RustChain_Rustcamp_Technical Solution

1. 架构概述

“RustChain”是一个采用 Rust 语言开发的轻量级区块链平台，主要面向学习与开发实践。系统采用模块化设计，将核心区块链功能、交易处理、UTXO 模型、钱包管理以及 P2P 网络通讯分离成多个独立模块，各模块通过清晰的接口协同工作。产品不仅支持 CLI 交互，还内置了基于 TCP Socket 的节点间通讯，确保整个网络中区块、交易与节点信息的互通。

2. 系统模块设计

2.1 区块模块（Block）

- **数据结构与实现：**
  - 定义了区块数据结构，包含时间戳、交易列表、前一区块哈希、当前区块哈希、随机数（nonce）以及区块高度。
  - 实现了区块的构造、创世区块生成、以及区块哈希计算。
- **工作量证明（PoW）：**
  - 使用 SHA256 对区块内容及 nonce 进行哈希运算，通过比对哈希值前 TARGET_HEXS 位是否满足“0”的要求来验证 PoW 结果。
  - 利用 Merkle 树对区块内所有交易进行汇总，确保交易数据不可篡改。

2.2 区块链模块（Blockchain）

- **数据库与存储：**
  - 采用 sled 数据库存储区块数据及链上索引信息，利用键值对存储实现快速读取和写入。
  - “LAST”键用于记录当前链顶区块的哈希，方便后续区块的迭代与查询。
- **链操作：**
  - 提供创建新区块链、挖矿生成新区块、区块迭代遍历等功能。
  - 包含对交易签名验证、UTXO 查找、重建 UTXO 集合等操作，确保数据一致性。

2.3 交易模块（Transaction）

- **交易结构：**
  - 定义了交易输入（TXInput）和输出（TXOutput），以及收集输出的结构（TXOutputs）。
  - 交易包含唯一标识符（id）、输入集合和输出集合。
- **签名与验证：**

- 基于 ed25519 算法实现交易的签名和验证，利用公私钥机制确保交易不可伪造。
- 在构造交易时，会先创建一个“剪裁版”副本用于签名，防止在签名过程中受到其他数据的干扰。

2.4 UTXO 模型模块（UTXOSet）

- **UTXO 查找与管理：**
  - 通过遍历整个区块链及数据库中存储的交易输出，查找并维护未花费交易输出集合。
  - 提供查找可支配余额、统计 UTXO 数量、重建 UTXO 集合以及在新区块生成时更新 UTXO 数据等功能。

2.5 钱包模块（Wallets）

- **密钥生成与地址生成：**
  - 采用 ed25519 生成密钥对，结合 bitcoincash_addr 实现钱包地址生成。
  - 使用 Ripemd160 与 SHA256 对公钥进行哈希，确保生成的地址符合加密要求。
- **钱包管理：**
  - 实现钱包的创建、存储及检索，支持将所有钱包数据保存在 sled 数据库中，实现持久化管理。

2.6 网络服务模块（Server）

- **P2P 网络设计：**
  - 基于 TCP Socket 实现节点间通信，定义了多种消息类型（如：addr、version、tx、getdata、block、inv 等）。
  - 节点通过消息广播与请求，实现区块、交易数据及节点信息的同步。
- **消息处理机制：**
  - 采用固定长度命令码（CMD_LEN）对消息进行区分，并通过 bincode 序列化/反序列化实现数据传输。
  - 处理逻辑覆盖区块获取、交易广播、节点地址同步以及版本对比等操作。

2.7 CLI 接口模块（Cli）

- **命令行交互：**
  - 提供创建区块链、钱包管理、查询余额、发送交易、打印链、启动节点/矿工等多种命令。
  - CLI 工具为开发者和运维人员提供了快速调试与演示的接口，降低了上手门槛。

3. 数据存储设计

- **sled 数据库：**

    - 整个区块链、UTXO 集合及钱包数据均存储在 sled 数据库中，利用其嵌入式存储特性提供高效的读写性能。
    - 数据库采用键值对存储方式，使用字符串或字节数组作为键，存储经过 bincode 序列化后的对象。

4. 共识算法与安全机制

- **Proof-of-Work：**
    - 每个区块在创建时都需要进行工作量证明，通过不断调整 nonce 使区块哈希满足预设难度。
    - 难度由常量 TARGET_HEXS 决定，实际可根据测试情况调整以平衡计算复杂度与区块生成速度。
- **加密与数据完整性：**
    - 使用 SHA256 和 Merkle 树确保区块数据的完整性，任何篡改都将导致哈希值不匹配。
    - 基于 ed25519 实现交易签名与验证，确保只有拥有相应私钥的用户才能发起合法交易。
- **钱包安全：**
    - 钱包地址生成采用双重哈希（SHA256 后跟 Ripemd160），防止暴力破解和伪造。

5. 网络通信协议

- **消息格式设计：**
    - 定义统一的命令码格式（固定长度 CMD_LEN），后接消息数据。
    - 消息类型涵盖：
        - **version/addr：** 用于节点版本信息及地址同步。
        - **tx：** 交易数据广播，确保新交易能在网络中尽快传播。
        - **inv/getdata/block：** 实现区块和交易数据的请求与传输，保证节点数据一致性。
- **节点发现与数据同步：**
    - 节点启动后首先通过 version 消息比对区块链高度，若存在落后则自动请求最新区块。
    - 通过 inv 消息广播待同步区块和交易，实现跨节点数据传递。

6. 开发、测试与部署

- **开发环境：**
    - 基于 Rust 编程语言，使用最新稳定版 Rust 编译器。
    - 利用 Cargo 管理依赖及构建流程，确保项目模块化与可测试性。
- **测试策略：**
    - 提供单元测试覆盖钱包、交易签名验证、区块生成、UTXO 更新以及网络消息解析等核心功能。
    - 利用集成测试验证节点间通讯及整体系统流程，确保数据一致性与安全性。
- **部署方案：**

- 支持单机部署以及多节点 P2P 集群部署。
- CLI 工具可直接用于启动节点、挖矿及日常操作；后续可扩展 Web API 或图形化界面以提升用户体验。