Aokiji

A Rust Library and Wallet for Nano Group Transactions Using FROST

Introduction

Current *Nano* cryptocurrency wallets like *Nault* provide limited multisignature capabilities that are technically challenging for users. Specifically, *Nano* lacks accessible threshold signature tools, and existing solutions such as *Nault* only implement basic group signatures without threshold security features.

Additionally, group transaction processes are often manual, insecure, or overly complex for typical users. This project seeks to enhance the group transaction experience through modern cryptographic protocols, FROST, and systems programming approaches.

The primary objectives include implementing threshold signatures, developing a proof-of-concept wallet GUI, and ensuring both blockchain compatibility and robust security.

FROST and Nano

Flexible Round-Optimized Schnorr Threshold (FROST) is a threshold signature protocol that reduces the number of communication rounds needed to produce Schnorr signatures based on Shamir's secret sharing.

Like similar protocols, it remains secure against misbehaving participants, but handles them by identifying and blacklisting rather than attempting recovery - it aborts when misbehavior is detected.

Our implementation uses a semi-trusted participant called a *Signature Aggregator* that handles the signature aggregation process. This approach reduces communication overhead while simplifying wallet application architecture and enabling asynchronous operation. Additionally, it uses Lagrange interpolation for threshold signing, allowing any subset of *t* out of *n* participants to generate a group signature that's valid on the *Nano* blockchain.

Nano is a decentralized digital payment protocol designed to be accessible and lightweight in contrast with cryptocurrencies like *Bitcoin*. It uses a *Block Lattice*, a DAG (Directed Acyclic Graph) based architecture. In this type of data structure, individual accounts control their own blockchain, making it ideal for higher-level signature schemes like FROST.

Implementation

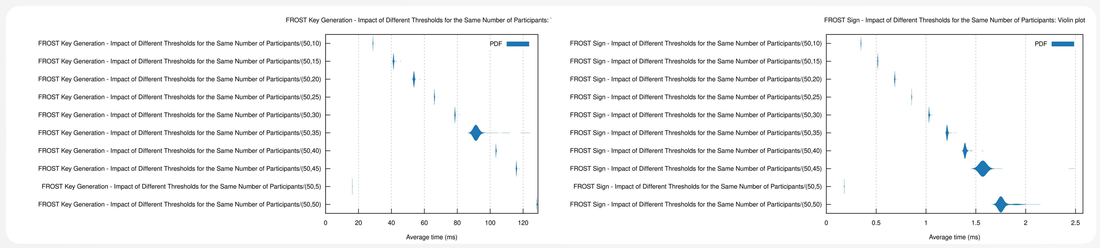
1. ***Technology Study:*** An analysis of the necessary technologies was conducted, namely *Nano* (cryptocurrency used), *Rust* (for security and performance), *Tokio* (for asynchronous IO networking) *Dioxus* (for graphical interface in *Rust* via *WebAssembly*), *Dalek* (for elliptic curve cryptography on ED25519), and the cryptographic concepts required for the project (FROST and *Blake2b*).
2. ***FROST Library Development:*** A *Rust* library was created that enables shared account creation on the *Nano* network and collective transaction signing using a custom FROST implementation for threshold signing. To facilitate remote communication between participants, sockets were used for secure cryptographic message exchange, implemented with *Tokio*.
3. ***Desktop Application Development:*** A desktop application was developed in *Rust*, using *Dioxus* and *WebAssembly* for dynamic components, allowing users to view balances, transaction history, and sign group operations in an intuitive and secure manner.

Benchmarking

Two benchmarking tests where conducted to evaluate the library's performance for both *Sign* and *Key Generation* operations.

