A Plug-and-Play Long-Range Defense System for Proof-of-Stake Blockchains

<u>Lucien K. L. Ng</u>, Panagiotis Chatzigiannis, Duc V. Le,
Mohsen Minaei, Ranjit Kumaresan, Mahdi Zamani

Georgia Tech
VISA Research

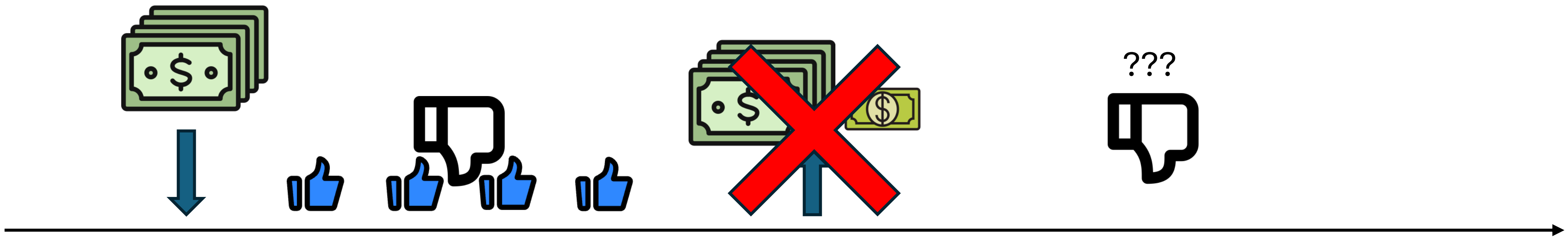
From PoW to PoS

- Proof-of-Work (PoW) consensus is highly energy-inefficient
  - Validators (Miners) use ton of electricity just for reaching consensus

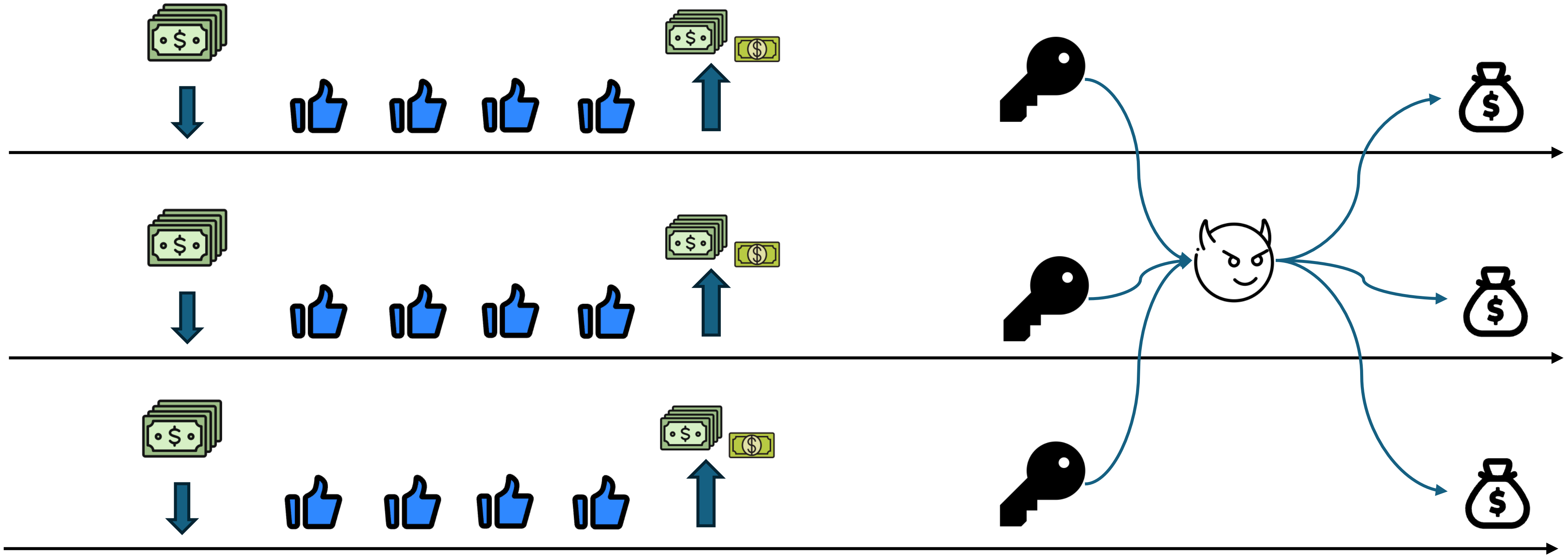
- Proof-of-Stake (PoS) is more energy-efficient
  - Leaders are selected based on their staked wealth on-chain

How (Penalty-Based) PoS works?

