Finding Consensus Bugs in Ethereum via Multi-transaction Differential Fuzzing

Youngseok Yang[1] Taesoo Kim[2] Byung-Gon Chun[1,3]

[1]Seoul National University [2]Georgia Institute of Technology [3]FriendliAI

Nov 11th, 2020 hard-fork

Ethereum ecosystem went down

- Infrastructure: Infura(largest), ...
- Exchanges: Binance(largest),
- DApps: Metamask, Uniswap, ...

Around 30 blocks abandoned

- $8.6M worth of ETH

Considered as Ethereum's greatest challenge since the 2016 DAO hack

Nasdaq

"Unannounced" Ethereum Hard Fork Proves Not All Blockchain Networks Are Built The Same

Earlier today, a change to the underlying Ethereum code made by developers some time ago resulted in an unannounced hard fork, effectively splitting the ...

2020. 11. 11.

Yahoo Finance

Ethereum's 'Unannounced Hard Fork' Was Trying to Prevent the Very Disruption It Caused

The split resulted from a code change that was surreptitiously inserted into a previous Geth update; some Ethereum node operators ignored the update, which ...

2020. 11. 11.

Cointelegraph

Binance briefly pauses Ethereum withdrawals as network suffers 'minor hard-fork'

The Ethereum (ETH) network has suffered what looks like a hard fork today as reports emerged of outages and irregularities on infrastructure providers Infura ...

2020. 11. 11.

Nov 11th, 2020 hard-fork

July, 2020

We found and reported two consensus bugs in the most popular Geth client

July~Nov, 2020

Bugs silently fixed in new Geth client releases, but not all users upgraded

Nov 11th, 2020

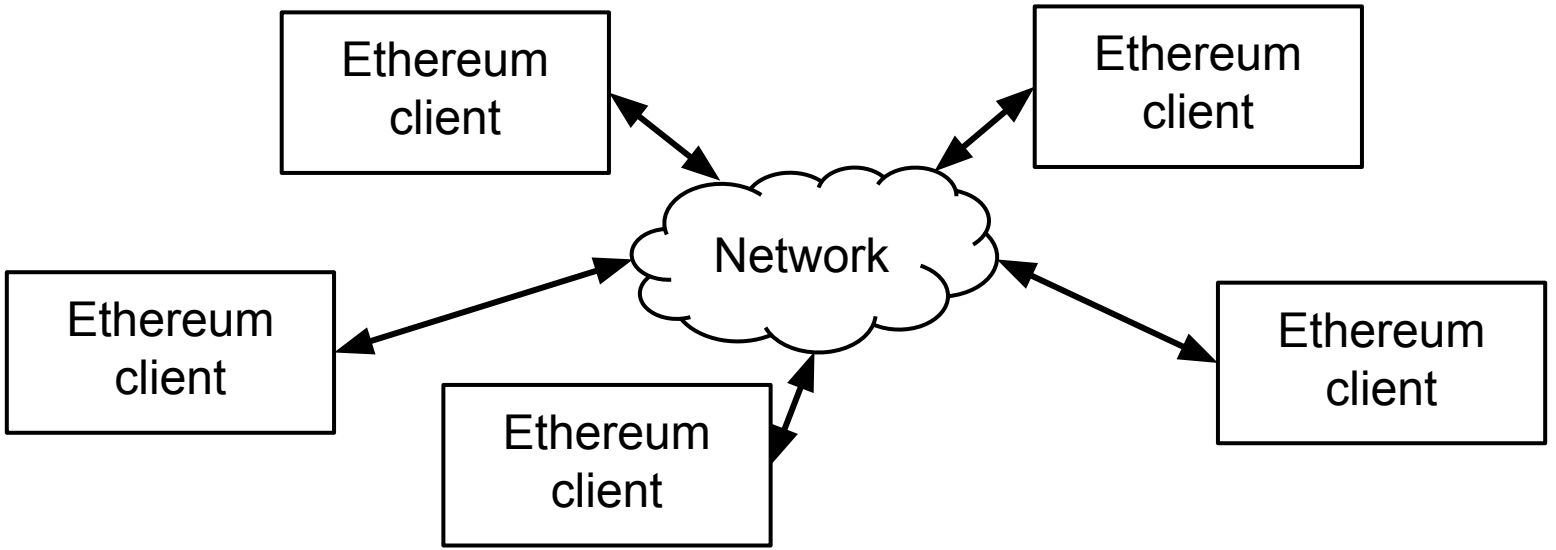
An Ethereum transaction triggered one of the bugs we reported

Our paper describes this

Background

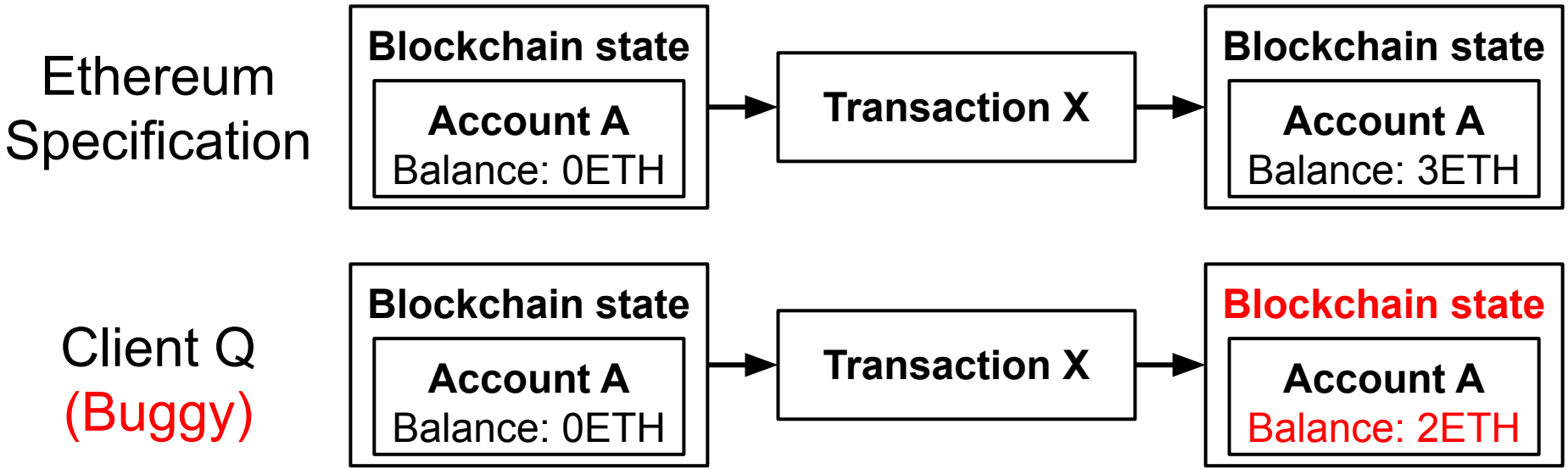
Ethereum

Consensus is reached by decentralized clients that implement the Ethereum Virtual Machine (EVM) specification



Consensus bugs

Implementation bugs in Ethereum clients
that lead to incorrect blockchain states



Consensus bugs

Consensus bugs are extremely rare

- Since Ethereum launched in July 2014, only 13 consensus bugs have been found in the most popular Geth and OpenEthereum clients
- Only 6 of them would have been exploitable on the live mainnet

Preventing consensus bugs is a top priority

- Consensus bugs have high impacts
  - Network split: Reliability issues (e.g., delaying transactions)
  - Theft: Security-critical issues (e.g., stealing ETH)
- Heavy investments in auditing, testing, and fuzzing Ethereum clients

Existing Differential Fuzzers

Existing fuzzers

Differential fuzzers have found most of the consensus bugs in Ethereum

Overview:
Step 1. Generate an input blockchain state and a single transaction
Step 2. Initialize multiple Ethereum clients with the blockchain state
Step 3. Invoke the clients with the transaction
Step 4. Compare the output blockchain states
Step 4. If the outputs are the same, GOTO Step 1.
If the outputs are not the same, a consensus bug is found