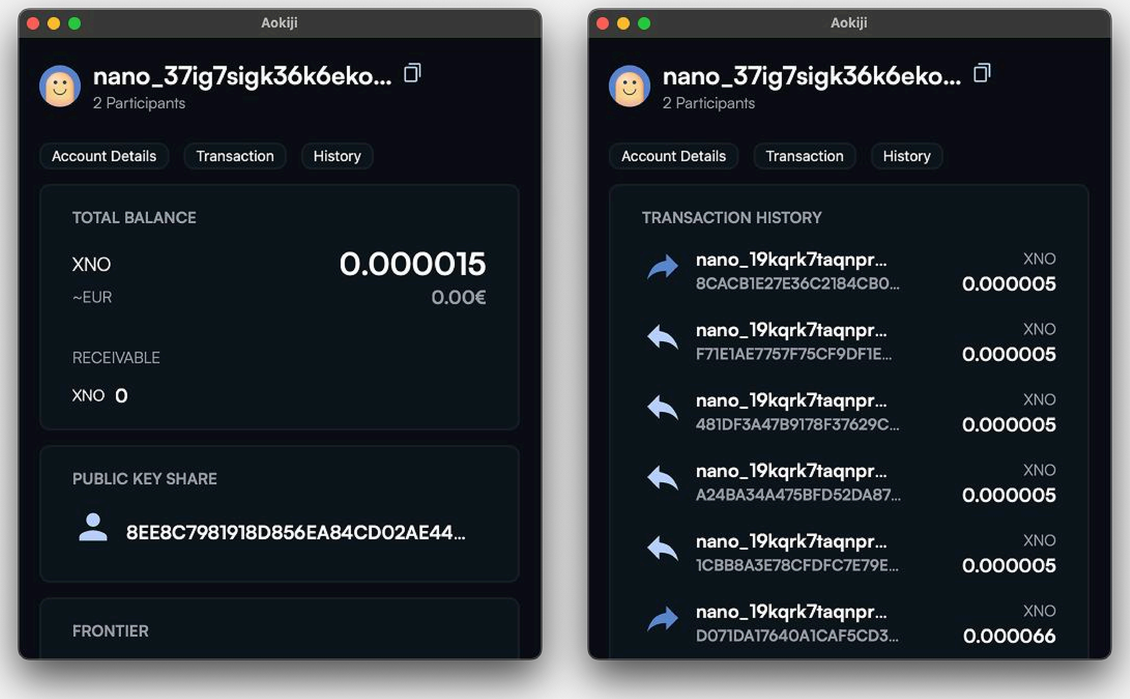
1. ***Impact of the Number of Total Participants (n):*** The results demonstrate that *key generation* in FROST grows with the number of participants but happens only once during setup, so its impact is minimal. In contrast, *signing* remains constant for each participant, ensuring efficiency regardless of group size.

2. ***Impact of the Threshold (t):*** The results show that higher thresholds increase the latency in *key generation* and *signing* operations, with *key generation* still being the slowest. This is expected, as more participants and computations are involved, but the trade-off improves security by requiring more parties to authorize actions.



Discussion and Results

This project demonstrates a successful implementation of FROST that seamlessly integrates with the *Nano* blockchain ecosystem. The custom threshold signing solution effectively handles the signing of *open*, *send*, and *receive* blocks through *Nano*'s RPC interface, while the application provides an accessible platform for participant coordination. It enables users to collectively manage accounts and execute transactions, showcasing the practical viability of threshold cryptography in real-world blockchain applications.



It also mitigates the risk of invalid signatures by verifying participant commitments and aborting upon detecting misbehaviour. However, it currently lacks protection against DoS attacks, such as those caused by malicious or unresponsive participants. The application can be improved by implementing mechanisms to prevent nonce reuse or weak randomness, both of which could lead to private key recovery by tracking nonce usage, ensuring each signing session uses fresh, securely generated nonces.

Finally, the *Github* repositories for both the library (*frost-sig*) and the application (*Aokiji*) can be accessed through the following QR codes.

Library's Repository **Application's Repository**

Research conducted by **Diogo Araújo**, **contact@diogoaraujo.com**
Under the scientific supervision of **Professor Dr. Paul Crocker**

At the (Rel)ease Lab in Universidade da Beira Interior / Instituto de Telecomunicações