Publicly Verifiable Generalized Secret Sharing Schemes and Their appplications

Particularly, The newly proposed cryptographic primitive Publicly Verifiable Generalized Secret Sharing (PVGSS) is implemented using GoLang and a proof of concept implementation about a decentralized exchange (DEX) based on Ethereum is presented.

Pre-requisites

- `Golang` https://go.dev/dl/ (https://go.dev/dl/)
- `Solidity` https://docs.soliditylang.org/en/v0.8.2/installing-solidity.html (https://docs.soliditylang.org/en/v0.8.2/installing-solidity.html) Version: 0.8.20
- `Solidity compiler (solc)` https://docs.soliditylang.org/en/latest/installing-solidity.html (https://docs.soliditylang.org/en/latest/installing-solidity.html) Version: 0.8.25-develop
- `Ganache-cli` https://www.npmjs.com/package/ganache-cli (https://www.npmjs.com/package/ganache-cli)
- `Abigen` Version: v1.14.3

  ```
  go get -u github.com/ethereum/go-ethereum
  go install github.com/ethereum/go-ethereum/cmd/abigen@v1.14.3
  ```

File description

- `crypto` The folder includes detailed implementation of LSSS, Shamir SS, GSS (on Group G) and PVGSS.
- `bn128` The folder contains the source codes of curve BN128, which is compatible with EVM.
- `main.go` run this file to test the functionalities of the framework.

- `test/dex_test.go` run this file to test the dex contract and get gas usage.
- `compile/contract/` The folder stores contract source code file (.sol) and generated go contract file.
- `compile/compile.sh` The script file compiles solidity and generates go contract file.
- `genPrvKey.sh` The script file generates accounts and stores in the `.env` file.

How to run

1. Generate private keys to generate the `.env` file

```
bash genPrvKey.sh
```

2. start ganache

```
ganache --accounts 20 --mnemonic "pvgss" -l 90071992547 -e 100
```

3. Compile the smart contract code

```
npm install && bash compile.sh
```

4. Test dex gas usage

```
cd test
go test -v -timeout 30m -run TestDexGasSSS
```

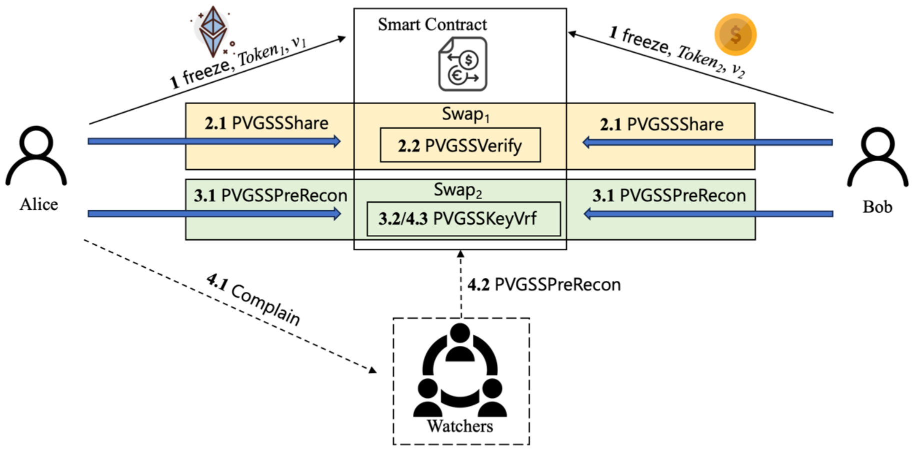
5. Run the main.go

```
go run main.go
```

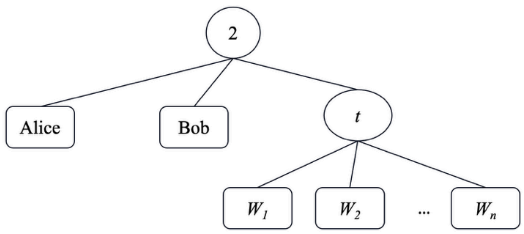
Introduction to the application of DEX

Based on the proposed Publicly Verifiable Generalized Secret Sharing (PVGSS) scheme, we design a Decentralized Exchange (DEX) to allow exchangers to swap tokens fairly and simultaneously. In this research, we merely focus on ERC-20 token exchange. The DEX involves two roles: exchangers and watchers. Anyone can be exchangers and watchers. Each exchanger holds specific types of ERC-20 tokens, while multiple watchers collectively form a passive notary committee to address potential disputes.

Take two exchangers, i.e., Alice and Bob, and `n` watchers as an example. The DEX optimistically runs in two communication rounds for Alice and Bob, as shown by below Figure.



In the first round, each exchanger commits to a secret using `PVGSSShare`, where all the `n+2` entities are considered as shareholders. The correctness of the commitment is guaranteed by the `PVGSSVerify` algorithm. The access structure is designed as `(2 of (Alice, Bob, (t of (W_1, W_2, ... , W_n))))`, as shown by below Figure.



In the second round, each exchanger reveals its decrypted share using `PVGSSPreRecon`. The correctness of share decryption is ensured by `PVGSSKeyVrf`. Then, both Alice and Bob jointly recover each other's secrets using `PVGSSRecon`.

In the pessimistic occasion where a player complains to the watchers, who will be involved to resolve dispute using `PVGSSPreRecon`. Note that the access structure of the DEX not only tolerates a faulty exchanger but also tolerates `n-t` faulty watchers.