# 杭州电子科技大学

## 毕业设计（论文）开题报告

| | |
|---|---|
| 题　　目 | 基于多方安全计算的区块链钱包的设计与实现 |
| 学　　院 | 计算机学院 |
| 专　　业 | 软件工程 |
| 姓　　名 | |
| 班　　级 | |
| 学　　号 | |
| 指导教师 | |

# 一、综述本课题国内外研究动态，说明选题的依据和意义

## 1.1 国内外研究

近年来，区块链技术的快速发展催生了以数字钱包为代表的多种去中心化应用。然而，传统数字钱包面临一系列安全和用户体验的挑战。其核心问题在于私钥的安全管理：私钥一旦丢失或泄露，将直接导致资产的不可挽回损失。尽管硬件钱包和助记词备份方案能够一定程度上缓解这一问题，但它们在安全性、便利性和技术易用性之间存在取舍，难以满足普通用户，尤其是新手用户的需求。

在完全去中心化的客户端私钥存储、派生的插件钱包或物理冷钱包方案，与托管钱包为主的私钥、注记词托管派生的中心化方案中，多方安全计算（MPC）技术提供了一种全新的私钥管理方式，通过将私钥分片存储在多个节点之间，并通过安全计算协议完成签名任务，在提高安全性的同时无需用户直接接触私钥。在保持中心化托管便利性的同时，最大限度的降低托管服务的私钥泄露风险。

此外，微信和支付宝作为中国乃至全球范围内的两大主流社交与支付平台，其 Oauth 登录功能能够提供便捷的用户身份认证方案。结合 MPC 与 Oauth 登录，开发基于社交账户的去中心化钱包，既能提升用户的资产管理安全性，又能降低用户使用门槛，可谓是当前 MPC 主流应用场景在中文互联网的应用案例延申。

目前，国内外主流 MPC 钱包（如 ZenGo、Web3Auth）虽然在去中心化钱包中应用了 MPC 技术，并且在诸如 Google、Apple、Github 等主流应用平台做了良好的适配接入，并开源有完善的 SDK，但由于相关的法律法规以及市场大小问题，对微信、支付宝等中国特定社交生态支持不足。

此外，这些钱包大多集中于以太坊和比特币等链的应用支持，对国内公链（如上海树图链 CONFLUX）、联盟链缺少签名适配性。因此，开发基于 MPC 技术的分布式去中心化钱包，并集成微信和支付宝 Oauth 登录 JWT 鉴权功能，或基于微信、支付宝小程序构建小程序签名器，同时适配中国大陆合规化程度较高的公链、联盟链，将成为国内区块链生态的重要创新点，填补现有市场空白。

## 1.2 研究价值

• 技术价值

通过集成 MPC 技术，能够实现高安全性、去中心化的私钥管理方案，将用户私钥拆分存储在多个独立节点中，从根本上解决传统钱包的私钥管理痛点。此外，将微信和支付宝 Oauth 登录能力融入 MPC 钱包，为用户提供一种无缝、安全的身份认证方式，同时降低开发者在实现钱包安全认证过程中的复杂度。

• 市场价值

上海树图链 CONFLUX 作为国内自主创新的公链，在技术架构和应用领域具有独特

优势。开发以该链为核心的 MPC 钱包，能够加速链上应用场景的落地，特别是在社交和支付场景中具有高度契合性。支持微信和支付宝的 Oauth 登录功能，可帮助国内用户快速接入区块链世界，同时为开发者生态赋能，降低开发者教育用户的门槛。