- Validators stake their coins on the blockchain
- If they comply with the protocol, they will earn reward
  - And they can withdraw their stake and reward after a lock-up period
- If they misbehave, their stake will be forfeited
- What if they misbehave after they have withdrawn their coins?

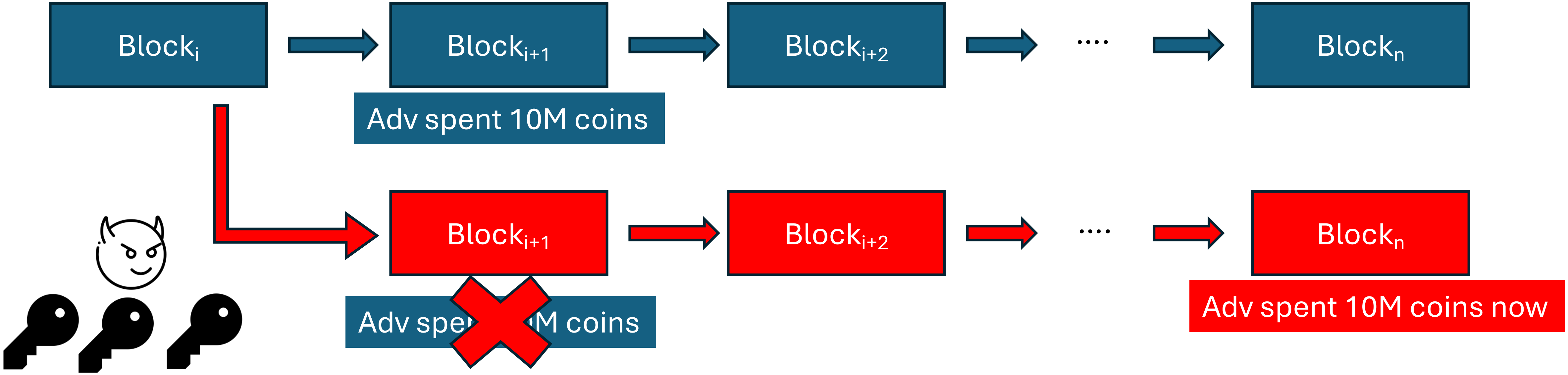


Posterior Key Corruption



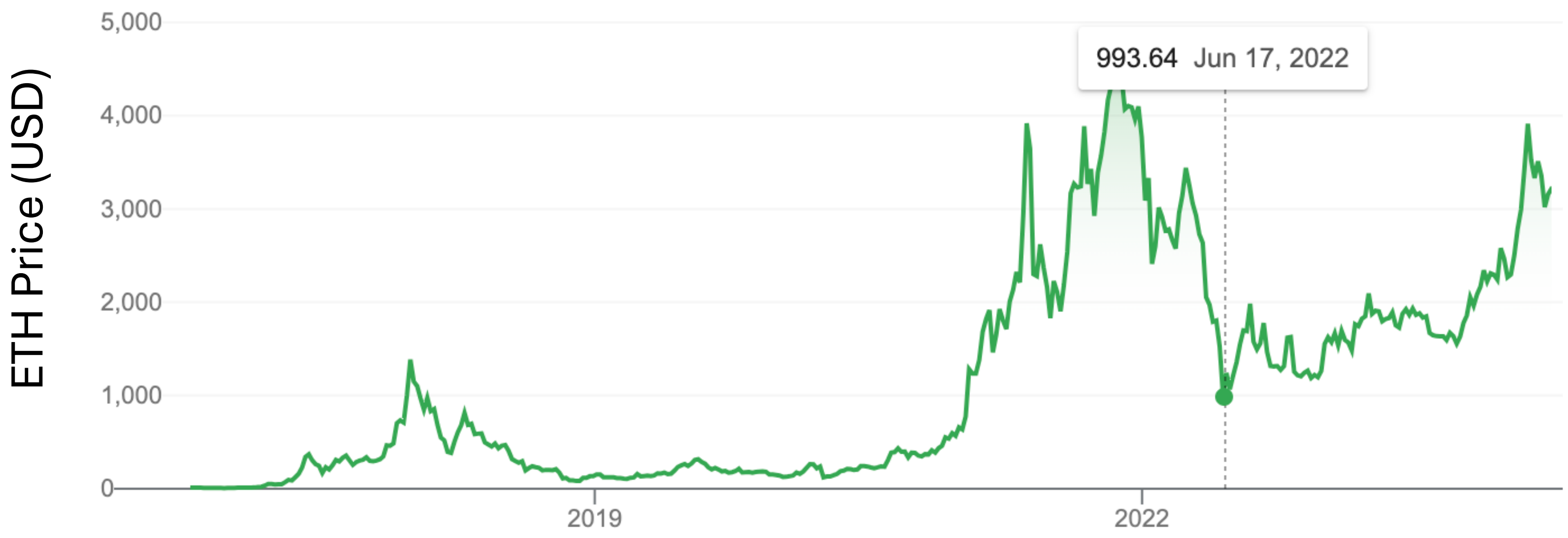
Long-Range Attack

- Once an adversary has gathered enough old validation keys
  - it can fork another valid chain and double-spend!



