同 CA 之间的信任模型, 利用区块链的共识机制简化多信任域之间的认证流程, 设计了基于网关的跨域认证机制. 张珠君等人[4]基于区块链构建动态可信轻量级认证机制, 结合设备物理特征与区块链记录的 DID[24]实现用户身份绑定, 通过智能合约实现设备的自动注册、验证和撤销, 建立面向智能家居的动态可信认证管理. Cui 等人[25]提出利用“公有链+联盟链”的混合区块链架构实现身份认证, 每个域生成自己的域根密钥对并为节点生成身份证书, 以公有链为信任锚点, 以联盟链为跨域协作核心平台, 构建面向无线传感器网络认证的信任模型, 提供了安全、高效、可扩展的认证方案.

另一方面, 在数据传输过程中, 数据信息在公开信道传输易遭受窃听敌手和恶意敌手篡改攻击, 威胁用户数据的机密性和完整性. 为保证传输数据在公开信道的安全性, 认证加密技术[26,27]、签密[28,29]等加密方案被广泛应用于安全数据传输系统. 典型地, 基于区块链的混合加密技术[1]使用公钥加密方法进行密钥封装并使用私钥加密技术加密数据, 实现数据的安全性与计算的高效性; 基于代理重加密[6]的跨域传输技术将信任域内用户的加密数据委托给代理机构, 由不可信代理服务器将数据重新加密后传输给目标信任域的用户; Signal 协议[16,30,31]提出 X3DH 密钥协商算法, 允许通信双方在非同时在线的情况下建立通信连接, 并构建双棘轮加密协议处理传输数据实现通信会话密钥持续更新, 确保每轮消息加密密钥是不同的, 达到前向与后向安全.

安全跨域通信系统在当前网络发展中已成为一大研究热点, 在当前主流的通用安全通信架构中, 基于证书的身份认证体系 (如 TLS/SSL[32]) 发挥着基石作用, 通过可信 CA 建立公钥信任链, 引入认证加密算法实现通信的机密性与完整性保障. 此外, 广泛应用的端到端加密系统 WhatsApp[33]采用传统公钥体制进行身份认证, 由服务器保存用户公钥副本, 通过主设备建立多终端独立密钥信任链, 从而实现多设备扩展, 通信过程基于 Signal 协议实现端到端安全的数据加密. Telegram[34]则采用混合式架构, 标准聊天模式基于服务器托管的密钥体系, 而秘密聊天模式启用端到端加密, 其私钥仅在本地生成与使用. SCDT 方案在以上研究基础上, 进一步将便携性融入身份认证过程中以适应现实使用场景, 在数据传输过程中实现抗密钥泄漏的安全性保障, 在保证高安全性的同时, 兼顾了系统可用性与终端便携性等现实需求.

我们将 SCDT 与现有方案[1,3,6,9,35]在功能性、安全性等方面进行对比, 结果如表 1 所示, 其中“Y”代表该方案满足所述性质, “N”代表该方案不满足所述性质, “⊥”代表该方案不关注所述性质.

表 1 相关工作对比

性质	Biplane[1]	BCDS[3]	BSCDDS[6]	WWCH[9]	BSCDA[35]	SCDT
便携性	Y	N	N	N	N	Y
密钥协商不要求通信双方同时在线	⊥	N	⊥	⊥	N	Y
前向安全性	N	⊥	N	Y	Y	Y
后向安全性	N	⊥	N	N	Y	Y
不可抵赖性	Y	Y	Y	N	Y	Y
加密机制	混合加密	对称加密+零知识证明	代理重加密+广播加密	⊥	对称加密	双棘轮加密

与 Biplane[1]、BCDS[3]、BSCDDS[6]、WWCH[9]、BCDSA[35]相比, 本文用户认证机制能够实现用户设备可移植性, 用户无需在本地设备存储任何信息, 只需要安全保存低熵口令, 即可在任意设备执行跨信任域数据传输. 在密钥协商过程中, SCDT 允许建立通信连接的双方不同时在线, 而 BCDS 与 BSCDA 都要求参与方均以在线交互式进行密钥协商. 在数据传输过程中, 现存方案分别采用不同的加密机制以实现不同程度的功能性与安全性, WWDH 聚焦于构建安全的双向认证机制, 然而无法实现用户端便携性, 事实上, 除 SCDT 与 Biplane[1]之外, 其余方案均需要用户本地存储私钥信息, 无法实现用户在任意设备任意时间使用通信系统的需求. 文献 [1,6,9] 无法满足后向安全性, 即在当前消息密钥泄露后, 未来通信无法继续保证安全. 文献 [3,6] 在密钥泄露后, 无法保证历史通信内容安全. SCDT 基于 Signal 协议实现, 能够同时实现前向及后向安全性与不可抵赖性.

2 预备知识

2.1 基本符号表示

本文 $\{0,1\}^*$ 表示所有有限长度的比特串组成的集合, $\{0,1\}^n$ 表示所有长为 n 的比特串组成的集合; $a \xleftarrow{\$} A$ 表示从集合 A 中均匀一致选取一个值 a; 对于两个字符串 s_1 和 s_2, $s_1 \| s_2$ 表示将这两个字符串链接在一起, 同时暗含了可以从 $s_1 \| s_2$ 中恢复出原始字符串; $\Pr[\epsilon]$ 表示事件 ϵ 发生的概率.

2.2 双线性映射

假设 G_1和G_2 是两个加法循环群, G_T 为乘法循环群, 其阶均为素数 p, 映射函数 $e: G_1 \times G_2 \to G_T$ 是一个双线性映射, 其对于 $\forall P \in G_1, Q \in G_2$, 满足以下性质.

(1) 双线性. 对于 $\forall a,b \in Z_p^*$, $e(aP,bQ) = e(P,Q)^{ab}$ 成立.

(2) 非退化性. 若 $P \neq Q$, $e(P,Q) \neq 1$ 成立, 其中 1 表示群 G_T 的单位元.

(3) 有效性. 存在高效算法计算 $e(P,Q)$.

2.3 BLS 短签名

BLS 签名[36]是一种基于双线性对的短签名方案, 首先借助哈希技术将签名消息进行压缩, 然后再对哈希值进行签名, 使用双线性对的性质进行签名验证. 在相同安全等级下, BLS 签名与传统签名方案相比 (例如 RSA 签名) 更加高效. SCDT 使用基于 BLS 的盲签名技术, 确保签名内容对签名者不可见.

基于 BLS 的盲签名方案[37]包含 3 个算法: KeyGen、Sign、Verify. $pp = (G_1, G_2, G_T, e, P, Q, \mathrm{H})$ 为各算法隐含输入, 其中 e 为双线性映射; $P \in G_1$, $Q \in G_2$ 分别是群 G_1, G_2 的生成元; 哈希函数 $\mathrm{H}: \{0,1\}^* \to G_1$.

(1) KeyGen. 签名者选取签名公私钥对为 (sk, pk).

(2) Sign. 签名算法包含以下 3 个子算法.

1) 盲化: 消息拥有者随机选取一个盲化因子 r, 对消息 m 进行盲化处理并将盲化消息 m^* 发送给签名者.

2) 签名: 签名者以 (m^*, sk) 为输入, 计算签名 σ^*.

3) 去盲化: 消息拥有者使用盲化因子 r 对签名 σ^* 进行去盲化处理, 得到关于 m 的签名 σ.

(3) Verify. 以消息 m、签名 σ 与签名公钥 pk 为输入, 判断 $e(\sigma,Q)\stackrel{?}{=}e(\mathrm{H}(m),pk)$, 若等式成立, 则 σ 为关于消息 m 的合法签名, 输出 1; 反之, 输出 0.

安全性: 基于 BLS 的盲签名技术提供盲化性及不可伪造性安全保证, 其中盲化性指的是恶意签名者无法根据用户输入获取关于签名内容的有效信息; 不可伪造性指恶意用户在未知签名密钥的情况下无法伪造关于任意消息的合法签名. 以下定义两个游戏分别说明签名方案的安全性.

(1) 定义一个不可区分的游戏 IND-BM 为敌手 $\mathcal{A}$ 与挑战者 $\mathcal{C}$ 之间的交互游戏, 如下所示.

1) 给定安全参数 ℓ, $\mathcal{C}$ 运行密钥生成算法获得签名公私钥对 (sk,pk), 将 sk 发送给 $\mathcal{A}$.

2) $\mathcal{A}$ (签名者) 选取消息 (m_0,m_1) 并发送给 $\mathcal{C}$.

3) $\mathcal{C}$ 均匀一致选取一个比特 $b\in\{0,1\}$, 选取随机数 $r_b\stackrel{\$}{\leftarrow}Z_p^*$, $r_{1-b}\stackrel{\$}{\leftarrow}Z_p^*$, 执行 Sign 签名算法中的盲化算法计算盲化消息 $m_b^*=r_b\cdot\mathrm{H}(m_b)$, $m_{1-b}^*=r_{1-b}\cdot\mathrm{H}(m_{1-b})$, 并将盲化消息随机变换顺序后返回.

4) $\mathcal{A}$ 对收到的盲化消息 (m_b^*,m_{1-b}^*) 签名, 计算 $\sigma_b^*=sk\cdot m_b^*$, $\sigma_{1-b}^*=sk\cdot m_{1-b}^*$ 并将 $(\sigma_b^*,\sigma_{1-b}^*)$ 按顺序发送给 $\mathcal{C}$.

5) $\mathcal{C}$ 对盲化消息的签名去盲化计算得到 $\sigma_b=r_b\cdot\sigma_b^*$, $\sigma_{1-b}=r_{1-b}\cdot\sigma_{1-b}^*$ 并将 (σ_b,σ_{1-b}) 按顺序发送给 $\mathcal{A}$.

6) $\mathcal{A}$ 输出一个比特 b' 作为对 b 的猜测.

7) 若 $b=b'$, 则敌手在游戏中获得胜利.

基于 BLS 的盲签名具有盲化性, 当且仅当对于任意概率多项式时间 (probabilistic polynomial-time, PPT) 敌手 $\mathcal{A}$ 存在一个可忽略函数 $negl(\ell)$, 使得:

$$Adv_{\mathcal{A}}^{\text{IND-BM}}(\ell)=|\Pr[b'=b]-1/2|\leqslant negl(\ell) \tag{1}$$

(2) 定义一个游戏 CMA-EUF 为敌手 $\mathcal{A}$ 与挑战者 $\mathcal{C}$ 之间的交互游戏, 如下所示.

1) 给定安全参数 ℓ, $\mathcal{C}$ 运行密钥生成算法获得签名公私钥对 (sk,pk), 将 pk 发送给 $\mathcal{A}$.

2) $\mathcal{A}$ (消息拥有者) 选取消息 m 与盲化因子 r, 访问随机谕言机获得 $\mathrm{H}(m)$ 并计算 $m^*=r\cdot\mathrm{H}(m)$, 将 m^* 发送给 $\mathcal{C}$.

3) $\mathcal{C}$ 计算 $\sigma^*=sk\cdot m^*$ 并返回给 $\mathcal{A}$.

4) $\mathcal{A}$ 将签名去盲化得到 $\sigma=\sigma^*\cdot r^{-1}$.

5) $\mathcal{A}$ 多项式每次执行操作 2)–3).

6) $\mathcal{A}$ 输出挑战消息 $(\tilde{m},\tilde{\sigma})$.

7) 若 $e(\tilde{\sigma},Q)\stackrel{?}{=}e(\mathrm{H}(\tilde{m}),pk)$, 则 $\mathcal{A}$ 成功伪造消息 $\tilde{m}$ 的签名, 在游戏中获得胜利.

基于 BLS 的盲签名具有不可伪造性, 当且仅当对于任意 PPT 敌手 $\mathcal{A}$ 存在一个可忽略函数 $negl(\ell)$, 使得:

$$\Pr[\text{CMA-EUF}_{\mathcal{A}}^{\text{BLS}}(1^{\ell})=1]\leqslant negl(\ell) \tag{2}$$

2.4 Diffie-Hellman (DH) 密钥交换协议

密钥交换协议旨在允许两个陌生的双方在不直接交换各自隐私密钥的情况下, 安全地共享一个通信密钥以建立通信连接, 确保后续秘密通信顺利进行. DH 密钥交换协议包含两个算法: KeyGen、ComSK, 具体如下.

(1) KeyGen. 通信双方 $\mathcal{U}_A,\mathcal{U}_B$ 提前共享参数 $<G,P,p>$, 其中 G 是 p 阶循环群, P 为 G 的生成元. $\mathcal{U}_A$ 选择随机数 $a\stackrel{\$}{\leftarrow}Z_p^*$ 作为私钥, 计算公钥 $A=a\cdot P$, 则 $\mathcal{U}_A$ 的公私钥对为 (a,A), $\mathcal{U}_B$ 执行类似计算获得公私钥对 (b,B).

(2) ComSK. $\mathcal{U}_A$ 获取 $\mathcal{U}_B$ 的公钥并计算 $K_A=a\cdot B=ab\cdot P$, $\mathcal{U}_B$ 执行类似计算获得 K_B.

● 正确性: DH 密钥交换协议是正确的当且仅当以下等式成立.

$$\mathrm{SK}=K_B=b\cdot A=ba\cdot P=K_A \tag{3}$$

● 安全性: 若决定性 DH (decisional Diffie-Hellman, DDH) 问题是困难的, 那么 DH 密钥交换协议在窃听者存在的前提下是安全的.

2.5 带附加值的认证加密

带附加值的认证加密方案包含 3 个 PPT 算法: KeyGen、Enc、Dec, 如下所示.

(1) $k\leftarrow\mathrm{KeyGen}(1^{\ell})$ 是密钥生成算法. 执行该算法获取一个加密密钥 k, 明文空间为 M, 密文空间为 C.

(2) $c \leftarrow \mathrm{Enc}(k,m,ad)$ 是加密算法. 使用密钥 k 与附加数据 ad 作为输入, 对明文 $m \in M$ 加密, 输出密文 c.