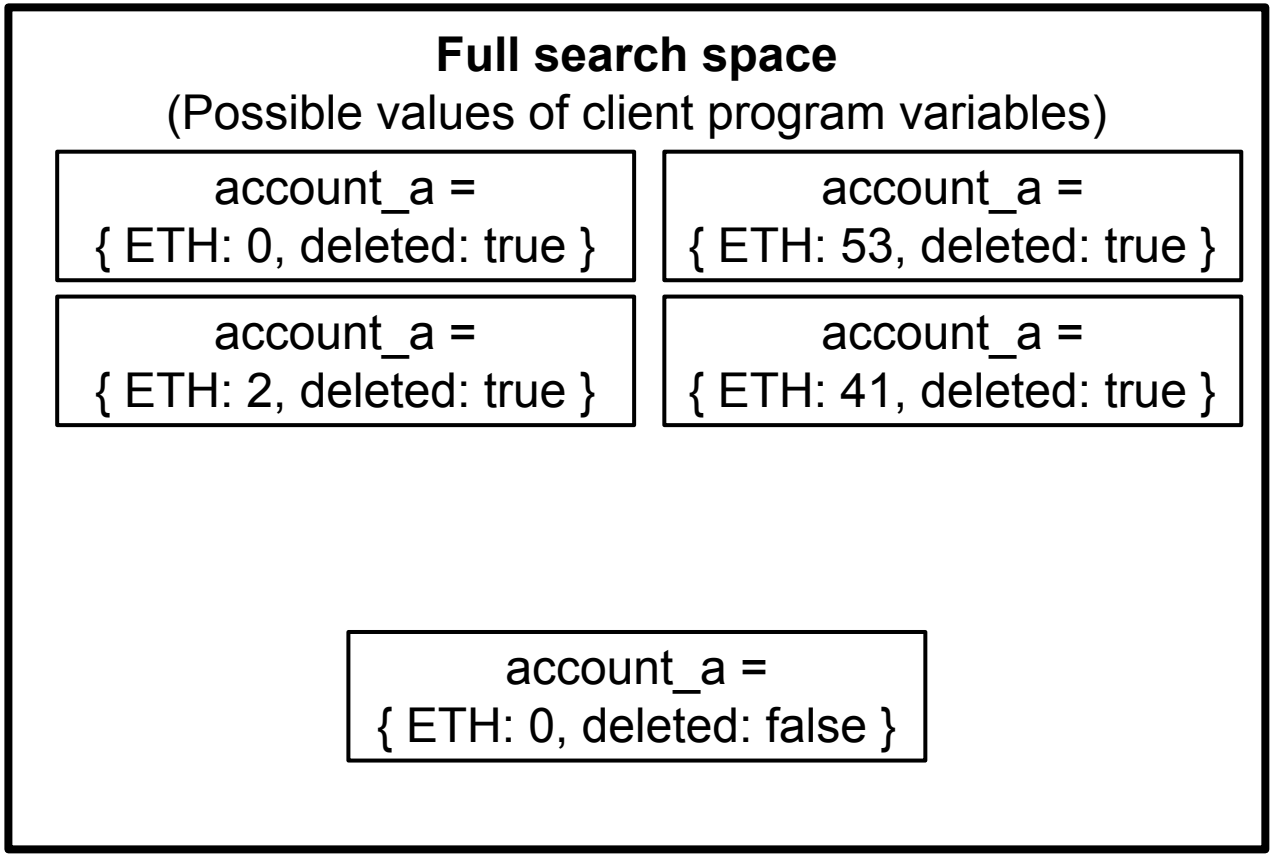
Existing fuzzers

Existing differential fuzzers test
only **a single transaction** in each iteration

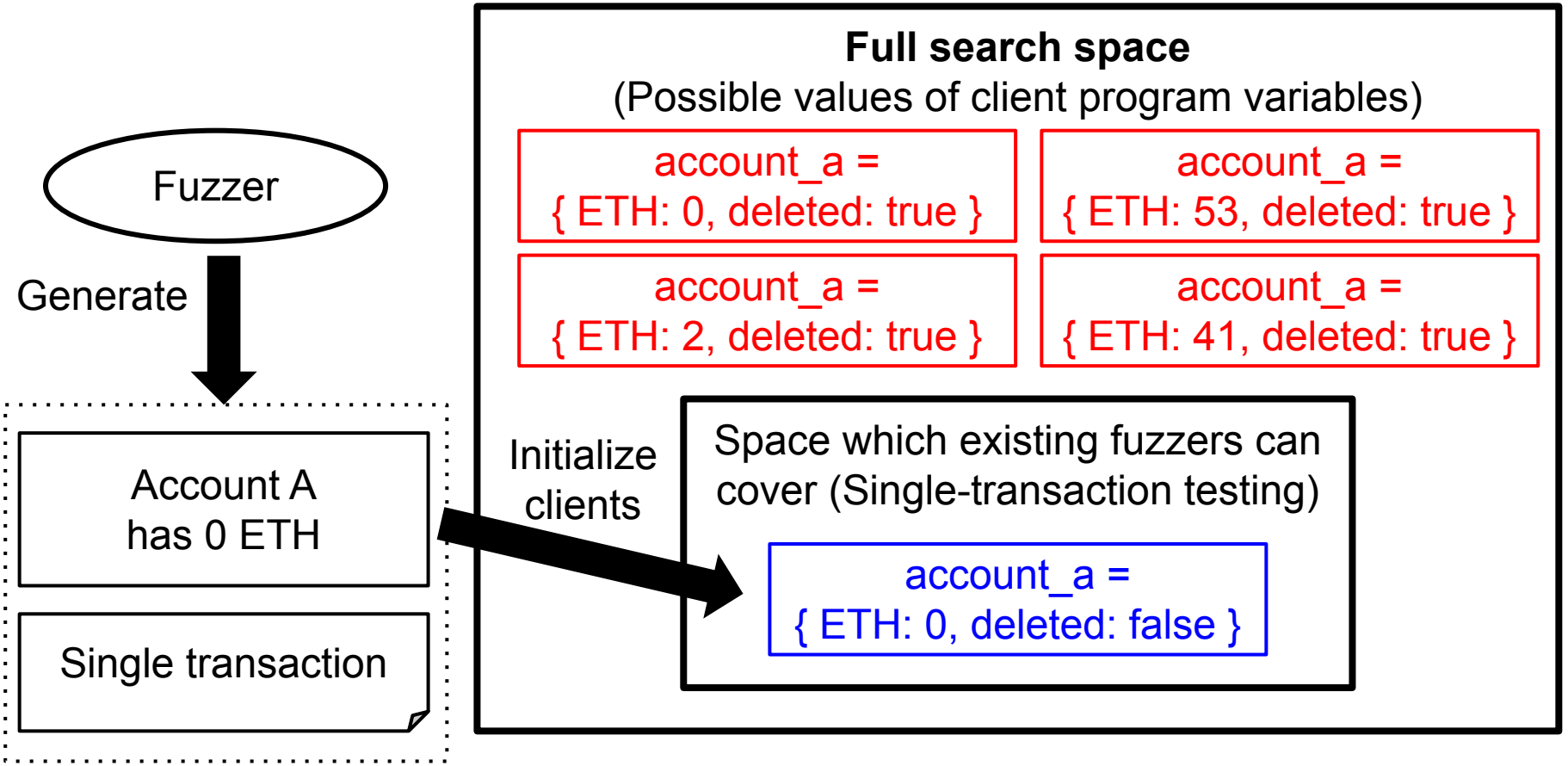
⇒ Cannot cover the “full search space”

Existing fuzzers

The blockchain state “A has 0 ETH” can be represented in multiple ways



Existing fuzzers



Our Key Idea

Key idea

Goal: Enable the fuzzer to cover the full search space

Test **a sequence of multiple transactions** ⇒
Test various pre-transaction **client program states**

Case Study

Bugs we found

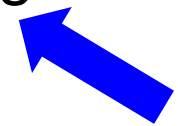
Shallow copy bug

Transfer-after-destruct bug

Bugs we found

Shallow copy bug

Transfer-after-destruct bug



In this talk

Transfer-after-destruct bug

Root cause
Geth “carries over” the balance of a deleted account object to the newly created account object under the same address

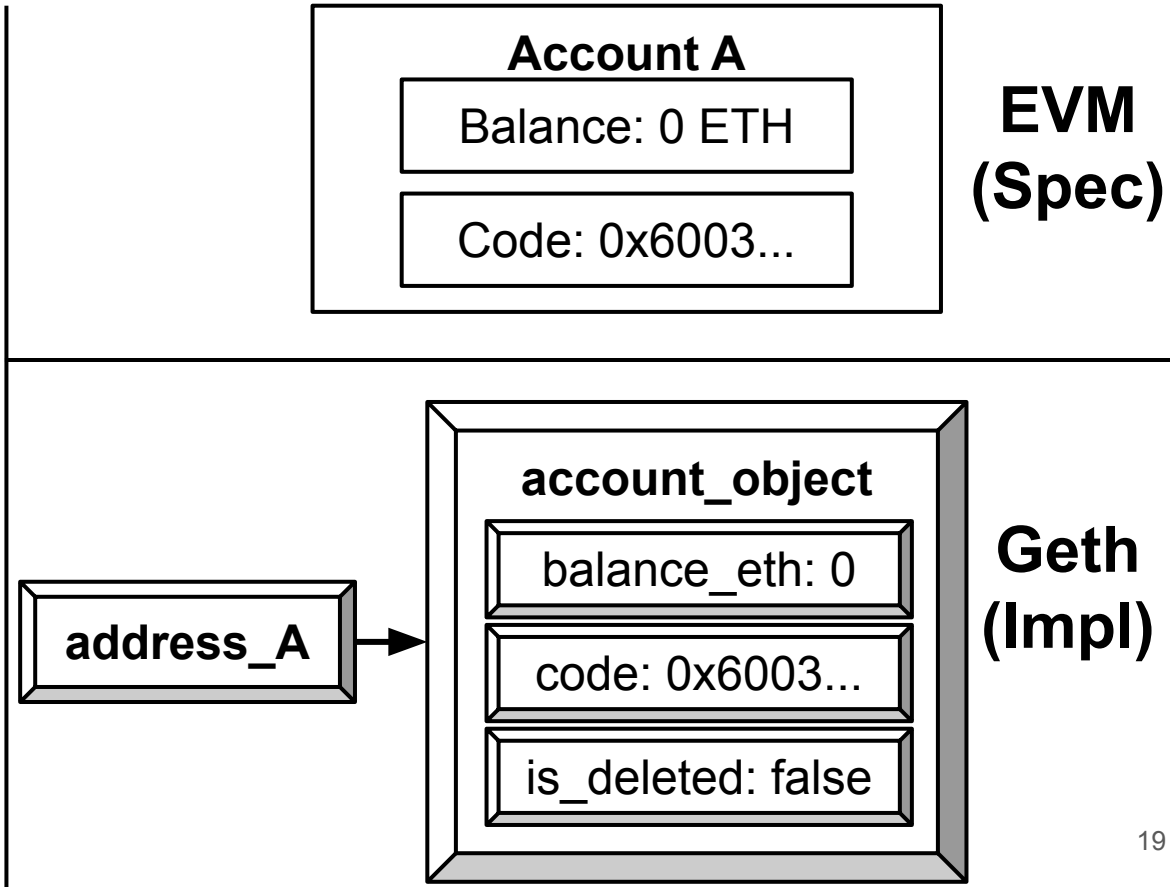
At least 2 transactions are required to trigger the bug
- Transaction 1: Destroys account A, and sends 2 ETH to A
- Transaction 2: Sends 1 ETH to A