• 用户价值

对普通用户而言，该方案既消除了对私钥管理的技术门槛，又通过熟悉的微信和支付宝账户快速登录，实现了更佳的用户体验。

同时，MPC 技术确保用户的资产安全，即使设备丢失或被盗，也能通过社交账户恢复资产，显著增强用户对区块链钱包的信任度。

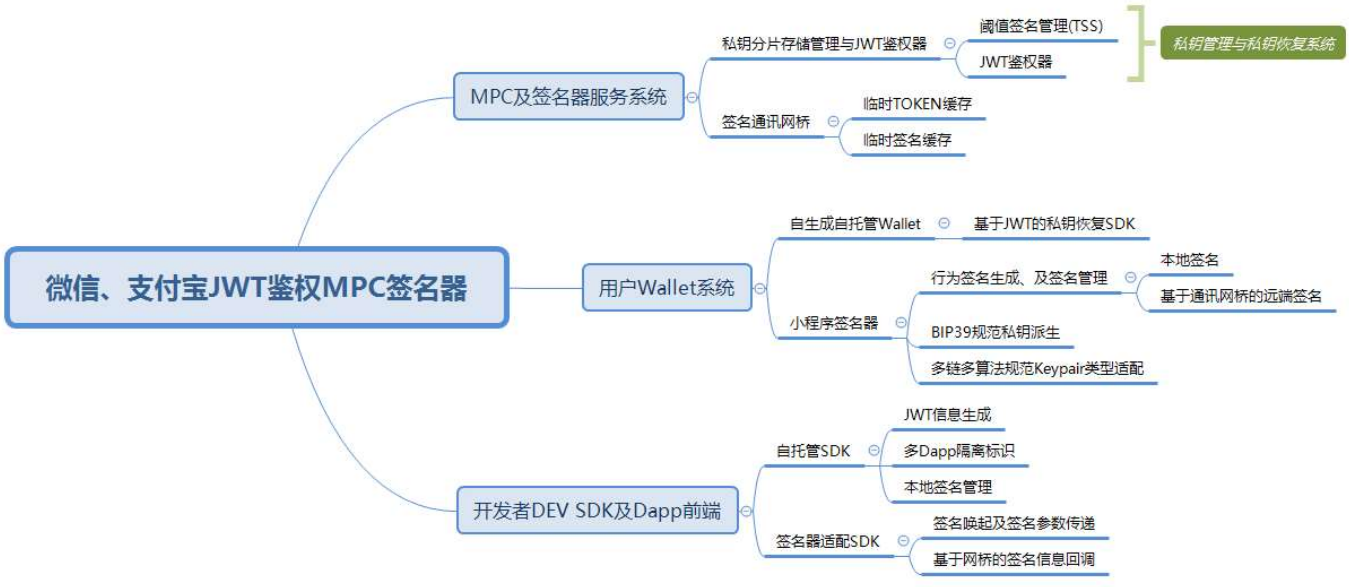
• 生态价值

在使用 MPC 方案解决私钥存储问题后，使得开发者可以使用 BIP39 规范对私钥进行派生，并根据不同公链的算法适配，实现将使钱包能够服务于更多公链和用户群体，为 ETH、BTC、SOLANA、TON 等链的多链生态创建统一接口，推动区块链生态进一步融合与发展。

## 二、研究的基本内容，拟解决的主要问题：

### 2.1 系统整体功能模块

本课题的研究内容聚焦于多方安全计算技术在去中心化钱包中的应用，特别是结合微信、支付宝 Oauth 登录与 JWT 鉴权，并通过微信、支付宝小程序的形式，为用户提供便捷、安全的数字资产管理解决方案。具体包括以下几个方面：



## 2.2 系统功能更详细的介绍

本系统是一套基于 多方安全计算（MPC） 和 JWT（JSON Web Token）鉴权 技术的签名服务系统，结合微信和支付宝等第三方鉴权手段，提供安全、去中心化的签名及密钥管理功能。该系统组成如下四大核心模块：

1. 用户钱包（Wallet）管理与签名服务：支持用户自托管钱包的创建、签名和恢复。
2. 多方安全计算（MPC）服务：将私钥分片存储，通过 MPC 进行去中心化签名，确保私钥安全。
3. JWT 鉴权服务：通过微信、支付宝等第三方平台鉴权，实现用户身份验证及密钥恢复。
4. 开发者集成（SDK）：提供开发者 SDK，便于 Dapp 应用无缝对接签名服务。

### 2.2.1 各系统详细功能：

#### 1）MPC 及签名器服务系统

该模块是系统的核心，主要负责私钥的分片存储管理、JWT 鉴权、阈值签名以及私钥的恢复等服务。

- **私钥分片存储管理与 JWT 鉴权器**

该功能结合 **MPC** 和 **JWT 鉴权** 技术，将用户的私钥分成多个“密钥份额”并分布式存储。通过微信、支付宝等第三方平台的 OAuth 服务，用户身份得到验证，从而保证用户私钥的安全和真实性。系统支持使用 Shamir's Secret Sharing 算法对私钥进行分片存储，每个密钥份额分散存储在用户设备、服务器等不同位置，避免单点存储的安全隐患。同时，系统通过 JWT 鉴权生成用户标识，确保用户的身份真实性。JWT 鉴权信息中的 sub 字段会被用作检索用户私钥份额的依据，支持私钥的分布式恢复。