(3) $m \leftarrow \mathrm{Dec}(k,c,ad)$ 是解密算法. 使用密钥 k 与附加数据 ad 作为输入, 对密文 c 进行解密, 若解密失败输出错误符号 $\perp$, 解密成功输出明文 m.

● 正确性: 一个 AEAD 加密算法是正确的当且仅当对于所有的 (k,ad), 满足:

$$\mathrm{Dec}(k,\mathrm{Enc}(k,m,ad),ad) = m \tag{4}$$

● 安全性: 加密算法中附加数据提供完整性保护, 无需考虑其机密性, AEAD 方案能够保证传输数据 m 的机密性, 抵御选择密文攻击 (chosen-ciphertext attack, CCA), 即在允许 PPT 敌手 $\mathcal{A}$ 访问加密谕言机与解密谕言机的情况下, $\mathcal{A}$ 在挑战阶段无法判断密文对应的明文, 即 IND-CCA 安全; 此外 AEAD 方案能够保证数据的认证性, 即敌手无法伪造消息的密文. 综上, AEAD 方案能够实现认证加密 (authenticated encryption, AE) 安全.

2.6 困难问题假设

● DDH 困难问题. 给定循环群 G, 其生成元为 P, 给定 $h_1,h_2,h_3 \xleftarrow{\$} G$, 其中 $h_1 = a \cdot P$, $h_2 = b \cdot P$, $h_3 = c \cdot P$, 在多项式时间内区分 h_3 与 $b \cdot h_1$ (或 $a \cdot h_2$) 是困难的.

● 计算性 DH (computational Diffie-Hellman, CDH) 困难问题. 给定循环群 G, 其生成元为 P, 给定 $h_1,h_2 \xleftarrow{\$} G$, 其中 $h_1 = a \cdot P$, $h_2 = b \cdot P$, 在多项式时间内计算 $ab \cdot P = a \cdot h_2 = b \cdot h_1$ 是困难的.

2.7 区块链

区块链是由多个节点构成的去中心化的分布式账本, 使用区块记录所有的交易, 并以哈希链进行存储. 一个群组的节点共同维护区块链, 在节点互不信任的情况下对新产生的区块达成共识, 并将其链接在区块链上, 区块链上的交易数据 (至少) 对于该群组内的所有节点都是公开可验证的, 但无法篡改已经上链的区块内容, 提供了不可篡改性和透明性. 每个区块链系统用户在注册时获得一个包含区块链地址和对应密钥对的账户, 每笔交易包含发送方/接收方账户地址、转账值、数据和交易签名等相关数据, 且交易上链后将获得唯一的区块地址, 用户可以根据该地址获取区块内容.

3 系统概述与设计目标

3.1 系统模型

本文旨在基于区块链设计一个面向轻量级节点的数据跨域安全传输方案, 方案共包含 4 类实体, 分别为密码服务器 $\mathcal{CS}$、身份认证服务器 $\mathcal{AS}$、用户 $\mathcal{U}$ 与区块链.

● 用户 $\mathcal{U}$. 在本系统中是进行通信的实体, 双方通过与本域内服务器交互实现基于区块链的公钥认证与密钥管理, 用户能够通过与域内服务器交互分布式存储一个备份密钥用于加密用户私钥, 将身份公钥与私钥密文上传至区块链. 欲发起通信的一方主动建立通信连接, 向域内服务器发起请求, 经过服务器之间的交互获得对方节点的身份信息, 本文用 $\mathcal{S}$ 表示消息发送方, 用 $\mathcal{R}$ 表示消息接收方, 用户利用对方节点与自身的身份密钥信息进行密钥协商计算获得共享密钥 SK, 该密钥用于派生加密过程依赖的消息密钥, 并使用 AEAD 算法进行数据加密与解密.

● 密码服务器 $\mathcal{CS}$. 为用户提供口令强化服务并存储用户备份密钥密文, 用户欲存储或恢复备份密钥时, 可先向 $\mathcal{CS}$ 请求口令强化服务并将计算结果作为用户在身份认证服务器处的登录口令, 保证服务器在不合谋的情况下能够抵御 DGA; 用户在 $\mathcal{CS}$ 端登录成功后获得预先存储的备份密钥密文, 通过身份认证服务器备份的盐值数据解密获取备份密钥. 服务器端存储一个服务请求列表与用户登录记录, 通过限速机制能够有效抵御敌手的在线猜测攻击.

● 身份认证服务器 $\mathcal{AS}$. 在备份密钥管理阶段存储用户产生的随机数据, 该随机数据作为用户生成加密密钥所需的盐值, 服务器端维护一个用户登录记录列表来存储用户认证信息, 并使用限速机制抵御敌手的在线猜测攻击; 在身份密钥管理阶段为用户提供身份认证服务, 用户生成的用户身份区块地址发送给 $\mathcal{AS}$, $\mathcal{AS}$ 对该身份区块进行认证, 认证通过后产生与服务器身份绑定的认证区块, $\mathcal{AS}$ 将认证区块地址以及用户中期预共享密钥信息存储在用户密钥数据列表; 在跨信任域用户欲进行通信时, 服务器通过检索列表找到目标用户的认证区块地址及中期密

钥信息并转发, 协助用户建立通信连接, 该过程无需目标节点参与, 实现了通信节点非同时在线的通信连接建立.

● 区块链. 区块链在 SCDT 中主要用来记录并认证用户身份. 区块内容包含发送方地址、接收方地址、转账金额、数据内容、区块签名等, 根据区块地址可以便捷地定位目标区块并读取其中的数据内容, 缓解用户的存储压力并保证用户身份的不可篡改性.

图 1 描述了 2 个信任域内实体进行身份认证并建立通信连接进行通信的过程, 其中 2 个信任域分别表示为信任域 1 与信任域 2, 通信连接发起方位于信任域 2 并表示为 $\mathcal{S}$, 通信连接目标用户位于信任域 1 并表示为 $\mathcal{R}$, 图中虚线箭头表示实体产生区块并上传至区块链.

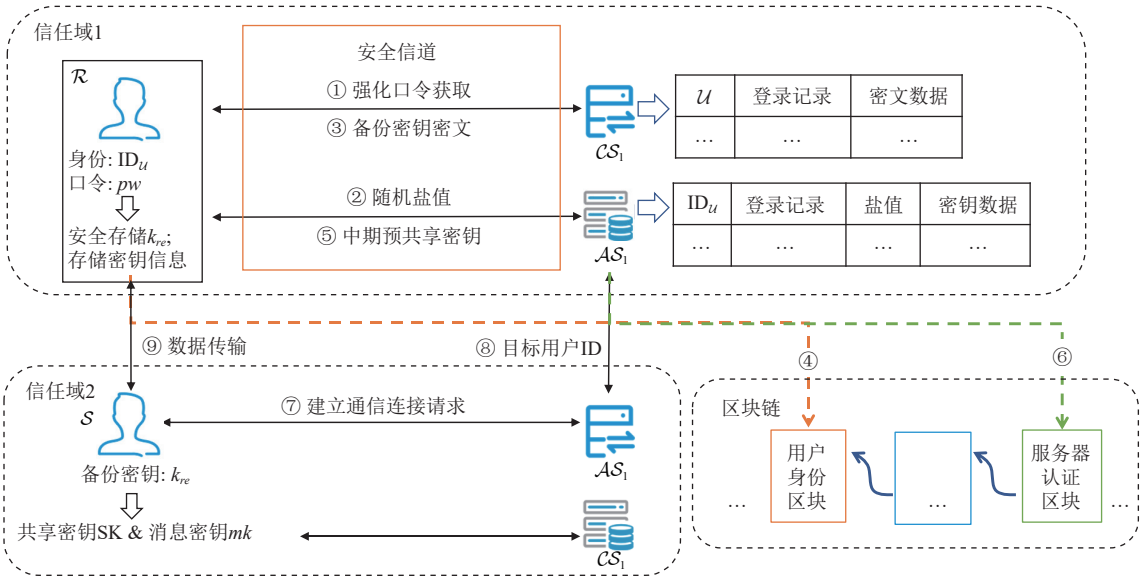


图 1 系统模型框架

SCDT 域内实体首先进行身份认证, 如图 1 中步骤①–⑥所示, ① $\mathcal{R}$ 向 $\mathcal{CS}_1$ 请求强化口令服务并使用计算结果登录 $\mathcal{AS}_1$; ②生成随机盐值存储在 $\mathcal{AS}_1$ 端, 该盐值用于派生用户在 $\mathcal{CS}_1$ 处的登录口令与备份密钥 k_{re} 的加密密钥; ③ $\mathcal{R}$ 在 $\mathcal{CS}_1$ 处登录并将备份密钥密文存储在 $\mathcal{CS}_1$ 端 (若 $\mathcal{R}$ 欲恢复 k_{re}, 可分别从 $\mathcal{AS}_1$ 与 $\mathcal{CS}_1$ 端获得盐值与备份密钥密文进行恢复); ④ $\mathcal{R}$ 使用 k_{re} 加密身份私钥, 并将身份私钥密文与公钥上传至区块链; ⑤ $\mathcal{R}$ 使用 k_{re} 加密预共享密钥对中的私钥, 并将预共享私钥密文与公钥存储在 $\mathcal{AS}_1$; ⑥ $\mathcal{AS}_1$ 认证 $\mathcal{R}$ 上传区块的合法性并将用户身份区块地址上传至区块链, 该区块作为服务器认证区块. 经过认证的实体可以在服务器协助下进行跨信任域通信, 如步骤⑦和⑧所示. ⑦ 发起通信连接的用户 $\mathcal{S}$ 向信任域内 $\mathcal{AS}_2$ 发送建立通信连接请求; ⑧ $\mathcal{AS}_2$ 与 $\mathcal{AS}_1$ 交互获得通信双方的身份区块地址并分别转发给 $\mathcal{S}$ 与 $\mathcal{R}$, 通信双方获得对方身份数据后进行计算获得共享密钥 SK; ⑨ 上述连接建立完成后, 通信双方可在公开信道进行数据传输.

本文用户间执行安全数据传输过程中使用的加密算法基于双棘轮算法思想实现, 通信参与方各自维护 3 种密钥, 分别为根密钥、发送密钥以及接收密钥. 根密钥用于通信双方身份发生转变时更新用户的发送/接收密钥; 发送密钥及接收密钥分别用于用户在发送/接收消息时作为密钥派生函数的输入以产生消息密钥. 每次数据处理时需要按照协议要求执行密钥更新操作以确保每次加密操作使用不同的消息密钥实现前向安全, 每当通信双方身份发生转换时, 双方共同执行一次根密钥的更新并产生新的发送/接收密钥以实现后向安全, 后续本文将密钥更新所得结果称为密钥链, 且发送方的发送密钥链与接收方的接收密钥链是对称的. 其中, 密钥更新操作使用基于哈希的密钥派生函数 (hash-based key derivation function, HKDF) 实现. 通信过程中, 若通信双方身份发生转换 (消息发送方转换为消息接收方), 执行加解密操作前需首先进行一次非对称棘轮步进, 根据先前保留的密钥参数与新获取的密钥参数进行 DH 密钥协商计算, 来更新根密钥链, 随后进行对称棘轮步进更新发送/接收密钥链, 并获取当前消

息密钥用以进行加密/解密操作.

3.2 变量设置

本节对方案中使用到的变量及存储数据表示进行描述, 表 2 展示了 SCDT 涉及的变量表示及对应含义.

表 2 变量表示及含义

变量	含义
k_{re}	用户备份密钥
l_{req}	一个周期内向 $\mathcal{CS}$ 请求同一ID强化口令的次数上限
$l_{\text{login}}^{\mathcal{AS}}$	在 $\mathcal{AS}$ 处登录口令验证失败次数上限
$l_{\text{login}}^{\mathcal{CS}}$	在 $\mathcal{CS}$ 处登录口令验证失败次数上限
$req_{\mathcal{U}}$	$\mathcal{CS}$ 记录同一ID在一个周期内请求强化口令的次数
$reg_{\mathcal{U}}^{\mathcal{AS}}$	用户 $\mathcal{U}$ 在 $\mathcal{AS}$ 处的登录记录
$reg_{\mathcal{U}}^{\mathcal{CS}}$	用户 $\mathcal{U}$ 在 $\mathcal{CS}$ 处的登录记录
$\mathcal{L}_{\mathcal{U}}$	$\mathcal{AS}$ 记录用户 $\mathcal{U}$ 的密钥数据列表
SK	通信双方建立通信连接的共享密钥

3.3 威胁模型

由于实际中用户口令具有内在低熵特性, 易遭受口令猜测攻击, 故众多现有认证系统中, 用户口令一旦被猜中, 那么用户的所有隐私信息将会毫无保留地泄露给敌手或恶意服务器. 本节中, 我们允许恶意外部敌手或恶意服务器执行 DGA 猜测用户口令. 以下分别从身份认证阶段与数据传输阶段分析系统面临的安全威胁.

在身份认证阶段, 用户与 $\mathcal{AS}$ 及 $\mathcal{CS}$ 交互式安全存储一个备份密钥, 并将自己的密钥信息记录在区块链与 $\mathcal{AS}$ 端. 在此过程中, 数据传输在安全信道进行, 能够有效抵御外部敌手的安全威胁, 因此系统面临: a) 恶意 $\mathcal{AS}$ 试图通过为用户提供备份密钥管理服务的过程猜测用户口令; 试图在为用户提供身份密钥管理服务过程中篡改用户身份密钥信息; b) 恶意 $\mathcal{CS}$ 试图通过为用户提供备份密钥管理服务的过程猜测用户口令; 试图恶意篡改用户存储的密钥密文以掌握用户的备份密钥并在后续身份密钥管理过程中威胁用户身份密钥的安全性.