EVM Specification says “A has 1 ETH”
Buggy Geth says “A has 3 ETH”

Transfer-after-destruct bug

```
// Contract (Address A)
1: If VALUE == 0
2:   SELFDESTRUCT
3: ELSE
4:   STOP
```

```
// Contract (Address: B)
1: CALL A with 0 ETH
2: CALL A with 2 ETH
```

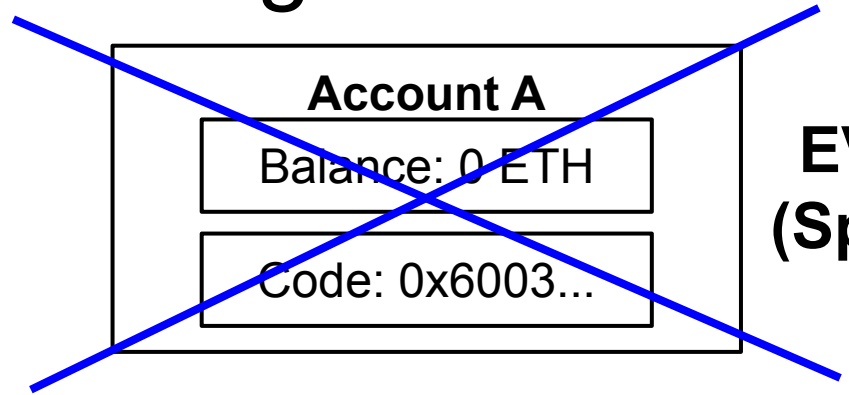


Transfer-after-destruct bug

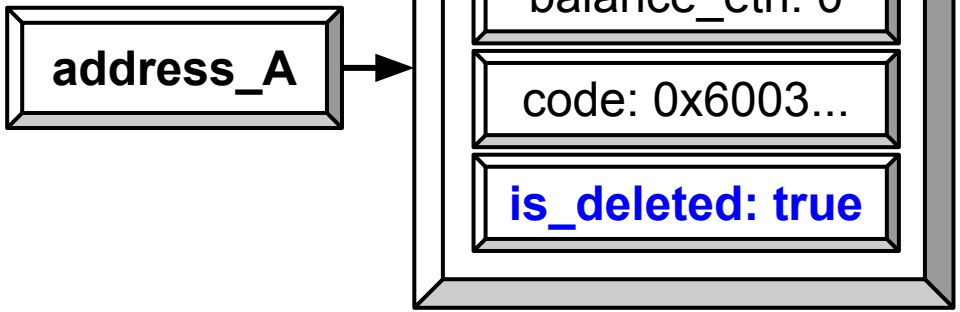
```
// Contract (Address A)
1: If VALUE == 0
2:   SELFDESTRUCT
3: ELSE
4:   STOP
```

```
// Contract (Address: B)
1: CALL A with 0 ETH
2: CALL A with 2 ETH
```

Transaction 1:
Call B with 5 ETH

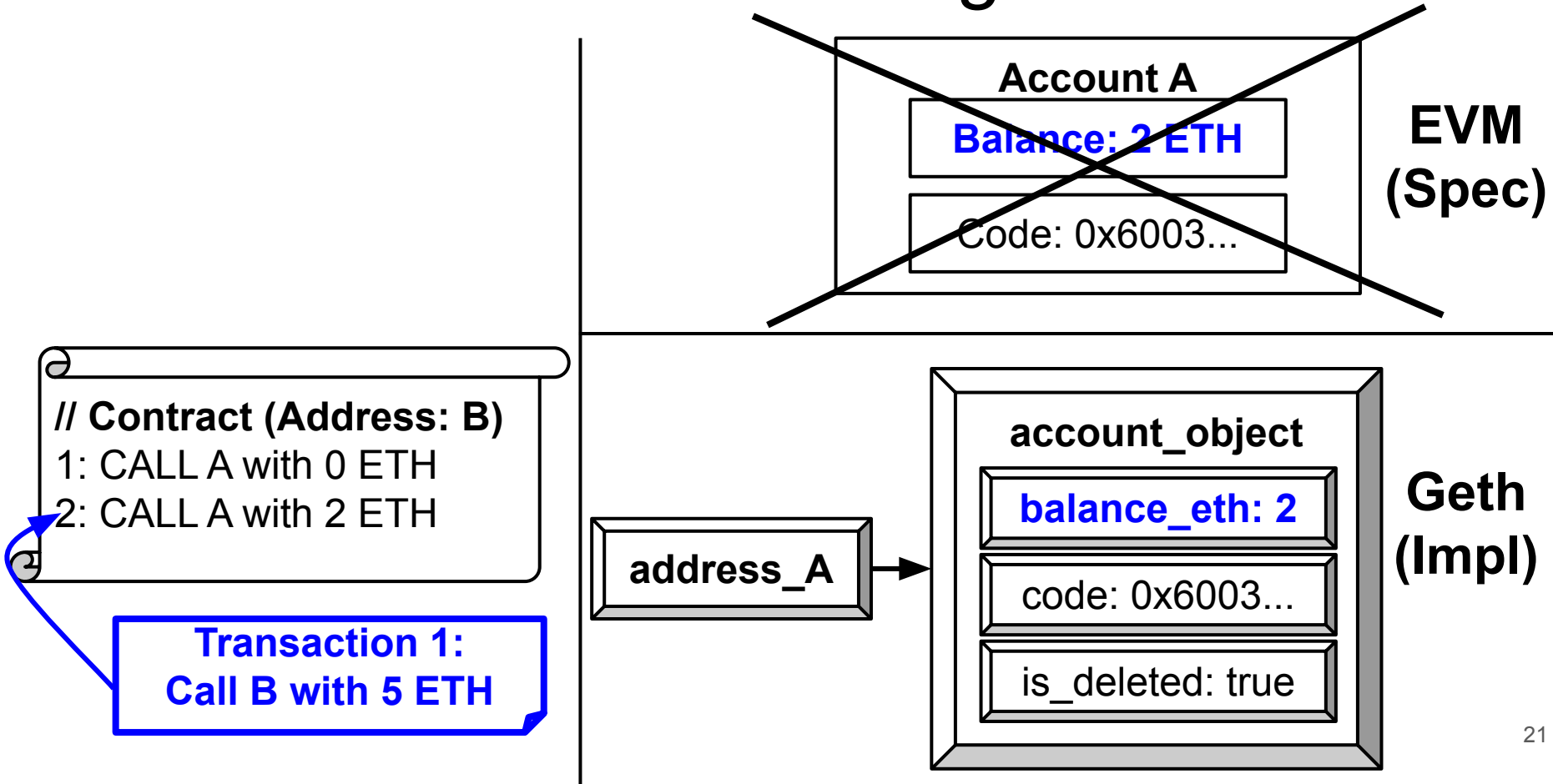


EVM (Spec)

