Ethereum Mechanics

Ronghui Gu

Fall 2025

Columbia University

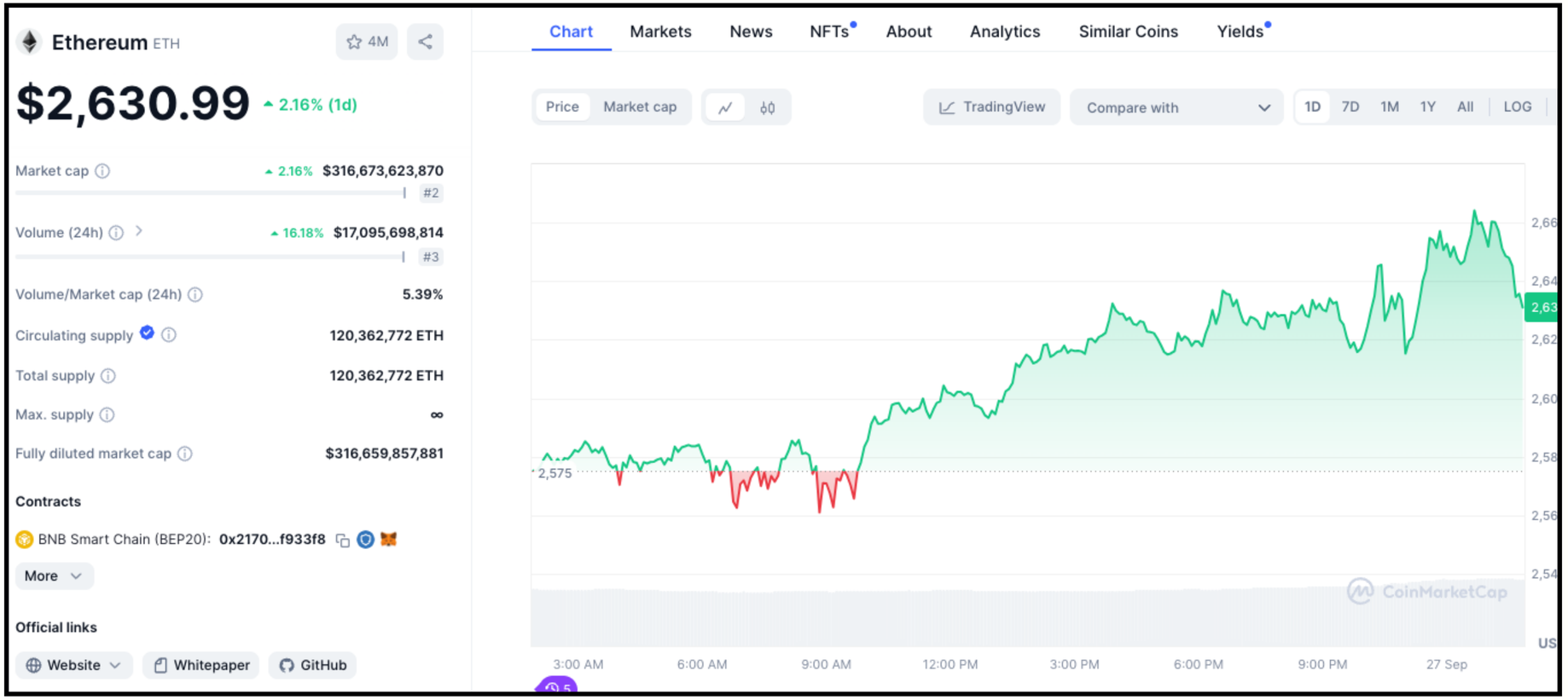
Course website: https://verigu.github.io/6998Fall2025/

Ethereum

Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.
By Vitalik Buterin (2014).

When Satoshi Nakamoto first set the Bitcoin blockchain into motion in January 2009, he was simultaneously introducing two radical and untested concepts. The first is the "bitcoin", a decentralized peer-to-peer online currency that maintains a value without any backing, intrinsic value or central issuer. So far, the "bitcoin" as a currency unit has taken up the bulk of the public attention, both in terms of the political aspects of a currency without a central bank and its extreme upward and downward volatility in price. However, there is also another, equally important, part to Satoshi's grand experiment: the concept of a proof of work-based blockchain to allow for public agreement on the order of transactions. Bitcoin as an application can be described as a first-to-file system: if one entity has 50 BTC, and simultaneously sends the same 50 BTC to A and to B, only the transaction that gets confirmed first will process. There is no intrinsic way of determining

ETH MarketCap



1 Limitations of Bitcoin

Review of Bitcoin Script: P2PKH (Pay to Public Key Hash)