在数据传输阶段, 数据在公开信道流通, 面临: a) 恶意外部敌手通过窃听信道或主动篡改数据内容威胁系统安全性; b) 通信参与方密钥信息泄露导致历史通信内容被批量恢复以及通信双方后续通信失去安全保障.

3.4 设计目标

本文的目标是基于区块链构建面向轻量级节点的安全数据跨域传输方案, 需要满足以下安全目标.

1) 功能性. 基于区块链实现去中心化公钥与身份认证机制, 采用基于口令的加密方法, 使得用户仅需安全的保存易于记忆的低熵口令, 即可进行安全的身份认证和数据跨域传输; $\mathcal{CS}$ 与 $\mathcal{AS}$ 能够分布式为用户提供备份密钥存储服务, $\mathcal{AS}$ 能够认证用户身份的合法性并辅助跨域节点建立安全的通信连接, 实现抗密钥泄露的数据传输.

2) 安全性. 在用户隐私方面, 方案能够保证诚实参与方的隐私不会被恶意获取, 用户身份密钥信息不会被服务器或外部敌手恶意篡改. 在数据传输方面, 方案能够达到以下安全性: a) 机密性: 在公开信道中, 攻击者无法获取除密文本身外关于传输数据的任何额外信息; b) 完整性: 在通信双方都未被捕获的情况下, 攻击者不能篡改合法用户发送的消息或注入新消息; 在通信双方有一方被捕获的情况下, 攻击者的消息无法通过合法参与方的认证; c) 前向安全性: 攻击者捕获通信中的任一方 (或双方), 捕获状态之前的任意时刻的信息对攻击者来说不可恢复; d) 后向安全性: 若攻击者捕获某个参与方, 当捕获状态结束后, 经过有限轮次的交互, 通信双方的行为能够重新恢复安全.

3) 性能效率. SCDT 实现全时域设备独立的系统访问; 在实现功能性与安全性的同时, 用户端在身份认证阶段计算总开销不超过 0.3 s, 通信过程中用户维护当前密钥的存储开销约 0.1 KB; 服务器端计算总开销不超过 0.1 s, 维护每条登录记录的存储开销不超过 0.5 KB. 实验结果表明系统在保持较低存储开销的同时, 还具备较高计算效率.

4 具体方案

4.1 系统初始化

1) 令 ℓ 为安全参数, 系统公共参数 $pp=\{p,G_1,G_2,P,\mathrm{H}_1,\mathrm{H}_2,\hat{h},\mathrm{DH}(\cdot),\mathrm{Sign/Vrfy}(\cdot),\mathrm{Enc/Dec}(\cdot),\mathrm{KDF},len\}$ 为各阶段算法的隐含输入, 其中 G_1,G_2 是阶为素数 p 的循环群, P,Q 分别是 G_1,G_2 的生成元, $e\colon G_1\times G_2\to G_T$ 是双线性映射, $\mathrm{H}_1,\mathrm{H}_2,\hat{h}$ 是安全的密码学哈希函数, 其中 $\mathrm{H}_1\colon\{0,1\}^*\to G_1$, $\mathrm{H}_2\colon\{0,1\}^*\to Z_p$, $\hat{h}\colon\{0,1\}^*\to\{0,1\}^{len}$, DH(·) 为 DH 密钥交换协议 (为便于叙述, 本文以黑盒方式调用 DH 密钥交换协议), Sign/Vrfy(·) 表示一对数字签名/验签算法, Enc/Dec(·) 表示一对 AEAD 加/解密算法, KDF 是一个密钥派生函数, *len* 是区块链网络中账户地址的比特长度.

SCDT 密钥生成算法为: $\mathrm{KGen}(1^\ell)$: $sk\xleftarrow{\$}Z_p^*$, $pk=sk\cdot Q$, 后续选取密钥对的操作默认均采用该算法.

2) $\mathcal{AS}$ 的区块链账户为 $(bsk_{\mathcal{AS}},Add_{\mathcal{AS}})$, 随机选取一个秘密 $s_1\xleftarrow{\$}Z_p$; $\mathcal{CS}$ 选取一对签名公私钥 $(sk_{\mathcal{CS}},pk_{\mathcal{CS}})$, 随机选取一个秘密 $s_2\xleftarrow{\$}Z_p$.

3) 用户 $\mathcal{U}$ 的区块链账户为 $(bsk_{\mathcal{U}},Add_{\mathcal{U}})$.

4.2 身份认证

方案身份认证阶段包含两个过程: 备份密钥管理与身份密钥管理. 用户分别与 $\mathcal{CS}$ 和 $\mathcal{AS}$ 交互, 获取备份密钥并将其安全地分布式存储在两个服务器上. 使用该备份密钥加密用户的身份私钥信息, 并将私钥密文与对应公钥整合上传至区块链, 而后经域内 $\mathcal{AS}$ 认证. 具体如下.

(1) 备份密钥管理阶段: 要求用户分别与 $\mathcal{CS}$ 和 $\mathcal{AS}$ 交互, 获取备份密钥并将其安全地分布式存储在两个服务器上. 备份密钥管理方案由一组算法构成: PassHard、Reg、KRestore、KRetri.

● PassHard. 口令强化算法由 $\mathcal{U}$ 与 $\mathcal{CS}$ 交互获得强化口令.

1) 用户拥有口令 $pw_{\mathcal{U}}$, 任意选取随机数 $r\xleftarrow{\$}Z_p^*$, 计算 $pw_{\mathcal{U}}^*=r\cdot\mathrm{H}_1(pw_{\mathcal{U}})$, 并将 $(ID_{\mathcal{U}},pw_{\mathcal{U}}^*)$ 发送给 $\mathcal{CS}$.

2) $\mathcal{CS}$ 维护一个服务请求列表 $\mathrm{REQ}_{\mathcal{CS}}$ 记录该用户在一个周期内请求服务的次数, 收到 $\mathcal{U}$ 的服务请求后, 查询 $\mathrm{ID}_{\mathcal{U}}$ 是否存在, 若不存在, 则为该用户创建一条注册记录 $req_{\mathcal{U}}=(\mathrm{ID}_{\mathcal{U}},l_{\mathcal{U}}^{req}=0)$; 若存在, 则检查用户是否达到服务请求上限, 若 $l_{\mathcal{U}}^{req}\geqslant l_{req}$, 则终止后续操作; 否则, 令 $l_{\mathcal{U}}^{req}=l_{\mathcal{U}}^{req}+1$ 并计算 $\sigma_{\mathcal{U}}^*=sk_{\mathcal{CS}}\cdot pw_{\mathcal{U}}^*$ 返回给用户.

3) $\mathcal{U}$ 首先通过 $e(\sigma_{\mathcal{U}}^*,Q)\stackrel{?}{=}e(pw_{\mathcal{U}}^*,pk_{\mathcal{CS}})$ 验证 $\sigma_{\mathcal{U}}^*$ 的有效性, 若相等则计算 $hpw_{\mathcal{U}}=\mathrm{H}_2(pw_{\mathcal{U}},\sigma_{\mathcal{U}}^*\cdot r^{-1})=\mathrm{H}_2(pw_{\mathcal{U}},sk_{\mathcal{CS}}\cdot\mathrm{H}_1(pw_{\mathcal{U}}))$; 否则终止后续操作.

● Reg. 注册阶段包含用户向 $\mathcal{CS}$ 与 $\mathcal{AS}$ 分别注册身份.

1) $\mathcal{U}$ 使用 $(\mathrm{ID}_{\mathcal{U}},hpw_{\mathcal{U}})$ 在 $\mathcal{AS}$ 处注册或登录.

2) $\mathcal{AS}$ 维护注册列表 $\mathrm{REG}_{\mathcal{AS}}$ 用于记录用户的登录信息及存储数据, 收到用户的注册请求后, 首先检查是否存在该用户的注册记录, 若不存在, 则为该用户创建注册记录 $reg_{\mathcal{U}}^{\mathcal{AS}}=(\mathrm{ID}_{\mathcal{U}},c_1=\mathrm{Enc}(s_1,hpw_{\mathcal{U}}\|s\|r),l_{\mathcal{U}}^{\mathcal{AS}}=0)$ 并登录, 其中 $s=t=\perp$; 若该用户已完成注册, 则使用 s_1 解密 c_1, 根据解密结果检查用户登录信息与记录是否匹配, 若不匹配, 则令 $l_{\mathcal{U}}^{\mathcal{AS}}=l_{\mathcal{U}}^{\mathcal{AS}}+1$, 若 $l_{\mathcal{U}}^{\mathcal{AS}}\geqslant l_{\mathrm{login}}^{\mathcal{AS}}$, 则终止后续操作.

3) $\mathcal{U}$ 在 $\mathcal{AS}$ 处登录成功后, 随机选取 $(s,r)\xleftarrow{\$}Z_p^2$, 并发送给 $\mathcal{AS}$.

4) $\mathcal{AS}$ 收到 (s,r), 查询注册列表并更新该用户的注册记录, 令 $c_1=\mathrm{Enc}(s_1,hpw_{\mathcal{U}}\|s\|r)$.

5) $\mathcal{U}$ 计算 $t=\mathrm{KDF}(s,pw_{\mathcal{U}})$, 并使用 $(\mathrm{ID}_{\mathcal{U}},t)$ 在 $\mathcal{CS}$ 处注册.

6) $\mathcal{CS}$ 维护注册列表 $\mathrm{REG}_{\mathcal{CS}}$ 用于记录用户的登录信息及存储数据, 收到用户的注册请求后, 首先检查是否存在该用户的注册记录, 若不存在, 则为该用户创建注册记录 $reg_{\mathcal{U}}^{\mathcal{CS}}=(\mathrm{ID}_{\mathcal{U}},c_2=\mathrm{Enc}(s_2,t\|ct_{\mathcal{U}}),l_{\mathcal{U}}^{\mathcal{CS}}=0)$, 其中 $ct_{\mathcal{U}}=\perp$, 若存在, 则返回该用户已注册的提示.

● KRestore. 备份密钥存储算法由 $\mathcal{U}$ 随机选取备份密钥并分布式存储在 $\mathcal{CS}$ 与 $\mathcal{AS}$ 处.

1) $\mathcal{U}$ 首先在 $\mathcal{AS}$ 处登录, 解密 c_1 来判断用户口令是否匹配, 若登录成功, $\mathcal{AS}$ 获取 (s,r) 返回给用户; 否则, 令 $l_{\mathcal{U}}^{\mathcal{AS}}=l_{\mathcal{U}}^{\mathcal{AS}}+1$ 并终止后续操作;

2) $\mathcal{U}$ 选取备份密钥 $k_{re} \xleftarrow{\$} Z_p^*$, 计算 $t = \mathrm{KDF}(s_{\mathcal{U}}, pw_{\mathcal{U}})$, $k_{\mathcal{U}} = \mathrm{KDF}(r_{\mathcal{U}}, pw_{\mathcal{U}})$, $ct = \mathrm{Enc}(k_{\mathcal{U}}, k_{re})$. 使用 $(\mathrm{ID}_{\mathcal{U}}, t, ct)$ 在 $\mathcal{CS}$ 处登录;

3) $\mathcal{CS}$ 获取到用户的登录请求, 解密 c_2 来判断用户口令是否正确, 若不正确, 令 $l_{\mathcal{U}}^{\mathcal{CS}} = l_{\mathcal{U}}^{\mathcal{CS}} + 1$ 并返回; 否则, 更新用户的注册记录, 令 $c_2 = \mathrm{Enc}(s_2, t\|ct)$, 更新用户的服务请求 $req_{\mathcal{U}}$, 令 $l_{\mathcal{U}}^{req} = 0$.

● KRetri. 备份密钥获取算法允许用户与服务器交互获取备份密钥 k_{re}.

1) $\mathcal{U}$ 首先在 $\mathcal{AS}$ 处登录, 登录成功后, $\mathcal{AS}$ 返回 (s, r) 给用户.

2) $\mathcal{U}$ 根据公式 (6) 计算 $(t, k_{\mathcal{U}})$, 并使用 $(\mathrm{ID}_{\mathcal{U}}, t)$ 在 $\mathcal{CS}$ 处登录.

3) $\mathcal{CS}$ 获取到用户的登录请求, 检查用户登录信息 t 与注册记录是否一致, 若不一致, 令 $l_{\mathcal{U}}^{\mathcal{CS}} = l_{\mathcal{U}}^{\mathcal{CS}} + 1$ 并返回. 否则, 更新用户的服务请求记录 $req_{\mathcal{U}}$, 令 $l_{\mathcal{U}}^{req} = 0$ 并将 ct 返回给用户.

4) $\mathcal{U}$ 解密 ct 得到 $k_{re} = \mathrm{Dec}(k_{\mathcal{U}}, ct)$, 若解密失败, 则密文被 $\mathcal{CS}$ 篡改, 终止后续操作.

(2) 身份密钥管理阶段: 涉及两类区块, 其中用户身份区块存储数据包含用户身份密钥信息, 服务器认证区块记录通过 $\mathcal{AS}$ 认证的用户公钥信息与该用户的身份区块地址.

1) $\mathcal{U}$ 执行 KRetri 函数获取备份密钥 k_{re}.

2) $\mathcal{U}$ 执行 $\mathrm{KGen}(1^{\ell})$ 获取长期身份密钥对 $(isk_{\mathcal{U}}, ipk_{\mathcal{U}})$, 计算 $cbsk_{\mathcal{U}} = \mathrm{Enc}(k_{re}, bsk_{\mathcal{U}})$, $cisk_{\mathcal{U}} = \mathrm{Enc}(k_{re}, isk_{\mathcal{U}})$, 创建一笔区块链交易 T_{key} 作为用户身份区块, 从地址 $Add_{\mathcal{U}}$ 向地址 $\hat{h}(\mathrm{ID}_{\mathcal{U}})$ 转账 0 代币, 嵌入数据 $cbsk_{\mathcal{U}}\|cisk_{\mathcal{U}}\|ipk_{\mathcal{U}}$, 区块地址为 $Add_{T_{\mathrm{key}}}$. 令 $Auth_{\mathcal{U}} = \mathrm{ID}_{\mathcal{U}}\|Add_{\mathcal{U}}\|Add_{T_{\mathrm{key}}}$, 计算签名 $\sigma = \mathrm{Sig}(isk_{\mathcal{U}}, Auth_{\mathcal{U}})$, 将 $(Auth_{\mathcal{U}}, \sigma)$ 发送给 $\mathcal{AS}$.