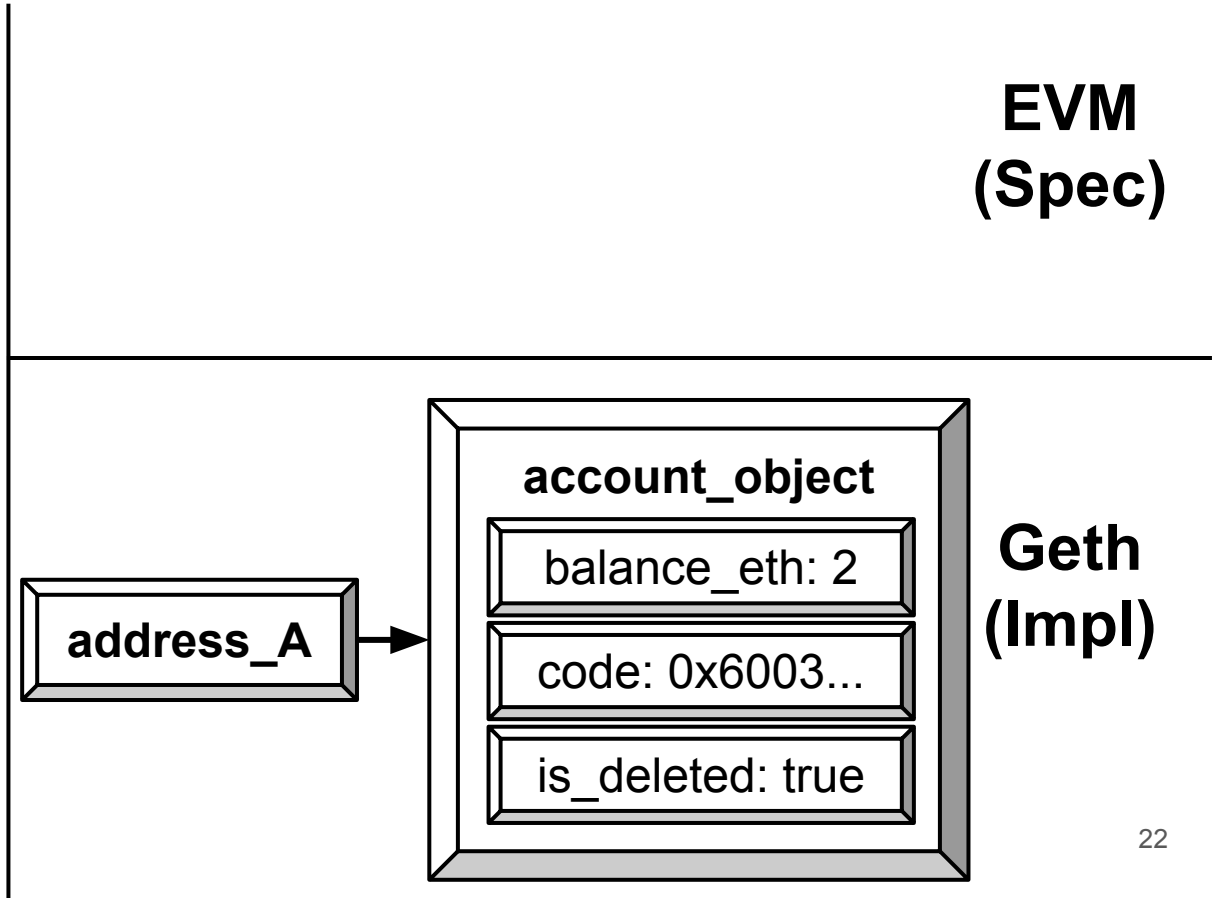


Geth (Impl)