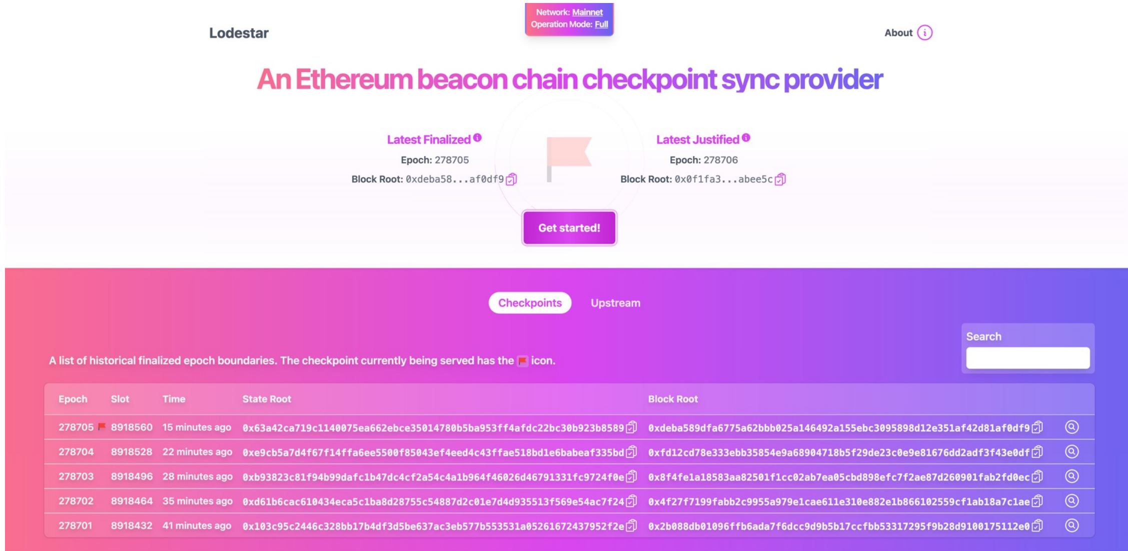
Possibility of Long-Range Attack

- Coin values can fluctuate a lot
- Selling old keys become more profitable (than protecting the assets)
- Attackers launch long-range attacks when coin value bounces back



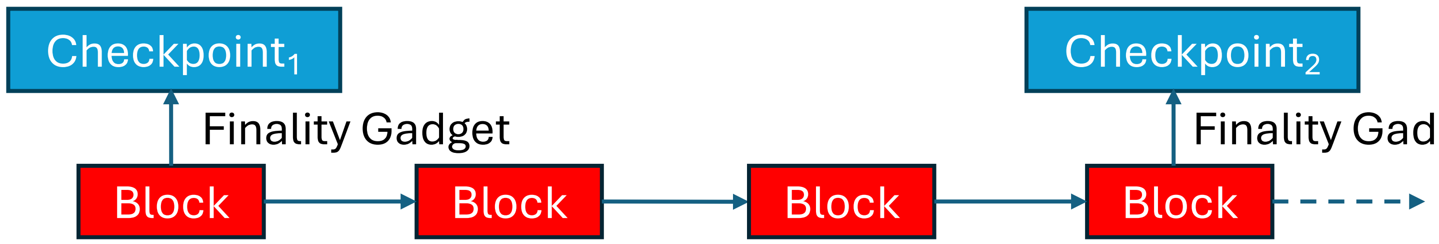
Unsatisfying Solution: Checkpointing

- Digests of old blocks are hardcoded in the node's software
  - Centralization Issue: the software developers can launch attacks
- A few centralized servers broadcast the digests
- When there are conflicts, who should the client trust?