3) $\mathcal{AS}$ 从 $Auth_{\mathcal{U}}$ 中提取 $Add_{T_{\mathrm{key}}}$ 定位用户的密钥区块, 进行以下验证: a) 用户密钥区块的发送方是否为 $Add_{\mathcal{U}}$; 接收方是否为 $\hat{h}(\mathrm{ID}_{\mathcal{U}})$; b) $\mathcal{AS}$ 从 T_{key} 中获取用户公钥 $ipk_{\mathcal{U}}$ 执行 $\mathrm{Vrfy}(ipk_{\mathcal{U}}, Auth_{\mathcal{U}}, \sigma)$ 验证签名 σ 是否合法, 若上述认证均通过, $\mathcal{AS}$ 产生一笔交易 T_{cer} 作为用户身份的认证凭证, 从地址 $Add_{\mathcal{AS}}$ 向地址 $\hat{h}(\mathrm{ID}_{\mathcal{U}})$ 转账 0 代币, 嵌入数据 $\mathrm{ID}_{\mathcal{AS}}\|Add_{T_{\mathrm{key}}}\|ipk_{\mathcal{U}}$, 区块地址为 $Add_{T_{\mathrm{cer}}}$; 反之, 若上述任一条件验证失败, 则中止后续操作.

4) $\mathcal{AS}$ 根据 $\mathrm{ID}_{\mathcal{U}}$ 检索 $\mathcal{L}_{\mathcal{U}}$ 是否存在, 若不存在, 则创建一条列表记录用户数据 $\mathcal{L}_{\mathcal{U}} = (\mathrm{ID}_{\mathcal{U}}, Add_{\mathcal{U}}, Add_{T_{\mathrm{cer}}}, \perp)$.

5) $\mathcal{U}$ 执行 $\mathrm{KGen}(1^{\ell})$ 获取中期预共享密钥对 $(ssk_{\mathcal{U}}, spk_{\mathcal{U}})$, 计算 $cssk_{\mathcal{U}} = \mathrm{Enc}(k_{re}, ssk_{\mathcal{U}})$, $\sigma_{spk_{\mathcal{U}}} = \mathrm{Sign}(isk_{\mathcal{U}}, spk_{\mathcal{U}})$, $\sigma_{spk_{\mathcal{U}}}$ 为用户对 $spk_{\mathcal{U}}$ 的签名, 令 $\mathcal{D}_{\mathcal{U}} = cssk_{\mathcal{U}}\|spk_{\mathcal{U}}\|\sigma_{spk_{\mathcal{U}}}$, 将 $(\mathrm{ID}_{\mathcal{U}}, \mathcal{D}_{\mathcal{U}})$ 发送给 $\mathcal{AS}$.

6) $\mathcal{AS}$ 从 $\mathcal{L}_{\mathcal{U}}$ 中查找 $Add_{T_{\mathrm{cer}}}$ 以查询用户身份公钥 $ipk_{\mathcal{U}}$, 执行 $\mathrm{Vrfy}(ipk_{\mathcal{U}}, spk_{\mathcal{U}}, \sigma_{spk_{\mathcal{U}}})$, 若验证通过则更新列表为 $\mathcal{L}_{\mathcal{U}} = (\mathrm{ID}_{\mathcal{U}}, Add_{\mathcal{U}}, Add_{T_{\mathrm{cer}}}, \mathcal{D}_{\mathcal{U}})$, 否则终止后续操作.

7) 用户需要定期更新中期密钥对, 首先执行步骤 1 获取备份密钥 k_{re}, 并向 $\mathcal{AS}$ 请求获取用户身份区块地址获得身份密钥信息, 解密获得 $isk_{\mathcal{U}} = \mathrm{Dec}(k_{re}, cisk_{\mathcal{U}})$, 再执行步骤 5)–6) 更新数据列表 $\mathcal{L}_{\mathcal{U}}$.

4.3 数据传输

方案数据传输阶段包含两个过程: 建立通信连接与数据处理. 欲通信双方经双方域内 $\mathcal{AS}$ 协助并进行密钥协商计算获取相同的共享密钥 SK; 在数据处理过程中, 为保证传输数据的安全性, 通信双方加/解密数据需要及时更新密钥, 密钥进行一次更新操作称为一次密钥链的步进, SCDT 通信双方各自维护 3 条密钥链 (根密钥链、发送密钥链、接收密钥链), 具体执行细节如下.

(1) 建立通信连接阶段: 使得跨信任域通信双方获得相同的共享密钥 SK.

1) 发送方 $\mathcal{S}$ 欲与接收方 $\mathcal{R}$ 建立通信 (以下将发送方所在的信任域称为信任域 1; 接收方所在的信任域称为信任域 2), $\mathcal{S}$ 首先将接收方身份 $\mathrm{ID}_{\mathcal{R}}$ 发送给信任域 1 的身份认证服务器 $\mathcal{AS}_1$, 提出通信请求;

2) $\mathcal{AS}_1$ 检索 $\mathcal{R}$ 所在信任域, 向 $\mathcal{AS}_2$ 请求 $\mathcal{R}$ 的认证区块地址 $Add_{T_{\mathrm{cer}}}^{\mathcal{R}}$ 与中期预共享公钥 $spk_{\mathcal{U}}$ 及其签名 $\sigma_{spk_{\mathcal{U}}}$, $\mathcal{AS}_1$ 定位区块 $T_{\text{cer-}\mathcal{R}}$ 获取到 $\mathcal{R}$ 身份区块地址 $Add_{T_{\mathrm{key}}}^{\mathcal{R}}$, 并将 $(Add_{T_{\mathrm{key}}}^{\mathcal{R}}, spk_{\mathcal{R}}, \sigma_{spk_{\mathcal{R}}})$ 转发给用户 $\mathcal{S}$;

3) $\mathcal{S}$ 从区块 $T_{\text{key-}\mathcal{R}}$ 提取 $ipk_{\mathcal{R}}$ 并对 $\sigma_{spk_{\mathcal{R}}}$ 进行验证, 若通过验证, 则产生一对临时密钥 $(esk_{\mathcal{S}}, epk_{\mathcal{S}})$, $\mathcal{S}$ 使用 $\mathcal{R}$ 的公开信息以及自己的私钥密钥信息进行 3 次 DH 密钥协商计算并派生出共享密钥 $\mathrm{SK} = \mathrm{KDF}(dh1\|dh2\|dh3)$, 其中, $dh1 = \mathrm{DH}(isk_{\mathcal{S}}, spk_{\mathcal{R}})$, $dh2 = \mathrm{DH}(esk_{\mathcal{S}}, ipk_{\mathcal{R}})$, $dh3 = \mathrm{DH}(esk_{\mathcal{S}}, spk_{\mathcal{R}})$. 计算完成后, $\mathcal{S}$ 删除临时私钥以及中间输出, 使用 SK 加密数据本身, 计算 $m_0 = (\mathrm{ID}_{\mathcal{S}}, epk_{\mathcal{S}}, c_{\mathrm{SK}} = \mathrm{Enc}(\mathrm{SK}, \mathrm{SK}))$, 并将其发送给 $\mathcal{AS}_1$.

4) $\mathcal{AS}_1$ 根据 $\mathrm{ID}_{\mathcal{S}}$ 查询用户的身份区块信息与中期预共享密钥信息, 将 $(Add^{\mathcal{S}}_{T_{\mathrm{cer}}}, spk_{\mathcal{S}}, \sigma_{\mathcal{S}}, m_0)$ 发送给 $\mathcal{AS}_2$, 并由 $\mathcal{AS}_2$ 转发给目标用户 $\mathcal{R}$.

5) $\mathcal{R}$ 接收到以上消息后, 首先从 $T_{\text{cer-}\mathcal{S}}$ 中提取 $ipk_{\mathcal{S}}$ 并验证 $spk_{\mathcal{S}}$ 的签名合法性, 之后定位自己的密钥管理区块 $T_{\text{key-}\mathcal{R}}$, 使用 k_{re} 解密私钥密文获取私钥. 根据自己密钥信息及收到的临时公钥, 重复公式 (7) 的计算得到共享密钥 SK, 并使用该密钥解密消息, 若 $\mathrm{Dec}(\mathrm{SK}, c_{\mathrm{SK}}) = \mathrm{SK}$, 则 $\mathcal{S}$ 与 $\mathcal{R}$ 获得共享密钥 SK, 通信连接建立成功.

(2) 数据处理阶段: 旨在通过加密机制保证通信双方跨域传输数据的安全性.

1) $\mathcal{S}$ 与 $\mathcal{R}$ 在通信过程中需要各自维护 state 数据结构, 双方分别初始化 state 变量如表 3 所示.

表 3 state 变量初始化

数据表示	数据含义	$\mathcal{S}$	$\mathcal{R}$
DHs	DH发送棘轮密钥对	$(esk_{ss}, epk_{ss}) \leftarrow \mathrm{KGen}(1^{\ell})$	$(ssk_{\mathcal{R}}, spk_{\mathcal{R}})$
DHr	DH接收棘轮公钥	$spk_{\mathcal{R}}$	epk_{ss}
rk	根密钥	$rk_{0\text{-}\mathcal{S}} = \mathrm{KDF}(\mathrm{SK}, \mathrm{DH}(DHs, DHr))$	$rk_{0\text{-}\mathcal{R}} = \mathrm{KDF}(\mathrm{SK}, \mathrm{DH}(DHs, DHr))$
CKs	发送链的链密钥	$CKs_{0\text{-}\mathcal{S}} = \mathrm{KDF}(\mathrm{SK}, \mathrm{DH}(DHs, DHr))$	—
CKr	接收链的链密钥	—	$CKr_{0\text{-}\mathcal{R}} = \mathrm{KDF}(\mathrm{SK}, \mathrm{DH}(DHs, DHr))$

注: (esk_{ss}, epk_{ss})表示当前会话的发送棘轮密钥对, 其中ss为session的简写

2) 数据发送: 假设此时通信双方身份不发生改变 (即发送方依然是 $\mathcal{S}$), state 数据初始化操作与数据处理操作如图 2 所示, 其中 DH 表示进行密钥协商计算, KDF 表示进行 HKDF 密钥派生计算, Enc/Dec 表示执行消息加/解密操作. 以传输消息 m_1 为例, $\mathcal{S}$ 进行一次发送密钥链步进, 以 $CKs_{0\text{-}\mathcal{S}}$ 为输入, 执行 KDF 计算获取消息密钥 $mk_{1\text{-}\mathcal{S}}$ 以及 $CKs_{1\text{-}\mathcal{S}}$, 用消息密钥 $mk_{1\text{-}\mathcal{S}}$ 加密消息获取密文: $c_1 = \mathrm{Enc}(mk_{1\text{-}\mathcal{S}}, m_1)$; 计算结束后, 本地删除 $mk_{1\text{-}\mathcal{S}}$ 及 $CKs_{0\text{-}\mathcal{S}}$, 将消息密文 c_1 发送给 $\mathcal{R}$.

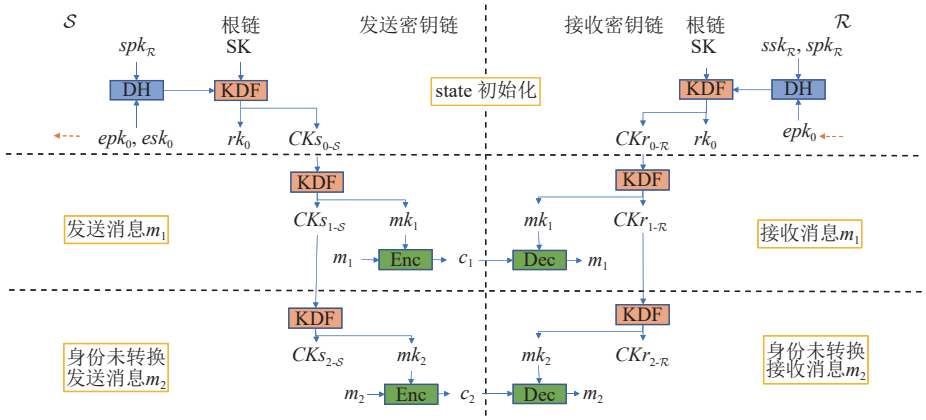


图 2 state 初始化与数据处理流程

3) 数据接收: $\mathcal{R}$ 收到密文后, 类似 $\mathcal{S}$ 执行一次接收密钥链步进, 如图 2 所示, 以 $CKr_{0\text{-}\mathcal{R}}$ 为输入, 执行 KDF 计算获取消息密钥 $mk_{1\text{-}\mathcal{R}}$ 以及 $CKr_{1\text{-}\mathcal{R}}$, 此时 $\mathcal{S}$ 发送密钥链与 $\mathcal{R}$ 接收密钥链对称, 即满足 $mk_{1\text{-}\mathcal{S}} = mk_{1\text{-}\mathcal{R}}$, 用消息密钥 $mk_{1\text{-}\mathcal{R}}$ 解密收到的密文 c_1 得到 $m_1 = \mathrm{Dec}(mk_{1\text{-}\mathcal{R}}, c_1)$; 计算结束后, 本地删除 $mk_{1\text{-}\mathcal{R}}$ 及 $CKr_{0\text{-}\mathcal{R}}$.