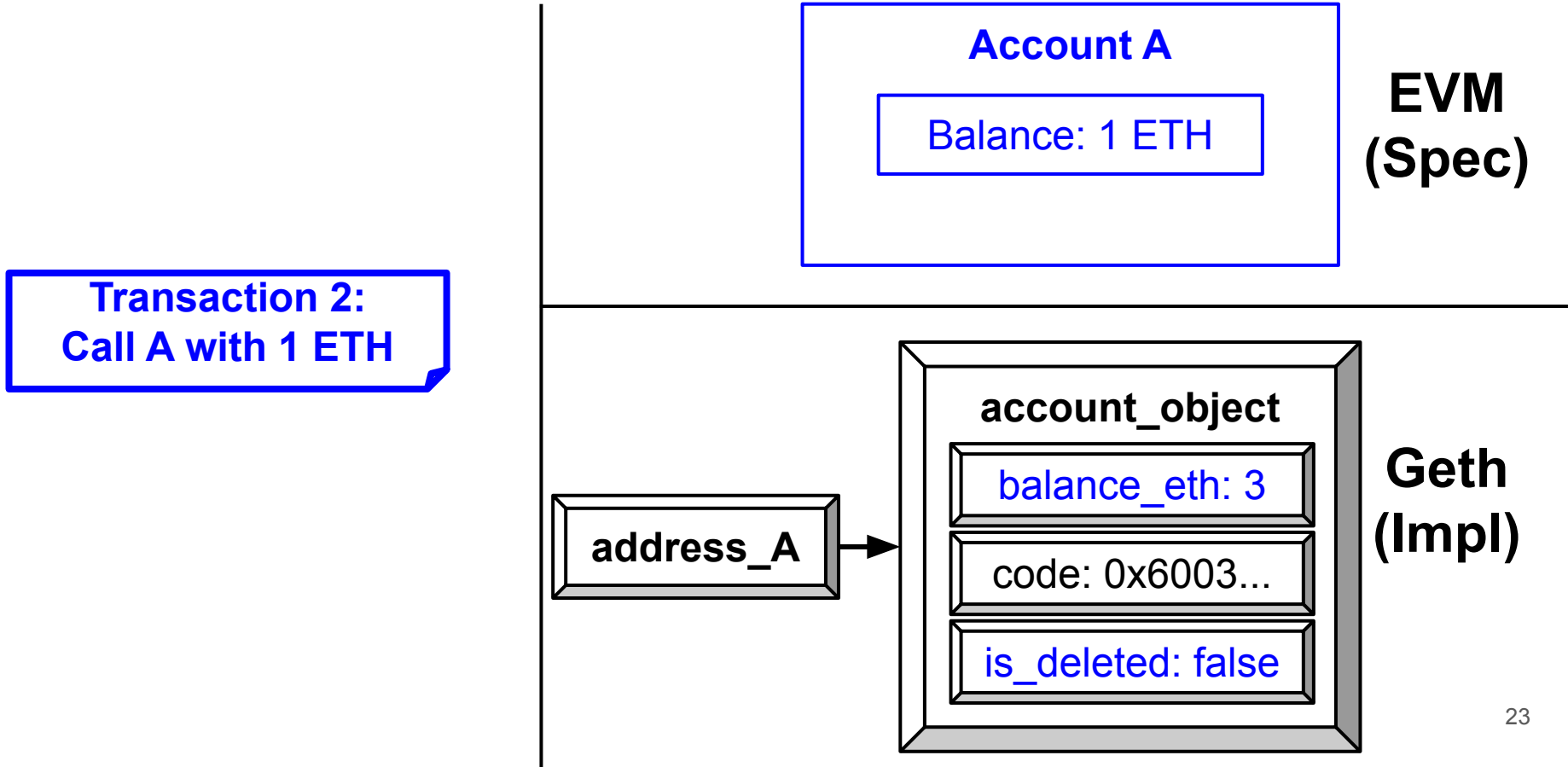
Transfer-after-destruct bug



Transfer-after-destruct bug



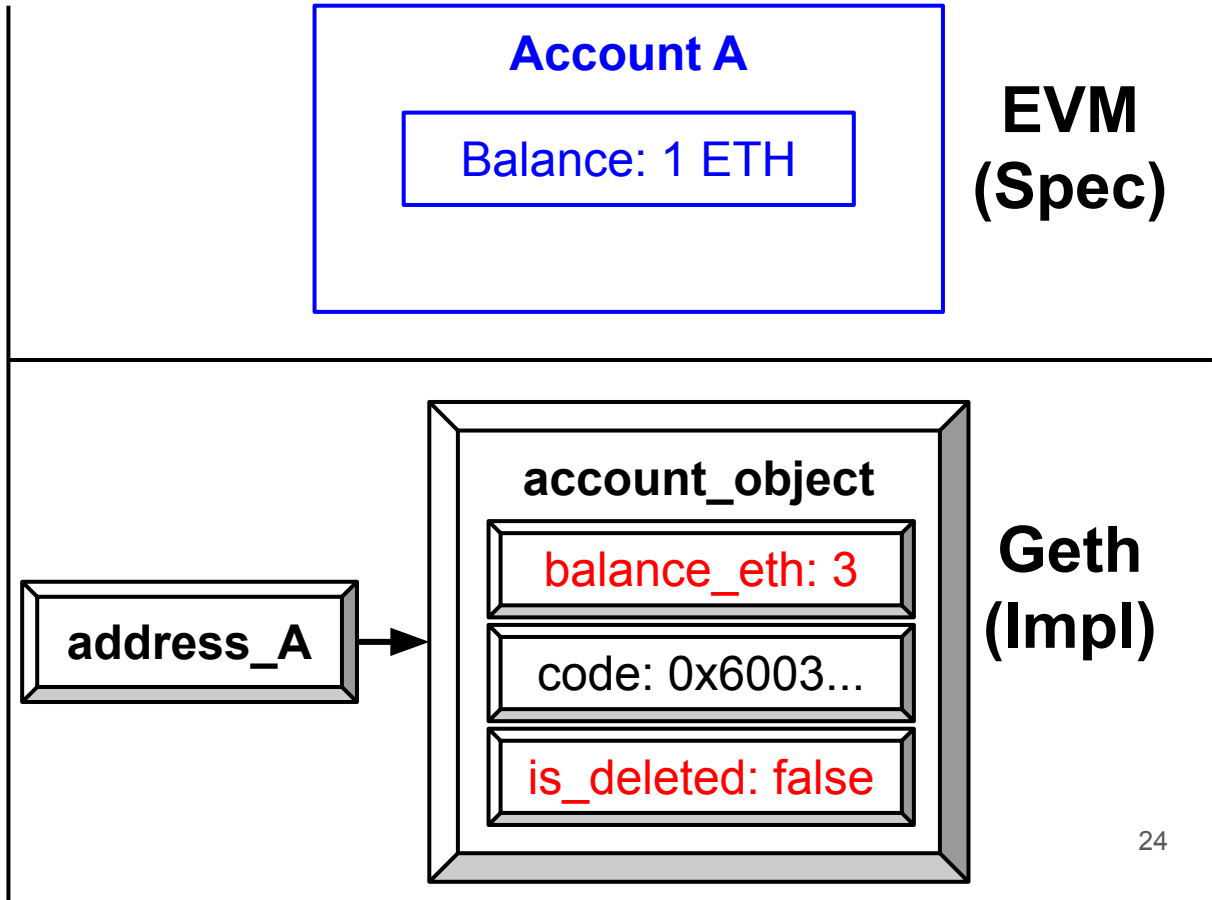
Transfer-after-destruct bug



Transfer-after-destruct bug

Spec says “1 ETH”

Geth says “3 ETH”
(Consensus bug!)



Our goal

Design a system that automatically generates and tests a sequence of **multiple transactions**

Fluffy Design

Design challenges

Challenge #1
How do we test multiple transactions efficiently?

Challenge #2:
How do we leverage intra-transaction dependencies?

Challenge #3
How do we generate high-quality multi-transaction test cases?

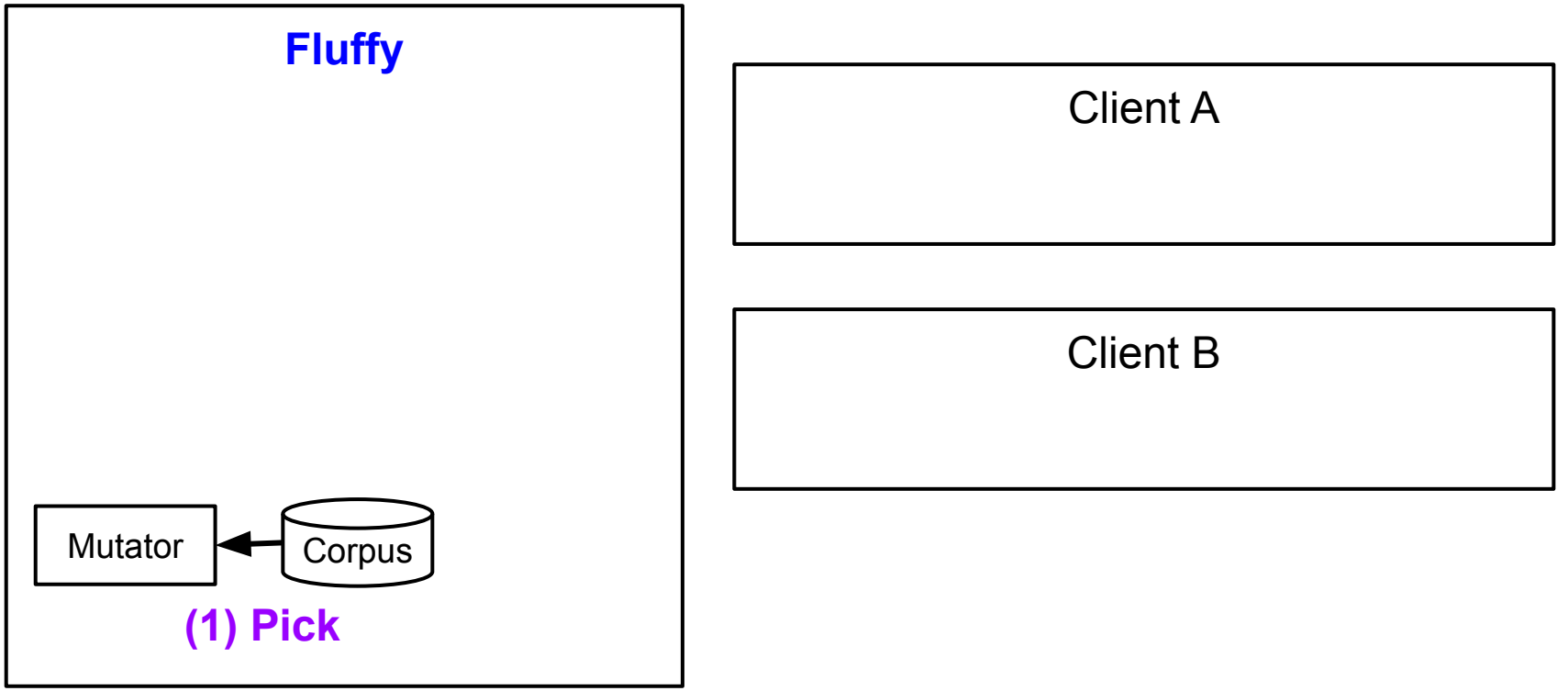
Fluffy (Our fuzzer)

Solution #1
Modifies existing clients to enable an efficient execution model

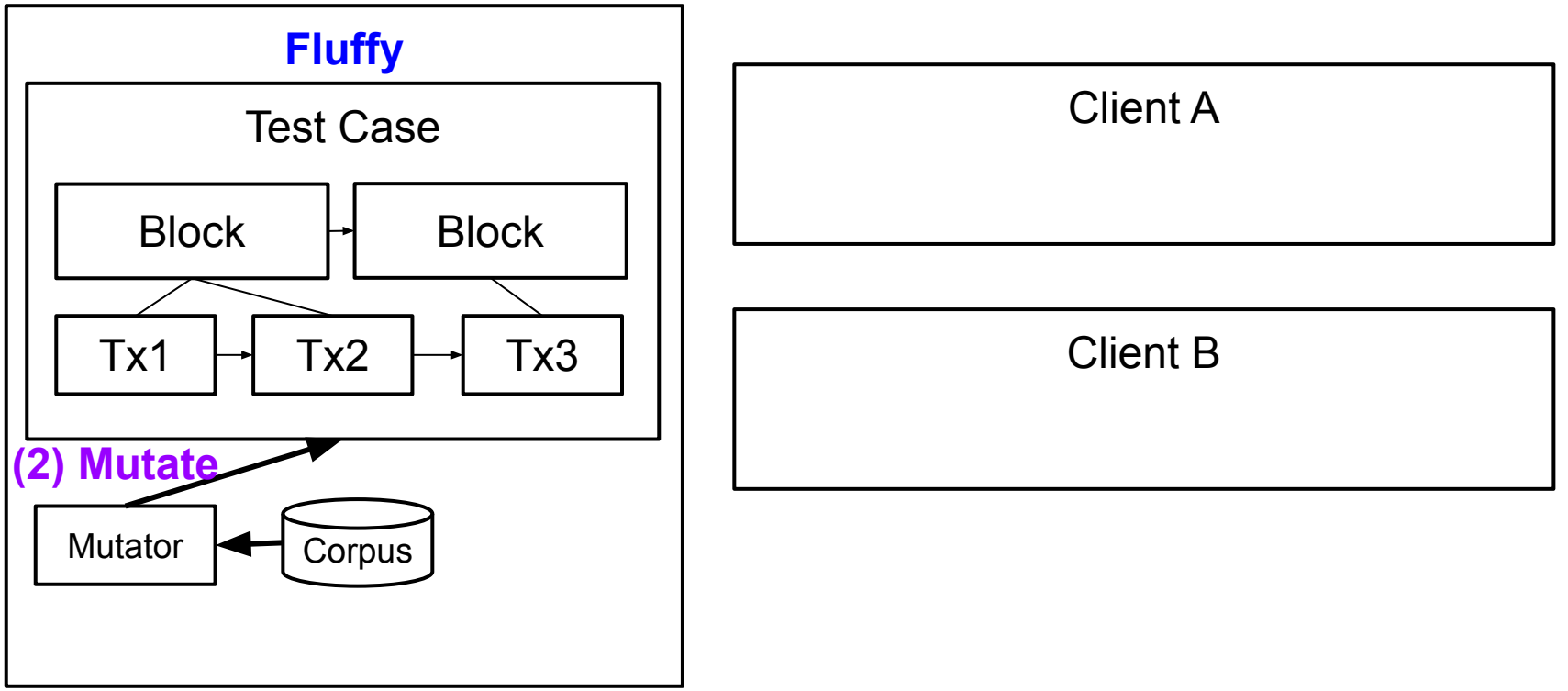
Solution #2
Test case design that encodes intra-transaction dependencies

