

Enterprise Zero-Knowledge Threshold Secret Sharing System

本项目提供了一个生产就绪的企业级零知识阈值秘密共享系统，集成多项高级安全特性、性能优化算法和完善的审计机制，适用于大规模分布式密钥管理与秘密恢复场景。

主要功能

- 阈值秘密共享 (Shamir Secret Sharing)**
通过多项式分割的方式，将秘密 s 分成 n 份，每份为 $(i, f(i))$ ，仅当至少 t 份有效分享聚合时，方可重构原始秘密。
- FFT 加速拉格朗日插值 (Optimized Lagrange FFT)**
采用并行化 Karatsuba 与 FFT 技术，在大规模多项式乘法与插值时显著提升性能，并提供完备的错误类型和性能度量。
- 企业级 BLAKE3 哈希适配器**
基于 BLAKE3，提供 64 字节增强输出、跨平台 SIMD 加速和审计上下文扩展，兼顾高吞吐与安全性。
- 全生命周期密钥管理**
从密钥生成、激活、退役到销毁，严格遵循 NIST SP 800-57，并在每一阶段记录安全审计事件。
- 零知识证明与 VSS 校验**
使用 Fiat-Shamir 变换生成非交互式零知识证明，验证每份分享的正确性；并实现批量并行恢复功能。
- 多方计算 (MPC) 集成示例**
演示多方多项式协议生成全局秘密切片，并进行聚合随机数示例，展现实际应用能力。
- 合规与审计**
支持 FIPS 140-2 Level 3、Common Criteria EAL4+ 等模式，可导出完整性能指标与审计日志，满足企业合规需求。

架构及模块

```
├─ src/
│  ├─ error.rs           // CryptoError、ErrorHandler 与审计日志
│  ├─ hash_adapter.rs    // Blake3Adapter 与 SecurityValidator
│  ├─ key_lifecycle.rs   // Key 生命周期管理
│  ├─ lagrange_fft.rs    // 优化多项式运算与插值
│  └─ sharing.rs        // Shamir 分享、更新与阈值调整
```

		mpc.rs	// MPC 协议模拟
		proof.rs	// 零知识证明生成与验证
		vss.rs	// Verifiable Secret Sharing 校验
		serialization.rs	// Scalar & RistrettoPoint 序列化
		utils.rs	// 随机数、常量与幂运算
		main.rs	// 企业演示与 CLI

数学正确性证明

1. Shamir 阈值秘密共享

选定素数域 \mathbb{F}_p ，秘密 $s \in \mathbb{F}_p$ 与阈值 $t \leq n$ 。构造随机多项式

$$f(x) = s + a_1x + a_2x^2 + \cdots + a_{t-1}x^{t-1}, \quad a_i \xleftarrow{\$} \mathbb{F}_p.$$

为第 i 位参与者分发分享 $(i, f(i))$ 。

- 阈值重建：**任取 t 个分享 $\{(i_j, y_j)\}_{j=1}^t$ ，可以唯一确定次数不超过 $t-1$ 的插值多项式，常数项即为原始秘密。
- 信息理论安全：**少于 t 个分享时，被丢弃的常数项可任意映射，秘而不宣。

2. Lagrange 插值公式

给定 t 个点 $\{(x_j, y_j)\}_{j=1}^t$ ，插值多项式

$$f(x) = \sum_{j=1}^t y_j \ell_j(x),$$

其中

$$\ell_j(x) = \prod_{\substack{1 \leq m \leq t \\ m \neq j}} \frac{x - x_m}{x_j - x_m}.$$

可验证：

$$\ell_j(x_k) = \begin{cases} 1, & k = j, \\ 0, & k \neq j, \end{cases}$$

从而保证 $f(x_j) = y_j$ 。

- 秘密重建：**取 x_1, \dots, x_t 与对应 $y_j = f(x_j)$ ，计算

$$s = f(0) = \sum_{j=1}^t y_j \ell_j(0).$$

3. BLAKE3 安全与性能

Blake3Adapter 输出长度 512 bit (64 字节)，具有以下安全性与性能保证：

- **碰撞抵抗**: 2^{128} 安全强度
- **前像抵抗**: 2^{256} 安全强度
- **树形 Merkle 结构**: 天然并行, 可横向扩展多线程与 SIMD 加速
- **PRF/MAC/KDF/XOF**: 一体化算法, 无额外分支
- **序列化一致性**: 使用 Zeroize 清除敏感状态, 满足审计合规.

快速开始

1. 克隆仓库并构建:

```
git clone https://github.com/your-org/enterprise-threshold-sdk.git
cd enterprise-threshold-sdk
cargo build --release
```

2. 演示运行:

```
cargo run --release
```

将依次执行安全验证、密钥生成、分享分发、秘密恢复等流程, 并输出性能指标与审计日志。

3. 在业务代码中集成示例:

```
use enterprise_threshold_sdk::{EnterpriseCryptoSystem, EnterpriseConfig, KeyState};

let config = EnterpriseConfig::default();
let mut system = EnterpriseCryptoSystem::new(config);

// 安全校验
system.validate_security()?;

// 生成并激活主密钥
let key = system.generate_enterprise_key("master-key-001"?);
assert_eq!(key.state, KeyState::Active);

// 创建并恢复秘密
let secret = key.secret;
let shares = system.create_secret_shares(secret, 3, 5, "op-123"?);
let recovered = system.recover_secret_enterprise(&shares[..3], "op-123"?);
assert_eq!(recovered, secret);
```

合规与审计

- 支持多种合规模式: Standard、FIPS 140-2 L3、Common Criteria EAL4+、自定义
- 可导出完整 PerformanceMetrics 与 SecurityEvent 列表, 用于上报与审计

- 敏感字段在 Drop 和 ZeroizeOnDrop 中清零，最大程度减少侧信道风险