Spectre Finance: A technical implementation DRAFT

Mrisho Lukamba
Protocol engineer & Tech Lead, Spectre Finance
MrishoLukamba@proton.me

April 15, 2024

Abstract

Blockchain technology capacitated Decentralized Finance (DeFi). This new financial paradigm democratized how financial players (i.e Traders, Exchanges, etc) access and interact with financial tools. We have seen Defi platforms move around $1.4T in volumes. Traders benefit from having options in interact with these platforms such as , Trading, Liquidity provision, Arbitrage, MEV, etc. In Defi traders with low and high amount of capital both benefit but the chances are more aligned with high capital traders. Spectre Finance is building a decentralized proprietary trading platform, connecting investors who will provide large trading capital to skilled and onchain proven traders with allocated capital to trade in Dexs and grow the capital. Traders will be tracked on their performance and have ranking, Spectre Finance will employ a proper risk management framework enforcing traders to have rational trades and control of the capital. We are helping moving forward how Defi can be used and accessed by empowering financial players in a more secure and trustless ways.

1 Introduction

To make Spectre Finance a robust and high throughput with lower latency platform for traders interacting with numerous Dexs with the allocated capital, It must focus on implementating a solution which employ cryptographically commitment schemes ensuring verification and proving of trade executions across Dexs and capital tracking, It must be interropable with other consensus systems and it must be secured under shared economic security.

We will be deploying the protocol in Polkadot, but we will be evaluating the asset inflows, Dex interactions and bridging performance. Our alternative ecosystem will be EigenLayer.

The interactions in Spectre Finance are defined with the following components. These components work in a modular architecture ensuring the failure of one

component does not affect the other.

1.1 Investor registration

Investors will first have to bridge native assets such as DOT, USDT, USDC as Spectre Finance will be first launched on Polkadot ecosystem. Bridging involves depositing assets to Spectre Derived Account on the Relay Chain and miniting equivalence of the assets per user account in Spectre parachain. The depositing of asset to SDA is what increases Total Value Locked in a protocol.

After depositing, the investor will have to allocate the balance to the related asset pool and the percentage ownership will be calculated.

1.2 Trader registration

Same as investors, traders will have to bridge asset and bond any amount above threshold to be registered. But traders have two registrations to make, one in the parachain and the other in the smart contract.

In the contract, traders will generate a pair of private and public key which will serve as onchain trading accounts linked to the trader. The onchain trading account private is opaque to traders as this account will be the one of which the funds will be deposited into for trading. The onchain trading private key will be used to sign trade transactions of which will be submitted to the target Dex network.

After generating the onchain tading keypairs, trader will have to register the public key to the spectre parachain for linking between trader sovereign account to onchain trading account

1.3 Oracles & light clients

The protocol for it to interact with non shared consensus Dexs/network it must have access to consensus commitments and state commitments proofs to prove the state transitions and data commited is final. The oracle or onchain light clients will be synced with the target Dex's network to provide block headers, consensus proofs and state proofs to Spectre parachain.

These data will be used to prove that the trade was executed in the Dex, the balance used and the returns made. This will be used to update the capital pool effeciently in a trustless manner.

1.4 Trade transaction inclusion, execution proof verification & balance updates

The protocol needs to process 2 trades (buy & sell). These trades are signed by onchain trading private key and submitted to the target Dex's network to be

executed. The buy trade involves deduction of allocated capital asset, resulting unrealized loss in the tota pool capital. As there are 2 distinctive accounts, total capital pool account and onchain trading account, the capital pool account tracks all allocated and non allocated capital, and onchain trading account tracks spent and unspent allocated trading capital.

After buy trade execution, the *consensus state proof at N block*, *transaction id* and *state (balance) proofs* must be verified and the capital pool balance shall be updated accordingly after reading state proofs. The capital pool will reflect a lesser amount than initially as the funds deducted was used in buy trade.