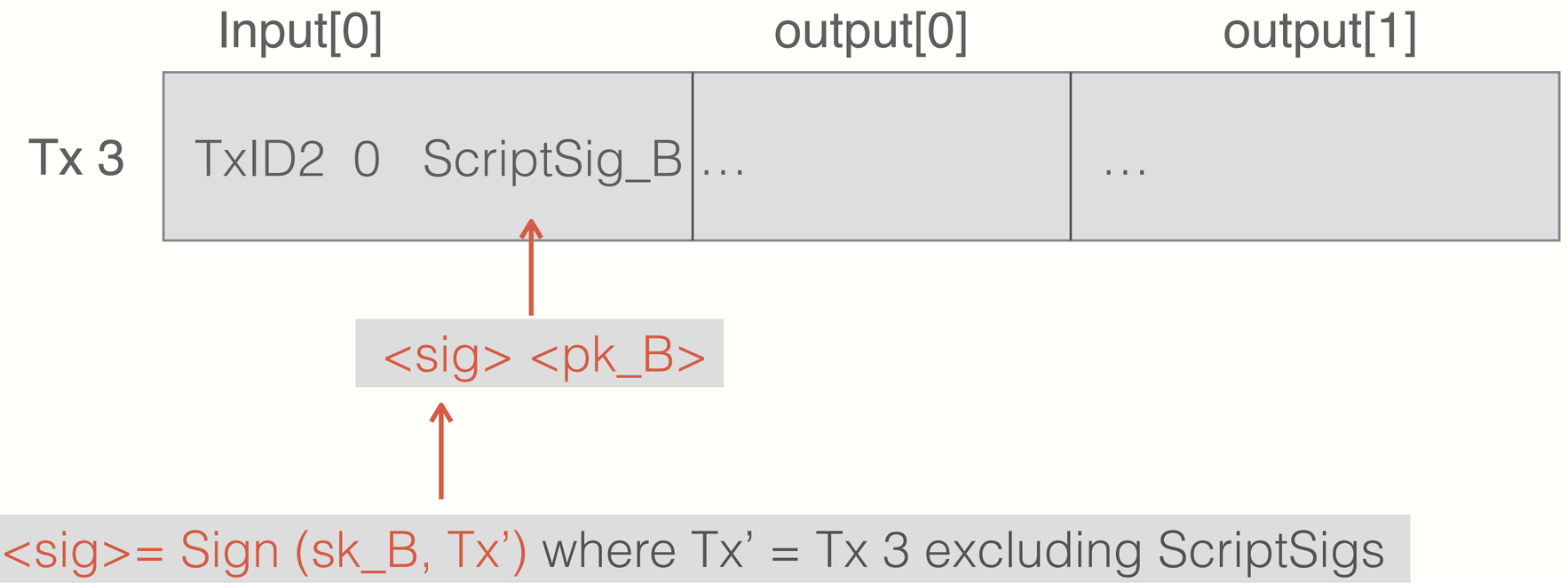
Alice want to pay Bob 5 BTC

- **Step1**: Bob generates key pair (pk_B, sk_B)
- **Step2**: Bob computes his BTC address as addr_B <— H(pk_B)
- **Step3**: Bob sends addr_b to Alice
- **Step4**: Alice broadcasts TX:

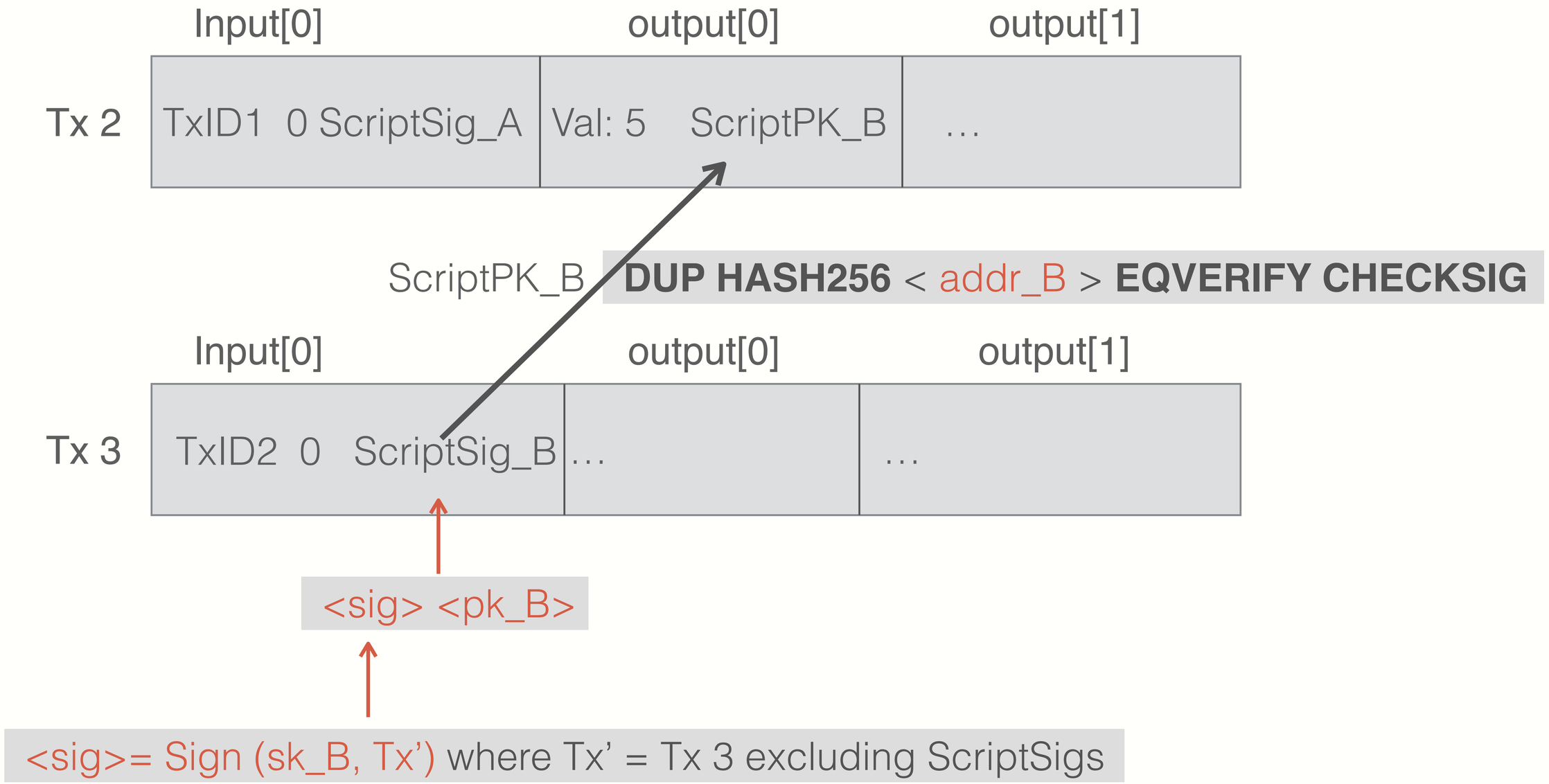
ScriptPK_B = **DUP HASH256** < addr_B > **EQVERIFY CHECKSIG**

Review of Bitcoin Script: P2PKH (Pay to Public Key Hash)

Later, when Bob wants to spend his UTXO, he creates **Tx 3**



Review of Bitcoin Script: P2PKH (Pay to Public Key Hash)



Review of Bitcoin Script: P2PKH (Pay to Public Key Hash)

<sig> <pk_B> **DUP HASH256** <addr_B> **EQVERIFY CHECKSIG**

Stack	
[ ]	Init
[<sig> <pk_B>]	Push values
[<sig> <pk_B> <pk_B>]	**DUP**
[<sig> <pk_B> <addr_B>]	**HASH256** addr_B <— H(pk_B)
[<sig> <pk_B> <addr_B> <addr_B>]	Push values
[<sig> <pk_B>]	**EQVERIFY**
[1]	**CHECKSIG** <sig>= Sign (sk_B, Tx')

Limitations of Bitcoin

UTXO contains (hash of) ScriptPK

- Simple script: indicates conditions when UTXO can be spent
- UTXO matches outputs and inputs, but does not track states explicitly

Limitations

- Difficult to maintain state in multi-stage contracts
- Difficult to enforce global rules on assets
- Example: rate limiting
  - Desired policy: can only transfer 2BTC per day out of my wallet

An Example: NameCoin

Domain name system on the blockchain: [certik.com —> IP Addr]

Need support for three operations:

- Name.new(OwnerAddr, DomainName): intent to register
- Name.update(DomainName, newVal, newOwner, OwnerSig)
- Name.lookup(DomainName)

An Example: NameCoin

Name.new and Name.update create a UTXO with **ScriptPK**:

DUP HASH256 <OwnerAddr> **EQVERIFY CHECKSIG VERIFY**
<NAMECOIN><DomainName> <IPAddr> <1>

only owner can spend this UTXO to update domain data:

Contract: if certik.com is registered, no on else can register the domain

Problem: this contract cannot be enforced just using Bitcoin script

An Example: NameCoin

Namecoin: fork of Bitcoin that implements this contract

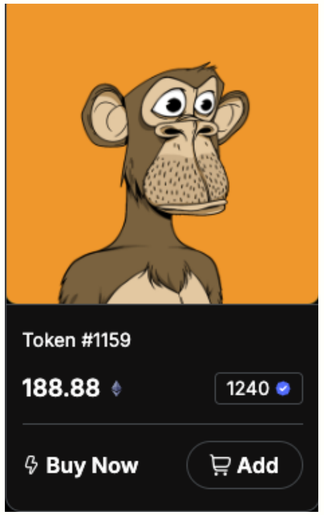
Can we build a blockchain that is programmable to support generic contracts?