Solution #3
Context, bytecode, and parameter mutation strategies that reduce erroneous test cases

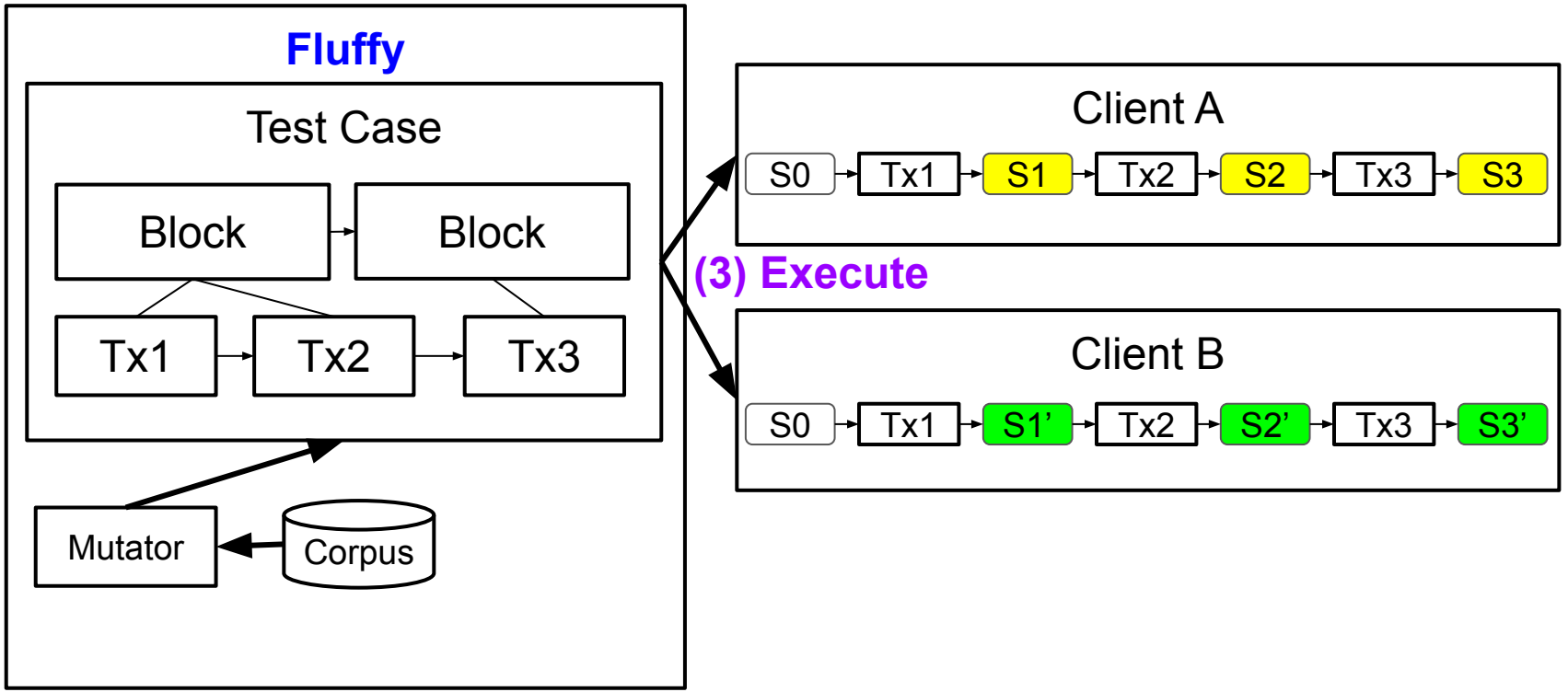
Fluffy overview



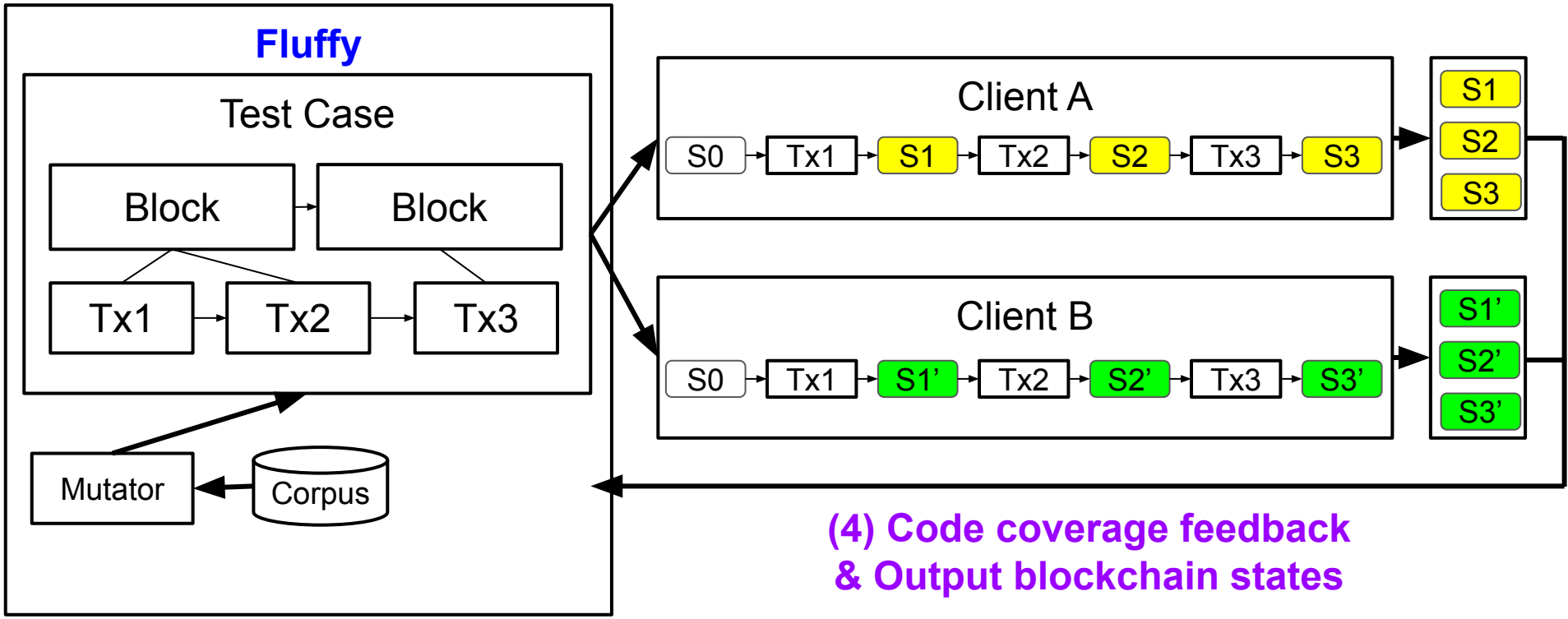
Fluffy overview



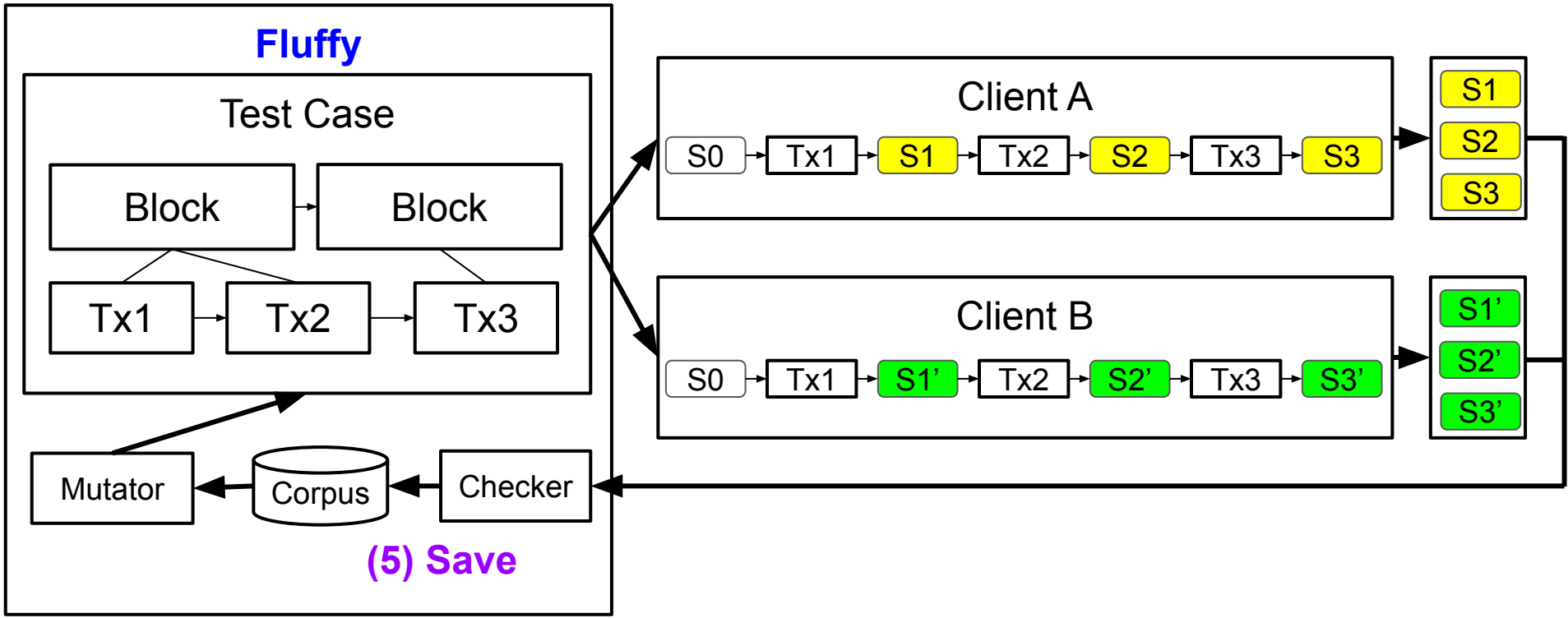
Fluffy overview



Fluffy overview



Fluffy overview



Implementation & Evaluation

Implementation

Integrations

- Built on top of libFuzzer using Rust and Go
- Supports fuzzing Geth and OpenEthereum (Used by 98% of nodes)

Fuzzing harnesses for optimized execution

- In-process fuzzing
- Skip transaction verification
- Disable JUMPDEST checking

Crash debugger for finding the root cause

Evaluation

Bug finding capability

Code coverage

Throughput

Evaluation setup

Single machine

- CPU: Intel(R) Xeon(R) CPU E5-2680 v3 (12 cores)
- Memory: 128 GB memory

Systems

- Fluffy: Our Fluffy implementation