Then after sell trade execution, all proofs neccessary will be verified and the capital pool will be updated reflecting unrealized profits, as it measures an increased pool amount. The actual addition of funds will be present in onchain trading account but the pool must have a representation in order to track and do the accounting.

All trades and returns shall be recorded per trader and per asset capital pool, as this data is essential for withdraw process.

1.5 Shared Security

For Spectre protocol to operate under high trust assumptions on verifying trade execution commitments and for the oracle to be able to provide high trusted data, they all should live under shared economic security. As this increases the cost of attack on the protocols.

1.6 Withdraw

The withdraw functionality needs to take into account the following parameters, **Investor capital allocation blocknumber**, **number of transactions executed post allocation per investor**, **percentage pool ownership**, **unrealized balance**, **blocknumber at which the withdraw is scheduled**, **pending unconfirmed sell trade transactions**, **pool fees** and **traders returns allocation**

For withdraw to be effective and to not pose any security concerns, it will be a scheduled event upon which all capital allocated per investor will be frozen to not be included in the allocation fo trading. And the current allocated funds in which the withdrawing investor is part, will be waiting until all sell trades associated with it are verified and the balance is reflected.

Withdrawal process needs to be reviewed upon implementation as it is critical to be subjected in attacks

2 Protocol implementation

We will dive in details how each component interact with other components and outside world.

2.1 Investor onchain registration

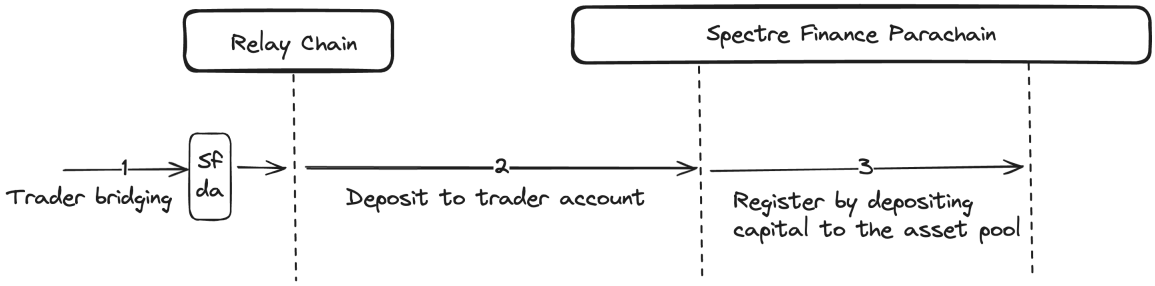

Figure 1: investor registration interactions

- Asset Pool capital contribution

$$P_{\mathrm{A}} = (P_{\mathrm{A\text{-}1}} + X_{\mathrm{i}})_{\,\mathrm{BT}} \tag{1}$$
$$\mathrm{BT} = BlockTime \tag{2}$$
$$P_{\mathrm{A}} = AssetPoolState \tag{3}$$
$$\mathrm{X} = InvestorAssetCapitalContribution \tag{4}$$

- All contributions will also be representated as percantage ownership of the pool