Unsatisfying Solution: Finality Gadgets

- Ethereum requires 2/3 of the validators to sign on the checkpoints
  - (Otherwise, the transactions in the checkpoint are not finalized)
  - It assumes <1/3 validators are malicious
- It is just asking the adversary to acquire more old keys



Unsatisfying Solution: Always-Online Nodes

- Servers monitoring the chain know which fork is authentic
  - The checkpoints produced first are the genuine ones
  - (More about it later)

- But how about clients?

Clients

- New clients have no knowledge about blockchain's history
- Existing Clients might be offline for a long period
- They may see two equally valid checkpoints when logging on

- The client are also "light"
  - with limited computation and communication capability

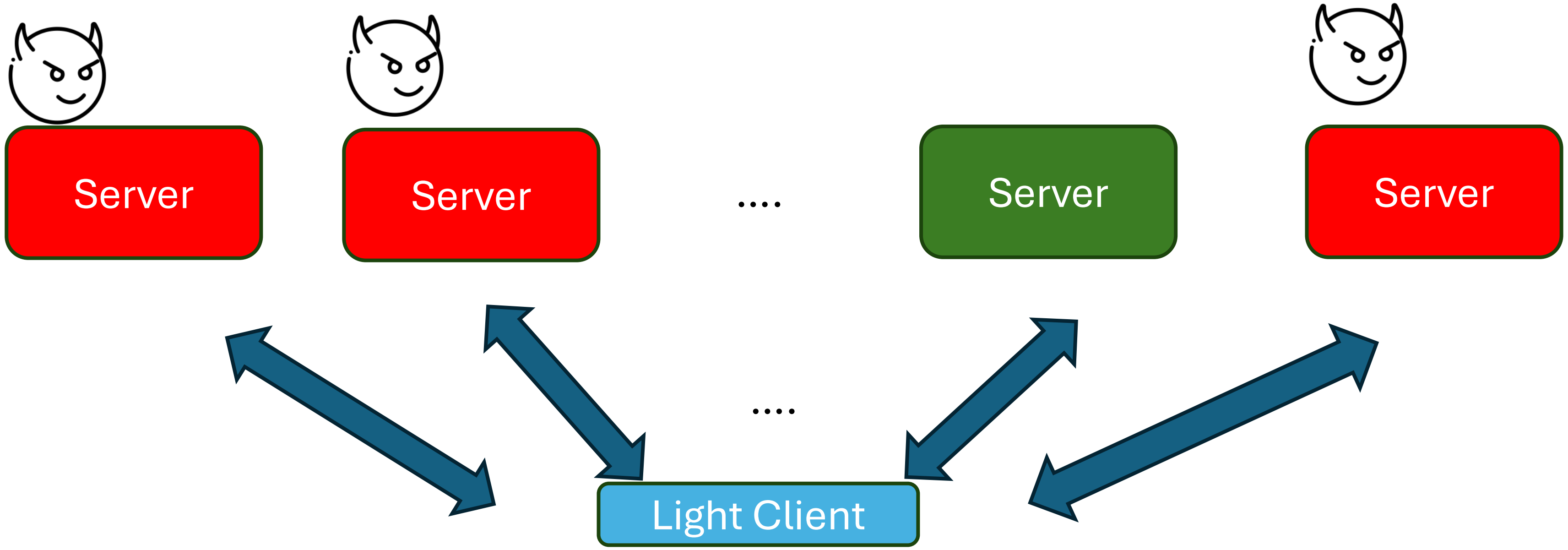
- Can the servers help them?
  - Wait… the servers can be malicious

Our Solution

- A defense system against long-range attacks
  - It helps light clients to distinguish which fork is authentic
- Advantages:
  - Plug-and-play: No soft nor hard-forks needed
  - Reasonable Assumptions: our defense works as long as one server is honest
  - Light-client friendly: Clients only need to verify succinct proofs