- Fluffy-Random-Bytecode: Modified Fluffy that randomly generates bytecode
- EVMLab: A state-of-the-art fuzzer for Ethereum

Ethereum clients

- OpenEthereum v3.0.0
- Geth v1.9.14

Bug finding capability

Total 15 consensus bugs found since Ethereum launched in 2014

- Bug #1 and Bug #2: New consensus bugs found by Fluffy
- Bug #3 ~ Bug #15: Consensus bugs that were reported to be found

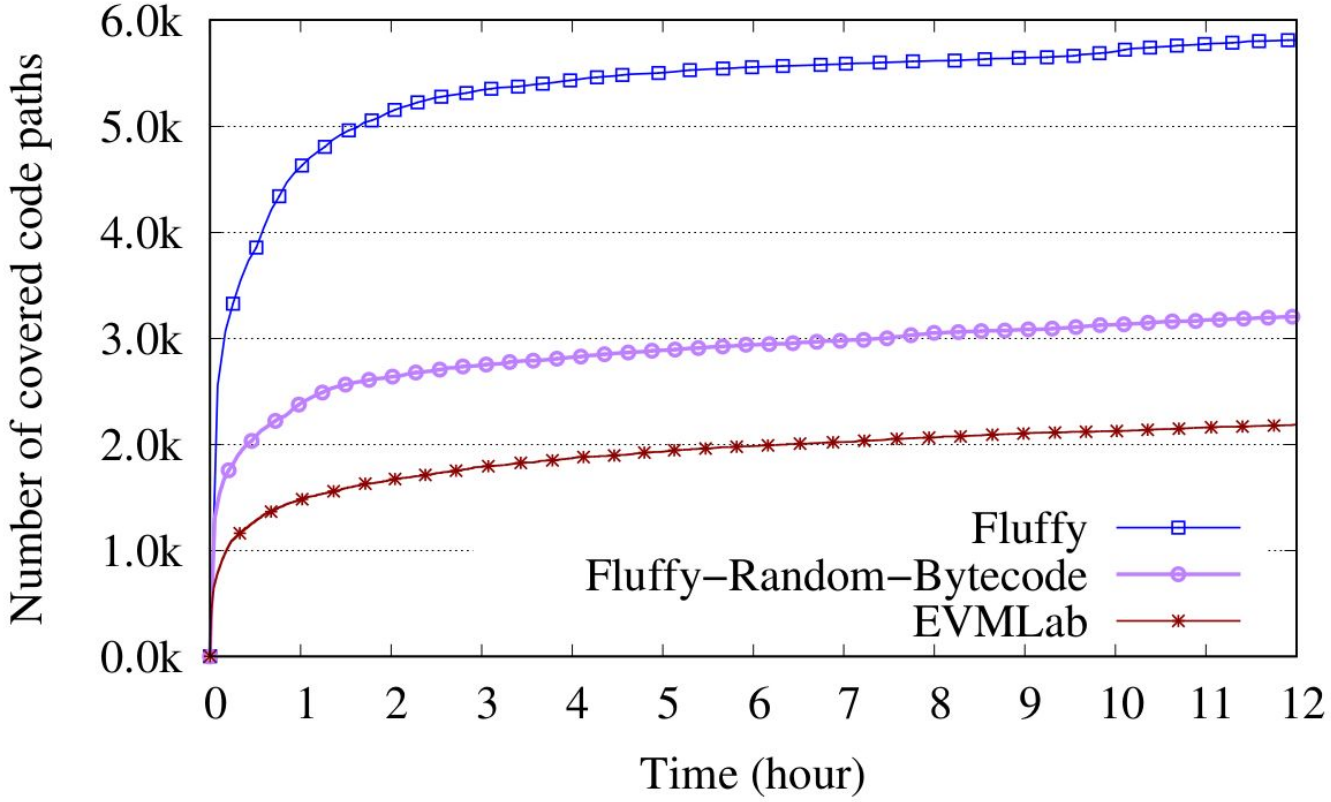
Bugs we do not experiment with

- Bug #3: Block mining, which Fluffy does not focus on
- Bug #5: Signature verification, which Fluffy does not focus on
- Bug #6: Was fixed by using a different library
- Bug #14: Details are undisclosed

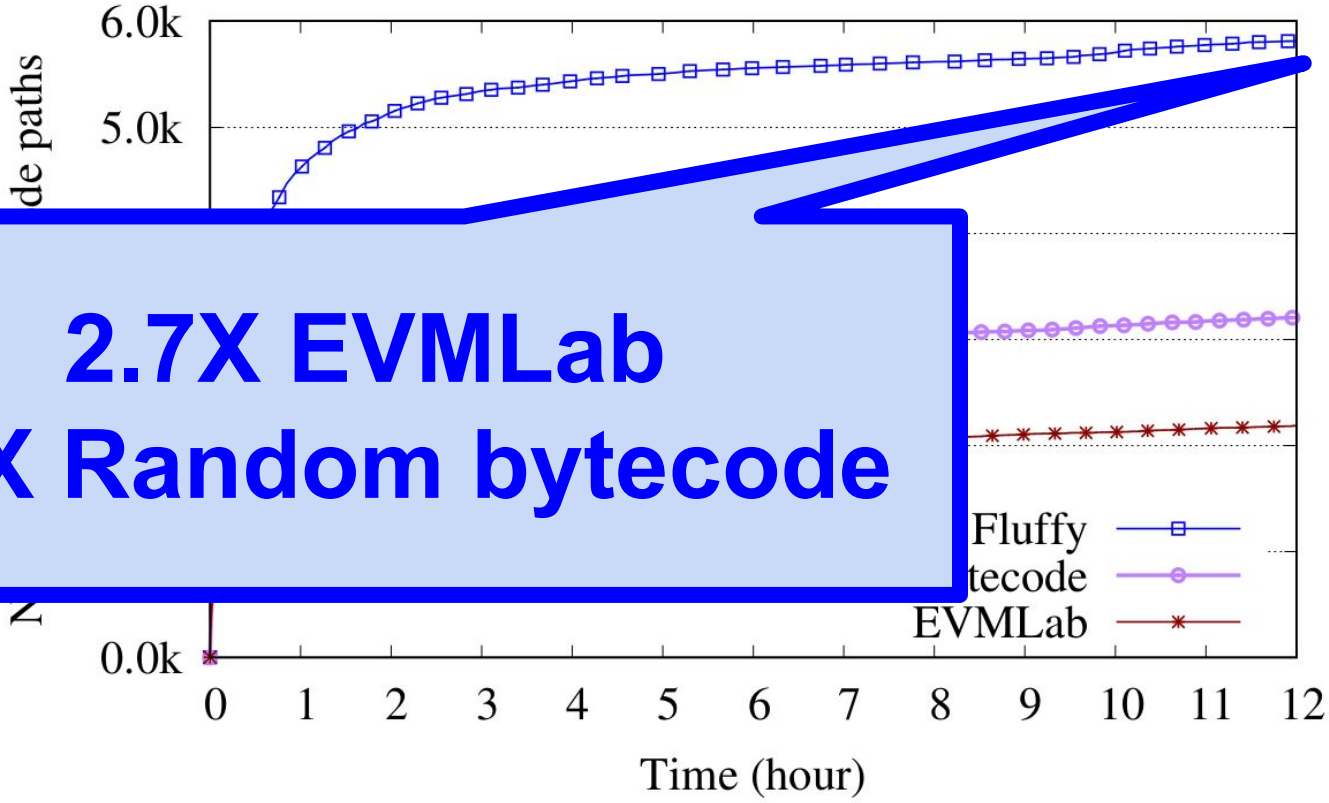
Result

- Out of 11 bugs, Fluffy finds 10 bugs within just 12 hours
- Fluffy fails to find Bug #9, which requires specific inputs that satisfy tight branch conditions to trigger (originally found with manual auditing)

