

Ethereum↔FreeTON Bridge concept



Three layers

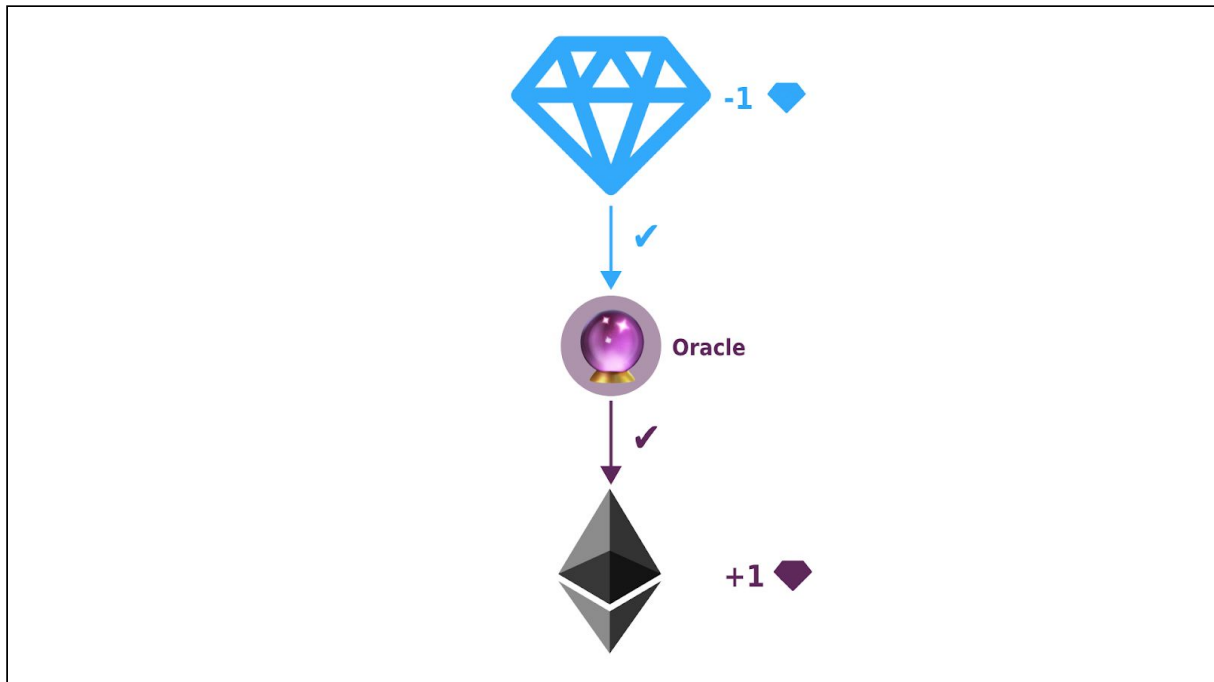


1. **Free TON layer** - smart-contracts on Free TON.
2. **Security layer** - system between Free TON and Ethereum.
3. **Free TON** - smart-contracts on Ethereum.


Let's talk about the **Security layer** and step by step going from simplest architecture to decentralized.


Security layer

Simplest architecture.

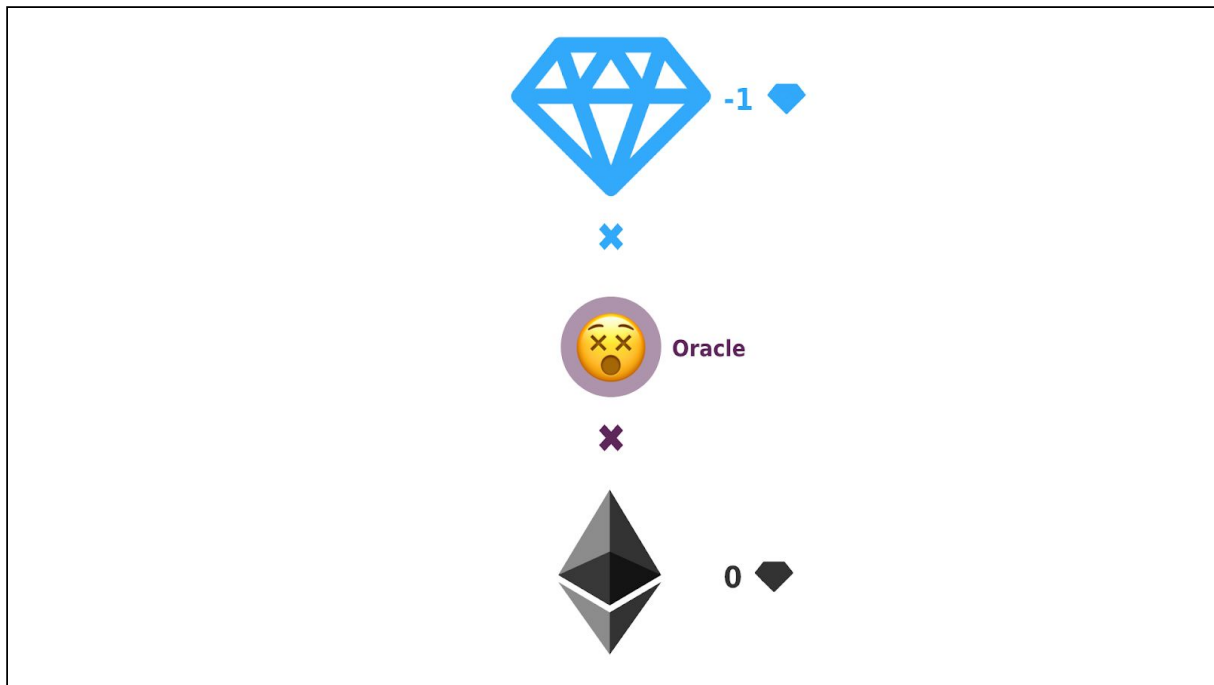


Simple bridge FreeTON > Ethereum

1. Send 1  to special smart-contract on FreeTON
2. **Oracle** check this.
3. **Oracle** generates some 1 ERC-20 FreeTON tokens in Ethereum.

What's the problem? What if the oracle is broken ?