ethereum

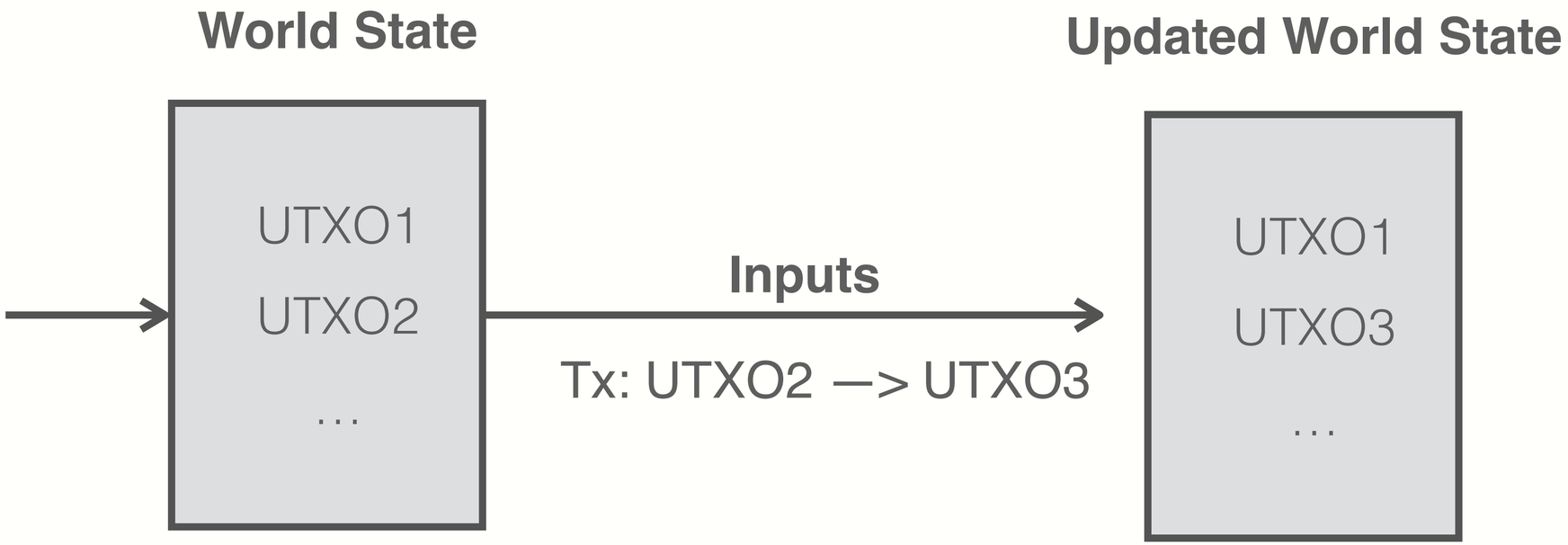
2 Ethereum

Ethereum: Enables a World of DApps

- New coins: ERC-20 Interface to DApps.
- DeFi: exchanges, lending, stablecoins, etc.
- NFTs: ERC-721 Interface to manage distinguished assets
- Games: assets managed on chain