Code coverage (Higher is better)

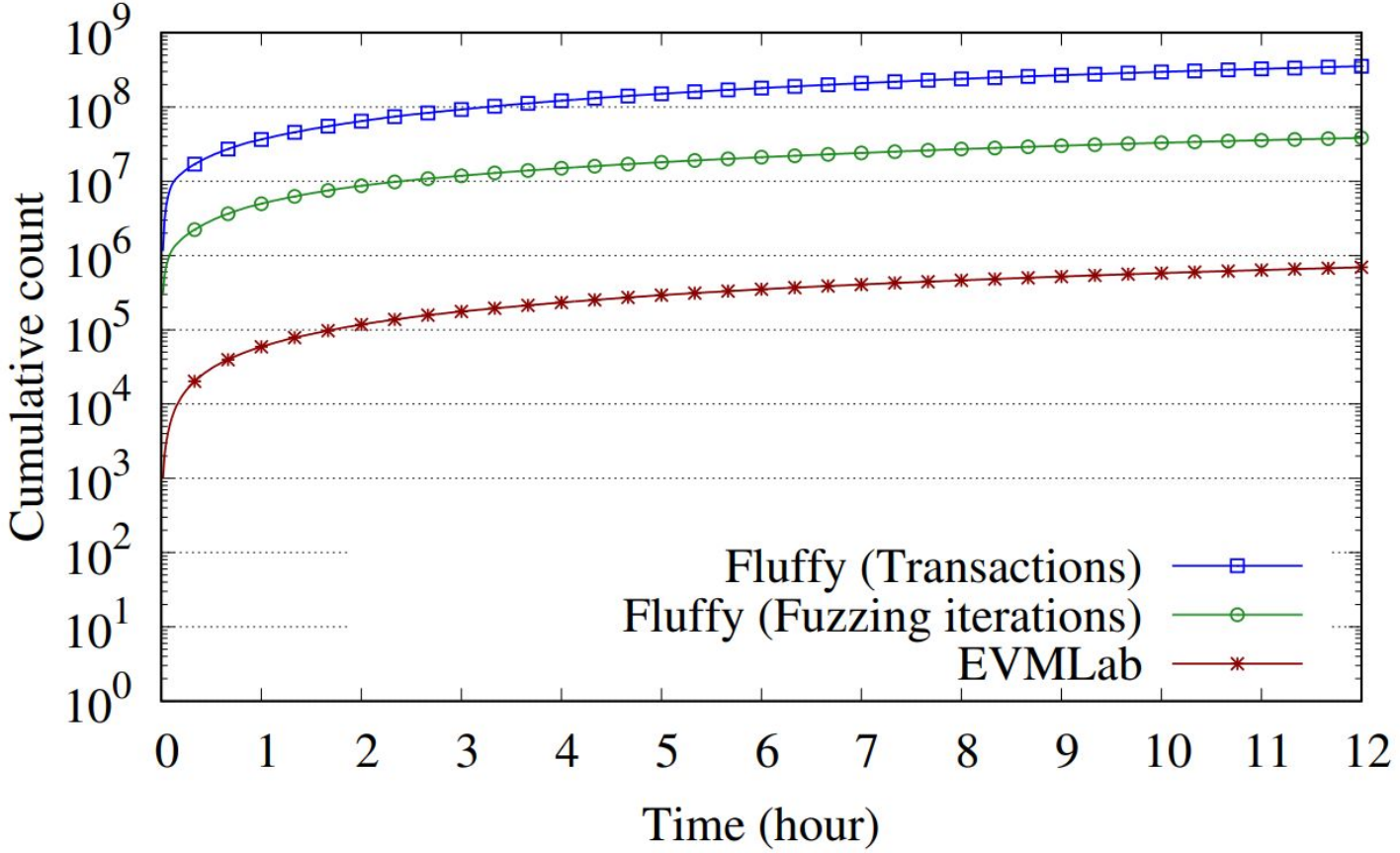


Code coverage (Higher is better)

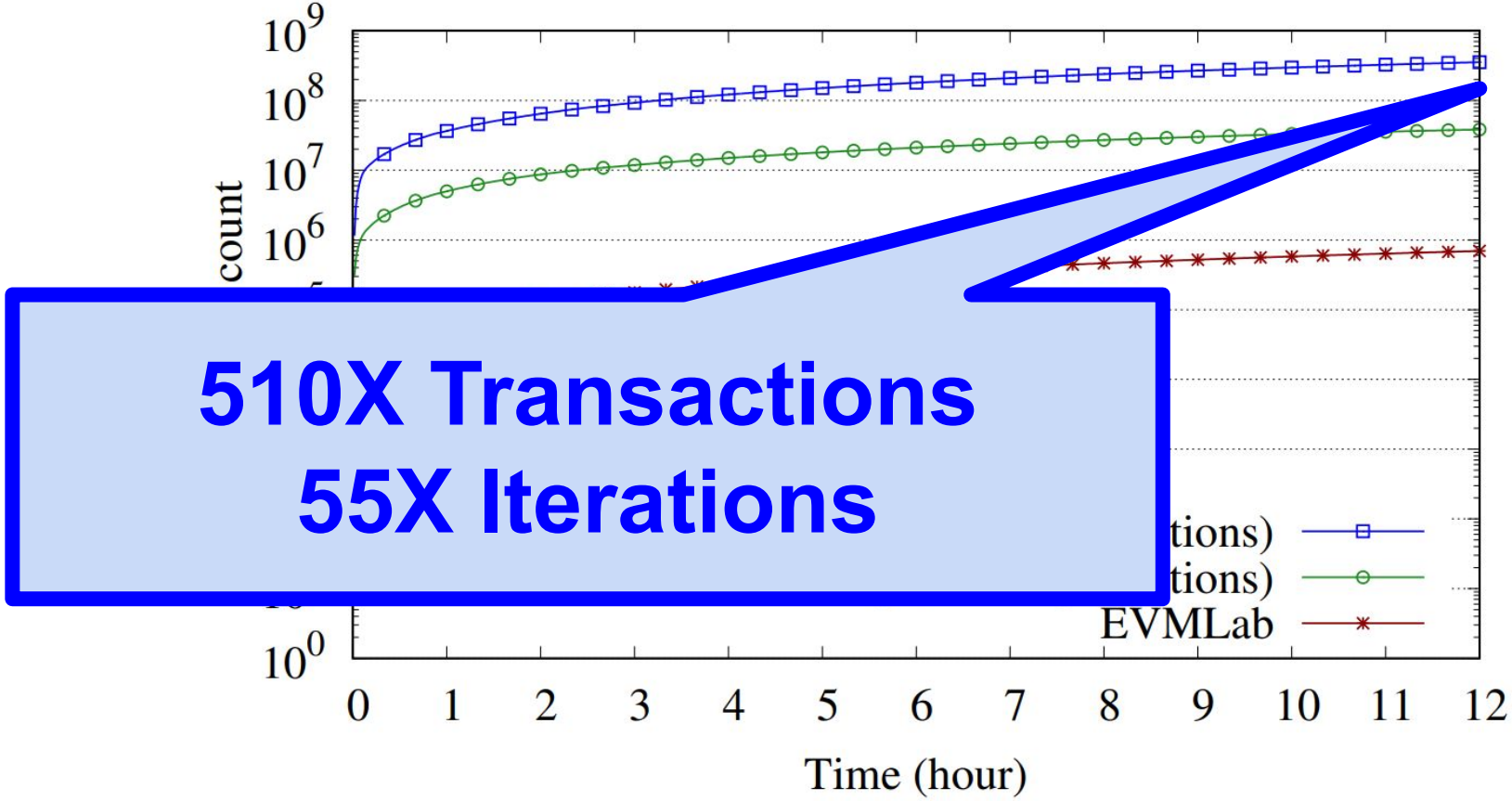


2.7X EVMLab
1.8X Random bytecode

Throughput (Higher is better)



Throughput (Higher is better)



510X Transactions
55X Iterations

Conclusion: Fluffy

- Problem: Find new consensus bugs in Ethereum
- Solution: Multi-transaction differential fuzzer
- Result
  - Found two new high-impact consensus bugs that were exploitable on the live Ethereum mainnet
  - Can find 10 out of 11 consensus bugs within 12 hours
  - vs. EVMLab: 2.7X code coverage, 510X throughput

https://github.com/snuspl/fluffy