Cryptocurrencies with Security Policies and Two-Factor Authentication

Florian Breuer (KIT); Vipul Goyal (CMU and NTT); Giulio Malavolta (MPI-SP)

inquiries@florian-breuer.de
giuliø.malavølta@høtmail.it (substituting ø with o)
vipul.goyal@gmail.com

Recovery is challenging with cryptocurrencies

- What happens if one user's key is lost or stolen?
  - No bank to call and no authority to rely upon to get your funds back
  - A number of high profile thefts have been carried out
- These issues are not unique
- Traditional banking has decades of experience in them
  - Different banks have different security policies
  - Security policies vary by account type (business vs private)
  - 2FA is widely deployed

Leading research question

"Can we take the lessons learnt in the traditional banking domain, and apply them fruitfully to blockchain-based systems, without compromising their decentralized nature?"

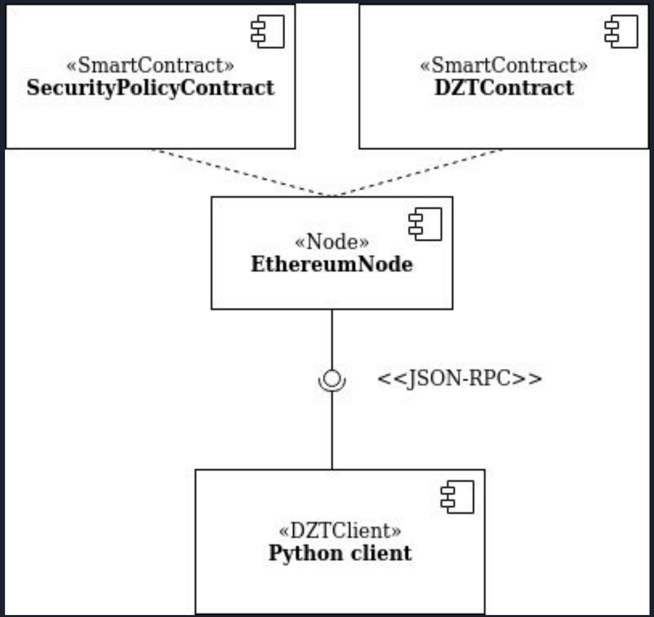
- Contributions:
  - The definition and instantiation of a distributed zero-tester (DZT)
  - The implementation of the DZT and U2F on the Ethereum blockchain

Design principles

- Our contributions shall have
  - Distributed Trust
  - Reliability and Guaranteed Output Delivery
  - Low Latency
  - Computational Efficiency
- In the implementation part we use our contributions to verify transactions that a user generates (similar to banking 2FA)

System Overview

- The 2FA method is implemented as smart contract on Ethereum
- We use a security policy contract to work out which transactions need to be verified



Security Policies

- The contracts supports
  - Flag for 2FA every transaction to any new beneficiary.
  - Flag a transaction if the sum of all coins transferred (since the last flag) is above a predefined threshold.
  - Flag a transaction if the sum of all funds that are transferred in a specific timeframe (a sliding window) exceeds a predefined threshold.
  - Flag a transaction if the sum of all coins sent to a specific beneficiary (since the last flag exceeds a predefined threshold.
- It adds minimal costs to transactions and works with the two 2FA implementations

Scenario - Security Question

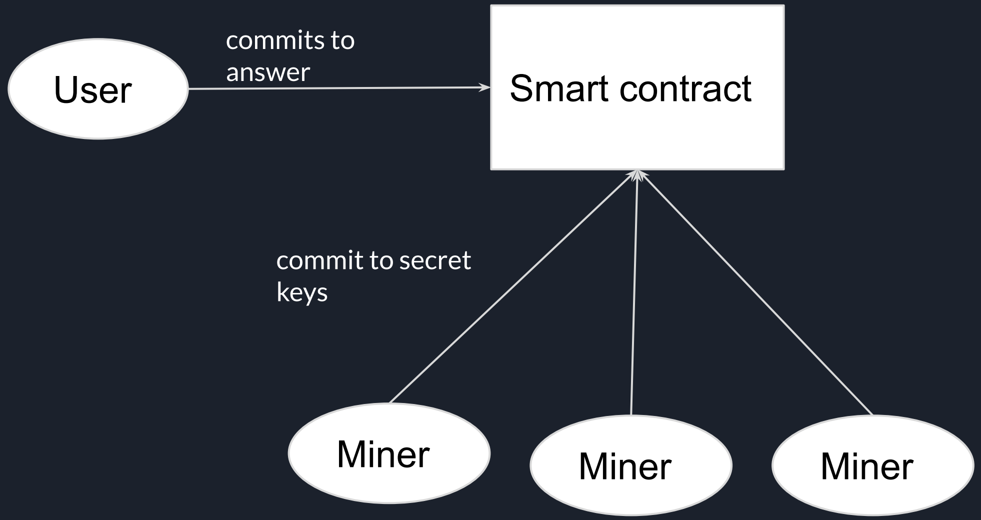
- A user wants to additionally secure their blockchain currency with an answer to a security question

