End-to-End Formal Verification of Ethereum 2.0 Deposit Smart Contract

Daejun Park   Yi Zhang   Grigore Rosu

July 22, 2020 @ CAV'20

runtime verification

ILLINOIS

Ethereum 2.0

- A new sharded *Proof-of-Stake* protocol
- Lives in parallel with the existing Proof-of-Work chain at its early stage
    - The Proof-of-Work chain is driven by miners
    - The Proof-of-Stake chain is driven by *validators*
- To be a validator, one needs to *deposit* a certain amount of Ether, as a "*stake*", by sending a transaction to the "*deposit contract*"

Ethereum 2.0 deposit contract

deposit ledger

Name	Amount
Alice	10 ETH
Bob	20 ETH
-	-
-	-

2^{32}

$\sim$4 billion

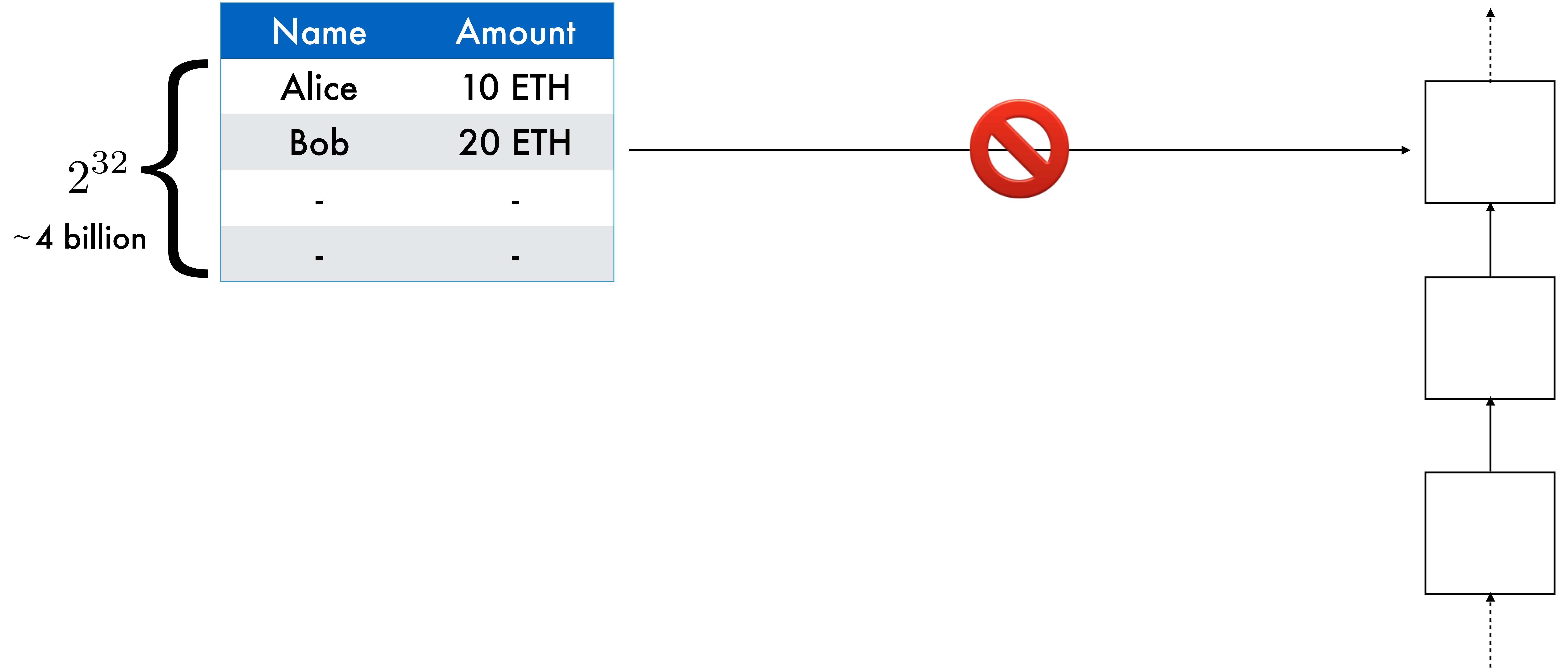
Ethereum 2.0 deposit contract

deposit ledger

Name	Amount
Alice	10 ETH
Bob	20 ETH
-	-
-	-

2^{32}

~4 billion



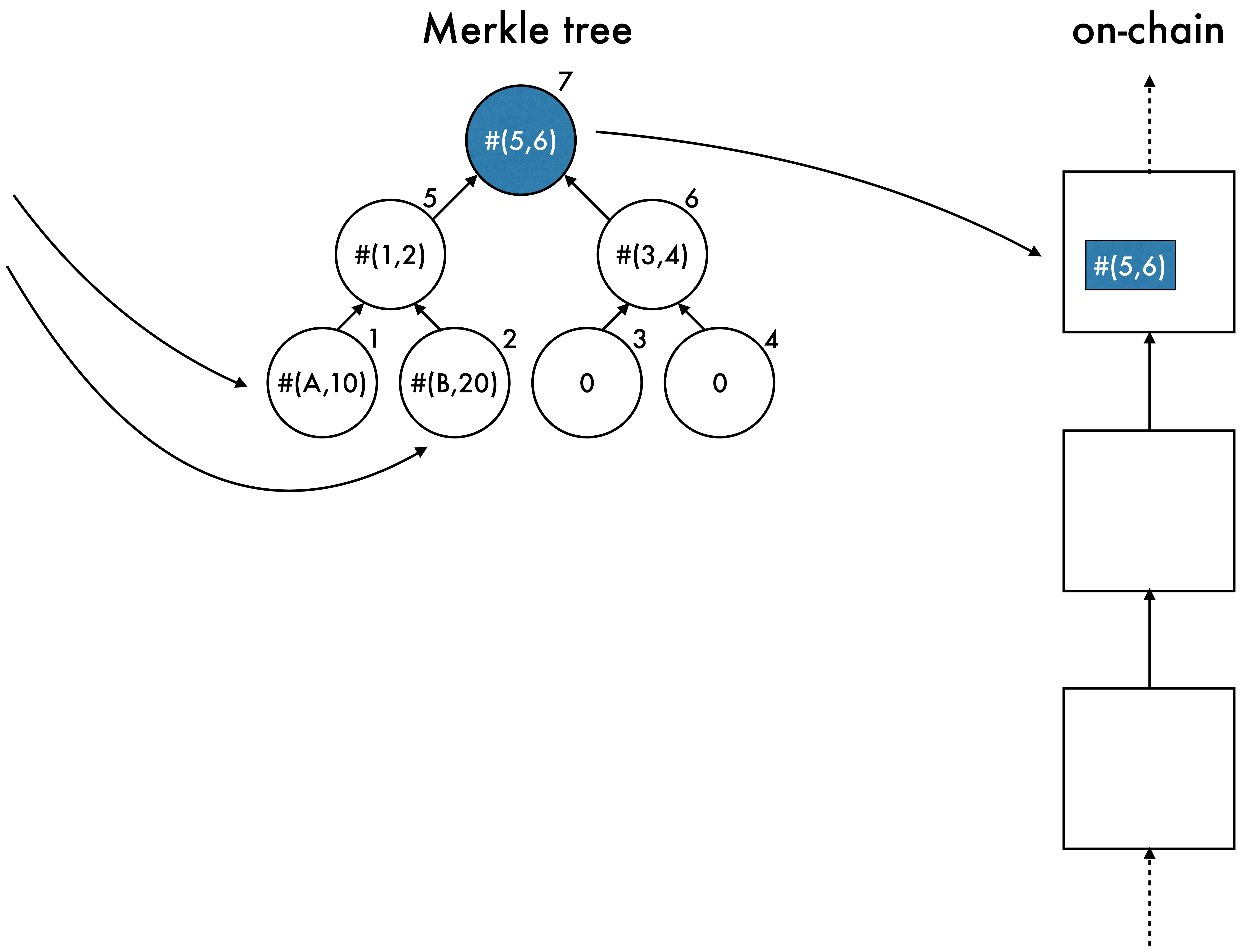
Ethereum 2.0 deposit contract

deposit ledger