2.2 Trader onchain registration

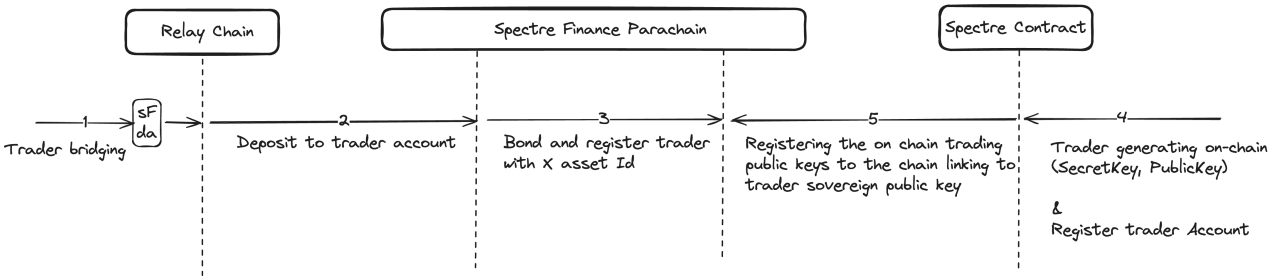

Figure 2: trader registration interactions

- oncontract

$$\text{register}(A_T) = (\text{sr25519}(S_k), \text{ed25519}(S_k), \text{EDCSA}(S_k)) \quad (1)$$

- onparachain

$$\text{bridging deposit \& bond} = lock(X_{\text{t}}) \quad (2)$$
$$\text{complete registration } T = lock(X_t) + (register(A_t)(P_k) + \pi) \quad (3)$$
$$\text{where } \pi = \text{proof of onchain trading account ownership} \quad (4)$$

2.3 Trade inclusion and execution proof verification

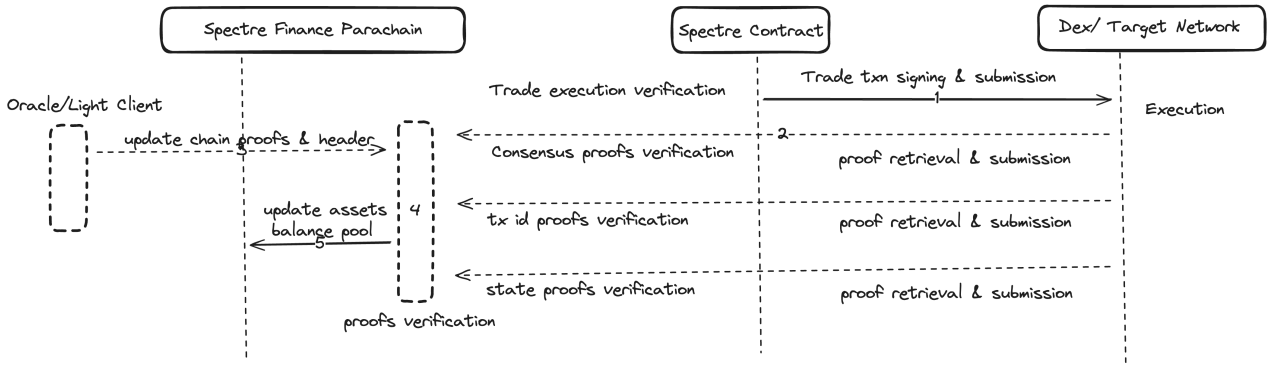

Figure 3: consensus, tx, state proofs verification

- execution & verification

$$(S_k, T) = \alpha_T => \text{signing transaction} \quad (1)$$
$$Execute(\alpha_T) = (C_p, S_p, T_p) \quad (2)$$
$$oracle/lightClient = (C_r, S_r, T_p) \quad (3)$$
$$verify\,(C_p, C_r, S_p, Sr, T_p, T_r) = bool \quad (4)$$

- updating pools for buy trades

$$readProofs(S_T, S_r) = balance \quad (5)$$
$$pool = Nassetbalance \quad (6)$$
$$updatePool(pool, balance) = UnrealiazedLossPool(pool - balance) \quad (7)$$

- updating pools for sell trades

$$readProofs(S_T, S_r) = balance \quad (8)$$
$$pool = assetbalance \quad (9)$$
$$updatePool(pool, balance) = UnrealiazedProfitPool(pool + balance) \quad (10)$$