- Problems that arise with blockchains
    - Answer can have low entropy (e.g. first car brand) -> offline bruteforce is possible
    - Answer can’t be hashed -> other primitives need to be used

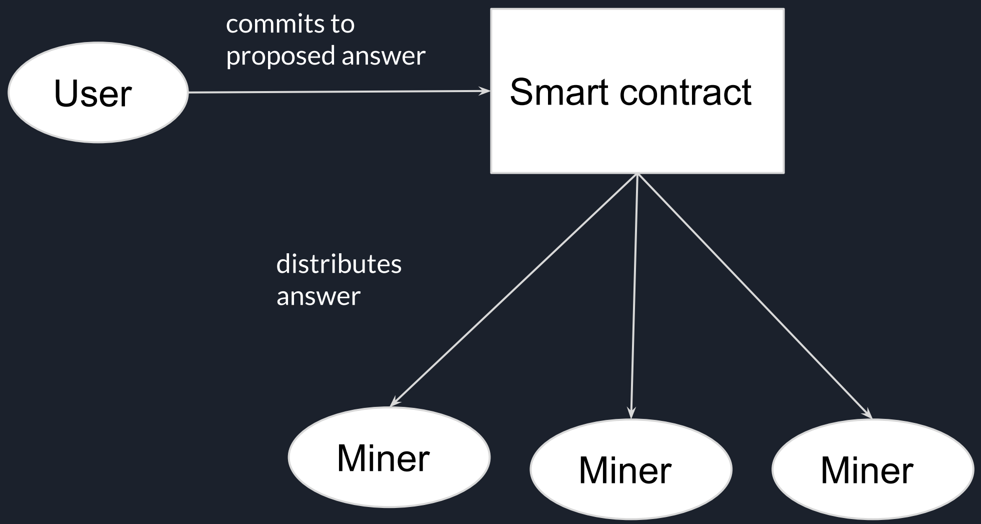
Distributed Zero-Tester (DZT)

- We present the notion of a DZT which solves the security question challenge
- The protocol is executed by:
  - The user (owner of the currency)
  - Miners, which form a group that is used for verification
  - A smart contract that verifies that all parties behaved according to the protocol