4) 后续通信中, 若通信双方的收发地位发生转换 (即发送方变为 $\mathcal{R}$), 如图 3 所示, 以传输消息 m_3 为例. $\mathcal{R}$ 首先执行 $\mathrm{KGen}(1^{\ell})$ 获取 (esk_1, epk_1) 更新 DHs, 使用更新后的 DHs 与 DHr 进行 DH 计算更新根密钥链并开启新的发送密钥链: $(rk_{1\text{-}\mathcal{R}}, CKs_{0\text{-}\mathcal{R}}) = \mathrm{KDF}(rk_{0\text{-}\mathcal{R}}, \mathrm{DH}(DHs, DHr))$; 以 $CKs_{0\text{-}\mathcal{R}}$ 为输入, 执行一次 KDF 计算获取消息密钥 $mk_{3\text{-}\mathcal{R}}$ 以及 $CKs_{1\text{-}\mathcal{R}}$, 用消息密钥 $mk_{3\text{-}\mathcal{R}}$ 加密消息获取密文: $c_3 = \mathrm{Enc}(mk_{3\text{-}\mathcal{R}}, m_3)$; 计算结束后, 本地删除 $mk_{3\text{-}\mathcal{R}}$ 及 $CKs_{0\text{-}\mathcal{R}}$, 将 (c_3, epk_1) 发送给 $\mathcal{S}$.

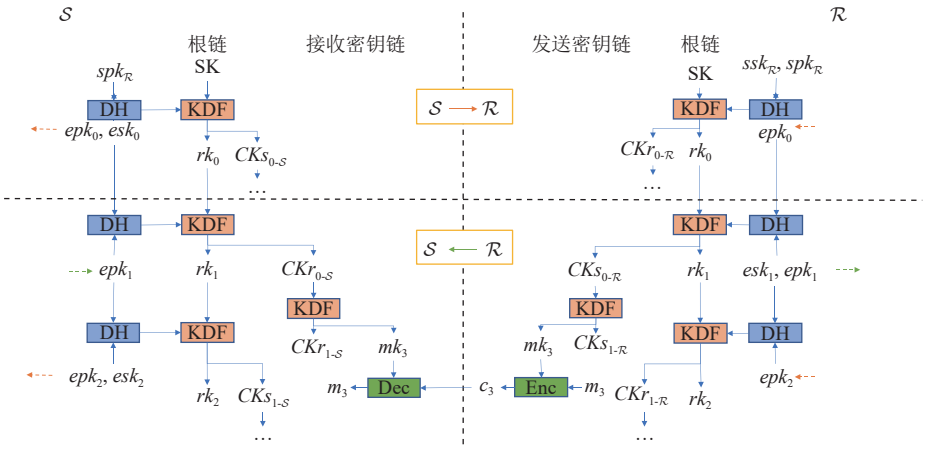


图 3 通信身份转换数据处理流程

$\mathcal{S}$ 接收到数据 (c_3, epk_1), 首先令 $DHr = epk_1$, 使用 DHs 与 DHr 进行 DH 计算更新根密钥链并开启新的接收密钥链: $(rk_{1\text{-}\mathcal{S}}, CKr_{0\text{-}\mathcal{S}}) = \mathrm{KDF}(rk_{0\text{-}\mathcal{S}}, \mathrm{DH}(DHs, DHr))$; 类似地, 以 $CKr_{0\text{-}\mathcal{S}}$ 为输入, 执行 KDF 计算获取 $mk_{3\text{-}\mathcal{S}}$ 与 $CKr_{1\text{-}\mathcal{S}}$, 此时 $\mathcal{S}$ 接收密钥链与 $\mathcal{R}$ 发送密钥链对称, 即满足 $mk_{3\text{-}\mathcal{S}} = mk_{3\text{-}\mathcal{R}}$, 解密 c_3 获得 $m_3 = \mathrm{Dec}(mk_{3\text{-}\mathcal{S}}, c_3)$, 本地删除 $mk_{3\text{-}\mathcal{S}}$ 及 $CKr_{0\text{-}\mathcal{S}}$.

5 安全性分析

SCDT 安全性依赖于身份认证和数据传输两个阶段的安全. 身份认证阶段包含备份密钥管理与身份密钥管理两个过程, 用户首先与 $\mathcal{AS}$ 及 $\mathcal{CS}$ 交互安全存储备份密钥 k_{re}, 合法用户可使用口令 $pw_{\mathcal{U}}$ 随时获得 k_{re}, 该过程中我们假定服务器之间不合谋, 而服务器可以任意创建新用户执行方案, 故恶意服务器与用户合谋无法获得额外优势, 故本节只考虑该过程能够抵御恶意 $\mathcal{AS}$ 与恶意 $\mathcal{CS}$ 以保证用户口令安全; 随后用户使用该备份密钥加密身份密钥信息, 将长期身份密钥对的私钥以密文形式与公钥明文记录在区块链上, 并将中期预共享密钥对的私钥以密文形式与公钥明文存储在 $\mathcal{AS}$ 端, 我们保证该过程中恶意 $\mathcal{AS}$ 无法篡改或冒充合法用户身份; 为抵御身份认证过程中外部敌手的威胁, 我们在安全信道进行用户与服务器之间的交互; 数据传输阶段包含通信连接建立与数据处理两个过程, 通信双方在服务器协助下获取对方身份信息并进行密钥协商计算得到共享密钥 SK, 由于上述身份认证过程能够保证合法用户身份不可篡改与不可冒充, 因此该过程能够抵御恶意服务器与外部敌手的威胁; 在数据处理过程中, 本方案使用能够保证前向安全与后向安全的加密方案以保证公开信道的数据内容能够抵御被恶意外部敌手的窃听及篡改攻击.

根据文献 [38] 对认证限速机制的可用性影响分析可知, 若限速参数设置过大, 将导致用户的口令猜测次数过多, 从而威胁系统安全性; 相反, 若参数设置过小, 则会显著降低系统的可用性. 综合权衡系统的安全性与可用性, 多数研究在实验与部署的限速参数多设置为 10, 以实现对暴力攻击的有效抑制并同时保持较高的系统响应性能.

SCDT 方案安全性基于 $\mathcal{AS}$ 与 $\mathcal{CS}$ 不合谋假设, 理论上, 单一服务器架构往往难以同时满足隐私保护与服务可用性需求, 若用户敏感数据均由单一服务器管理, 该服务器一旦被攻破, 将导致整体系统安全性失效; 事实上, 多数主流平台均通过部署独立运行的服务器进行日常管理 (如用户管理等), 因此多服务器架构在实际中已作为成熟的系统模型被广泛应用, 例如 Google Cloud KMS、Dropbox 等均采用多实体分布式密钥管理设计, 通过职能分离或信任拆分实现密钥保护.

5.1 身份认证阶段安全性分析

5.1.1 备份密钥管理过程安全性分析

在备份密钥管理过程中, 恶意 $\mathcal{CS}$ 试图通过为用户提供口令强化服务或从用户备份密钥密文中试图恢复合法

用户口令. 直观来看, 恶意 $\mathcal{CS}$ 能够通过猜测用户口令并计算强化口令值, 在线向 $\mathcal{AS}$ 请求登录来验证猜测结果, SCDT 采用限速机制抵御敌手的在线猜测攻击; 用户备份密钥以密文形式存储在 $\mathcal{CS}$ 端, 而派生其加密密钥的盐值同样以密文形式存储在 $\mathcal{AS}$ 端, $\mathcal{CS}$ 无法获取关于加密密钥的信息, 故而无法执行离线猜测攻击.

恶意 $\mathcal{AS}$ 能够通过两种途径获取合法用户口令: a) $\mathcal{AS}$ 通过 *hpw* 猜测用户口令; b) $\mathcal{AS}$ 通过猜测攻击获取用户口令. 直观来看, $\mathcal{AS}$ 可以通过猜测用户口令并向 $\mathcal{CS}$ 请求强化口令值与本地数据库进行比对来猜测用户口令, SCDT 采用限速机制抵御敌手的在线猜测攻击; 用户将密钥派生所使用的盐值存储在 $\mathcal{AS}$ 端, 而用户信息以密文形式存储在 $\mathcal{CS}$ 端, 因此 $\mathcal{AS}$ 无法执行离线猜测攻击.

定理 1. 假设 SCDT 使用 IND-CCA 安全的加密算法, KDF 是安全的密钥派生函数, 在恶意 $\mathcal{CS}$ 存在的前提下, 用户在备份密钥管理过程中能够抵御恶意 $\mathcal{CS}$ 的口令猜测攻击.

证明: 本文使用一系列游戏证明定理 1.

● Game 0 游戏: 在这个游戏中, 敌手 $\mathcal{A}$ 与模拟器 $\mathcal{S}$ 交互, 其中 $\mathcal{S}$ 严格按照 SCDT 执行如下操作.

(1) $\mathcal{S}$ 根据安全参数 ℓ, 生成公共参数 *PP*, 均匀一致选取 $s_1 \xleftarrow{\$} Z_p$, 将 *PP* 发送给敌手 $\mathcal{A}$; $\mathcal{A}$ 选取签名密钥对 (sk, pk) 并将 *pk* 发送给 $\mathcal{S}$.

(2) 在请求阶段, $\mathcal{A}$ 选取 (pw_i, m_i) 发送给 $\mathcal{S}$.

(3) $\mathcal{S}$ 随机选取 $(r_{i,1}, r_{i,2}, r_{i,3}) \xleftarrow{\$} Z_p^3$, 将 $(pw_i^*, t_i, c_{i,1}, c_{i,2})$ 发送给 $\mathcal{A}$, 其中:

$$\begin{cases} pw_i^* = r_{i,1} \cdot \mathrm{H}_1(pw_i) \\ t_i = \mathrm{KDF}(pw_i \| r_{i,2}) \\ c_{i,1} = \mathrm{Enc}(s_1, r_{i,3}) \\ c_{i,2} = \mathrm{Enc}(\mathrm{KDF}(pw_i \| r_{i,3}), m_i) \end{cases} \tag{5}$$

(4) $\mathcal{A}$ 可以重复以上步骤 (2) 和 (3) 至多 $poly(\ell)$ 次.

(5) 在挑战阶段, $\mathcal{A}$ 选取 (pw_0, pw_1, m) 发送给 $\mathcal{S}$.

(6) $\mathcal{S}$ 均匀一致选取 $b \in \{0,1\}$ 以及 $(r_1, r_2, r_3) \xleftarrow{\$} Z_p^3$, 将 (pw^*, t, c_1, c_2) 发送给 $\mathcal{S}$, $\mathcal{S}$ 将其转发给 $\mathcal{A}$, 其中

$$\begin{cases} pw^* = r_1 \cdot \mathrm{H}_1(pw_b) \\ t = \mathrm{KDF}(pw_b \| r_2) \\ c_1 = \mathrm{Enc}(s_1, r_3) \\ c_2 = \mathrm{Enc}(\mathrm{KDF}(pw_b \| r_3), m) \end{cases} \tag{6}$$

(7) $\mathcal{A}$ 根据接收到的数据输出 $b' \in \{0,1\}$ 作为对 b 的猜测, 若 $b' = b$, 则 $\mathcal{A}$ 在游戏中获得胜利.

● Game 1 游戏: Game 1 游戏与 Game 0 游戏相同, 仅存在一点差异. 将 Game 0 游戏中的 pw^* 换成从 G_1 中随机选取, 若敌手能够正确输出 b 的猜测值, 则敌手在 Game 1 游戏中获得胜利. 若 $\mathcal{A}$ 在 Game 0 和 Game 1 中获胜的概率是不可忽略的, 那么能够构造有效敌手区分 $pw_0^* = r_1 \cdot \mathrm{H}_1(pw)$ 与 $pw_1^* \xleftarrow{\$} G_1$, 其中 $r_1 \xleftarrow{\$} Z_p$, 而对于群 G_1 中任意元素 pw_1^*, 亦存在 r_1' 使得 $pw_1^* = r_1' \cdot \mathrm{H}_1(pw)$, 由于敌手无法验证 r_1' 与 r_1 之间的一致性, 因此不存在有效敌手能够区分 pw_0^* 与 pw_1^*, 即 Game 0 与 Game 1 是不可区分的.

● Game 2 游戏: Game 2 游戏与 Game 1 游戏相同, 仅存在一点差异. 将 Game 1 游戏中的 t 换成从 KDF 输出空间中随机选取一个值, 若 $\mathcal{A}$ 能够正确输出 b 的猜测值, 则敌手在 Game 2 游戏中获得胜利. 若敌手在 Game 1 游戏和 Game 2 游戏中获胜的概率是不可忽略的, 那么能够构造有效敌手破坏 KDF 的伪随机性, 由于方案使用安全的 KDF 函数, 因此敌手在 Game 2 游戏和 Game 1 游戏中获胜的概率是可忽略的.

● Game 3 游戏: Game 3 游戏与 Game 2 游戏相同, 仅存在一点差异. 将 Game 2 游戏中 c_1 的加密计算换成随机选取一个值 $c_1 \xleftarrow{\$} Z_p^*$, 若 $\mathcal{A}$ 能够正确输出 b 的猜测值, 则敌手在 Game 3 游戏中获得胜利. 由于 SCDT 使用的加密算法是 IND-CCA 安全的, 且加密内容为 $\mathcal{S}$ 随机选取, 因此 r_3 的加密密文与从密文空间中随机选取的密文是不可区分的, 故 Game 3 与 Game 2 之间的差异是可忽略的.

● Game 4 游戏: Game 4 游戏与 Game 3 游戏相同, 仅存在一点差异. 将 Game 3 游戏中 c_2 的加密密钥换成从

KDF 输出空间中随机选取一个值, 若 $\mathcal{A}$ 能够正确输出 b 的猜测值, 则敌手在 Game 4 游戏中获得胜利. 由 Game 2 可知 KDF 函数具有伪随机性, Game 4 与 Game 3 之间的差异是可忽略的.

综上可知, 敌手在 Game 4 游戏和 Game 0 游戏中获胜的概率是可忽略的.