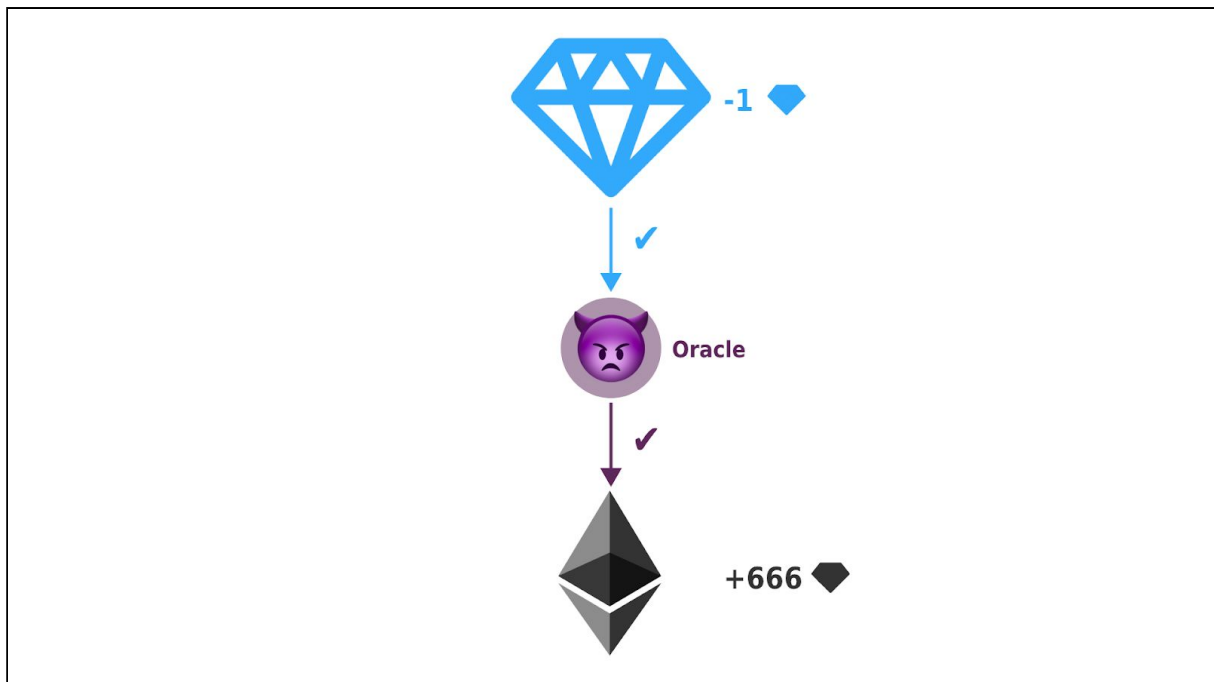
Broken Oracle. 🤡




- Send 1 💎
- Gain 0 💎

What is Oracle cheating? 😈

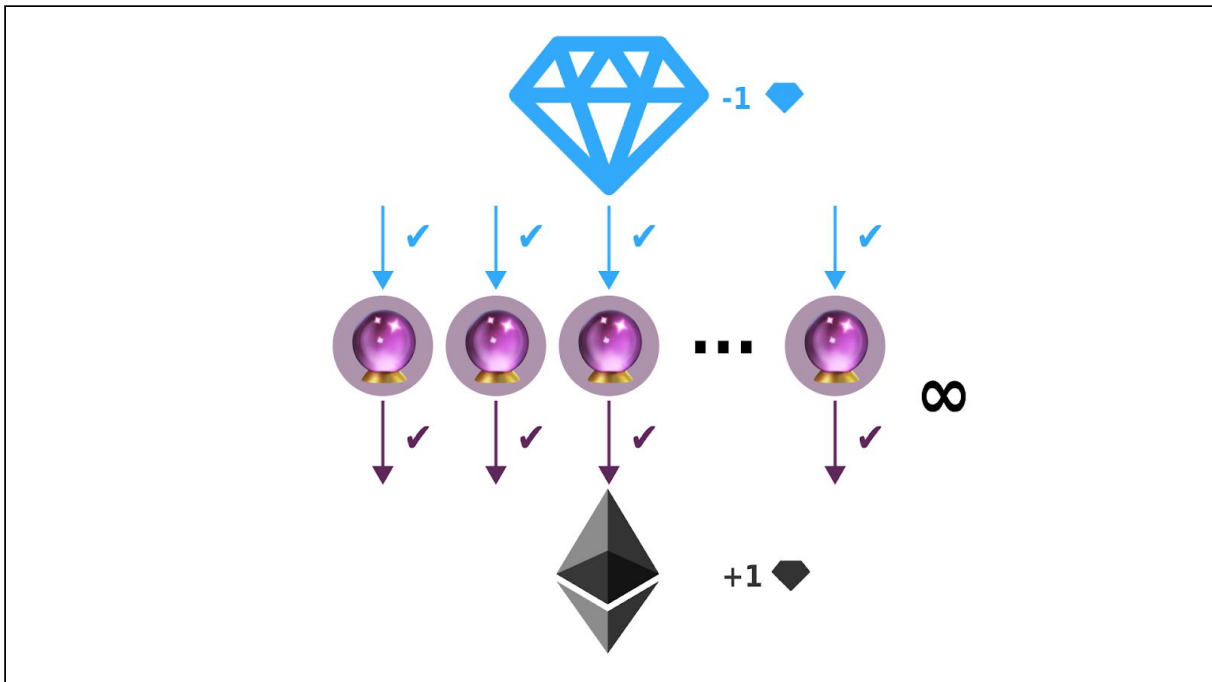
Simplest architecture. Oracle cheating. 😈



- Send 1 💎

- Gain 666 

Not good. How to fix it? Add more oracles. Let's try to decentralize it. Decentralize architecture. Infinity oracles for everyone