2^{32}

~4 billion

Name	Amount
Alice	10 ETH
Bob	20 ETH
-	-
-	-



Ethereum 2.0 deposit contract

deposit ledger

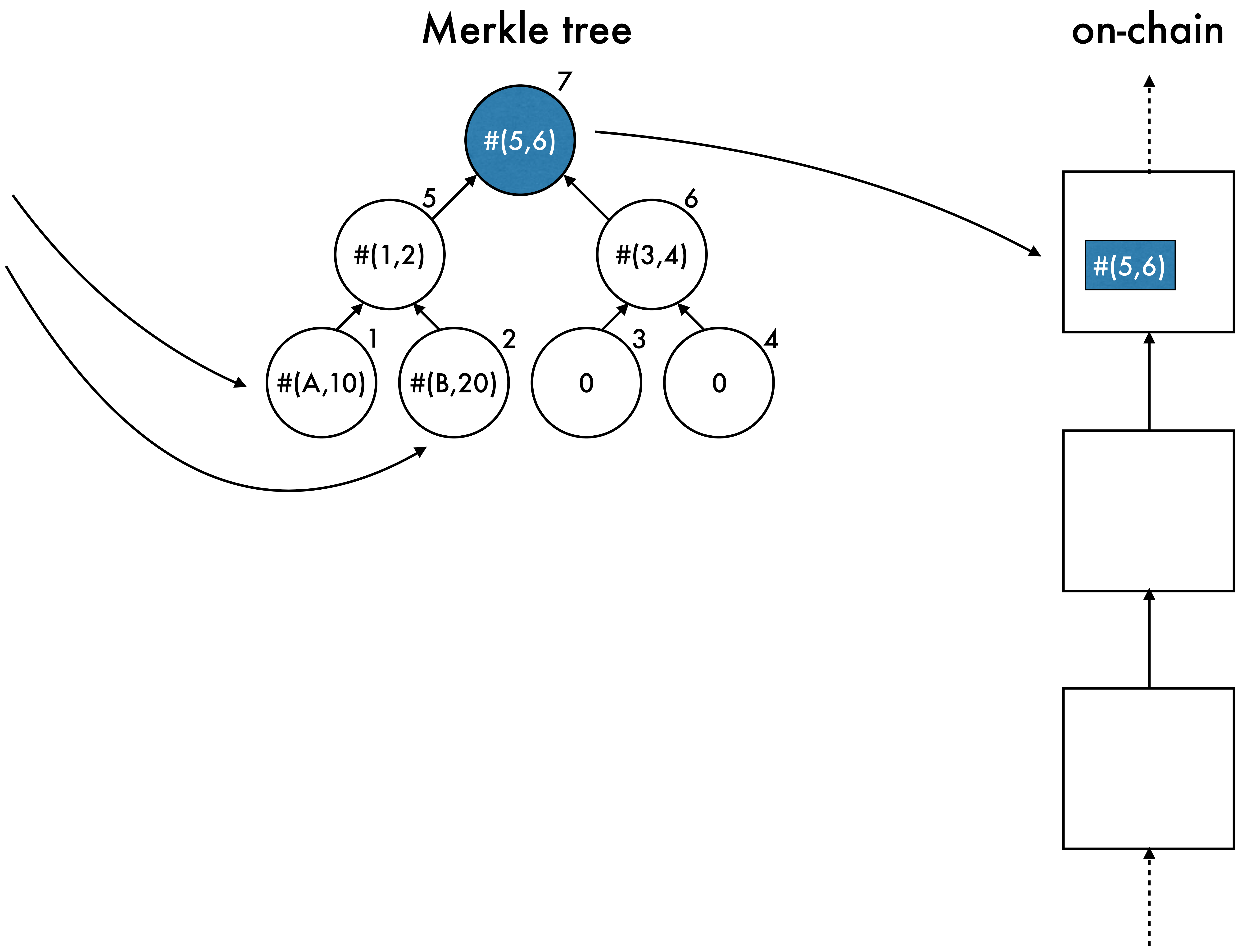
2^{32}

$\sim$4 billion

Name	Amount
Alice	10 ETH
Bob	20 ETH
-	-
-	-

Charlie joins

Name	Amount
Alice	10 ETH
Bob	20 ETH
Charlie	30 ETH
-	-



Ethereum 2.0 deposit contract

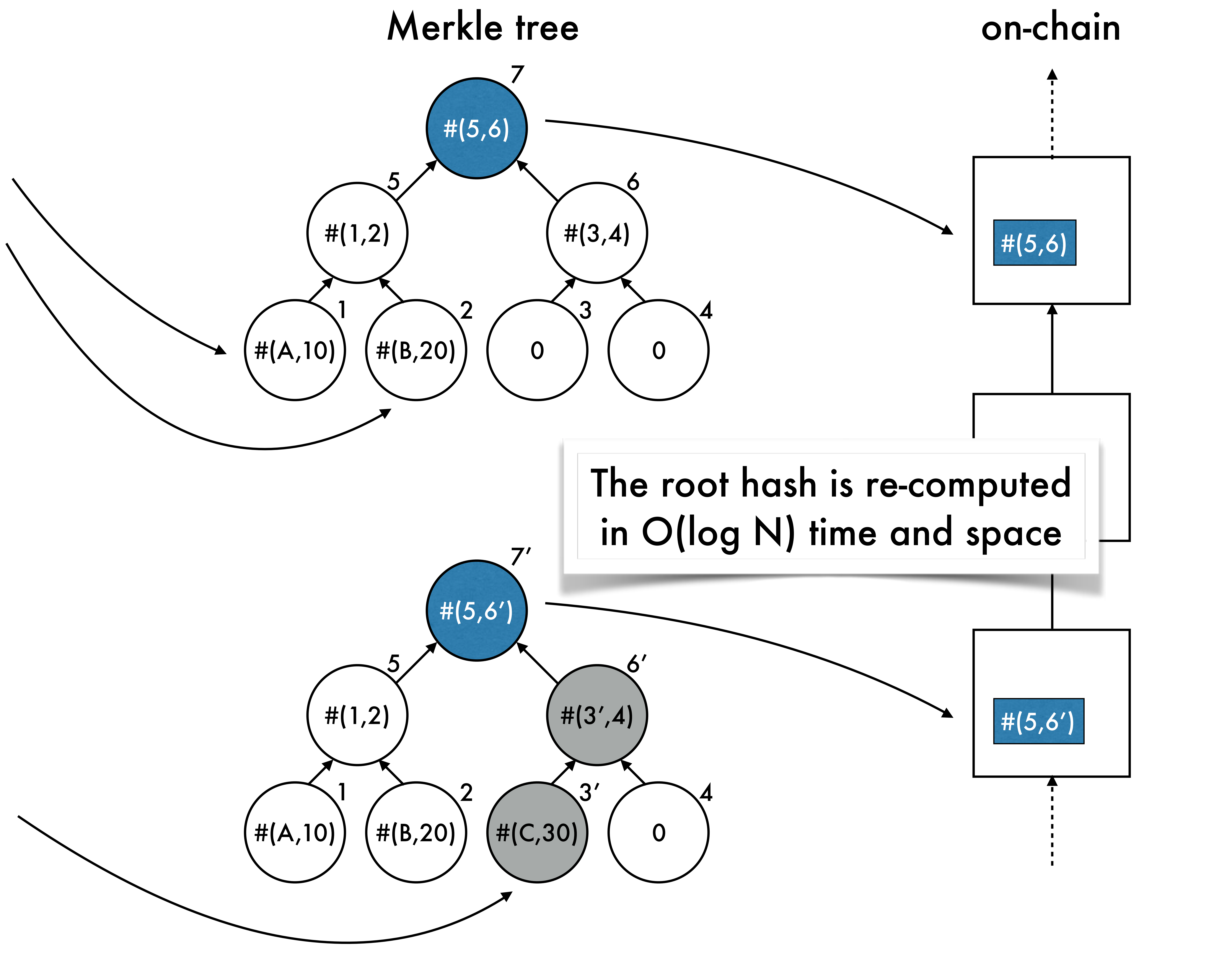
deposit ledger

2^{32} ~4 billion

Name	Amount
Alice	10 ETH
Bob	20 ETH
-	-
-	-

Charlie joins

Name	Amount
Alice	10 ETH
Bob	20 ETH
Charlie	30 ETH
-	-



The root hash is re-computed in O(log N) time and space

Formal verification of deposit contract

- *End-to-end* verification via *refinement-based* approach
  - Formalize and verify the incremental Merkle tree algorithm
  - Verify the contract *bytecode* faithfully implements the algorithm
    - No need to trust compiler
- This separation of concerns helped to reduce verification effort