System Setting and Threat Model

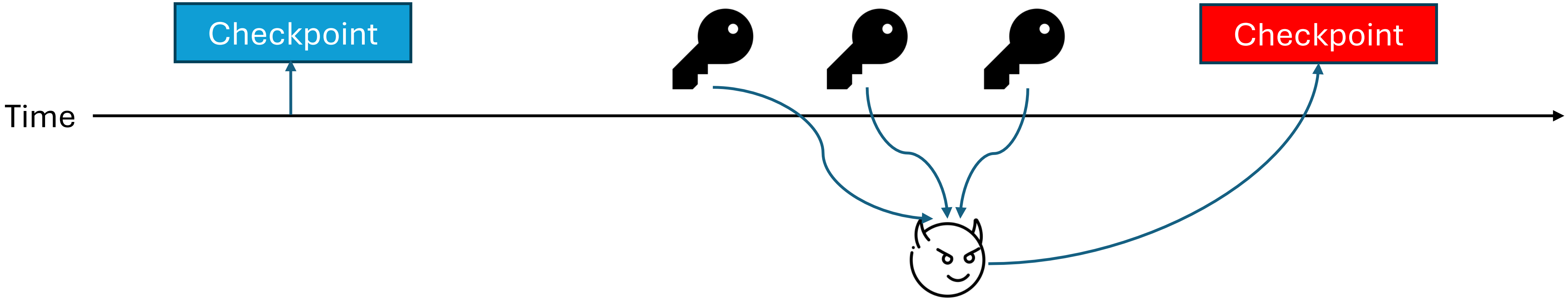
- For simplicity, we assume there are only two servers
  - One is honest, and the other is malicious
- The attacker can only corrupt keys of past (but not current) validators



without any knowledge of blockchain's history

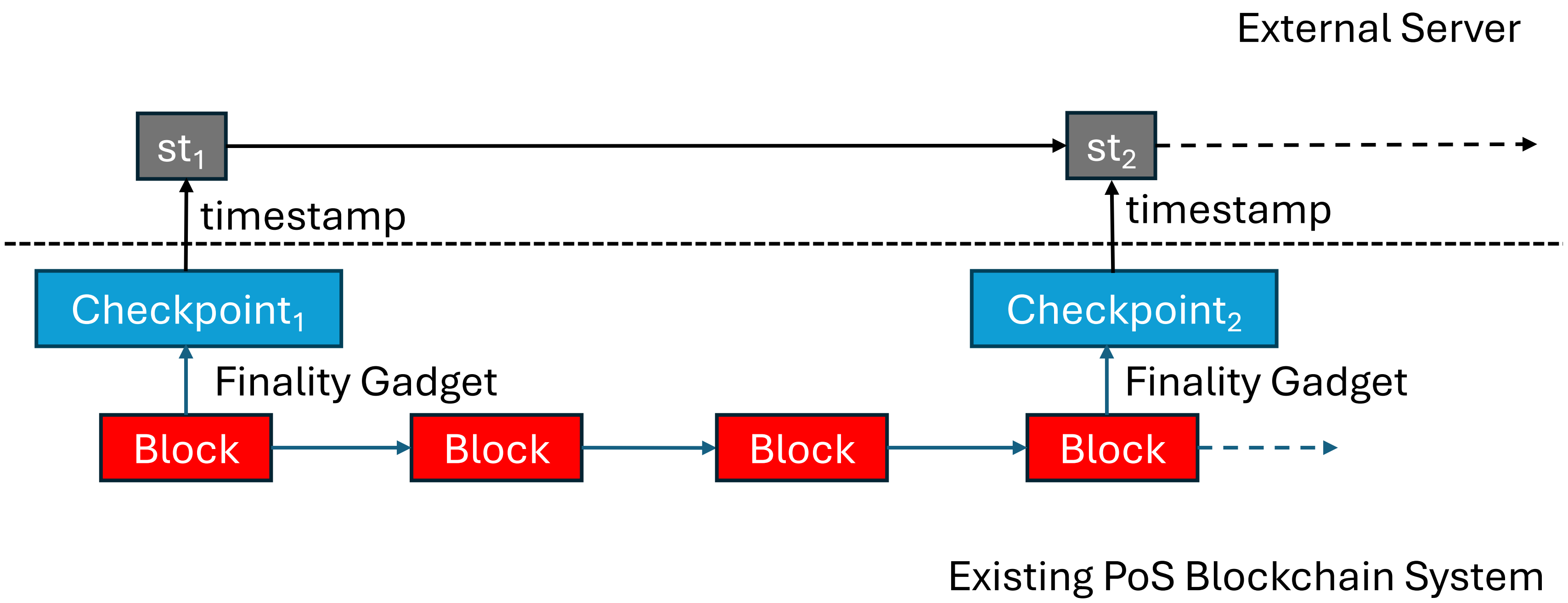
Timestamp

- The checkpoints produced first are the genuine ones
- The servers timestamps the checkpoints and their finality proofs
- How to timestamp? Verifiable Delay Functions (VDF)