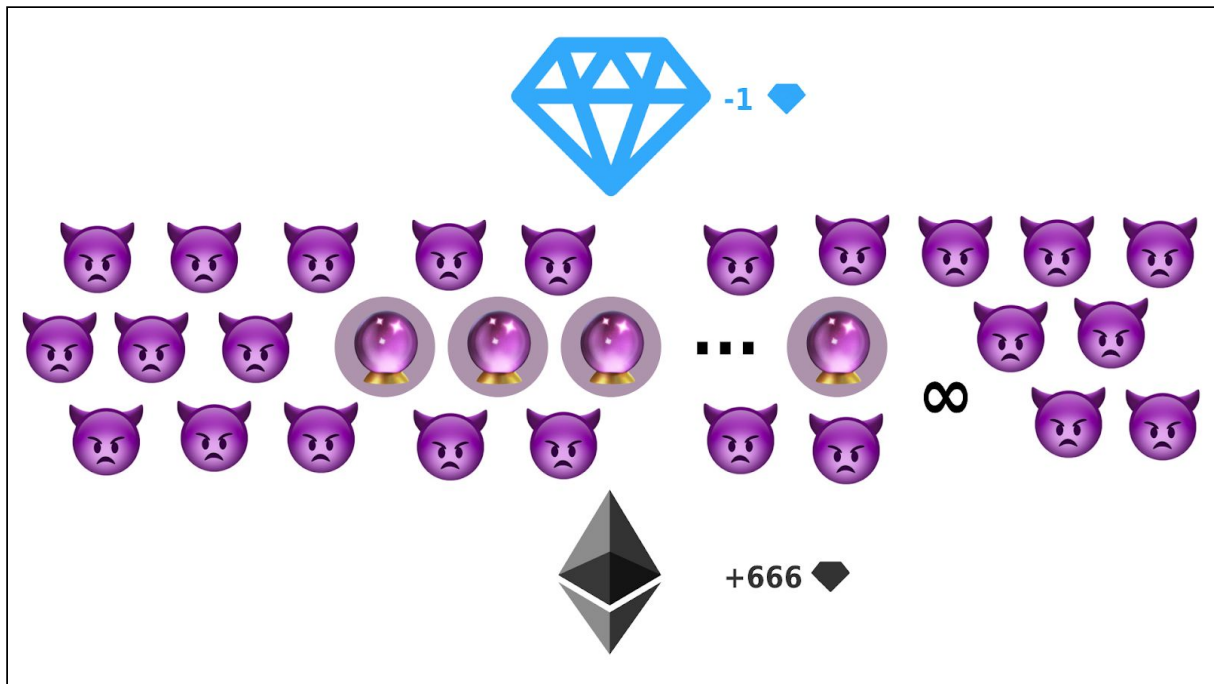


Easiest way is to allow everyone to send transactions to the blockchain. Everyone can become oracul. Everyone can confirm transactions or everyone can send some data to blockchain [SPV](#). The decision which data is correct is made by the majority. This is how bridges like [BTC Relay](#) work. Totally freedom.

What is the problem? Big problem: anyone can join the system. One hacker can create 1000 cheating oracles - the bridge will crash and users will lose money.

Decentralize architecture.

Infinity oracles and the hacker.

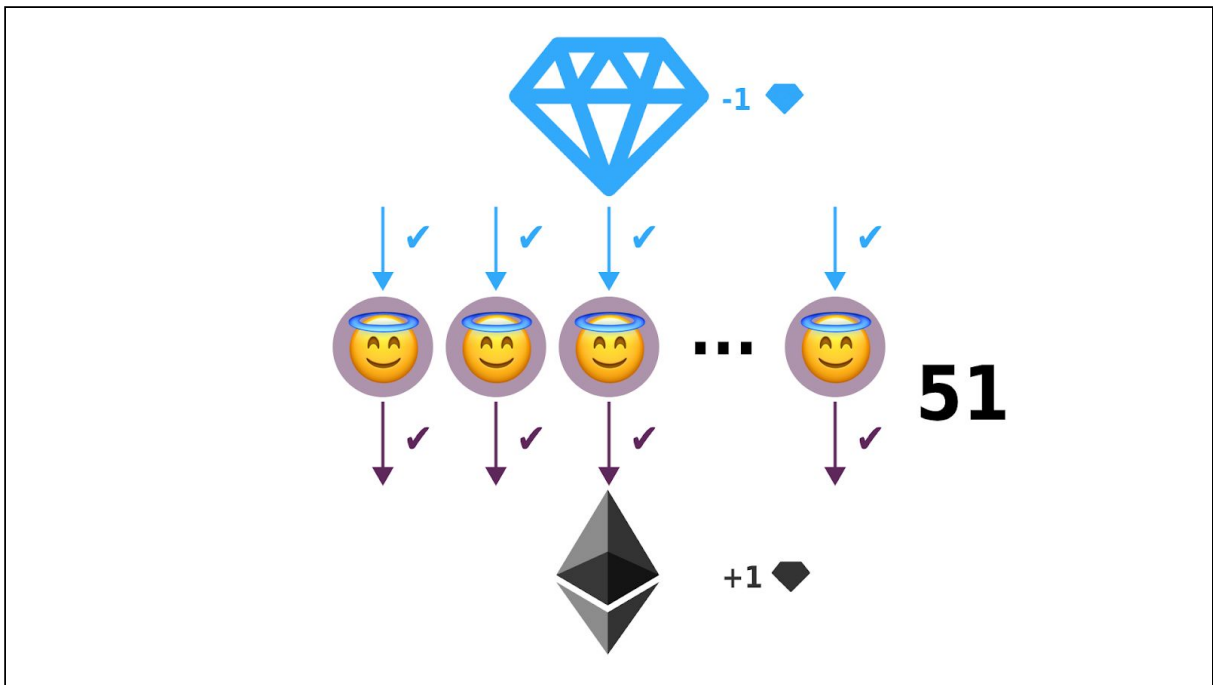


Need protection from this.

Solution

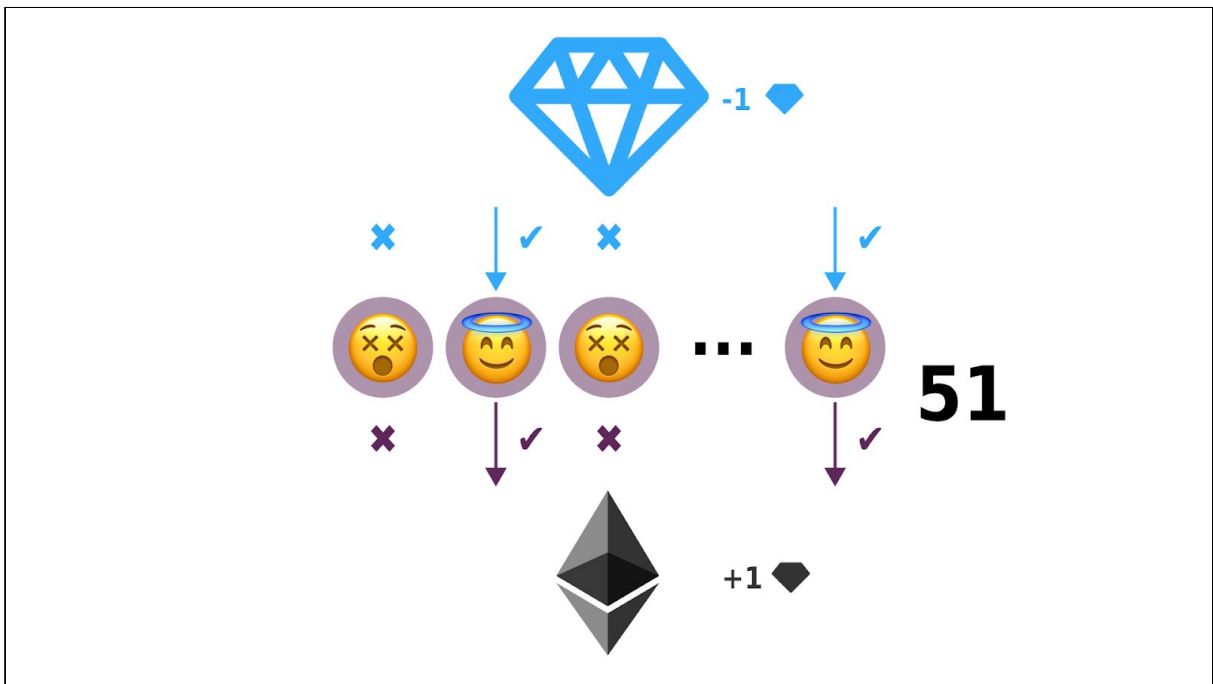
1. Limit count of oracles.
2. Give access to bridge smart-contracts only trusted peoples after private KYC
3. One person - one access. 😇

Limited count of oracles. 😊



Good. But what if some person loses access or stops work.