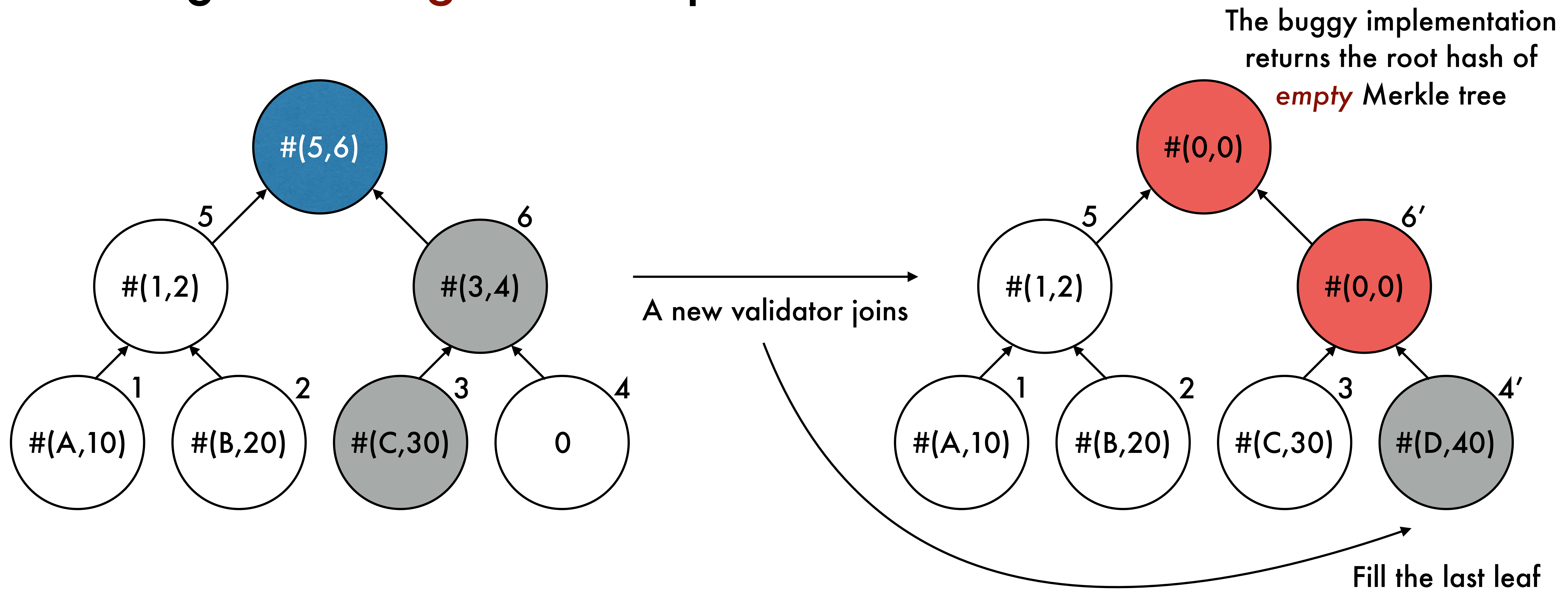
Verification effort

Correctness proof (over formal model)	2 person-weeks
Refinement proof (over bytecode)	5 person-weeks
Total	7 person-weeks