Bitcoin as a State Transition System



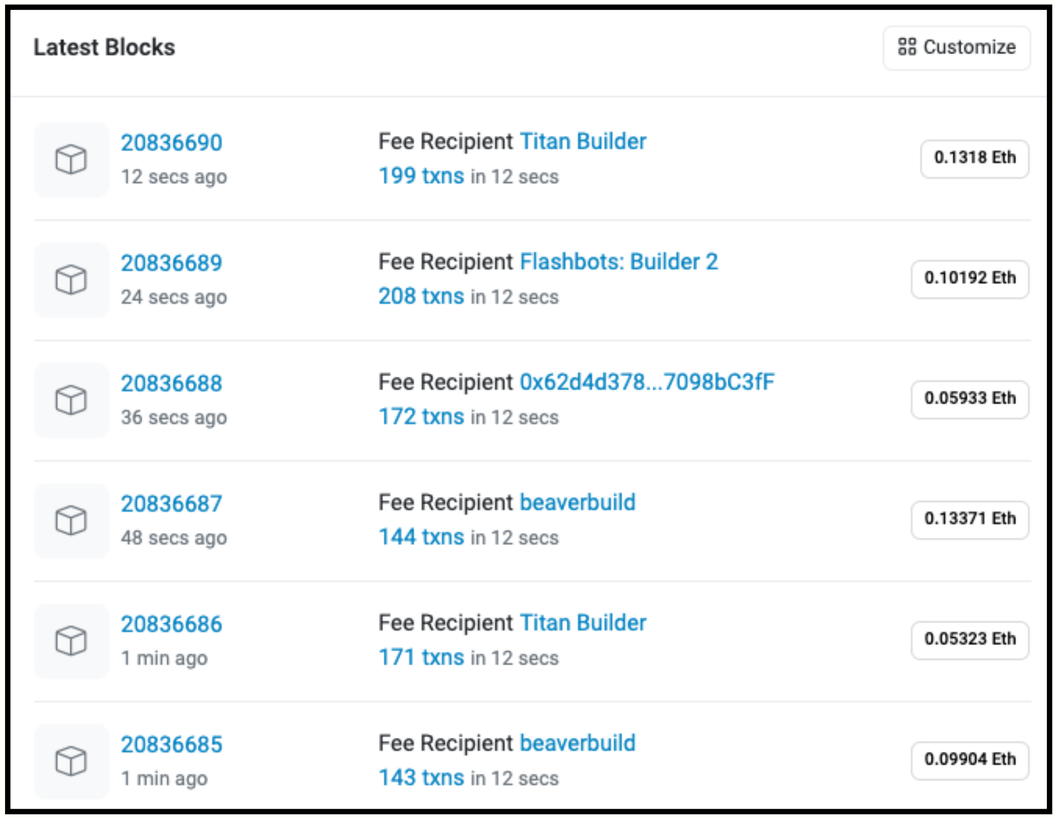
Ethereum System

Layer1 (ETHv1)

- PoW consensus
- Block reward = 2 ETH + Tx fees (gas)
- Avg block rate = 15s
- ~ 150 Tx per block

ETHv2:

- PoS (Proof of Stake) consensus
- Sept 15, 2022



Ethereum Compute Layer: the EVM

World State: set of accounts identified by 32-byte address

Two types of accounts:

- Owned accounts: controlled by signing key pair (PK, SK)
- Contracts: controlled by code
    - Code set at account creation time
    - Does not change

Ethereum Compute Layer: the EVM

Data	Owned	Contracts
Address	H(PK)	H(CreatorAddr, CreatorNonce)
Code	—	CodeHash
State	—	Storage Root
Balance	Balance	Balance
Nonce	Nonce	Nonce

= #Tx sent + #accounts created

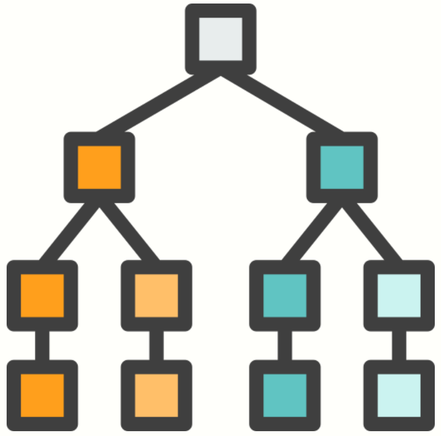
Account State: Persistent Storage

Every contract has an associated **storage array S[]**:

- S[0], S[1], …, S[2^256 -1]: each cell holds 32 bytes, init to 0.

Account storage root: Merkle Patricia Tree hash of S[]:

- Why not directly using Merkel Tree hash?



Account State: Persistent Storage

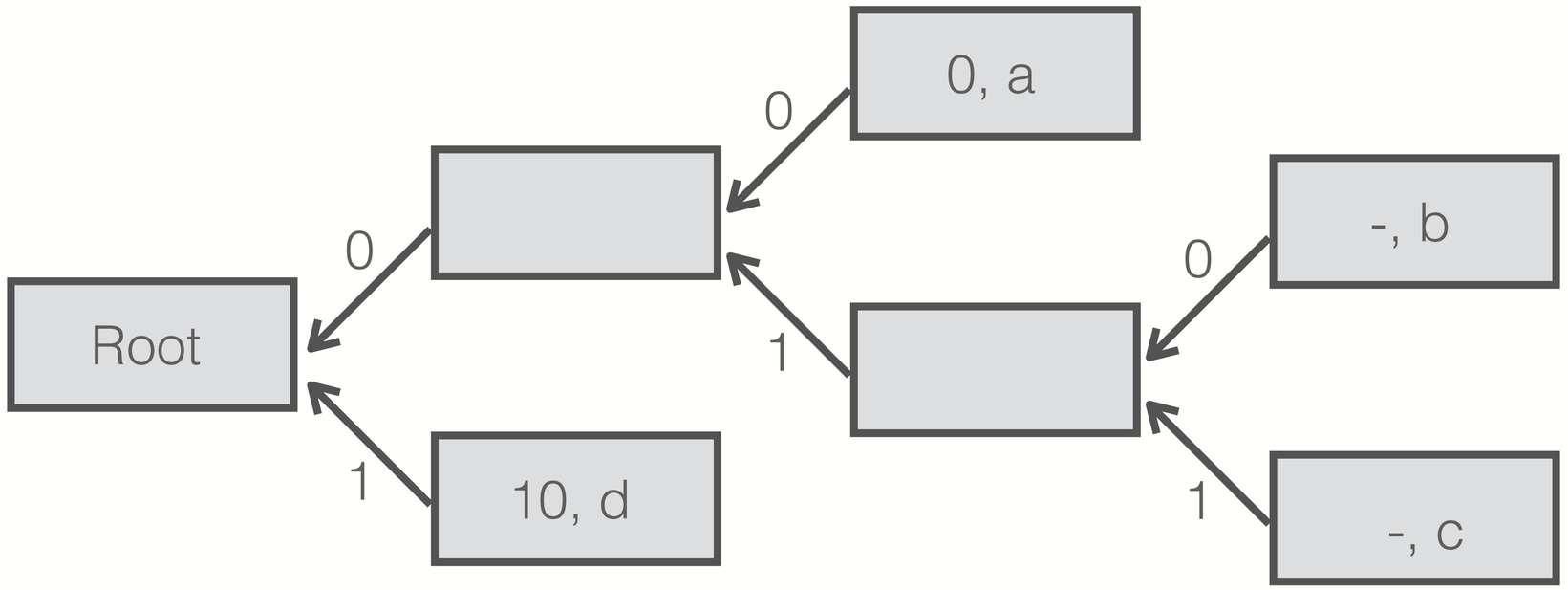
Every contract has an associated **storage array S[]**:

- S[0], S[1], …, S[2^256 -1]: each cell holds 32 bytes, init to 0.

Account storage root: Merkle Patricia Tree hash of S[]:

- Why not directly using Merkel Tree hash?

S[000] = a
S[010] = b
S[011] = c
S[110] = d



State Transitions: Tx and Messages

Transactions: signed data by initiator

- **To**: 32-byte address of target (0 means creating new contract)
- **From, [Sig]**: initiator address and [signature on Tx if owned accounts]
- **Value**: #Wei being sent with Tx
- **Tx fees** (EIP 1559): gasLimit, maxFee, maxPriorityFee
- **Code** (If To = 0): (init, body)
- **Data** (If To != 0): what function to call and args
- **Nonce**: must match current nonce of sender
  - Preventing Tx replay

State Transitions: Tx and Messages

Transaction Types

- Owned -> Owned: transfer ETH between users
- Owned -> Contract: call contract with ETH and data

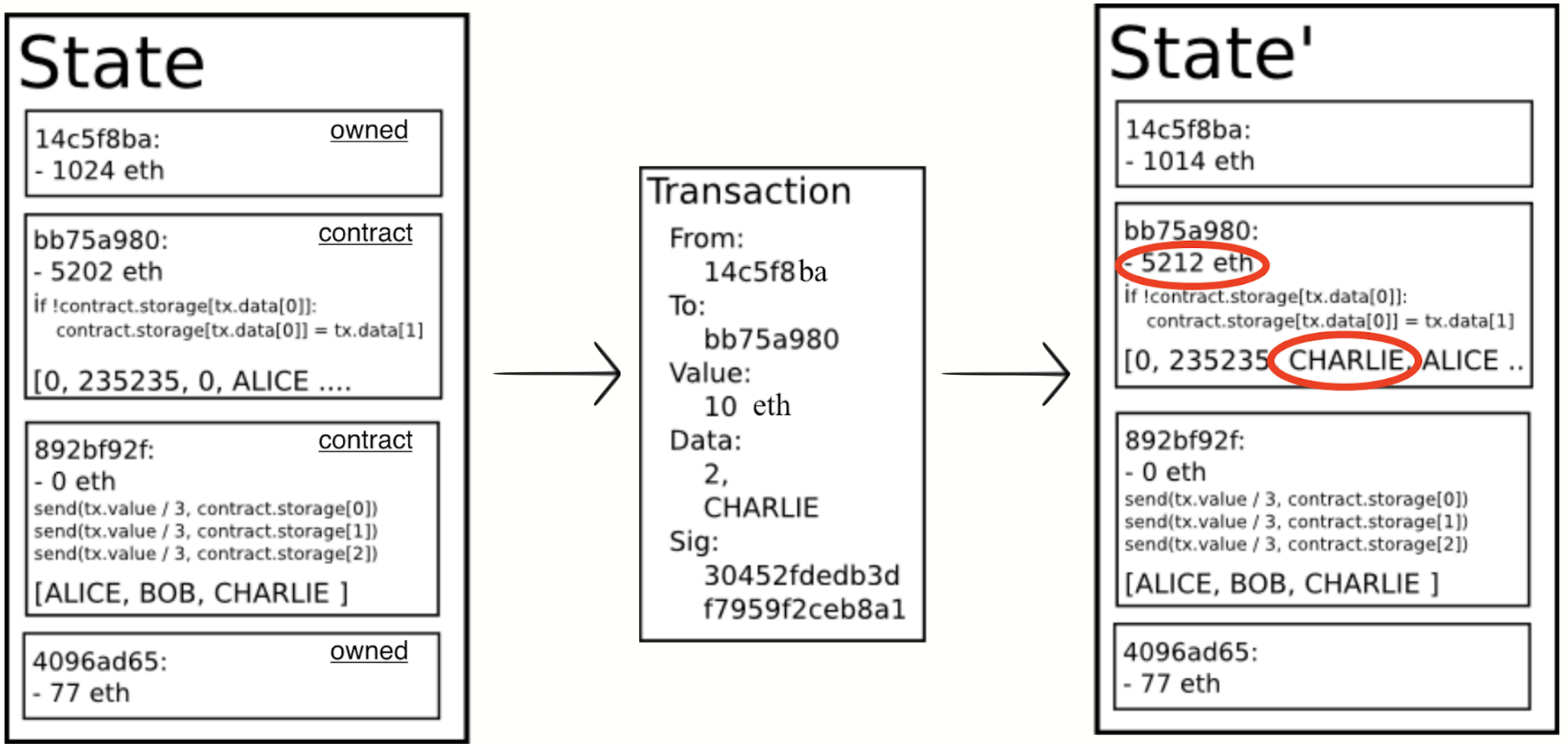
Messages: same as Tx, but no signature

- Contract -> Owned: contracts sends funds users
- Contract -> Contract: one program calls another (and sends funds)

One Tx from user: can lead to many Tx and messages

- Tx from Owned -> Contract -> another Contract -> Different Owned

State Transitions: Tx and Messages



An Ethereum Block

Miners: collect Txs from users

- For each Tx, execute state change sequentially
- Record updated world state in block

Leader: creates a block

Other miners: re-execute all Txs to verify the block

- Miners should only build on a valid block
- Miners are not paid for verifying block

Block Header Data (Simplified)

Consensus data: Prev hash, difficulty, PoW solution, etc

Address of gas beneficiary: where Tx fees will go

World state root: updated world state

- Merkle Patricia Tree has of all accounts in the system

Tx root: Merkel hash of all Tx in the block

Tx receipt root: Merkel hash of log messages generated in the block

Gas used: tells verifier how much work to verify block

3 Ethereum Contracts

An Example Contract: NameCoin

```
contract nameCoin {    // Solidity code