Broken oracles.



Need create reserve 😊

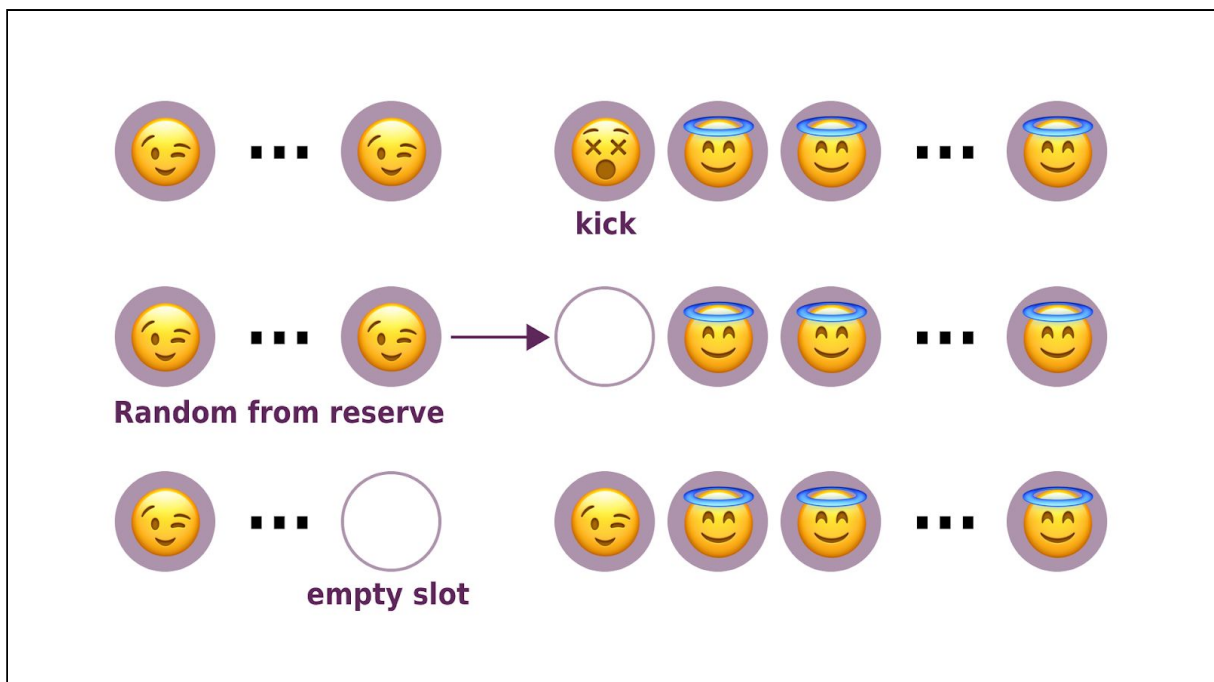
Reserve. 😊



Preservers do not have access to bridge contracts. They wait for their own time. They all receive **5%** of oracles' earnings. This is needed so they don't leave.

What happens if one oracle is broken or cheating.

Broken or cheating oracle and reserve. 🤡👹😊



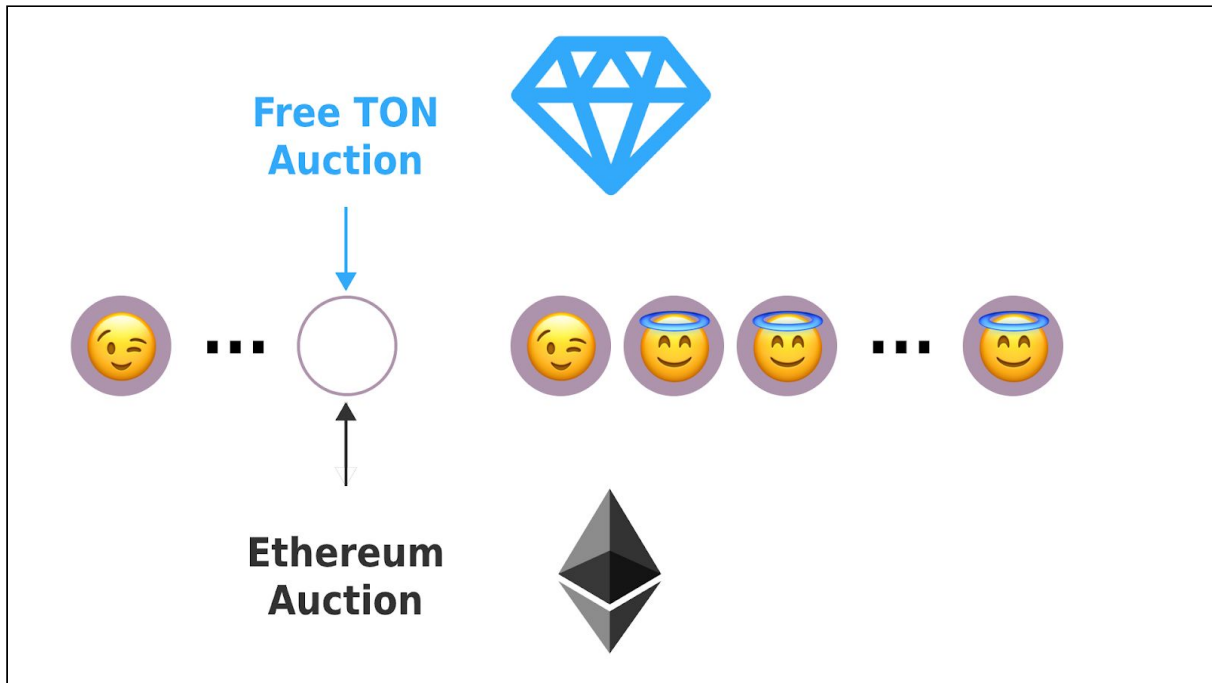
If the oracle is broken or cheating - all oracles send transactions but he doesn't. 🤡👹

1. Kick broken oracle and destroy his access.

2. Select random access from reserve.
3. Create an auction for a new reserve slot. 🏛️😬

How does auction work? 🏛️

Auction. 🏛️



Two block chains

There are two access to the auction:

- For Free TON bridge smart-contract.
- Same for Ethereum.

It's normal. It is normal for one oracle to have only one access. For example, an oracle can generate tokens in Ethereum but cannot in Free TON.