- **阈值签名管理（TSS）**

阈值签名方案（TSS）实现了无需重建完整私钥的去中心化签名功能。签名过程通过多方计算协同完成，私钥份额之间独立计算局部签名，最终聚合出有效签名，且签名结果与单一私钥签名一致。

系统支持灵活的阈值设置，如 2/3、3/5 等阈值签名模式，可满足不同的安全需求。TSS 方案适配了主流的签名算法，包括 ECDSA 和 EdDSA，确保系统在多链生态下的兼容性。

- **私钥管理与私钥恢复系统**

私钥管理与恢复功能保障了私钥的安全分发与找回。用户通过微信或支付宝登录进行 JWT 鉴权后，系统会检索与其标识相关的私钥份额，并进行安全恢复。通过分

布式存储机制，用户的私钥不会集中存储在某一位置。用户设备、服务端等多个节点协同恢复密钥，确保密钥找回过程的安全性和高效性。

### 2）签名通讯网桥

签名通讯网桥是签名操作的中间桥梁，负责签名过程中的数据传输、临时缓存及签名结果回调，确保签名操作的流畅性和安全性。

- **临时 TOKEN 缓存**

签名通讯网桥在 JWT 鉴权通过后会生成短时有效的临时 TOKEN，并进行缓存管理。用户进行签名请求时，系统会校验 TOKEN 的有效性，确保请求来源合法，防止恶意请求和签名滥用。

- **临时签名缓存**

该功能用于临时存储签名请求的相关数据和签名结果。在签名操作完成后，系统会将签名结果缓存并通过回调机制传输给调用方，如用户端或 Dapp 前端，确保签名结果的快速响应和交付。

### 3）用户 Wallet 系统

用户 Wallet 系统主要面向终端用户，提供钱包生成、自托管、签名与管理功能，确保用户在本地安全管理其私钥并实现签名操作。

- **自生成自托管 Wallet**

系统支持用户在本地生成自托管钱包，私钥仅在用户设备上存在，确保私钥安全无外泄风险。用户可以通过 JWT 鉴权结果恢复其密钥份额，找回钱包私钥。此外，系统提供基于 JWT 的私钥恢复 SDK，支持用户身份验证后快速找回私钥，简化了钱包找回流程。

- **小程序签名器**

微信或支付宝小程序场景中，系统提供轻量级的签名器服务，支持行为签名生成与签名管理功能。用户可以选择本地签名或通过签名通讯网桥完成远端签名请求，签名结果回传到本地进行验证。

同时，小程序签名器支持基于 BIP39 规范的私钥派生，用户可以通过助记词管理和派生私钥。此外，系统适配了多链多算法签名需求，支持包括以太坊、Solana、Polygon 等主流区块链网络，并兼容 ECDSA 和 EdDSA 等加密签名算法，确保多链生态的兼容性。

### 4）开发者 DEV SDK 及 Dapp 前端

开发者模块提供灵活的 SDK 及工具，支持开发者将系统功能快速集成到 Dapp 应用中，实现钱包管理和签名操作。

**自托管 SDK**

自托管 SDK 主要支持用户自托管的密钥管理与 JWT 信息生成功能。开发者可以通过 SDK 为用户生成合法的 JWT 信息，完成身份鉴权。此外，SDK 提供多 Dapp 隔离标识功能，确保不同 Dapp 之间的数据隔离，保护用户隐私。同时，SDK 支持本地签名管理，用户无需网络即可完成签名操作，进一步提升安全性。

**签名器适配 SDK**

签名器适配 SDK 提供签名操作的无缝对接服务，包括签名唤起与参数传递功能。开发者可以通过调用 SDK 将签名请求发送至系统，由系统进行签名操作。同时，签名结果会通过签名通讯网桥的回调机制快速传输至调用方，确保签名请求高效响应。此外，SDK 提供灵活的签名参数传递方案，支持开发者根据业务需求定制签名逻辑和流程