2.4 Asset flow onchain

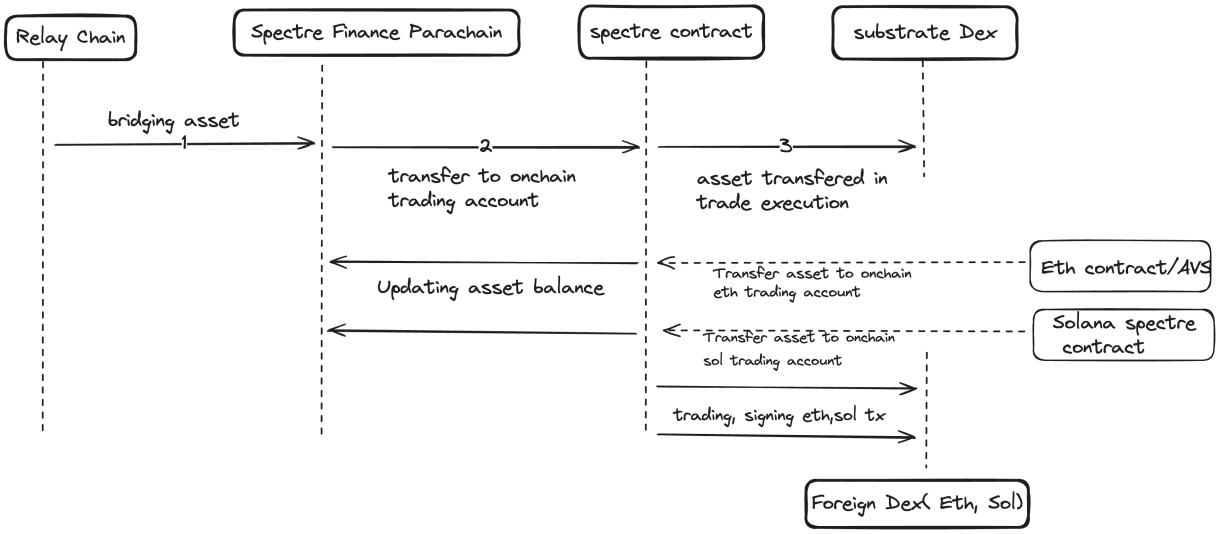

Figure 4: assets flow

- bridging involves depositing balance to Spectre Sovereign Derived Account (SDA) on RelayChain and minting equivalence of token in spectre parachain $bridge(Asset) = SDA(Asset)$
- Tranfering to onchain trading account involves taking balance from SDA and depositing to onchain trading account $SDA(asset) = OnChainTradingAccount(asset+)$ and updating the bridged asset to reflect the remaining asset in SDA
- The balance in onchain trading account can be used to trade assets in Dexs
- After trading the balance in onchain trading account will change and we are only tracking the main token and not the swapped one, so in updating pool balance, the actual tokens will be in onchain trading account while in the assetpools will be a reflection of them in totality.
- In interacting with foreign Dexs (i.e Uniswap), its either we bridge the tokens inside onchain trading account or we deploy a contract in Ethereum, Solana, etc and allow pools to exist on the contracts and the funds will be moving from those contracts to onchain trading account specific for that network. But for all that we need to maintain the reflection of those assets in spectre parachain

2.5 Withdraw

2.6 Asset Pools rebalancing

2.7 Bridges for moving assets from onchain trading accounts

3 Exploring zk proofs

Conclusion

Spectre Finance aims to have high throughput transaction verification processing, allowing traders to interact with Dexs and make N numbers of trades in asynchronous with trade execution verification in Spectre chain.

With this architecture we can achieve good user experience, low latency trades as users interact with Dex directly offchain securely and later verify their trades. We will be exploring which ecosystem is best of on chain asset liquidity and also exploring how we can interact with non Polkadot Dexs such as Uniswap in secured manner. As interaction with other foreign Dexs involve movement of asset to onchain trading account of that Dex's network account type (eg, EDCSA, ED25519).

We will also be exploring how we can use Zk proofs to verify trader sovereign account with trader onchain trading account keypairs, And also verification of trade execution We aim Spectre Finance to be a canonical multichain secured onchain proprietary trading protocol.