Game 4 游戏中模拟器 $\mathcal{S}$ 的计算为 $pw^* \xleftarrow{\$} Z_p$, $t \xleftarrow{\$} \mathcal{K}_{\mathrm{KDF}}$, $c_1 \xleftarrow{\$} \mathcal{C}$, $c_2 = \mathrm{Enc}(k, m)$, 其中 $k \xleftarrow{\$} \mathcal{K}_{\mathrm{KDF}}$, 且 $\mathcal{K}_{\mathrm{KDF}}$ 表示 KDF 函数的输出空间, $\mathcal{C}$ 表示加密算法的输出空间, 敌手在游戏中获胜的概率等于 1/2, 因此敌手在 Game 0 游戏中获胜的概率 $\Pr[\mathrm{Game0}_{\mathcal{A}}(\ell) = 1] = 1/2 + negl(\ell)$. 定理 1 证毕.

定理 2. 在随机谕言机模型下, 本方案中用户强化口令满足盲化性, 即在已知 hpw 的情况下, 恶意 $\mathcal{AS}$ 无法获得关于用户口令 pw 的额外信息.

证明: 我们定义盲化性游戏 Blind 如下.

(1) 环境 $\mathcal{E}$ 根据安全参数 ℓ, 生成公共参数 PP, 选取签名密钥对 (sk, pk) 并将 (PP, pk) 发送给 $\mathcal{S}$, $\mathcal{S}$ 将 (PP, pk) 转发给敌手 $\mathcal{A}$.

(2) 在请求阶段, 允许 $\mathcal{A}$ 访问谕言机: $\mathcal{O}^{\mathrm{H}_1}(x)$, $\mathcal{O}^{\mathrm{Sign}}(x)$, $\mathcal{O}^{\mathrm{H}_2}(x, \sigma_x)$, 允许 $\mathcal{S}$ 访问谕言机 $\mathcal{O}^{\mathrm{Vrfy}}(x, \sigma_x)$.

(3) $\mathcal{S}$ 收到 $\mathcal{A}$ 的谕言机请求后, 分别进行如下计算.

$\mathcal{O}^{\mathrm{H}_1}(x)$: 若 $\mathcal{Q}_1[x] \neq \perp$, 返回 $\mathcal{Q}_1[x]$; 否则选取 $y_1 \xleftarrow{\$} G_1$, 并令 $\mathcal{Q}_1[x] = y_1$ 并返回 y_1.

$\mathcal{O}^{\mathrm{Sign}}(x)$: 该谕言机维护一个列表 $\mathcal{L}$ 记录用户请求签名的输入, 根据输入 x, 令 $\mathcal{L}[i] = x$ 并访问谕言机 $\mathcal{O}^{\mathrm{H}_1}$ 获得 $\mathrm{H}_1(x)$ 并将结果转发给 $\mathcal{E}$, $\mathcal{E}$ 计算 $\sigma_x = sk \cdot \mathrm{H}_1(x)$ 后将其返回给 $\mathcal{S}$, $\mathcal{S}$ 将 σ_x 转发给 $\mathcal{A}$.

$\mathcal{O}^{\mathrm{Vrfy}}(x, \sigma_x)$: 该谕言机验证 $\mathcal{S}$ 提供的签名的有效性, 根据输入 x 访问谕言机 $\mathcal{O}^{\mathrm{H}_1}$ 得到 $\mathrm{H}_1(x)$, 验证 $e(\sigma_x Q) \stackrel{?}{=} e(\mathrm{H}_1(x), pk)$, 若相等, 则返回 1 给 $\mathcal{S}$, 否则, 返回 0.

$\mathcal{O}^{\mathrm{H}_2}(x, \sigma_x)$: 根据输入 (x, σ_x) 访问谕言机 $\mathcal{O}^{\mathrm{Vrfy}}$, 若谕言机返回 1, 则继续执行后续操作, 否则返回 $\perp$; 若 $\mathcal{Q}_2[x, \sigma_x] \neq \perp$, 返回 $\mathcal{Q}_2[x, \sigma_x]$; 否则选取 $y_2 \xleftarrow{\$} Z_p$, 并令 $\mathcal{Q}_2[x, \sigma_x] = y_2$ 并返回 y_2.

(4) 在挑战阶段, 敌手选取 $(\hat{x}, \hat{y}_2)$ 发送给 $\mathcal{S}$.

(5) $\mathcal{S}$ 对于 $\mathcal{A}$ 的挑战消息应答如下.

(a) 若 $\hat{x} \in \mathcal{L}$, 输出 0, 即敌手在游戏中失败.

(b) 否则, 如果 $\mathcal{Q}_2$ 中存在 $(\hat{x}, \cdot, \cdot)$ 的条目, 判断 $\hat{y}_2 \stackrel{?}{=} \mathcal{Q}_2[\hat{x}, \cdot]$, 若等式成立, 那么 $\mathcal{S}$ 将该条目中对应的 $(\hat{x}, \hat{\sigma}_x)$ 发送给 $\mathcal{E}$ 作为 BLS 签名的伪造, 并输出 1 表示 $\mathcal{A}$ 在游戏中胜利.

(c) 否则, 如果 $\mathcal{Q}_2$ 中不存在 $(\hat{x}, \cdot, \cdot)$ 的条目, 选取 $\hat{y}_2' \xleftarrow{\$} Z_p$, 并令 $\mathcal{Q}_2[\hat{x}, \cdot] = \hat{y}_2'$, 判断 $\hat{y}_2 \stackrel{?}{=} \hat{y}_2'$, 若不相等, 则输出 0; 否则输出 1.

令 Query 表示 $\mathcal{A}$ 向 $\mathcal{O}^{\mathrm{H}_2}$ 查询 $(\hat{x}, \cdot)$, $q(\ell)$ 表示 $\mathcal{A}$ 向 $\mathcal{S}$ 请求的次数, 则 $\mathcal{A}$ 在上述游戏中获胜的概率为:

$$\begin{aligned}\Pr[\mathrm{Blind}_{\mathcal{A}}(1^\ell) = 1] &= \Pr[\mathrm{Blind}_{\mathcal{A}}(1^\ell) = 1 \wedge \overline{\mathrm{Query}}] + \Pr[\mathrm{Blind}_{\mathcal{A}}(1^\ell) = 1 \wedge \mathrm{Query}] \\ &= \Pr[\mathrm{Blind}_{\mathcal{A}}(1^\ell) = 1 | \overline{\mathrm{Query}}] \cdot \Pr[\overline{\mathrm{Query}}] + \Pr[\mathrm{Blind}_{\mathcal{A}}(1^\ell) = 1 | \mathrm{Query}] \cdot \Pr[\mathrm{Query}] \\ &\leqslant \Pr[\mathrm{Blind}_{\mathcal{A}}(1^\ell) = 1 | \overline{\mathrm{Query}}] + \Pr[\text{CMA-EUF}_{\mathcal{A}}^{\mathrm{BLS}}(1^\ell) = 1] \cdot \frac{1}{q(\ell)} \end{aligned} \tag{7}$$

当 $\overline{\mathrm{Query}}$ 发生时, $\mathcal{S}$ 随机选取 $\hat{y}_2'$ 作为敌手挑战消息的 $\mathcal{O}^{\mathrm{H}_2}$ 应答, 该随机结果与 $\hat{y}_2$ 相等的概率为 $\frac{1}{2^\ell}$ (对应情况 (a)), 因此 $\Pr[\mathrm{Blind}_{\mathcal{A}}(1^\ell) = 1 | \overline{\mathrm{Query}}] = negl_1(\ell)$; 当 Query 发生时, 说明敌手在请求阶段获得了 $(\hat{x}, \sigma_{\hat{x}})$ 对应的 $\mathcal{O}^{\mathrm{H}_2}$ 输出 $\hat{y}_2$, 因此在挑战阶段, 敌手输出 $(\hat{x}, \hat{y}_2)$ 作为盲化性游戏 Blind 的应答 (对应情况 (b)), 游戏胜利的条件为敌手在请求阶段提供的 $\sigma_{\hat{x}}$ 为针对消息 $\hat{x}$ 有效的签名伪造, 而 SCDT 使用安全的 BLS 方案, 由第 2.3 节可知 $\Pr[\text{CMA-EUF}_{\mathcal{A}}^{\mathrm{BLS}}(1^\ell) = 1] = negl_2(\ell)$, 综上, $\Pr[\mathrm{Blind}_{\mathcal{A}}(1^\ell) = 1] = negl_1(\ell) + negl_2(\ell) \cdot \frac{1}{q(\ell)} = negl(\ell)$. 定理 2 证毕.

下面我们证明方案能够抵抗恶意 $\mathcal{AS}$ 的离线口令猜测攻击.

定理 3. 假设 SCDT 使用 IND-CCA 安全的加密算法, 在恶意 $\mathcal{AS}$ 存在的前提下, 用户在备份密钥管理过程中能抵御恶意 $\mathcal{AS}$ 的口令猜测攻击.

证明: 我们定义 ASDGA 游戏是敌手 $\mathcal{A}$ 与模拟器 $\mathcal{S}$ 的交互游戏, 其中 $\mathcal{S}$ 严格按照 SCDT 执行操作.

(1) $\mathcal{S}$ 根据安全参数 ℓ, 生成公共参数 PP, 选取签名密钥对 (sk, pk) 以及待加密消息 m, 均匀一致选取 $s_2 \xleftarrow{\$} Z_p$, 将 (PP, pk) 发送给敌手 $\mathcal{A}$; $\mathcal{A}$ 均匀一致选取 $s_1 \xleftarrow{\$} Z_p$.

(2) 在请求阶段, $\mathcal{A}$ 选取 pw_i 发送给 $\mathcal{S}$.

(3) $\mathcal{S}$ 随机选取 $(s_i, r_i) \xleftarrow{\$} Z_p^2$, 将 (hpw_i, s_i, r_i, c_i) 发送给 $\mathcal{A}$, 其中:

$$\begin{cases} hpw_i = \mathrm{H}_2(pw_i \| sk \cdot \mathrm{H}_1(pw_i)) \\ c_{i,1} = \mathrm{KDF}(pw_i \| s_i) \\ c_{i,2} = \mathrm{Enc}(\mathrm{KDF}(pw_i \| r_i), m) \\ c_i = \mathrm{Enc}(s_2, c_{i,1} \| c_{i,2}) \end{cases} \tag{8}$$

(4) $\mathcal{A}$ 可以重复以上步骤 (2) 和 (3) 至多 $poly(\ell)$ 次.

(5) 在挑战阶段, $\mathcal{A}$ 选取 (pw_0, pw_1) 发送给 $\mathcal{S}$.

(6) $\mathcal{S}$ 均匀一致选取 $b \in \{0,1\}$ 与 $(s, r) \xleftarrow{\$} Z_p^2$, 将 (hpw_b, s, r, c_b) 发送给 $\mathcal{S}$, $\mathcal{S}$ 将其转发给 $\mathcal{A}$, 其中:

$$\begin{cases} hpw_b = \mathrm{H}_2(pw_b \| sk \cdot \mathrm{H}_1(pw_b)) \\ c_{b,1} = \mathrm{KDF}(pw_b \| s) \\ c_{b,2} = \mathrm{Enc}(\mathrm{KDF}(pw_b \| r), m) \\ c_b = \mathrm{Enc}(s_2, c_{b,1} \| c_{b,2}) \end{cases} \tag{9}$$

(7) $\mathcal{A}$ 根据接收到的数据输出 $b' \in \{0,1\}$ 作为对 b 的猜测, 若 $b' = b$, 则 $\mathcal{A}$ 在游戏中获得胜利.

由定理 2 可知, 用户强化口令具有盲化性, 故而 $\mathcal{A}$ 无法以一个不可忽略的概率根据 hpw_b 正确猜出 b; 在敌手已知 (s, r, pw) 的情况下, $\mathcal{A}$ 执行 KDF 函数可以得到 (c_1, k), 其中 $k = \mathrm{KDF}(pw \| r)$, 而 SCDT 使用安全的加密方案对 (c_1, c_2) 进行加密得到 c, 由于加密密钥是 $\mathcal{S}$ 随机选取的, 若 $\mathcal{A}$ 在上述游戏中获得胜利, 则破坏了加密方案的安全性, 与假设矛盾. 定理 3 证毕.

5.1.2 身份密钥管理过程安全性分析

根据第 5.1.1 节分析, 用户能够安全的存储并基于口令恢复备份密钥 k_{re}, 在身份密钥管理阶段, 用户安全产生一个身份密钥对与一个中期预共享密钥对, 并使用 k_{re} 加密密钥对中的私钥, 将身份密钥对记录在区块链, 记为身份区块, 由 $\mathcal{AS}$ 协助认证用户身份区块, 并在 $\mathcal{AS}$ 端安全的存储用户中期预共享密钥对. 在该过程中, 恶意 $\mathcal{AS}$ 试图篡改合法用户的中期预共享密钥对以冒充用户身份与其他合法用户进行通信.

定理 4. 假设 SCDT 使用的加密算法能够达到 AE 安全, 签名算法能够满足不可伪造性, 那么用户的私钥信息能够保证机密性, 且用户密钥对能够抵御篡改攻击.

证明: SCDT 使用区块链记录用户身份信息, 由于区块链系统具有不可篡改性, 一旦用户产生区块上传至区块链并通过 $\mathcal{AS}$ 认证, 那么用户身份密钥区块与用户 ID 绑定, 能够抵御篡改或冒充攻击; 而用户身份密钥对中的私钥经过备份密钥加密以密文形式记录, 由定理 1–定理 3 可知, 方案能够保证用户口令的安全性, 进而保证 k_{re} 的安全性, 且加密算法满足 AE 安全, 因此方案能够保证用户密钥对中私钥的安全性.

关于用户预共享密钥对的安全性, 我们定义 *Auth* 游戏为 $\mathcal{A}$ 与 $\mathcal{S}$ 之间的交互游戏.