## 2.3 拟解决的主要问题

（1）私钥管理安全性问题：

传统数字钱包的核心痛点在于私钥的安全管理。私钥是用户访问和管理区块链资产的唯一凭证，一旦私钥丢失、泄露或被盗，用户资产将无法恢复，带来不可挽回的损失。目前的解决方案主要包括助记词备份、硬件钱包以及中心化托管钱包，但这些方案各有弊端。助记词备份要求用户手动保存，既不方便也容易丢失；硬件钱包虽然安全性较高，但价格昂贵且操作复杂，难以被普通用户接受；中心化托管钱包虽然便利，但私钥存储在第三方服务器上，存在单点故障与被攻击的风险，安全隐患较大。

为了解决这一问题，本研究引入多方安全计算（MPC）技术，将私钥进行分片管理。系统通过 MPC 技术将用户的私钥拆分成多个“密钥份额”，分别存储在用户设备和服务器等节点上，并在签名操作时进行计算，生成有效的签名结果。这一过程无需重建完整的私钥，从而确保密钥在整个生命周期内不会被完整存储或泄露。此外，系统支持阈值签名（TSS）机制，允许用户灵活设置阈值（如 2/3、3/5 等），即使部分节点被破坏，也不会影响签名的正常执行。该方案兼顾了去中心化的安全性与中心化的便利性，消除了用户手动管理私钥的负担，显著提升了钱包的安全性与可靠性。

（2）用户体验的复杂性问题：

传统去中心化钱包在用户体验上存在明显不足，尤其对于新手用户而言，操作流程过于复杂。用户不仅需要妥善保存助记词，还需要掌握繁琐的私钥管理知识。此外，完全去中心化的钱包缺乏统一的身份验证机制，用户在多个应用平台间切换时体验较差。而对于开发者来说，实现钱包的身份认证与安全管理功能需要投入大量的时间与技术成本，增加了 DApp 开发的难度。

本研究通过集成微信、支付宝 OAuth 登录与 JWT 鉴权机制，有效降低了用户操

作门槛。用户可以通过微信或支付宝账户一键登录钱包系统，系统会基于 JWT（JSON Web Token）技术生成用户的唯一身份标识，并将其与 MPC 密钥分片管理机制结合，实现安全的私钥恢复与管理。此外，系统在微信和支付宝小程序中提供轻量级签名器服务，支持本地签名与远端签名，用户无需了解复杂的加密技术即可完成签名操作。这一设计不仅提升了用户的操作便捷性，也显著降低了新手用户进入区块链世界的技术门槛。

（3）多链生态的兼容性问题：

当前大部分去中心化钱包主要支持以太坊、比特币等国际主流公链，对国内公链（如上海树图链 Conflux）和联盟链的支持不足，无法满足国内区块链生态发展的需求。此外，不同公链间的签名算法不统一，导致跨链适配难度较大，开发者与用户面临较高的技术壁垒。本研究充分考虑多链兼容性，支持基于 BIP39 规范的私钥派生，能够实现统一的多链钱包管理功能。

在签名适配方面，系统兼容主流的签名算法，包括 ECDSA 和 EdDSA，同时支持以太坊、比特币、Solana、Polygon、TON 等国际主流公链和国内公链，为开发者与用户提供高度兼容的多链生态解决方案。此外，系统通过开发者 SDK 封装多链签名与派生接口，简化了多链适配的技术流程，降低了开发者的技术难度，进一步推动了多链生态的融合发展。

（4）身份验证与私钥恢复问题：

在传统去中心化钱包中，身份验证与私钥恢复是两个难以解决的难题。用户一旦丢失私钥或助记词，几乎无法找回资产，这也成为区块链钱包难以被大众普及的重要原因。本研究将微信和支付宝 OAuth 登录的能力与 JWT 鉴权机制结合，解决了这一问题。用户通过微信或支付宝账户登录时，系统生成 JWT 标识，并基于 MPC 技术将用户密钥份额与身份标识关联。即使用户设备丢失或被盗，用户仍然可以通过社交账户快速恢复私钥，从而找回资产。

这种设计将传统复杂的私钥恢复过程简化为熟悉的社交账号登录，既降低了用户的技术门槛，又有效提高了钱包的安全性和可用性。同时，私钥的分布式存储和恢复机制确保了整个私钥管理流程的安全可靠，避免了单点存储带来的风险。