    struct nameEntry {

        address owner; // address of domain owner
        bytes32 value;  // IP address

    }

    // array of all registered domains

    mapping (bytes32 => nameEntry) data;
```

An Example Contract: NameCoin

```
function nameNew (bytes32 name)  {
    // registration costs is 100 Wei
    if (data[name] == 0 && msg.value >= 100) {
        data[name].owner = msg.sender;   // record domain owner
        emit Register(msg.sender, name);  // log event
    }
}
```

An Example Contract: NameCoin

```
function nameUpdate (
                    bytes32 name,  bytes32 newValue, address newOnwer)  {
    // check if message is from domain owner, and update if 10Wei is paid
    if (data[name].owner == msg.sender && msg.value >= 10) {
        data[name].value = newValue;      // record new value
        data[name].owner = newOwner;   // record new owner
    }
}
```

An Example Contract: NameCoin

```
    function nameLookup (bytes32 name)  {

        return data[name];

    }

} // end of contract
```

EVM Mechanics: Execution

Write code in **Solidity** (or another front-end language)

=> Compile to EVM bytecode

=> Miners use the EVM to execute contract bytecode

The EVM

Stack machine with JUMP

- Max stack depth = 1024
- Program aborts if stack size exceeds; miner keeps gas
- Contract can create or call another contract

Two types of zero initialized memory

- Persistent storage (on blockchain): SLOAD, SSTORE (expensive)
- Volatile memory (for single Tx): MLOAD, MSTORE (cheap)
- LOG0(data): write data to log

Every Instruction Costs Gas

SSTORE addr (32bytes)**, value** (32bytes)

- Zero -> Non-Zero: 20,000 gas
- Non-Zero -> Non-Zero: 5,000 gas (for a cold slot)
- Non-Zero -> Zero: 15,000 gas refund

Refund: is given for reducing size of blockchain state

Every Instruction Costs Gas

SELFDESTRUCT addr (32bytes)

- Kill current contract
- In the past, 24,000 gas refund

CREATE

- 32,000 + 200 * (code size) gas

Gas Calculation

Why charge gas?