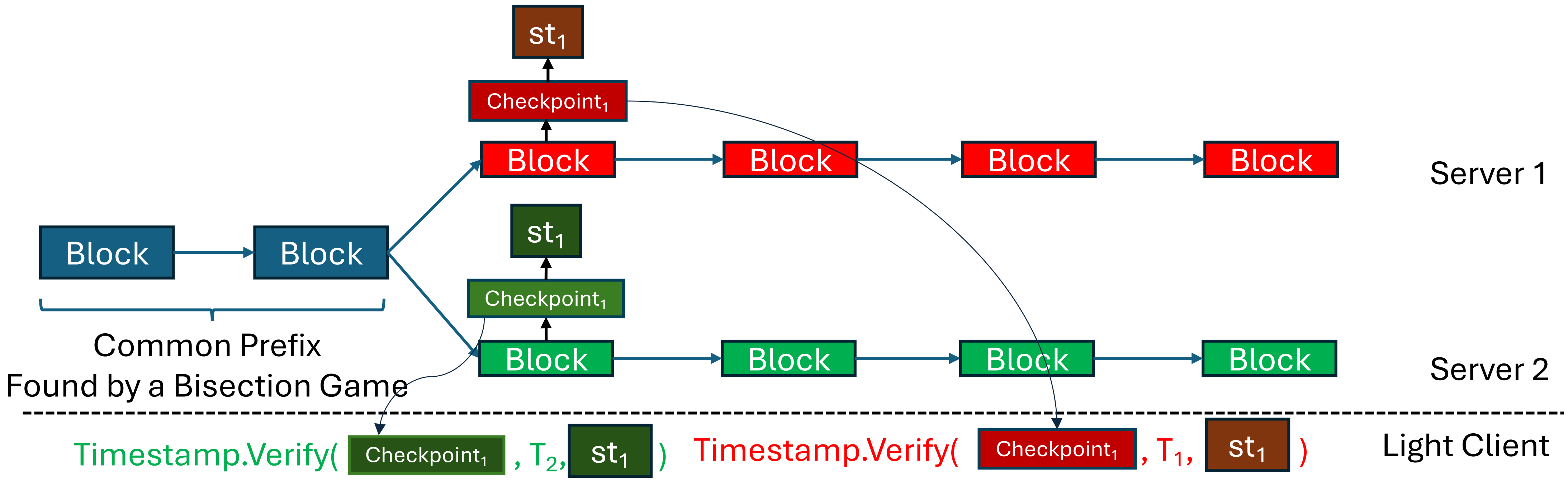


Server's Workflow

- No change to the blockchain's consensus protocol



Client's Workflow



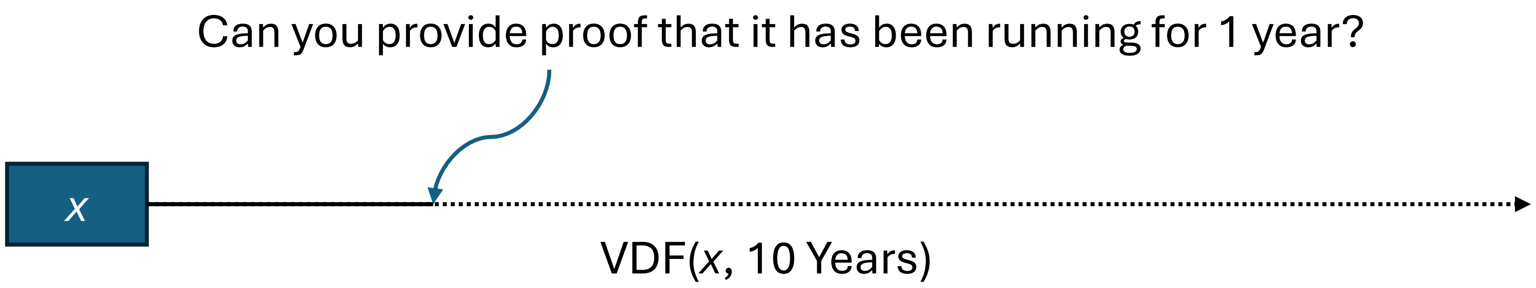
- Accept the block with an earlier timestamp

Background: Verifiable Delay Function (VDF)

- Informal Definition:
  - VDF(x, *t*) can only be computed with *t* unit of time
  - It can be succinctly proven
- It usually is based on repeated squaring assumption
  - x^{2^t} mod N is most efficiently computed by sequential squaring
    - The group order is unknown, e.g., N is an RSA modulus
  - Some schemes relies on other assumptions, e.g., lattice-based.