（5）开发者接入成本问题：

目前，开发者在构建 DApp 应用时需要自行开发钱包签名与身份认证功能，这不仅增加了开发成本，还提高了应用的安全风险。本研究通过提供灵活的开发者 SDK，简化了钱包功能的接入过程。开发者可以快速集成钱包签名、JWT 鉴权与密钥管理功能，实现用户身份验证、私钥恢复及多链签名等操作。此外，SDK 支持多 DApp 数据

隔离与签名管理，保障用户隐私与数据安全，同时提升 DApp 的整体安全性与用户体验。

通过 MPC、JWT 与 OAuth 技术的结合，本研究为开发者提供了一套安全、易用的工具链，降低了 DApp 开发的技术难度，促进了区块链生态的进一步发展。

## 三、研究步骤、方法及措施：

### 3.1 研究步骤

3.1.1 总体工作流程如下

| 工作流程 | 日程安排 |
| --- | --- |
| 系统分析和设计,TSS 算法 demo 实现 | 2.24-3.10 |
| 完成 MPC 分片存储模块与私钥管理系统 | 3.10-3.14 |
| 构建数据通讯网桥，接口化 MPC 服务，接入 Wechat/Alipay Oauth 系统。 | 3.14-3.21 |
| 构建 Wallet 小程序前端，实现自托管 wallet 模型的 demo 用例。 | 3.22-3.30 |
| 构建小程序签名器 Deeplink 支持的 dev SDK，并封装自托管 wallet SDK。 | 3.30-4.05 |
| 构建一个基于上海树图链 Conflux、Solana、EVM-testnet 的多链 Dapp 调用 SDK 进行签名操作、交易广播操作的基础 Dapp demo | 4.05-4.15 |

3.3.2 流程中的具体研究步骤

(1) 系统分析与设计

在项目初期，明确系统的总体架构和技术路线。研究多方安全计算（MPC）和阈值签名（TSS）算法的适配性，设计基于分布式私钥分片存储的签名服务框架。构建系统的核心模块架构图，包括私钥分片管理、签名通讯网桥、Wallet 用户端以及开发者 SDK 模块的相互关系。

TSS 算法的初步实现是本阶段的重点，旨在验证算法的可行性和性能。同时，研究微信和支付宝 OAuth 登录与 JWT 鉴权的集成方案，为后续开发奠定基础。

(2) MPC 分片存储模块与私钥管理系统的开发

构建私钥分片存储系统，将用户的私钥分片存储在多个节点上，确保私钥的安全性与分布式恢复能力。构建支持 JWT 鉴权的用户身份认证模块，通过微信、支付宝等第三方 OAuth 登录生成用户的唯一身份标识，并将其与密钥分片关联，实现分布式密钥恢复功能。

同时，完成阈值签名管理（TSS）功能开发，使系统能够支持多种阈值配置，并适配主流区块链的签名算法（如 ECDSA 和 EdDSA）。

(3) 数据通讯网桥与 OAuth 系统集成

构建签名通讯网桥，支持签名数据的安全传输与缓存管理。该模块包括临时 TOKEN 缓存与签名结果缓存的功能，确保签名请求和结果的实时性与安全性。同时，开发与 MPC 服务接口化的模块，使多方计算服务能够通过标准化接口调用，提高系统的模块化与扩展性。

本阶段还需接入微信和支付宝 OAuth 登录系统，完成用户身份验证与 JWT 生成功能的无缝集成，为后续用户端与开发者工具提供完整的鉴权服务。

(4) Wallet 小程序前端开发与自托管 Wallet 模型实现

开发用户 Wallet 系统的小程序前端，支持用户本地生成自托管 Wallet 并完成基本的密钥管理功能。通过整合前端与后端的私钥恢复逻辑，验证分布式私钥恢复功能的完整性。

此外，小程序前端需实现 BIP39 规范的助记词管理和私钥派生功能，支持用户多链钱包的管理需求。本阶段的目标是构建一个可用的自托管 Wallet 模型 Demo。