(1) $\mathcal{S}$ 生成公共参数 PP, 选取身份密钥对 (isk, ipk) 以及备份密钥 k_{re}, 将 (PP, ipk) 发送给敌手 $\mathcal{A}$;

(2) 在请求阶段, $\mathcal{A}$ 选取 (ssk_i, spk_i) 发送给 $\mathcal{S}$.

(3) $\mathcal{S}$ 计算 $c_i = \mathrm{Enc}(k_{re}, ssk_i), \sigma_i = \mathrm{Sign}(isk, spk_i)$, 并将 (c_i, σ_i) 发送给 $\mathcal{A}$.

(4) $\mathcal{A}$ 可以重复以上步骤 (2) 和 (3) 至多 $poly(\ell)$ 次.

(5) $\mathcal{A}$ 输出挑战消息 $(cssk, spk, \sigma_{spk})$ 发送给 $\mathcal{S}$.

(6) $\mathcal{S}$ 对 $cssk$ 进行解密并验证签名 σ_{spk} 的合法性如下.

(a) $\mathrm{Dec}(k_{re}, cssk) \neq \perp \wedge \mathrm{Dec}(k_{re}, cssk) \cdot P = spk$;

(b) $\mathrm{Vrfy}(ipk, spk, \sigma_{spk}) = 1$.

若验证通过, 那么输出 1, $\mathcal{A}$ 在游戏中获得胜利.

在上述游戏中, $\mathcal{A}$ 获胜的条件要求其满足: a) 能够在未知加密密钥的情况下伪造一个消息的密文; b) 能够在未知签名密钥的情况下伪造一个消息的签名, 由于 SCDT 使用 AE 安全的加密方案使得 $\mathcal{A}$ 无法以不可忽略的概率实现前者, 此外 SCDT 使用安全的签名方案, 使得 $\mathcal{A}$ 无法以不可忽略的概率实现后者. 定理 4 证毕.

5.2 数据传输阶段安全性分析

基于以上分析, 合法用户在身份认证阶段能够抵御内部服务器威胁完成安全的身份认证. 在数据传输阶段, 通信双方在认证服务器 $\mathcal{AS}$ 的协助下获得对方身份信息并建立安全信道进行数据传输. 根据定理 5.4 可知, 恶意 $\mathcal{AS}$ 无法篡改参与方身份, 即通信双方可以安全的获取对方正确的身份信息并建立安全的通信连接, 获取共享密钥 SK; 在通信过程中, 敌手能够被动窃听信道中的通信内容或主动篡改通信内容, 此外, 参与方密钥泄露会导致敌手能够根据密钥解密通信内容. SCDT 能够保证数据传输过程中抵御外部敌手的窃听或篡改攻击, 能够一定程度上保证在参与方密钥泄露后的系统安全. 以下给出证明.

引理 1. 假设 CDH 问题是困难的, KDF 是安全的密钥派生函数, SCDT 消息密钥与随机字符串不可区分.

由于 SCDT 使用数据处理加密算法为 AE 安全的加密算法, 且由引理 1 可知, 消息密钥与随机字符串不可区分, SCDT 数据传输过程能够实现消息的机密性与认证性; 根据文献 [30] 第 5.3 节可知, 双棘轮加密方案能够实现前向安全与后向安全, 当参与方通信状态未发生改变时, 每次传输消息前对加密密钥进行 KDF 棘轮步进, 因此, 若通信双方密钥泄露, 敌手仅可用其解密当前轮次通信内容, 无法获取历史通信信息, 保证系统前向安全性; 当参与方通信状态发生改变时, 双方进行 DH 棘轮进行新一轮密钥协商, 当前密钥链将失效, 敌手无法根据持有的密钥派生后续通信过程的有效解密密钥, 即当其中一个参与方密钥泄露时, 若通信双方身份发生转换, 系统将恢复安全性保证, 保证了后向安全性.

6 性能分析

6.1 计算开销

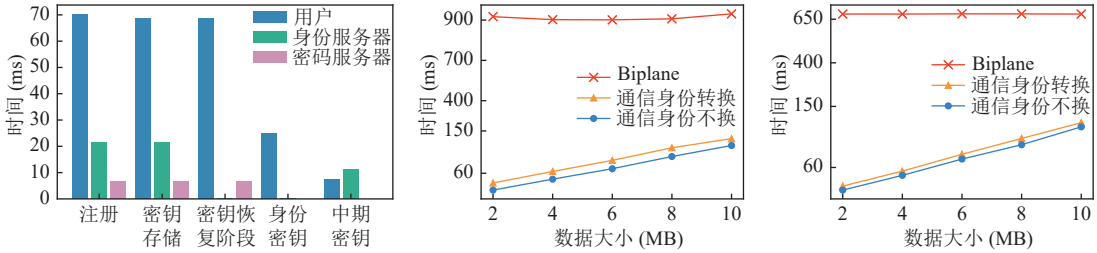
本节基于 13 代 i5 CPU, 32 GB 运行内存, 使用 IntelliJ IDEA 平台, 调用 JPBC 库对本文方案原型系统进行实现, 并对其计算开销和通信开销分别进行实验评估, 选取安全级别为 128 位, 使用 JPBC 库中的 A 类型配对来实例化配对类型, 选取 SHA3-256 算法执行本方案的哈希操作, 并基于 Signal 协议基本思路实现本文方案的数据传输. 具体源代码可在 https://github.com/jinglewe/SCDT 查看. 经过多次测试, 密码基本操作的平均时间如表 4 所示.

表 4 密码基本操作平均时间

符号	操作描述	时间 (ms)
T_M	椭圆曲线上乘法计算	10.27
T_H	哈希计算	0.89
T_P	双线性配对计算	9.34
T_{Enc}	加密计算	38.97
T_{Dec}	解密计算	0.20
T_D	密钥派生计算	0.13

本实验中由于用户身份区块只需执行一次, 我们重点关注各实体的链下开销, 省略用户及服务器产生区块以及定位区块的计算. 在身份认证阶段, 用户端的计算开销来源于强化口令获取、签名合法性验证、密钥派生等, $\mathcal{AS}$ 计算开销来源于用户数据加/解密计算以及密文区块合法性验证, $\mathcal{CS}$ 的开销来源于计算盲化口令的签名以及用户数据加/解密计算; 在注册阶段, 用户需要执行 $2T_M+2T_P+T_H+T_D$, 服务器端各自执行 T_M+T_{Enc}; 在密钥存储阶段, 用户在注册阶段基础上额外执行 T_D+T_{Enc}, 服务器端各自执行 T_M+T_{Enc}; 在密钥恢复阶段, 用户在注册阶段基础上额外执行 T_D+T_{Dec}, 服务器端各自执行 T_M+T_{Dec}; 在身份密钥管理阶段, 用户执行 $2T_{\text{Enc}}+T_H+T_M$, $\mathcal{AS}$ 执行 T_H+2T_P, $\mathcal{CS}$ 不参与该阶段; 在中期密钥管理阶段, 用户执行 $T_{\text{Dec}}+2T_M+T_{\text{Enc}}$, $\mathcal{AS}$ 执行 $2T_P$, $\mathcal{CS}$ 不参与该阶段. 各

实体在身份认证各子阶段的具体开销如图 4(a) 所示. SCDT 的数据传输基于双棘轮加密的基本思想, 用户首先进行 1 次 KDF 棘轮步进, 获取最新的加/解密密钥, 然后使用 AES-GCM 认证加密标准进行加解密操作, 若通信双方身份发生转换 (发送方转换为接收方), 需要首先进行 1 次 DH 棘轮步进. 加密 10 MB 数据时, 加密时间开销约 120 ms, 在用户可接受范围内, 通信双方身份发生转换所额外执行的 DH 棘轮步进时间开销约 10 ms. 而 Biplane[1]在数据处理阶段使用混合加密算法并产生一笔交易记录密文数据, 因此在数据处理中面临较高的链上开销. 图 4(b) 与图 4(c) 展示了随数据量增加, 加/解密的计算开销变化及与 Biplane[1]在数据处理过程中的时间开销对比. 需要说明的是, 为清晰描述 SCDT 在数据处理过程中的时间开销, 我们在图中适当将 0–150 ms 范围等比例放大.



(a) 各阶段各实体时间开销 (b) 加密时间随数据量变化 (c) 解密时间随数据量变化

图 4 各阶段各实体时间开销及加/解密时间随数据量变化

6.2 存储开销

本方案实现设备节点轻量化管理, 在身份认证阶段, 用户只需保存低熵口令, 无需本地存储身份数据或密钥信息, 备份密钥安全的分布式存储在服务器端, 身份密钥记录在区块链上, 定期更新的中期密钥及认证签名存储在 $\mathcal{AS}$ 端; 数据处理阶段, 参与方需在会话过程中管理 state 数据结构, 包含 DH 棘轮计算所需公私钥以及加密所需密钥链当前密钥 (消息密钥在数据处理结束后删除, 过期密钥在每次更新后删除, 均不占用设备本地存储空间), 故设备节点在会话期间产生存储开销约 0.1 KB. 本文 $\mathcal{CS}$ 为用户提供口令强化服务, 为每个 ID 创建一个服务请求记录 $req_{\mathcal{U}} = (\mathrm{ID}_{\mathcal{U}}, l_{\mathcal{U}}^{req})$, 存储用户登录信息与备份密钥密文信息 $reg_{\mathcal{U}}^{\mathcal{CS}} = (\mathrm{ID}_{\mathcal{U}}, c_2, l_{\mathcal{U}}^{\mathcal{CS}})$, 本地存储开销约 0.12 KB, $\mathcal{AS}$ 存储用户登录信息与盐值数据信息 $reg_{\mathcal{U}}^{\mathcal{AS}} = (\mathrm{ID}_{\mathcal{U}}, c_1, l_{\mathcal{U}}^{\mathcal{AS}})$, 存储用户身份认证区块地址与中期密钥信息 $\mathcal{L}_{\mathcal{U}} = (\mathrm{ID}_{\mathcal{U}}, Add_{\mathcal{U}}, Add_{T_{\mathrm{cer}}}, \mathcal{D}_{\mathcal{U}})$.本地存储开销约 0.47 KB.

6.3 通信开销

本节分析用户数据传输以外的通信开销, 身份认证阶段包括口令强化算法计算中间参数 pw^* 并发送给 $\mathcal{CS}$ 请求服务、备份密钥管理算法使用 (ID, hpw, s, r) 向 $\mathcal{AS}$ 注册、使用 (ID, t, ct) 向 $\mathcal{CS}$ 注册、身份密钥管理算法计算认证参数及其签名 $(Auth_{\mathcal{U}}, \sigma)$ 并发送给 $\mathcal{AS}$ 请求认证、用户身份区块开销、发送 $(\mathrm{ID}_{\mathcal{U}}, \mathcal{D}_{\mathcal{U}})$ 定期更新中期密钥产生的周期性通信开销; 数据处理阶段包括通信连接建立算法发送 $(epk_{\mathcal{S}}, Add_{T_{\mathrm{key}}}, c_{\mathrm{SK}} = \mathrm{Enc}(\mathrm{SK}, \mathrm{SK}))$ 进行密钥协商, 以及数据加密过程中引入的密文扩张. 通信各阶段的通信开销如表 5 所示.

表 5 各阶段通信开销

阶段	通信开销 (KB)
身份认证	0.38
密钥更新	0.30
连接建立	0.15
数据传输-身份不换	0.03
数据传输-身份转换	0.15

6.4 区块链开销

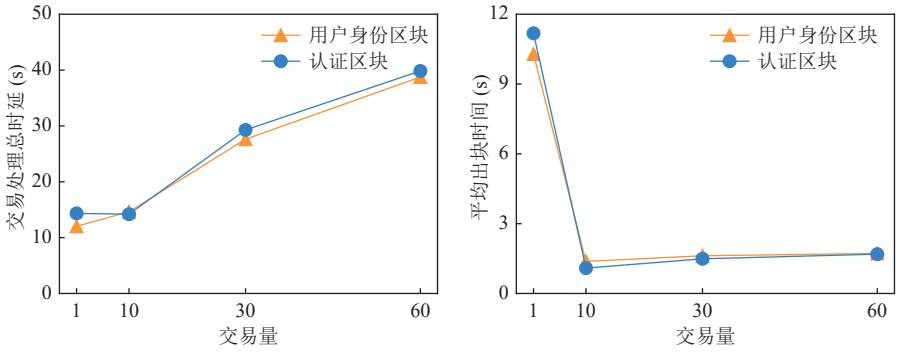
本节基于以太坊测试网 Sepolia 为测试平台, 身份认证过程中产生的交易数据结构如表 6 所示, 本文对用户身

份绑定及服务器认证过程产生的用户身份区块与服务器认证区块造成的时延、平均确认时间、写入成本进行测试与分析, 假定每单位 Gas 成本为 2 Gwei (测试网中约 0.001 Gwei), 用户身份区块写入成本为 40 200 Gas, 即 8.04×10^{-5} (4.02×10^{-8}) 以太币, 认证区块写入成本为 37 800 Gas, 即 7.56×10^{-5} (3.78×10^{-8}) 以太币. 具体时延结果如图 5 所示.

