ZKP applied to portfolio risk reporting

Submission for #32 Groth16 zkSNARK Proof Verification Use Cases Part II contest

Author: @zealot72

Repository: https://github.com/zealot72/ton-proof-verification-contest

Date: 31/08/2021

1. Introduction

Nowadays there is an information asymmetry between investment funds and their clients. Clients don't have the full information on their portfolio content and are not able to verify if the asset manager is going beyond their preferred risk limit. Funds keep their investment strategies confidential in order not to put their competitive advantage at risk.

We want to propose a Zero Knowledge Proof based solution for funds to perform risk reporting in a way that enables clients to verify the proof of portfolio weighted risk and be sure it's not beyond their limit.

2. Problem

Individuals and corporations often give their savings to hedge funds or other investment funds for higher returns on their investments. It's possible for clients to track the fund's performance by the amount of returns but the portfolio content is usually unknown as hedge funds prefer to keep it confidential. Investors have different risk thresholds based on their risk appetite. Currently, clients have almost no means to check that their preferences are actually taken into account by the asset managers. As for the funds they cannot make their portfolio and investing strategies public as this would eliminate their competitive advantage.

3. Solution

3.1 Idea

To solve the above mentioned problem we want to propose a prototype of a Zero Knowledge Proof based tool for investors and funds to check the compliance with the risk threshold set by the clients without disclosing the portfolio content. This solution is aimed at eliminating the need to make a trade-off between transparency and confidentiality.

3.2 Description

One of the most common ways to measure risk is by volatility. In most cases, the higher the volatility, the riskier the security. Information on volatility of the assets trading at stock market is publicly available.

The proposed solution will enable individual and corporate investors to verify that their portfolio risk is within the desired limit (r_{min}, r_{max}). For the purposes of risk reporting, asset managers can inform clients on the weighted average risk of their portfolios (R) by adding the individual risk (r_i) of the assets composing the portfolio multiplied by the relative weight (ω_i) of each asset in the portfolio:

$$r_{min} \leq R = \sum_{i=1}^{N} \omega_i \times r_i \leq r_{max}$$

3.3 Implementation

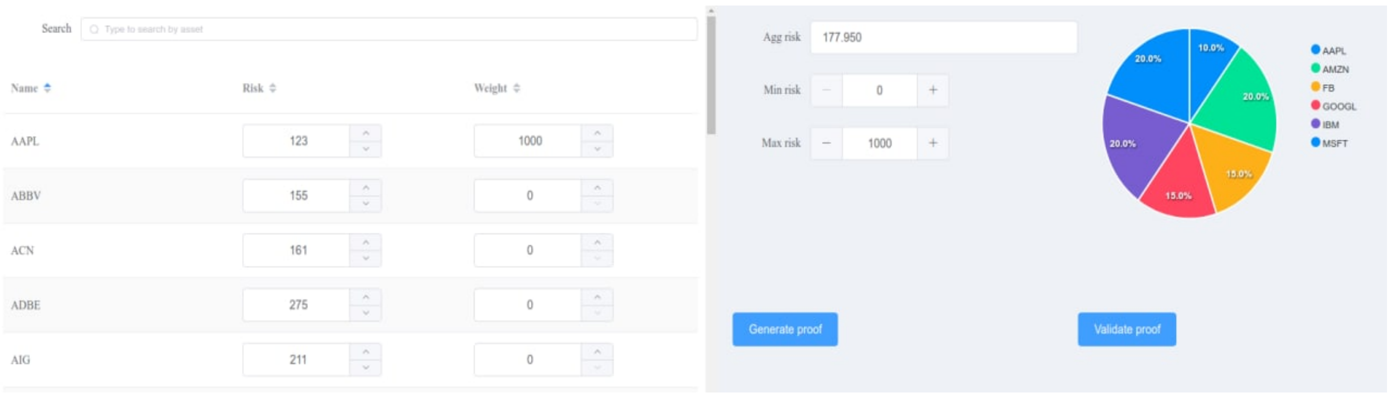
As the funds need to keep their portfolios private we are going to make sure the assets and their weights in the portfolio remain undisclosed. In the proposed prototype we will focus on Exchange Traded Funds (ETF), those funds usually follow a particular index, we are going to assume that all the assets in the portfolio are from the S & P 100 Index. Thus, we are going to calculate the aggregated risk for the stocks in that index.

In the course of implementation we created a .csv file with assets belonging to S & P 100 Index, we took their volatility from tradingview.com and assigned dummy weights to each asset in the portfolio. The aggregated risk of the portfolio was calculated according to the formula above. In the proposed prototype N = 101, which corresponds to the number of assets composing S & P 100 Index.

After the aggregated risk is calculated, the investment fund can generate a proof of the portfolio risk being within a certain range and send this proof to the client. The client can verify the received proof and be sure the asset manager is not going beyond the agreed risk limit.

3.4 Web application

In order to illustrate the idea, a small web application was created using vue.js for the frontend and Flask for the backend. It can configure risks and weights of portfolio assets, calculate average risk, generate and validate proofs.



Search: Type to search by asset

Name	Risk	Weight
AAPL	123	1000
ABBV	155	0
ACN	161	0
ADBE	275	0
AIG	211	0

Agg risk: 177.950
Min risk: 0
Max risk: 1000

4. Usage instructions

Clone contest repo:
```
git clone
https://github.com/zealot72/ton-proof-verification-contest
--recursive
```

Build cli tool:
```
mkdir build && cd build
cmake ..
make cli
```

Go to contracts directory:
```
cd ../contracts
```

Generate proof using prepared assets.csv file with risks and weights
```
../build/bin/cli/cli --generate --assets "../assets.csv" -l
101
```

Verify generated proof offchain using Blueprint library:
```
../build/bin/cli/cli --verify
```

Compile verification contract with Nil's tvm-solidity compiler:
```
tondev sol compile verification.sol
```

Verify generated proof in blockchain emulator using TestSuite4 library:
```
python verify.py
```

All detailed instructions are in the submission's GitHub repository.

5. Limitations

NIL network was unavailable during the contest, so all contracts were tested locally using TestSuite4 and tonos-se from NIL Foundation.