Auction math

Auction is reverse. Price starts from 1,000,000 coins (Crystals or ETH) and decreases to 15% every 10 minutes until 0.001

Time h:m	Coins
0:00	1,000,000.000000000
0:10	850,000.000000000
1:00	377,149.515624999

2:00	14,2241.757136172
10:00	58.228491993
20:00	0.003390557
21:20	0.001000000
24:00	auction closed

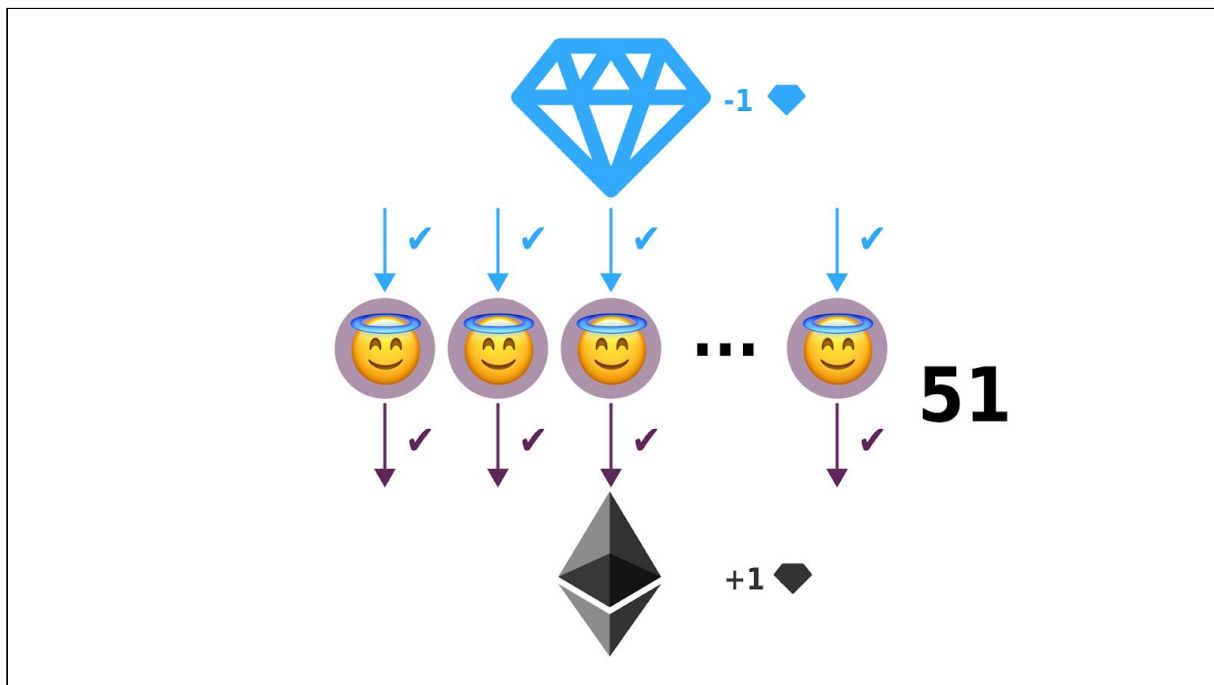
Auction restart

If no one buys access, auction restart. Restart every 24 hours. More than one auction can be started at the same time.

Delay between auction closing and oracle starting

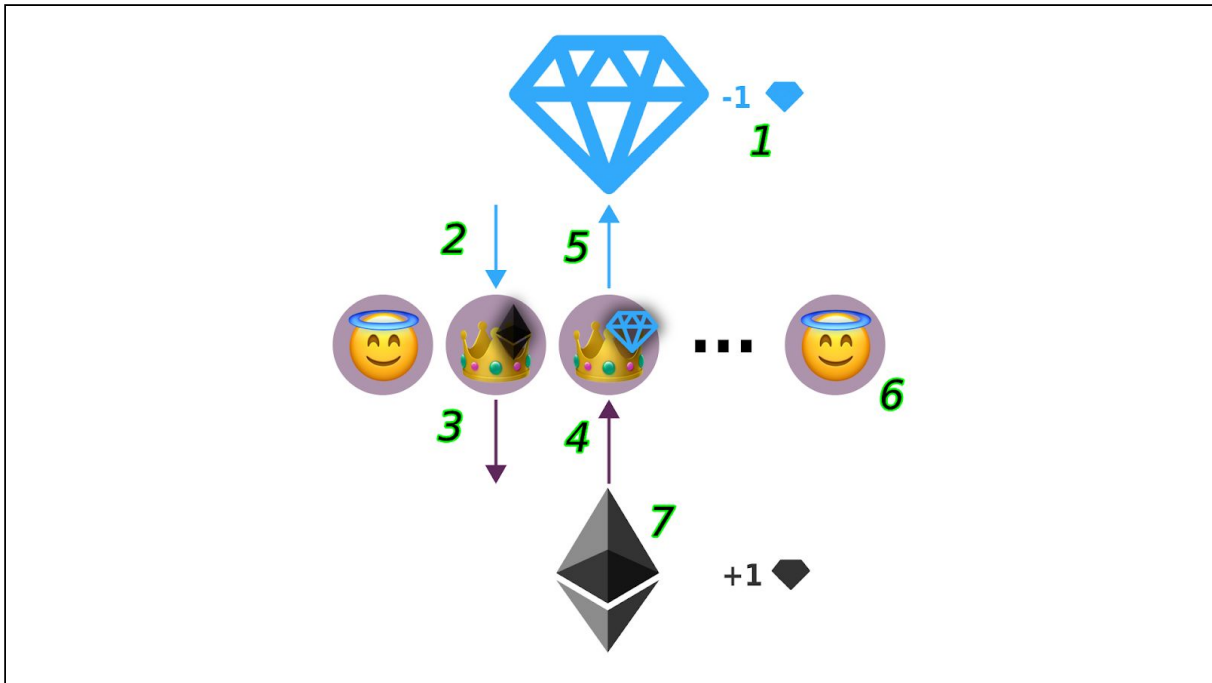
One week. It's enough time to set up the server and run oracle node.

Leaders. 👑




Look at this again. If every oracle sends transactions to blockchain they spend a lot of money on fees. This is not good.

How to solve it? Need to select from oracles two leaders who can send info into bridge smart-contract. First for Free TON, second for Ethereum. All other oracles become validators. Validators check info and reject wrong transactions. If a transaction is valid, oracles are silent. Silent is agreement.