表 6　交易数据结构

字段名	类型	用户身份区块字段内容	认证区块字段内容	长度 (字节)	说明
From	Address	$Add_{\mathcal{U}}$	$Add_{\mathcal{AS}}$	20/20	发送方地址
To	Address	$\hat{h}(\mathrm{ID}_{\mathcal{U}})$	$\hat{h}(\mathrm{ID}_{\mathcal{U}})$	20/20	接收方地址
Value	Uint256	0	0	32/32	转账金额
Data	bytes	$cbsk_{\mathcal{U}}\|\|cisk_{\mathcal{U}}\|\|ipk_{\mathcal{U}}$	$\mathrm{ID}_{\mathcal{AS}}\|\|Add_{T_{\mathrm{key}}}\|\|ipk_{\mathcal{U}}$	240/180	交易数据字段
Signature	bytes	$\sigma_{T_{\mathrm{key}}}$	$\sigma_{T_{\mathrm{cer}}}$	118/118	发送方签名

注: "/"前后分别表示用户身份区块字段长度和认证区块字段长度



(a) 交易处理总时延随交易量变化　(b) 平均出块时间随交易量变化

图 5　交易处理总时延及平均出块时间随交易量变化

6.5　对比分析

SCDT 实现了节点轻量化及便携性管理, 节点只需安全存储低熵口令即可在多设备使用 SCDT 的通信服务, 且数据传输过程能够满足前向安全与后向安全. 表 7 展示了传输一条消息时, SCDT 与其他跨域数据传输方案各实体的性能对比 (文献 [1] 中令 $n=t=1$; 文献 [6] 中令 $n=1$). 表中 T_{mtp} 表示将数据映射到椭圆曲线点的计算; T_{exp} 表示指数运算; T_A 表示点加法计算; 加/解密操作各方案开销类似, 不列入比较. 本方案与文献 [1] 相比, 用户端计算开销显著降低, 文献 [3,6,35] 用户端开销约等于或小于 SCDT, 然而无法实现用户端便携性, 需用户本地存储私钥信息, 且服务器端开销较大, 相比之下, SCDT 能在满足节点便携性管理的同时, 平衡用户端与服务器端计算开销.

表 7　各实体时间开销对比 (ms)

方案	发送节点	接收节点	服务器
Biplane[1]	$4T_P+9T_M+2T_H$ ≈131.578	$4T_P+9T_M+2T_H$ ≈131.578	$5T_M$ ≈51.35
BCDS[3]	$8T_M+T_A+2T_{\mathrm{exp}}+2T_P+2T_H$ ≈105.612	$10T_M+3T_A+T_{\mathrm{exp}}+2T_P+T_H$ ≈124.141	$2T_M$ ≈20.54
BSCDDS[6]	$2T_M+T_{\mathrm{exp}}+T_{\mathrm{mtp}}+3T_H$ ≈27.941	$5T_H+2T_{\mathrm{exp}}+T_M$ ≈17.574	$2T_P+T_{\mathrm{mtp}}+11T_M+4T_H+5T_A$ ≈297.917
BSCDA[35]	$3T_M+2T_P+8T_H$ ≈56.642	$6T_M+2T_P+9T_H$ ≈88.346	$9T_M+12T_P+5T_H$ ≈208.98
SCDT	$8T_M+2T_P+2T_H+6T_D$ ≈103.408	$8T_M+2T_P+2T_H+6T_D$ ≈103.408	$T_M+3T_P+T_H$ ≈39.184

7 结束语

现实任务复杂化使得多域网络系统互联互通、相互协作, 本文在强对抗环境下, 针对跨信任域网络中的轻量级节点设计了数据跨域安全传输方案, 包括基于区块链的公钥认证机制与基于口令的密钥管理方法, 使得节点能够全时域、设备独立地访问系统, 以及高效安全的数据跨域传输方案, 对传输数据进行加密处理保证数据安全性. SCDT 基于双服务器相互制约框架, 若身份服务器与密码服务器合谋, 用户信息安全将无法得到保证, 为进一步强化系统安全性, 简化系统架构, 后续工作将探索由区块链节点分布式执行密码服务器功能的可能性; 此外, 未来将对数据传输处理过程进行优化, 考虑数据丢失、乱序等情况, 进一步提升方案鲁棒性.

References

[1] Li SY, Zhang Y, Song YQ, Cheng N, Yang K, Li HW. Blockchain-based portable authenticated data transmission for mobile edge computing: A universally composable secure solution. IEEE Trans. on Computers, 2024, 73(4): 1114–1125. [doi: 10.1109/TC.2024.3355759]

[2] Wei X, Wang XY, Yu Z, Guo SY, Qiu XS. Cross domain authentication for IoT based on consortium blockchain. Ruan Jian Xue Bao/Journal of Software, 2021, 32(8): 2613–2628 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6033.htm [doi: 10.13328/j.cnki.jos.006033]

[3] Zeng SL, Cao B, Sun Y, Sun C, Wan ZG, Peng MG. Blockchain-assisted cross-domain data sharing in industrial IoT. IEEE Internet of Things Journal, 2024, 11(16): 26778–26792. [doi: 10.1109/JIOT.2023.3329577]

[4] Zhang ZJ, Fan W, Zhu DL. Lightweight blockchain authentication mechanism for smart home. Ruan Jian Xue Bao/Journal of Software, 2022, 33(7): 2699–2715 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6288.htm [doi: 10.13328/j.cnki.jos.006288]

[5] End-of-train and head-of-train remote linking protocol. 2025. https://www.cisa.gov/news-events/ics-advisories/icsa-25-191-10

[6] Wang FQ, Cui J, Zhang QY, He DB, Zhong H. Blockchain-based secure cross-domain data sharing for edge-assisted industrial Internet of Things. IEEE Trans. on Information Forensics and Security, 2024, 19: 3892–3905. [doi: 10.1109/TIFS.2024.3372806]

[7] Lei A, Cruickshank H, Cao Y, Asuquo P, Ogah CPA, Sun ZL. Blockchain-based dynamic key management for heterogeneous intelligent transportation systems. IEEE Internet of Things Journal, 2017, 4(6): 1832–1843. [doi: 10.1109/JIOT.2017.2740569]

[8] He BJ, Li Y. Blockchain-based key management and security decisions in Internet of Vehicles. IEEE Internet of Things Journal, 2025, 12(11): 17456–17472. [doi: 10.1109/JIOT.2025.3536480]

[9] Wu LB, Wang J, Choo KKR, He DB. Secure key agreement and key protection for mobile device user authentication. IEEE Trans. on Information Forensics and Security, 2019, 14(2): 319–330. [doi: 10.1109/TIFS.2018.2850299]

[10] Cheng L, Li YN, Tang Q, Yung MT. End-to-same-end encryption: Modularly augmenting an App with an efficient, portable, and blind cloud storage. In: Proc. of the 31st USENIX Security Symp. Boston: USENIX Association, 2022. 2353–2370.

[11] Zhang Y, Xu CX, Cheng N, Shen XM. Secure password-protected encryption key for deduplicated cloud storage systems. IEEE Trans. on Dependable and Secure Computing, 2022, 19(4): 2789–2806. [doi: 10.1109/TDSC.2021.3074146]

[12] Lai RWF, Egger C, Reinert M, Chow SSM, Maffei M, Schröder D. Simple password-hardened encryption services. In: Proc. of the 27th USENIX Security Symp. Baltimore: USENIX Association, 2018. 1405–1421.

[13] Perrin T, Marlinspike M. The double ratchet algorithm. Revision 1. 2016. https://whispersystems.org/docs/specifications/doubleratchet/doubleratchet.pdf

[14] Marlinspike M, Perrin T. The X3DH key agreement protocol. Revision 1. 2016. https://signal.org/docs/specifications/x3dh/x3dh.pdf

[15] He XY, Zhang Y, Song YQ, Qiu WD, Li HW, Tang Q. What makes a good exchange? Privacy-preserving and fair contract agreement in data trading. IEEE Trans. on Information Forensics and Security, 2025, 20: 4265–4279. [doi: 10.1109/TIFS.2025.3558590]

[16] Davies GT, Faller S, Gellert K, Handirk T, Hesse J, Horváth M, Jager T. Security analysis of the WhatsAPP end-to-end encrypted backup protocol. In: Proc. of the 43rd Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 2023. 330–361. [doi: 10.1007/978-3-031-38551-3_11]

[17] Zhang Y, Xu CX, Li HW, Yang K, Cheng N, Shen XM. PROTECT: Efficient password-based threshold single-sign-on authentication for mobile users against perpetual leakage. IEEE Trans. on Mobile Computing, 2021, 20(6): 2297–2312. [doi: 10.1109/TMC.2020.2975792]

[18] Lamport L. Password authentication with insecure communication. Communications of the ACM, 1981, 24(11): 770–772. [doi: 10.1145/358790.358797]

[19] Chatterjee R, Athayle A, Akhawe D, Juels A, Ristenpart T. Password typos and how to correct them securely. In: Proc. of the 2016 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2016. 799–818. [doi: 10.1109/SP.2016.53]

[20] Adams C, Lloyd S. Understanding PKI: Concepts, Standards, and Deployment Considerations. Boston: Addison-Wesley Professional, 2003.

[21] Li SY, Zhang Y, Cheng N, Song YQ. Badge: Blockchain-assisted secure authenticated data transmission in mobile edge computing. In: Proc. of the 2022 ICC IEEE Int'l Conf. on Communication. Seoul: IEEE, 2022. 4757–4762. [doi: 10.1109/ICC45855.2022.9838606]

[22] Faller S, Handirk T, Hesse J, Horváth M, Lehmann A. Password-protected key retrieval with (out) HSM protection. In: Proc. of the 2024 ACM SIGSAC Conf. on Computer and Communications Security. Salt Lake City: ACM, 2024. 2445–2459. [doi: 10.1145/3658644.3690358]

[23] Song YQ, Xu CX, Zhang Y, Li SY. Hardening password-based credential databases. IEEE Trans. on Information Forensics and Security, 2024, 19: 469–484. [doi: 10.1109/TIFS.2023.3324326]

[24] Reed D, Sporny M, Longley D, Allen C, Grant R, Sabadello M, Holt J. Decentralized identifiers (DIDs) v1.0. 2020. https://www.ossbig.at/wp-content/uploads/2023/05/Decentralized-Identifiers-DIDs-v1.0-w3c.pdf

[25] Cui ZH, Xue F, Zhang SQ, Cai XJ, Cao Y, Zhang WS, Chen JJ. A hybrid blockchain-based identity authentication scheme for multi-WSN. IEEE Trans. on Services Computing, 2020, 13(2): 241–251. [doi: 10.1109/TSC.2020.2964537]

[26] Bellare M, Namprempre C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Proc. of the 6th Int'l Conf. on the Theory and Application of Cryptology and Information Security on Advances in Cryptology. Kyoto: Springer, 2000. 531–545. [doi: 10.1007/3-540-44448-3_41]

[27] Dworkin M. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. Gaithersburg: Department of Commerce, 2007.

[28] An JH, Dodis Y, Rabin T. On the security of joint signature and encryption. In: Proc. of the 2002 Int'l Conf. on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology. Amsterdam: Springer, 2002. 83–107. [doi: 10.1007/3-540-46035-7_6]

[29] Hsu RH, Lee J, Quek TQS, Chen JC. GRAAD: Group anonymous and accountable D2D communication in mobile networks. IEEE Trans. on Information Forensics and Security, 2018, 13(2): 449–464. [doi: 10.1109/TIFS.2017.2756567]

[30] Alwen J, Coretti S, Dodis Y. The double ratchet: Security notions, proofs, and modularization for the signal protocol. In: Proc. of the 38th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology. Darmstadt: Springer, 2019. 129–158. [doi: 10.1007/978-3-030-17653-2_5]

[31] Collins D, Riepel D, Oliver Tran SA. On the tight security of the double ratchet. In: Proc. of the 2024 on ACM SIGSAC Conf. on Computer and Communications Security. Salt Lake City: ACM, 2024. 4747–4761. [doi: 10.1145/3658644.3690360]

[32] Rescorla E. RFC 8446 The transport layer security (TLS) protocol version 1.3. 2018. https://www.rfc-editor.org/rfc/pdfrfc/rfc8446.txt.pdf

[33] WhatsApp Inc. WhatsApp security whitepaper. 2016. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

[34] Miculan M, Vitacolonna N. Automated symbolic verification of telegram's MTProto 2.0. arXiv:2012.03141, 2020.

[35] Chai BB, Yu JG, Yan BW, Yu Y, Wang SL. BSCDA: Blockchain-based secure cross-domain data access scheme for Internet of Things. IEEE Trans. on Network and Service Management, 2024, 21(4): 4006–4023. [doi: 10.1109/TNSM.2024.3385777]

[36] Boneh D, Lynn B, Shacham H. Short signatures from the weil pairing. Journal of Cryptology, 2004, 17(4): 297–319. [doi: 10.1007/s00145-004-0314-9]

[37] Freedman MJ, Ishai Y, Pinkas B, Reingold O. Keyword search and oblivious pseudorandom functions. In: Proc. of the 2nd Theory of Cryptography Conf. on Theory of Cryptography. Cambridge: Springer, 2005. 303–324. [doi: 10.1007/978-3-540-30576-7_17]

[38] Lu B, Zhang XK, Ling ZM, Zhang YQ, Lin ZQ. A measurement study of authentication rate-limiting mechanisms of modern websites. In: Proc. of the 34th Annual Computer Security Applications Conf. San Juan: ASM, 2018. 89–100. [doi: 10.1145/3274694.3274714]

附中文参考文献

[2] 魏欣, 王心妍, 于卓, 郭少勇, 邱雪松. 基于联盟链的物联网跨域认证. 软件学报, 2021, 32(8): 2613–2628. http://www.jos.org.cn/1000-9825/6033.htm [doi: 10.13328/j.cnki.jos.006033]

[4] 张珠君, 范伟, 朱大立. 面向智能家居的区块链轻量级认证机制. 软件学报, 2022, 33(7): 2699–2715. http://www.jos.org.cn/1000-9825/6288.htm [doi: 10.13328/j.cnki.jos.006288]

作者简介

芦静文, 博士生, 主要研究领域为数据安全, 隐私保护.

宋雅晴, 硕士, 主要研究领域为数据安全, 应用密码学.

张源, 博士, 研究员, 博士生导师, CCF 专业会员, 主要研究领域为密码学, 信息安全, 区块链, 云计算安全.

何欣雨, 博士生, 主要研究领域为应用密码学, 数据交易, 数据安全.

龚竞, 高级工程师, 主要研究领域为智慧城市建设与运营, 数字技术研发与应用推广.

李洪伟, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为网络空间安全, 数据安全.