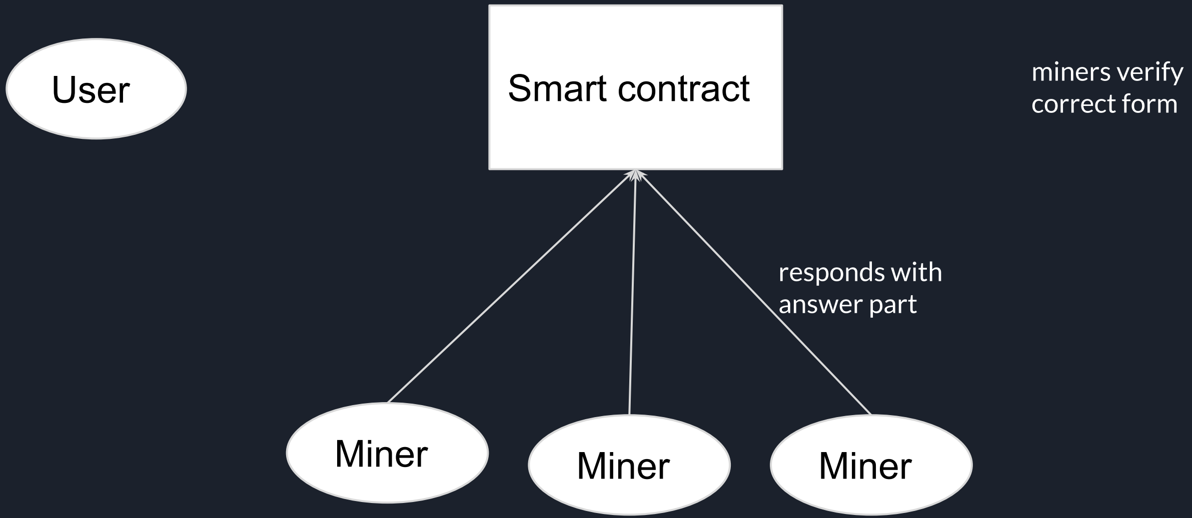
DZT Round: Setup



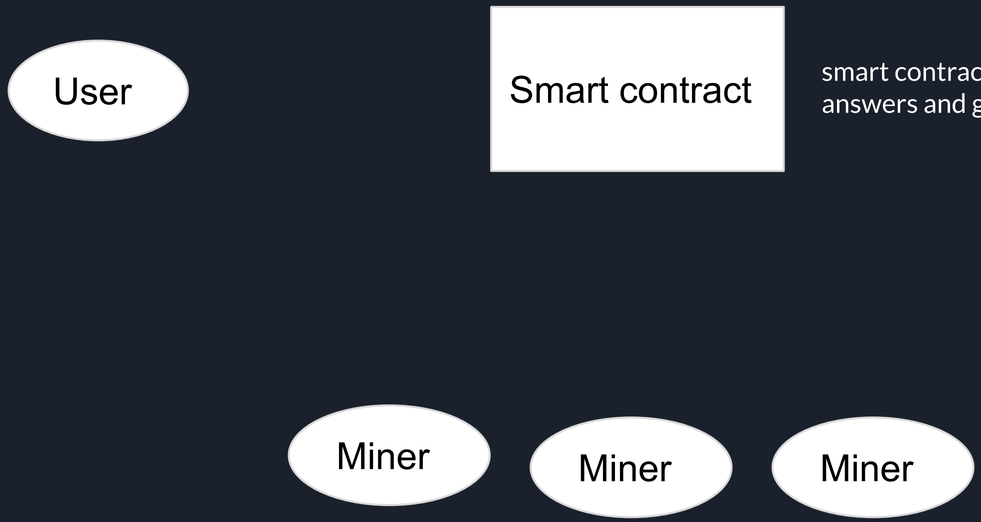
DZT Round: Query



DZT Round: Response



DZT Round: Verdict



smart contract verifies correct form, sums up answers and gives verdict

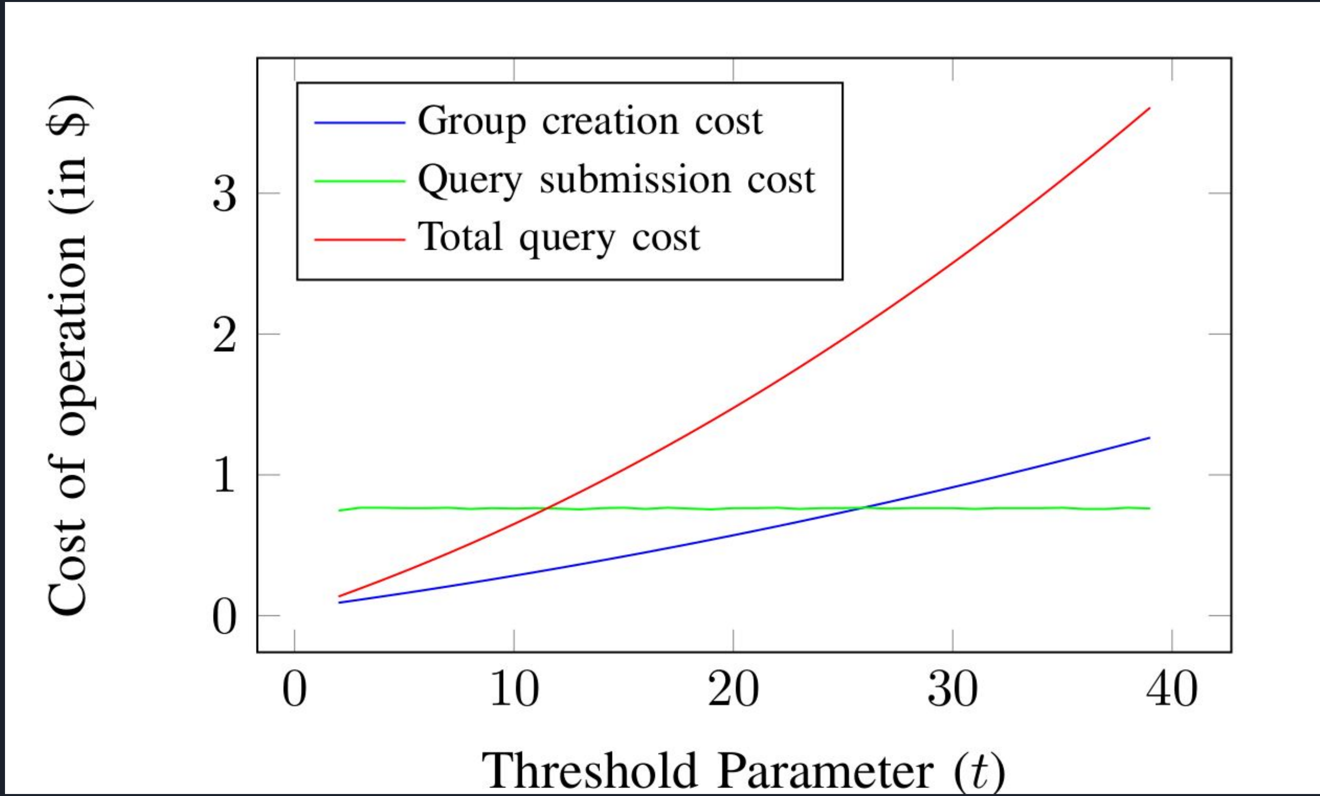
DZT

- The DZT guarantees
  - Round Optimality
  - Guaranteed Output Delivery
  - Public Verifiability
  - Resilience Against Offline Dictionary Attacks
  - Resilience Against Replay Attacks

- We instantiate the DZT using
  - a threshold homomorphic encryption scheme
  - bilinear maps