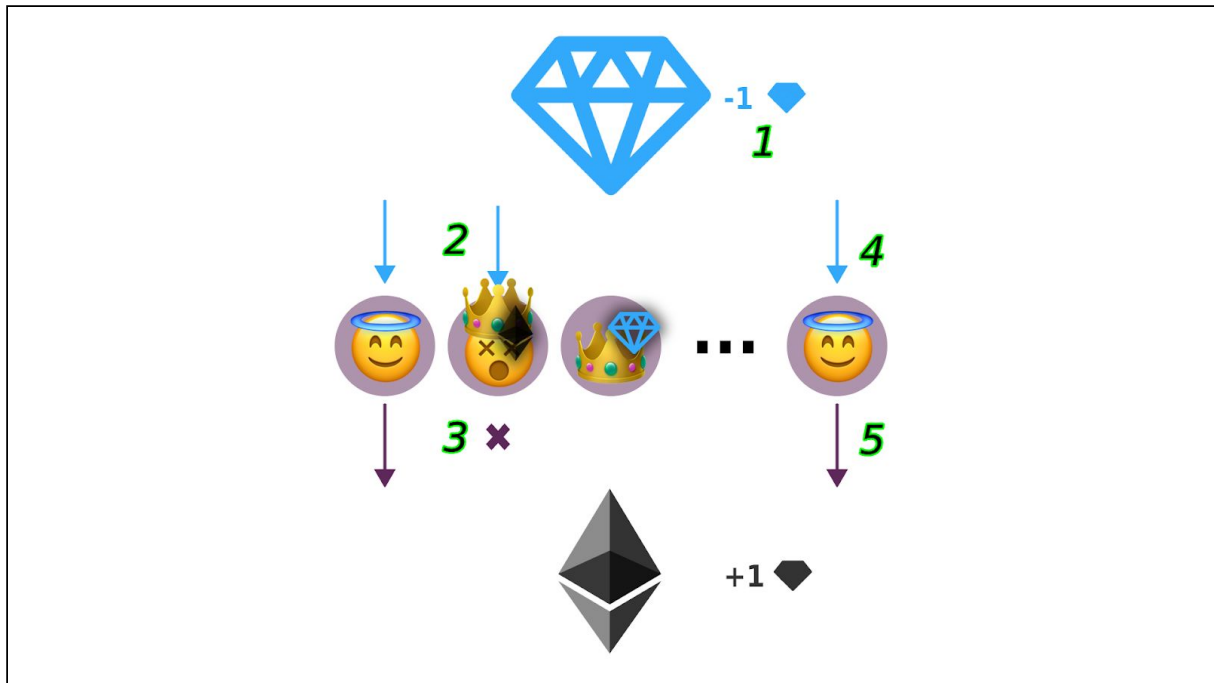


How it's works.

1. The user send 1  to a special smart-contract in Free TON. Contract lock money at 3 hours.
2. Ethereum Leader catches a transaction.
3. Ethereum Leader makes a request to create an ERC-20 Free TON token. He must do it in 1 hours.
4. Free TON Leader catches a request.
5. Free TON Leaders send transactions to lock Ethereum by all time. He must do it in 1 hours.
6. Validators check what all is right and don't do anything. After 3 hours Ethereum is totally locked and ERC-20 tokens can be free.
7. After 3 hours the user can receive tokens on their own address.

What is something wrong? For example, a leader is broken or cheating.

Broken or cheating leaders. 👑



1. The user sent 1 💎 to a special smart-contract in Free TON. Contract lock money at 3 hours.
2. Ethereum Leader catches a transaction.
3. Ethereum the Leader does not send a transaction to Ethereum in 1 hour.
4. Validators catch Free TON a transaction and do not find the same in Ethereum after 1 hour.
5. Validators send valid transactions and add one fail-point in Ethereum smart-contract to the leader. Data of valid transaction data selected by majority.
6. Next steps like in a valid transaction.

If the Leader sends wrong data, validators send rejection of transaction.

How to check what validators do their work?

Validate validators.

The only way to verify that validators are actually working is to send incorrect data transactions sometimes.

In **1%** cases leaders generate additional **fake transactions**. If some validators do not reject transactions he gains **fail-points**.

Fail-points system

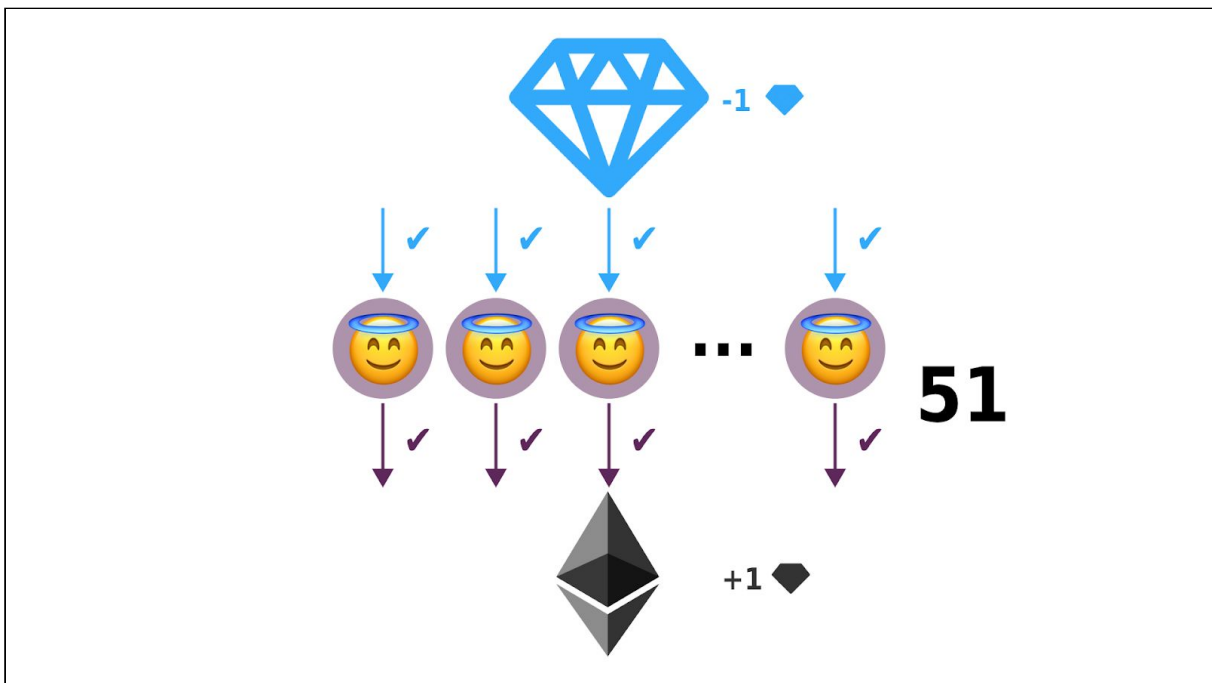
If oracle do not reject or confirm a transaction if >50% do it he gains a fail-point. Normally all oracles have 1% fails from all transactions. If fail-point / All_transaction_count > 0.01 need check probability. If probability > 0.000 000 001 it is ok. If no, any transaction kicks oracle.

Success-points system

Validators gain success-points for rejection and confirmation of transactions. First early validator gets 50 points. All others gain -1 points. Second validator gets 49 points etc. Validators sorted by points. First validator with biggest score gain x2 from all earned coins. Second x1.98. Next 1.96 etc. Points set to zero every week.