(5) 签名器 Dev SDK 与 Wallet SDK 的开发

构建支持 Deeplink 的小程序签名器开发者 SDK，使其能够通过标准化接口完成签名唤起、参数传递与结果回调功能。同时，封装 Wallet SDK，支持开发者集成自托管 Wallet 的创建、JWT 鉴权和签名功能，确保系统的开发者友好性。

SDK 开发阶段的关键在于优化系统性能，提升签名调用的响应速度与安全性，并提供灵活的签名参数定制接口。

(6) 基于多链的 Dapp 调用与 Demo 构建

开发一个基础的 Dapp Demo，集成多链调用和交易广播功能，验证系统在多链环境中的兼容性与性能表现。Demo 需支持上海树图链 Conflux、Solana、EVM-testnet 等链的签名与交易广播操作，确保 Wallet 和签名服务在多链生态下的适配性。

此外，通过开发者工具包的集成测试，验证系统的可用性与稳定性，并收集用户反馈以优化系统功能。

## 3.3 措施和实施

### 3.3.1 技术和工具选型

小程序前端：UNIAPP、VUE、NACL、Axios

Dapp 前端：React、Nextjs、NextUI

NPM-SDK 封装：TS

MPC&网桥接口后端：Nodejs、Express、MongoDB、Docker、Nginx、Redis

编程工具：Visual Studio Code

3.3.2 采用基于原型的敏捷编程进行系统设计实现

首先，团队进行需求调研与分析，明确系统的核心功能需求，如私钥管理的安全性、MPC 签名效率、OAuth 鉴权集成等。通过快速构建低保真原型（如流程图或线框图），直观展现系统架构与关键模块设计，并在此基础上与利益相关者充分沟通，确保功能需求清晰且可行。

随后，快速开发最小可行产品（MVP），将私钥分片存储管理、JWT 鉴权集成和 MPC 签名服务作为首批实现模块。通过敏捷迭代的方式，每个周期完成功能实现、用户测试和性能验证，及时根据反馈调整系统设计与实现策略。

最后，基于用户体验和开发者反馈，对原型进行逐步完善，优化界面交互、提升性能并扩展功能适配性，实现符合实际应用场景的高质量系统。敏捷编程结合原型验证的方式，确保系统在开发全程中高效迭代和贴合需求。

## 四、主要参考文献：（所列出的参考文献不得少于 10 篇，其中外文文献不得少于 2 篇，发表在期刊上的学术论文不得少于 4 篇。）


[1]《多方安全计算研究综述》—蒋凯元

[2] D. Evans, V. Kolesnikov and M. Rosulek. A Pragmatic Introduction to Secure Multi-Party Computa- tion. NOW Publishers, 2018.

[3] I. Damgård and J. Nielsen. Scalable and Unconditionally Secure Multiparty Computation. In CRYPTO 2007, Springer (LNCS 4622), pages 572–590, 2007.

[4] Yehuda Lindell . Secure Multiparty Computation (MPC)

[5] Anders Dalskov, Partisia Daniel Escudero, JP Morgan AI Research + AlgoCRYPT CoE Ariel Nof, Bar-Ilan University . Fully Secure MPC and zk-FLIOP Over Rings: New Constructions, Improvements and Extensions

[6]《安全多方计算：理论、实践和应用》 赵胜男，赵明浩，陈振祥，高崇志 ,李宏伟, 唐玉安

[7] Chen Gao , Jia Yu . SecureRC: A system for privacy-preserving relation classification using secure multi-party computation

[8] Wirawan Agahari, Hosea Ofe & Mark de Reuver . It is not (only) about privacy: How multi-party computation redefines control, trust, and risk in data sharing

[9] Chainlink . secure-multiparty-computation-mcp

[10] Michele Ciampi, University of Edinburgh Xiangyu Liu, Purdue University West Lafayette, Georgia Institute of Technology Ioannis Tzannetos, Purdue University West Lafayette, National Technical University of Athens Vassilis Zikas, Georgia Institute of Technology . Universal Adaptor Signatures from Blackbox Multi-Party Computation

[11] Marie Beth van Egmond1 , Vincent Dunning1 , Stefan van den Berg1 , Thomas Rooijakkers1 , Alex Sangers1 , Ton Poppe2 , and Jan Veldsink3 . Privacy-preserving Anti-Money Laundering using Secure Multi-Party Computation

[12] Web3auth . Harness the security of enterprise grade MPC