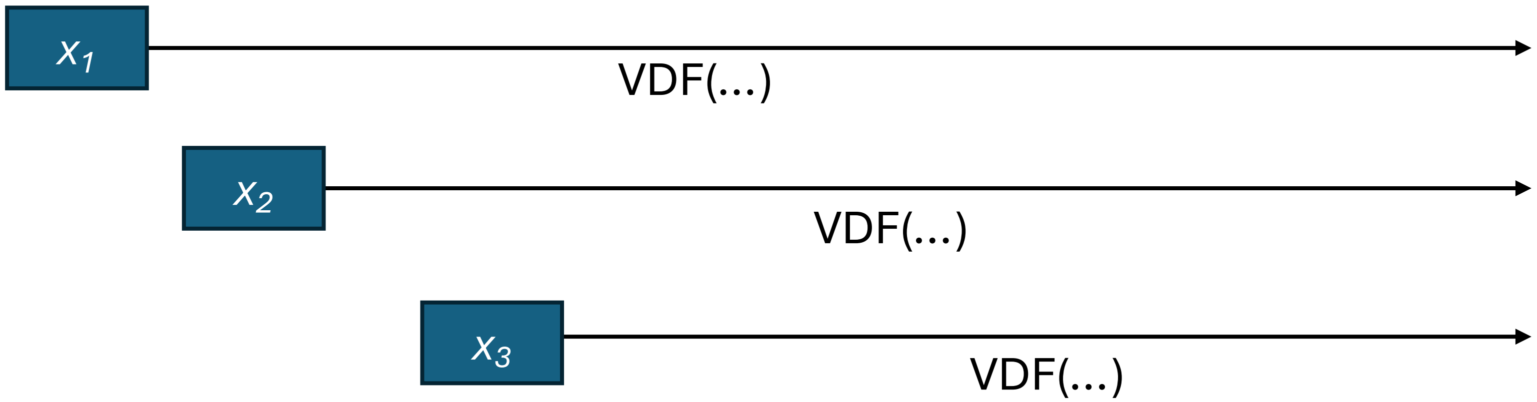
1st Issue about VDF

- VDF is not ever-going



2nd Issue about VDF

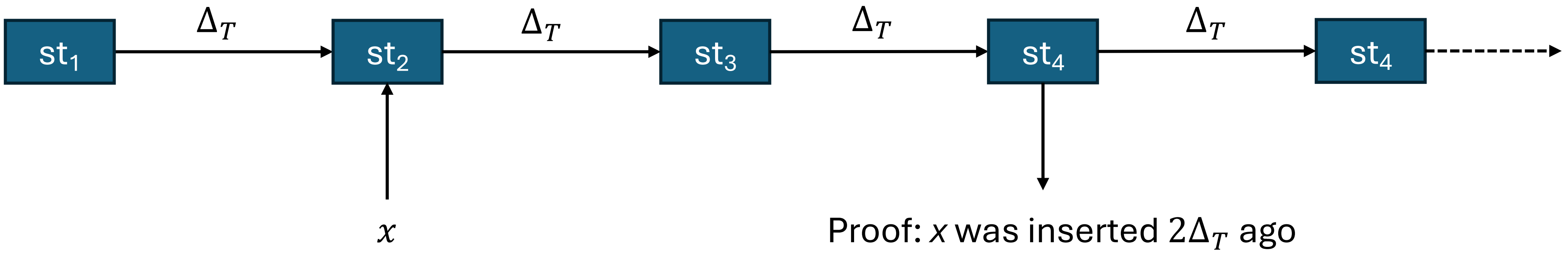
- The input *x* has to be committed in the beginning



- Ethereum has >250k checkpoints...