Source code	~100 LOCs
Bytecode	~3,000 instructions
Mechanized proofs	~1,000 LOCs

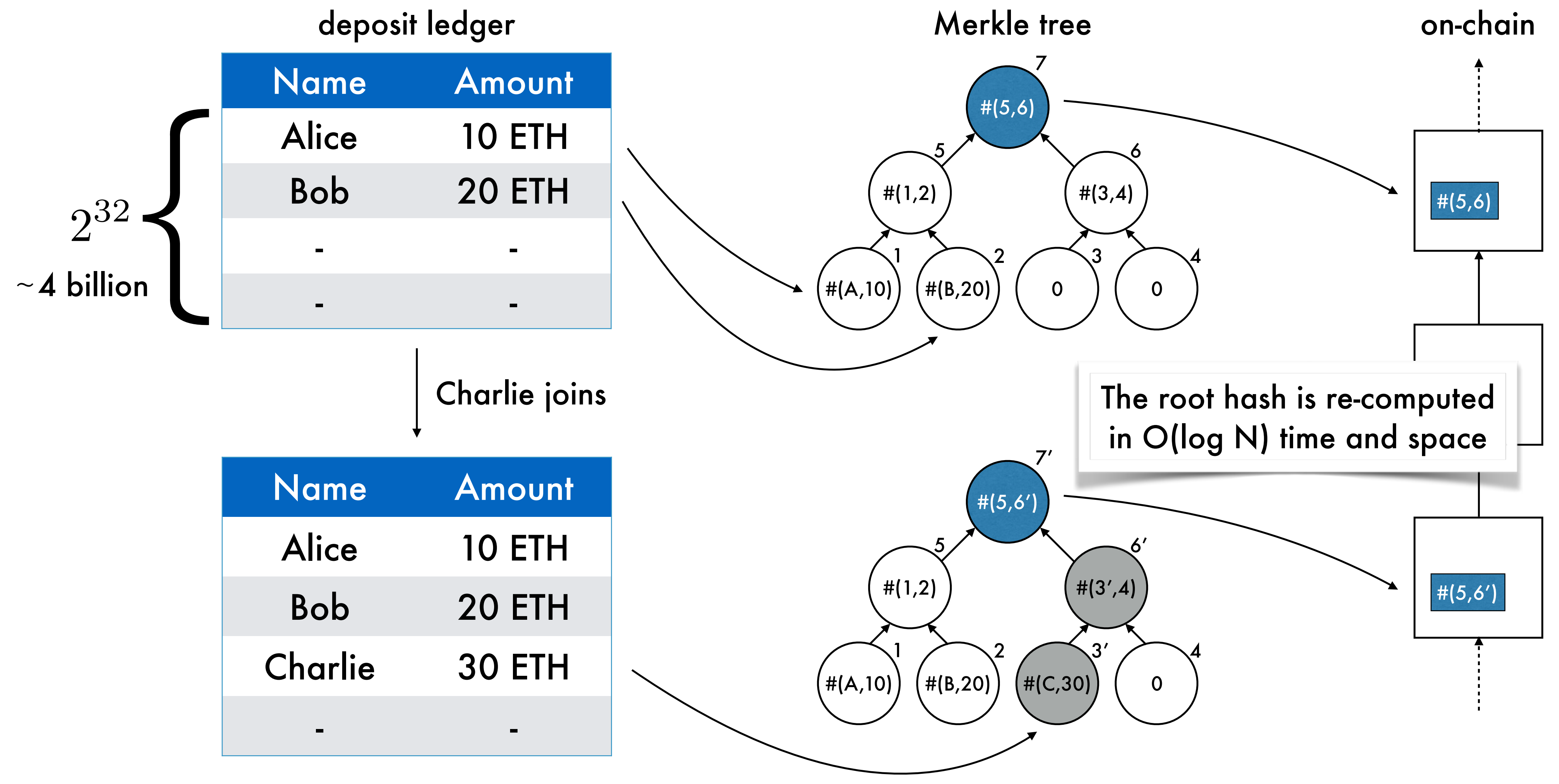
Findings

- Critical bug in the *algorithm* implementation



- Several bugs in the compiled *bytecode*
  - Mostly due to *compiler bugs*

Ethereum 2.0 deposit contract



deposit ledger

Name	Amount
Alice	10 ETH
Bob	20 ETH
-	-
-	-

Name	Amount
Alice	10 ETH
Bob	20 ETH
Charlie	30 ETH
-	-

Formal verification of deposit contract

- *End-to-end* verification via *refinement-based* approach
  - Formalize and verify the incremental Merkle tree algorithm
  - Verify the contract *bytecode* faithfully implements the algorithm
    - No need to trust compiler
- This separation of concerns helped to reduce verification effort