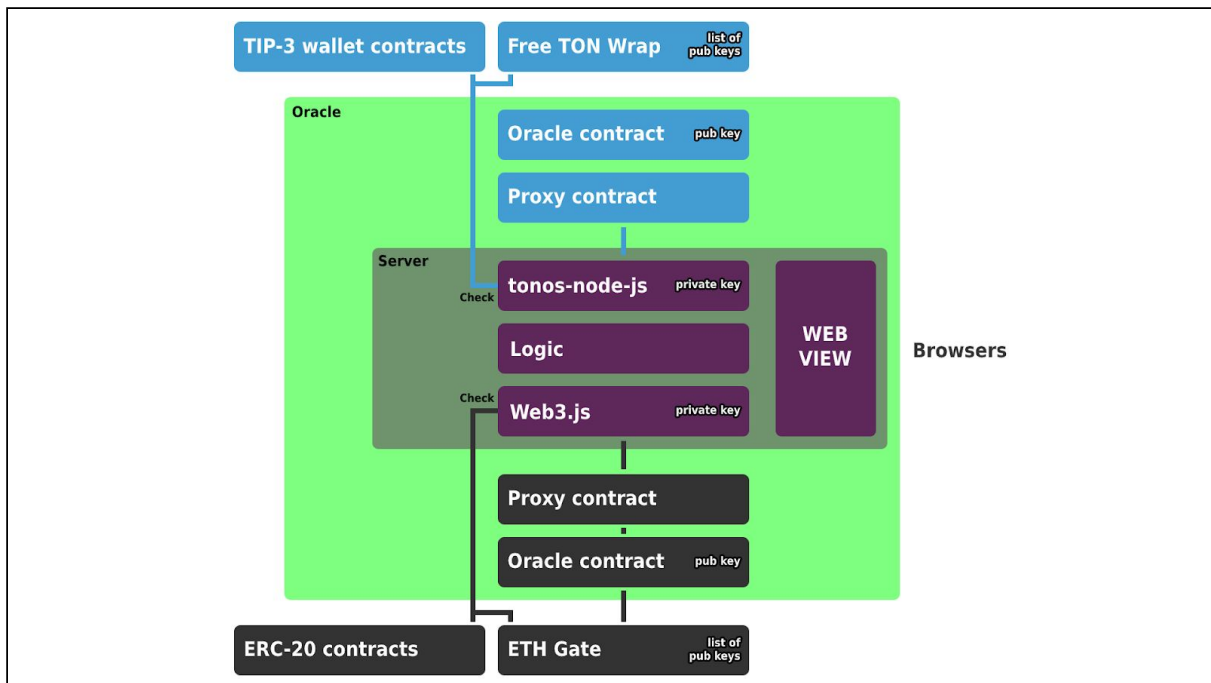
Momental transaction.

Users can send special transactions that confirm not only the leader. Add validators to confirm it like in the first version of decentralization architecture in this document.



This is faster. Users do not need to wait but cost **x51**.

Oracle



Server

Code that runs on VPS or dedicated server.

- Web3.js
 - Read settings from wrapper contract (list of tokens, who is leader now etc.)
 - Catch transactions in Free TON and process it.
- Logic
- Web3.js
 - Read settings from wrapper contract
 - Catch transactions in Ethereum and process it.
- Web view - site that show statistic and current status of oracle
- Main settings
 - Private key for Free TON Proxy contract
 - Private key for Ethereum Proxy contract
 - Address of Free TON Proxy contract
 - Address of Ethereum Proxy contract
 - Function of current Ethereum fee calculation
 - Ethereum node address
 - Free TON node address (If dapp server decentralization implemented)

Proxy contracts

Oracle can store private keys on PVS. Need to protect oracles from keys stealing. Proxy contracts do it. Oracle stores keys only for proxy but does not show keys for oracle

contracts. Proxy contract can run oracle contract methods. If someone steals proxy keys just create a new proxy and change your server provider.

Oracle contract

Only from oracle contracts you can call methods of bridge's contracts: Free TON Wrapper and ETH Gate. Public keys of oracle contracts are stored in bridge's contracts.

Free TON Wrapper and ETH Gate

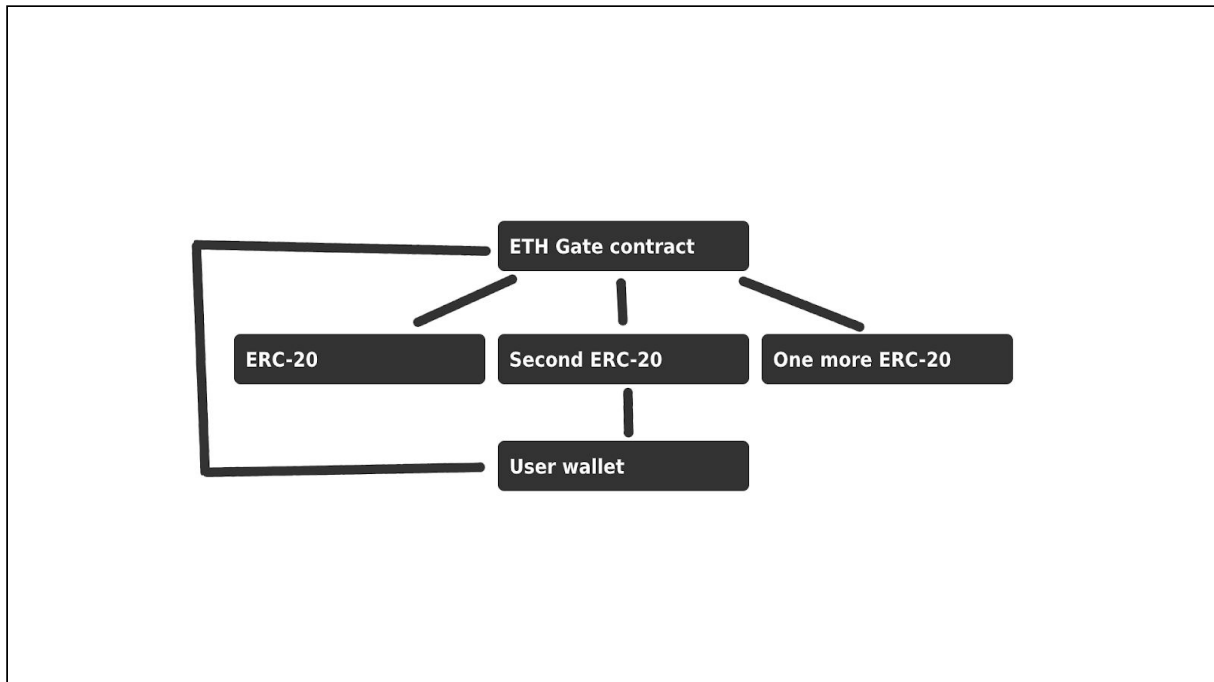
Contracts deployer by system architecture. It does all the dirty work.

How to become oracle

1. Clone contracts from github.
2. Deploy oracle contracts.
3. Get access to the bridge from your oracle contracts. Use an auction or get access at start.
4. Deploy proxy contracts.
5. Add proxy contracts address to oracle contract and oracle contract address to proxy contracts.
6. Setup server.
7. Clone oracle code from github.
8. Set settings.
9. Run.
10. Send a special transaction that you are ready to start.

Bridge smart-contract scheme

Ethereum



User can:

- Send coins directly to ETH gate contract.
- Send ERC-20 tokens to **ETH Gate contract** balance that is stored in **ERC-20 contract**.
- Unlock ETH from **ETH Gate contract** and back coins to **User wallet** if exchange failed.
- Unlock ERC-20 from **ETH Gate contract** balance and get tokens back to **User wallet** balance if exchange failed.