- We implement the DZT using the 256bit version of the Barreto-Naehrig curves

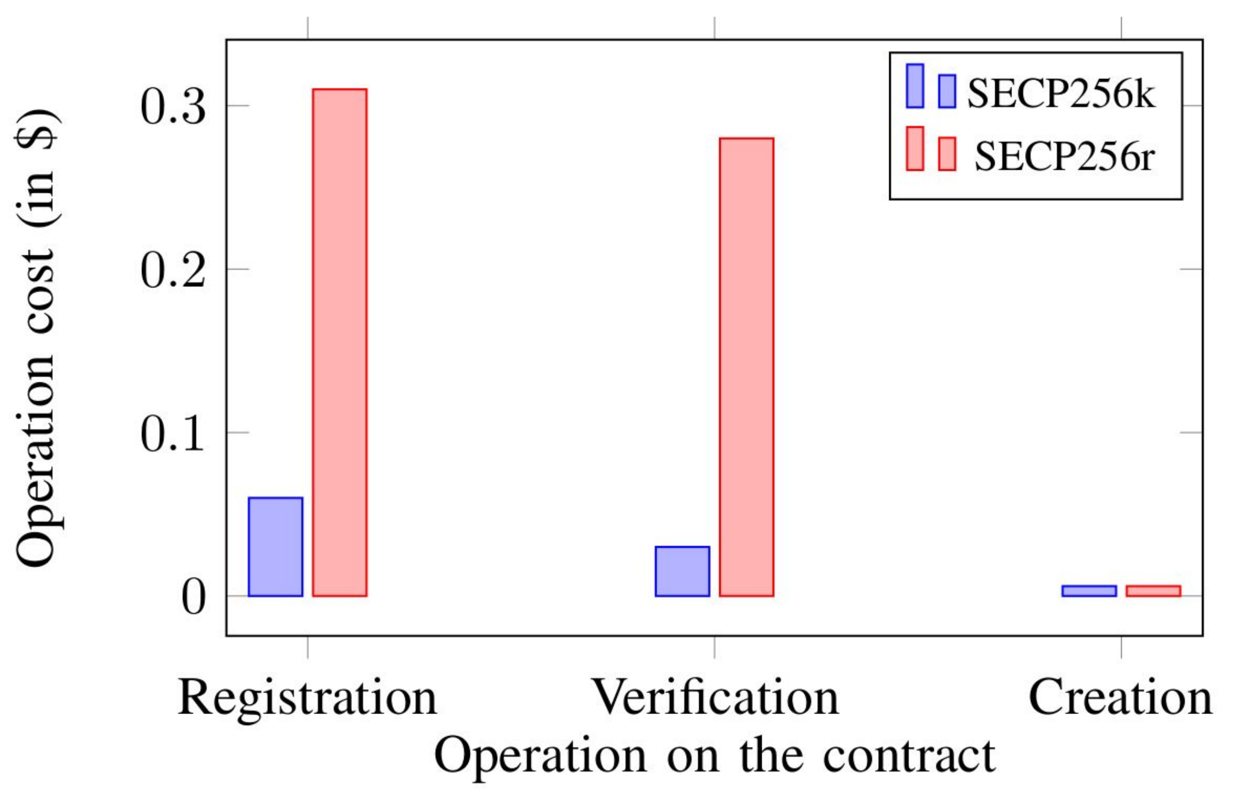
DZT implementation



Scenario - Token

- A user wants to additionally secure their blockchain currency with a hardware token
- We use U2F which is
  - a challenge response based protocol
  - based on ECC
- U2F consists of 2 different phases
  - Registration: Token is registered on the smart contract
  - Verification: Token is queried to verify a transaction

U2F implementation



Thank you for your attention!

The code can be found at https://github.com/FBreuer2/security-policies-in-crypto