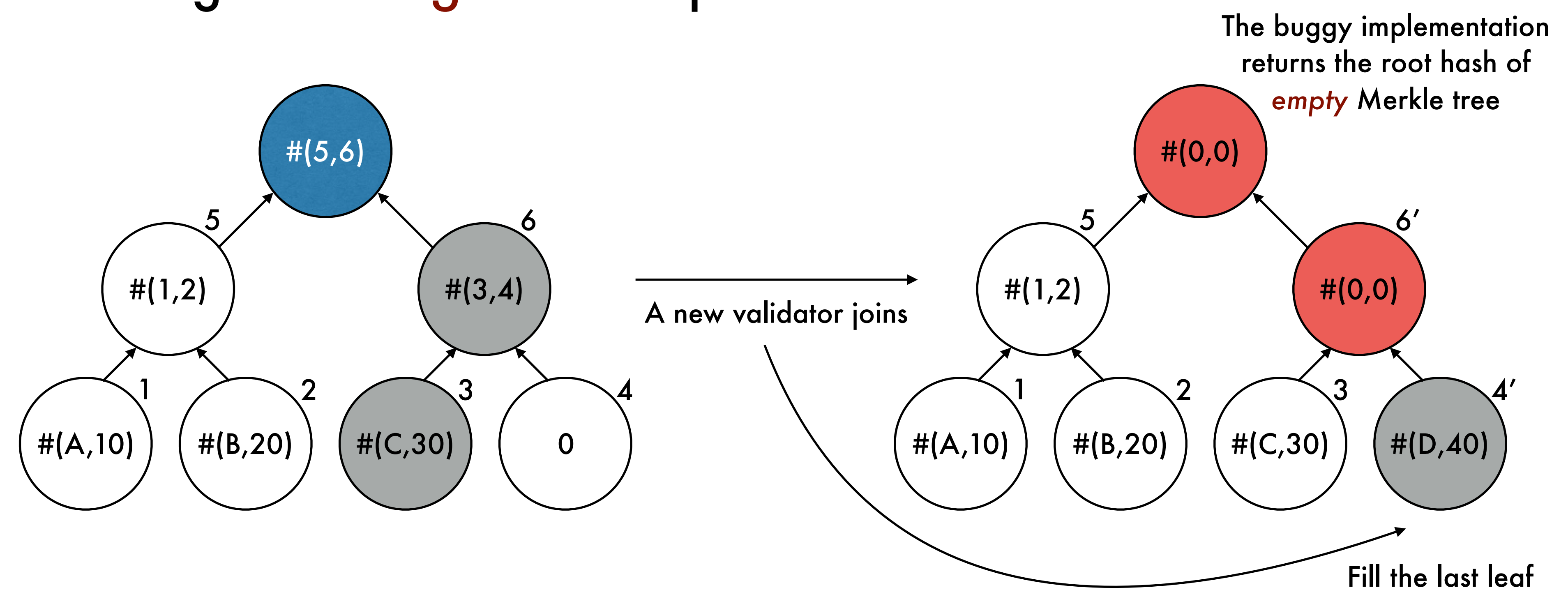
Verification effort

Correctness proof (over formal model)	2 person-weeks
Refinement proof (over bytecode)	5 person-weeks
Total	7 person-weeks