Insertable Proof-of-Sequential-Work (InPoSW)

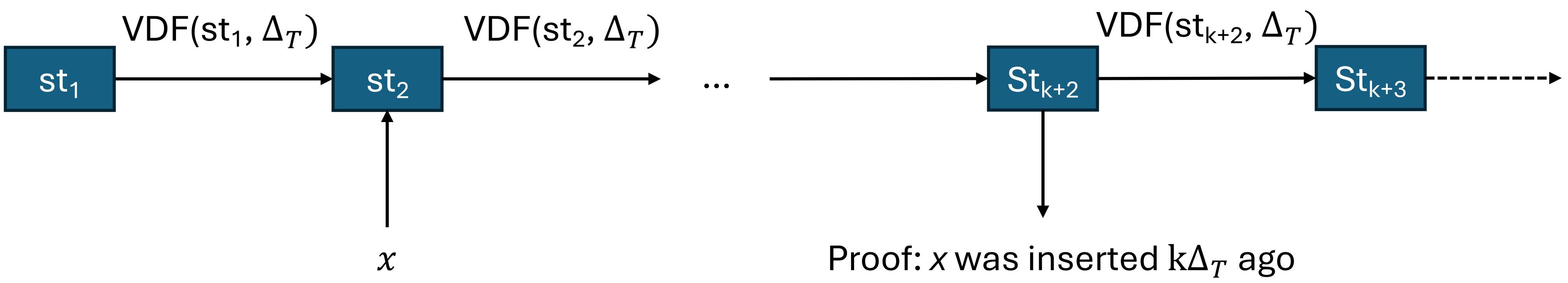
- At any time point (st_i), the prover can
  - Op 1: Insert data for timestamping
  - Op 2: Prove some data was insert $k \cdot \Delta_T$ ago



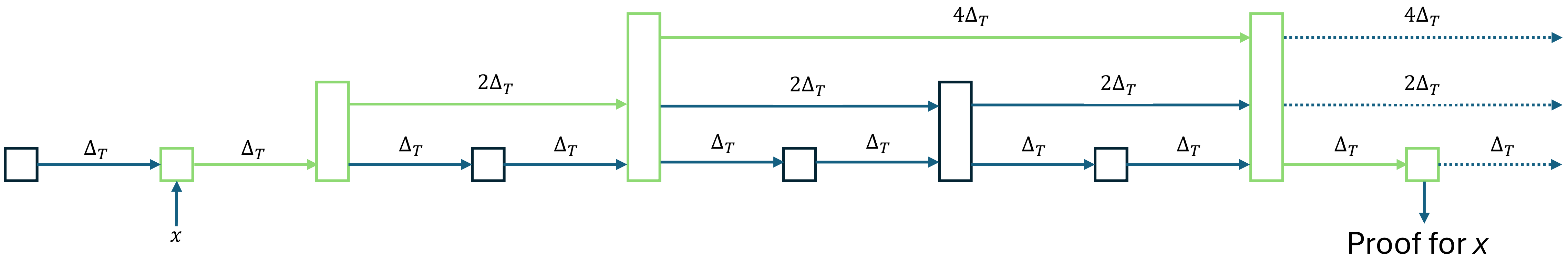
- Remark: Compared to PoW, PoSW cannot be parallelized

Strawman InPoSW Scheme

- 😭 Not succinct: The verifier needs to verify k VDF proofs



Our Skiplist-Style Construction for InPoSW



- Prover Storage: $O(N)$ VDF Proofs
- 😁 Verification Cost: $O(N) \rightarrow O(\log N)$ VDF Verification

Estimation of Concrete Cost

- We set $\Delta_T \approx 3.6$ minutes
  - which translates to 2^{33} repeated squaring
- Ethereum emits a checkpoint every 6 minutes

- We use Ethereum as our reference
- After 10 years of running our system
  - The server stores ≈ 546 GB of data
    - >22x less than adopting existing solution
  - The proof size is ≈ 20 KB
    - >17000x less than adopting existing solution
- Prior Solution that can be modified for InPoSW
  - An Incremental PoSW for General Weight Distributions [EC '23]
    - Graph-Labeling PoSW Scheme

Conclusion

- Long-range attack can bring devastating outcomes to PoS blockchains
  - And existing solutions are unsatisfying
- We propose a solution that
  - has reasonable assumption (at least 1 server being honest)
  - requires no soft/hardfork
  - is light-client friendly
- We propose a construction of InPoSW
  - It allows cost-efficient timestamping on data arriving at different times
  - It could be of independent interest for other timestamp applications

Questions?