ERC-20

Just simple implementation [ERC-20 standard](#)

ETH Gate Contract

- Store and change list of oracles addresses.
- Stores list of ERC-20 addresses.
- Has own balance in ERC-20 contracts.
- Can move tokens from its balance to another balance.

Methods

Oracles

getOracles() returns (address[] addr, uint[] exchanges, uint[] failPoints, uint[] successPoints)

Returns a list of active oracles with their parameters.

getReserveOracles() returns (address[] addr, uint[] exchanges, uint[] failPoints, uint[] successPoints)

Returns a list of reserve oracles with their parameters.

getCurrentLeader() returns (address addr)

Returns the address of the current leader.

ERC-20

getERC20Addresses() returns (address[] addr)

Returns the address of the current leader.

addERC20Address(address addr) returns (uint votingId)

Create voting. Only an oracle can call.

removeERC20Address(address addr) returns (uint votingId)

Only an oracle can call.

voteERC20Address(uint votingId, bool vote) returns (bool success)

Only an oracle can call.

getERC20AddressVotingList() returns (uint[] votingId, uint[] action, address[] addr)

Return list of address voting.

Change commission

changeCommission(uint commission) returns (uint votingId)

Create voting. Only an oracle can call.

voteChangeCommission(uint votingId, bool vote) returns (bool success)

Only an oracle can call.

getChangeCommissionVotingList() returns (int[] votingId, uint[] commission)

Return list of commission voting. Can call only by oracle.

Auction

getAuctions() returns (uint[] id, uint[] startTime, uint[] endTime, uint[] currentPrice)

Return list of auctions.

buyAccess(uint auctionId) returns (bool success)

Buy access from oracle contract.

User

sendETH(string freeTonAddress) returns (bool success)

Exchange ETH to Free TON tokens.

sendERC20(address tokenAddress, string freeTonAddress) returns (bool success)

Call to oracles exchange ERC-20 tokens to Free TON tokens. Call this after sending ERC-20 to gate balance

unlockETH() returns (bool success)

Try to unlock all your locked ETH in the gate contract.

unlockERC20() returns (bool success)

Try to unlock all your locked tokens on ERC-20 Gate balance.

getETHInfo() returns (uint[] count, uint[] timeToUnlock)

Get info about all locked coins.

getERC20Info() returns (address[] addr, uint[] count, uint[] timeToUnlock)

Get info about all locked tokens.

Generate tokens

releaseTokens(address addr, uint count) returns (uint votingId)

Returns vote id. Only the leader can call.

voteTokensReleasing(uint votingId, bool vote) returns (bool success)

Only validators can call.

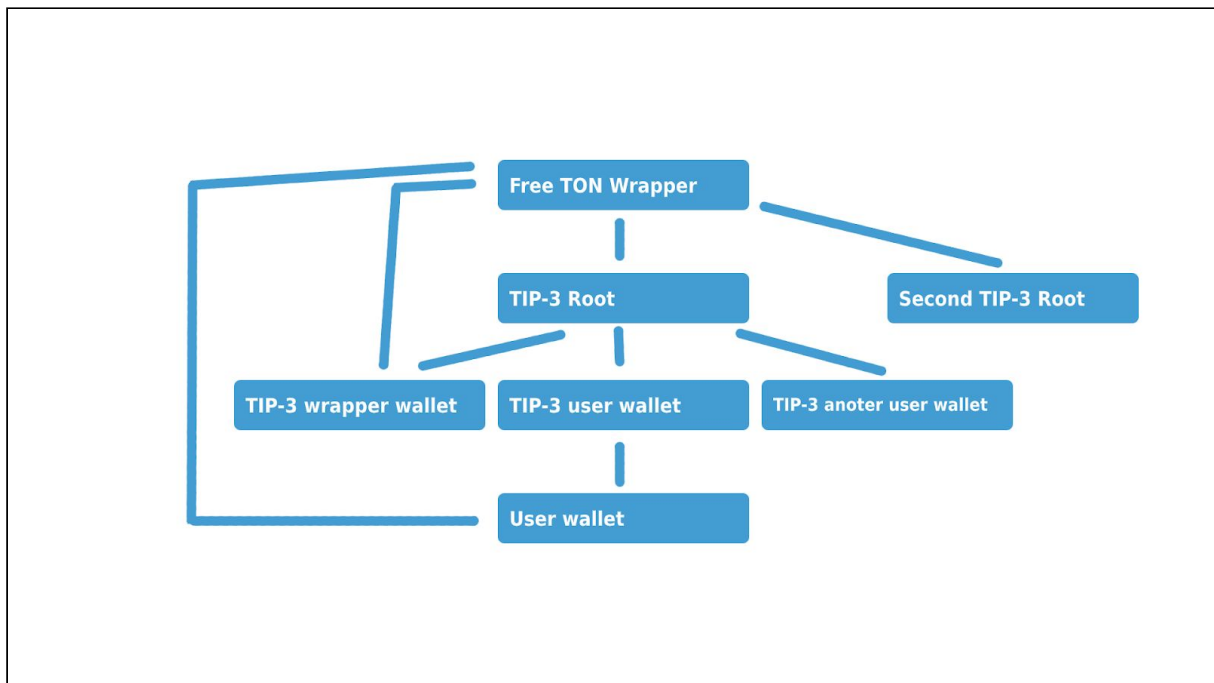
getTokensReleasingVotingLits() returns (uint[] votingId, address[] addr, uint[] count)

Return list of tokens voting.

`getTokensReleasingVotingInfo(uint[] votingId)` returns (address addr, uint count, address[] oracle, bool[] vote)

Return info about voting with a list of votes.

Free TON



User can:

- Send coins directly to the **Free TON Wrapper contract**.
- Send TIP-3 tokens to **TIP-3 wrapper wallet**.
- Unlock Crystals from **Free TON Wrapper contract** and back Crystals to **User wallet** if exchange failed.
- Unlock TIP-3 tokens from **TIP-3 wrapper wallet** and get tokens back to **TIP-3 user wallet** if exchange failed.

TIP-3 Root and TIP wallet

Just simple implementation of [TIP-3 standard](#)

Free TON Wrapper

- Store and change list of oracles addresses.
- Stores list of TIP-3 Root addresses and TIP 3 wrapper wallets.
- Has its own TIP 3 wrapper wallets.
- Can move tokens from TIP 3 wrapper wallets to other TIP wallets.

Methods

Very similar like in Ethereum, but we need to make corrections that must work with TIP-3 not with ERC-20. =)

```
deployTIP3Wallet(address rootAddress, uint walletPublickey) returns  
(bool success)
```

Get info about all locked tokens. Only an oracle can call.

Author

Nikita

Telegram: [@kokkekpek](#)

Forum freeton.org: [@kek](#)