Source code	~100 LOCs
Bytecode	~3,000 instructions
Mechanized proofs	~1,000 LOCs

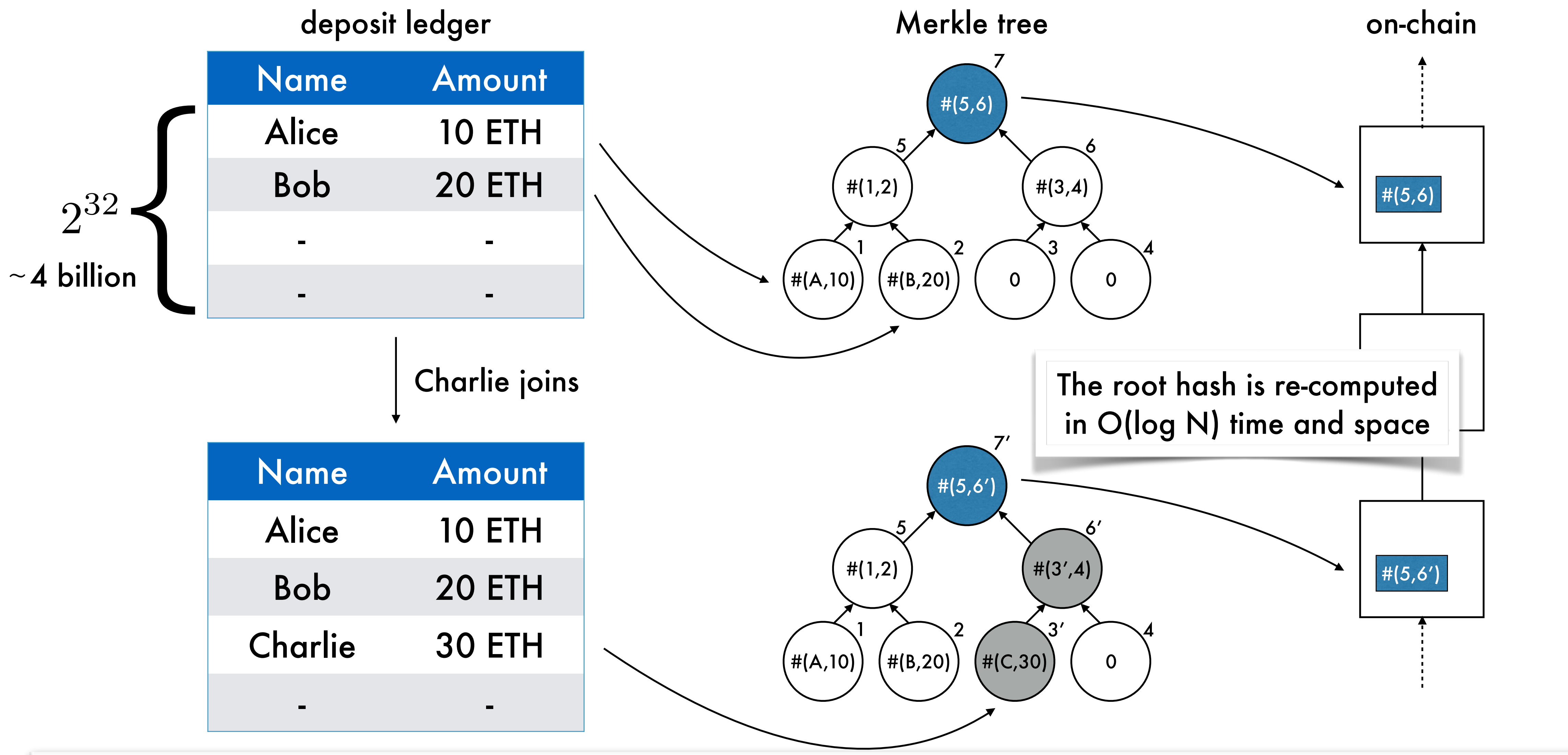
Findings

- Critical bug in the *algorithm* implementation



- Several bugs in the compiled *bytecode*
  - Mostly due to *compiler bugs*

Ethereum 2.0 deposit contract



Formal verification of deposit contract

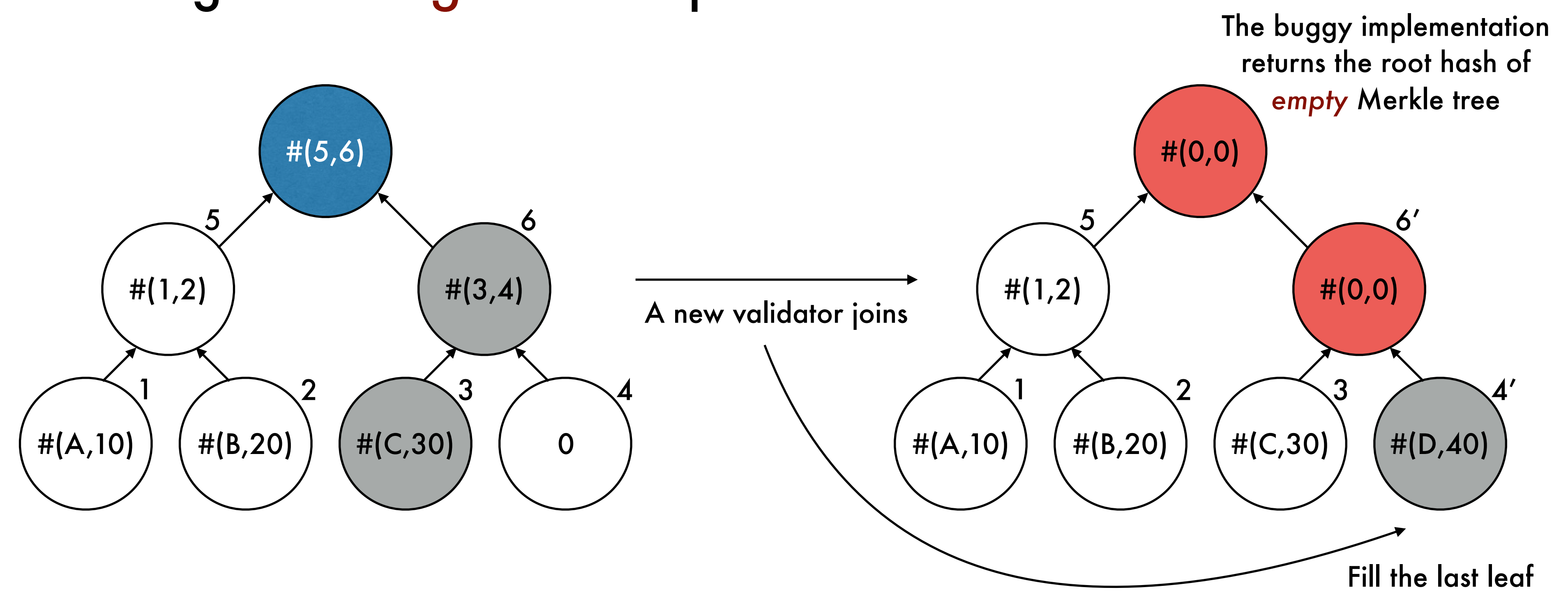
- *End-to-end* verification via *refinement-based* approach
  - Formalize and verify the incremental Merkle tree algorithm
  - Verify the contract *bytecode* faithfully implements the algorithm
    - No need to trust compiler
- This separation of concerns helped to reduce verification effort

https://github.com/runtimeverification/deposit-contract-verification

Correctness proof (over formal model)	2 person-weeks
Refinement proof (over bytecode)	5 person-weeks
Total	7 person-weeks

Source code	~100 LOCs
Bytecode	~3,000 instructions
Mechanized proofs	~1,000 LOCs

- Critical bug in the *algorithm* implementation



- Several bugs in the compiled *bytecode*
  - Mostly due to *compiler bugs*