- Prevents submitting Tx that runs for many steps
- During high load: miners choose Tx from the mempool with high gas

Old EVM (prior to EIP 1559)

- Every Tx contains a gasPrice “bid” (gas -> Wei conversion price)
- Miners choose Tx with highest gasPrice
  - First price auction
  - Not efficient

Gas Calculation (EIP 1559)

Every block has a “baseFee”:

- The minimum gasPrice for all Tx in the block

baseFee is computed from total gas in earlier blocks:

- If earlier blocks as “target size” (15M gas) => baseFee does not change
- If earlier blocks at gas limit (30M gas) => baseFee goes up 12.5%
- If earlier blocks empty => baseFee decreases by 12.5%

Gas Calculation (EIP 1559)

EIP1559 Tx specifies three parameters:

- **gasLimit**: max total gas allowed for Tx
- **maxFee**: max allowed gasPrice
- **maxPriorityFee:** additional “tip” to be paid to miner
- gasPrice = min (maxFee, baseFee + maxPriorityFee)

Let’s Look at a Transition

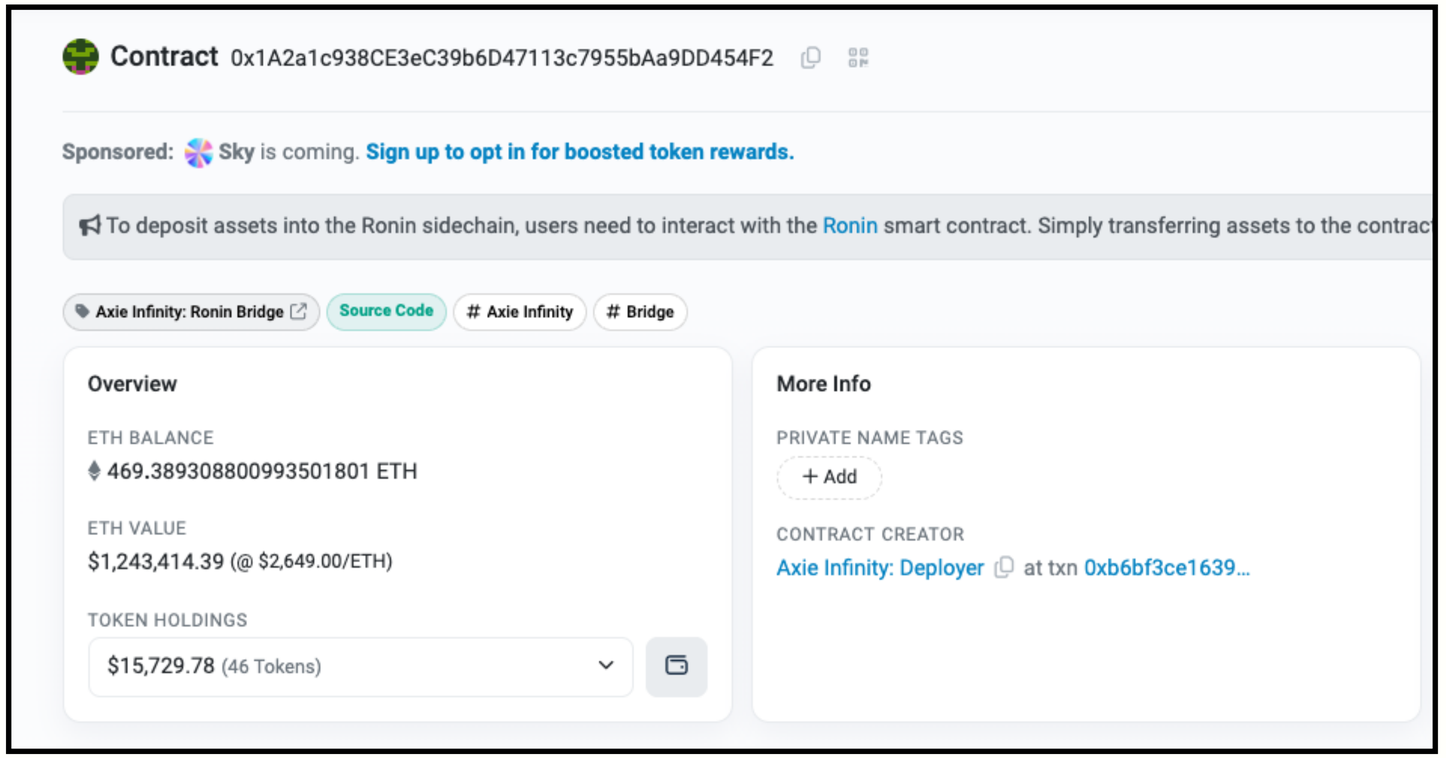
Transaction ID: 0xe3b0c810424edca4d07a00a84…

- **From**: 0x628ebe4e3fe7386da04a6f9a37ccb5e980c22ffc
- **To**: Contract 0x1a2a1c938ce3ec39b6d47113c7955baa9dd454f2
  - (Axie Infinity: Ronin Bridge)
- **Value:** 0.167 Ether
- **Data:** Function: depositEthFor [0]: d256119bb3ca86c7c9fcda4daba95bd233150e6

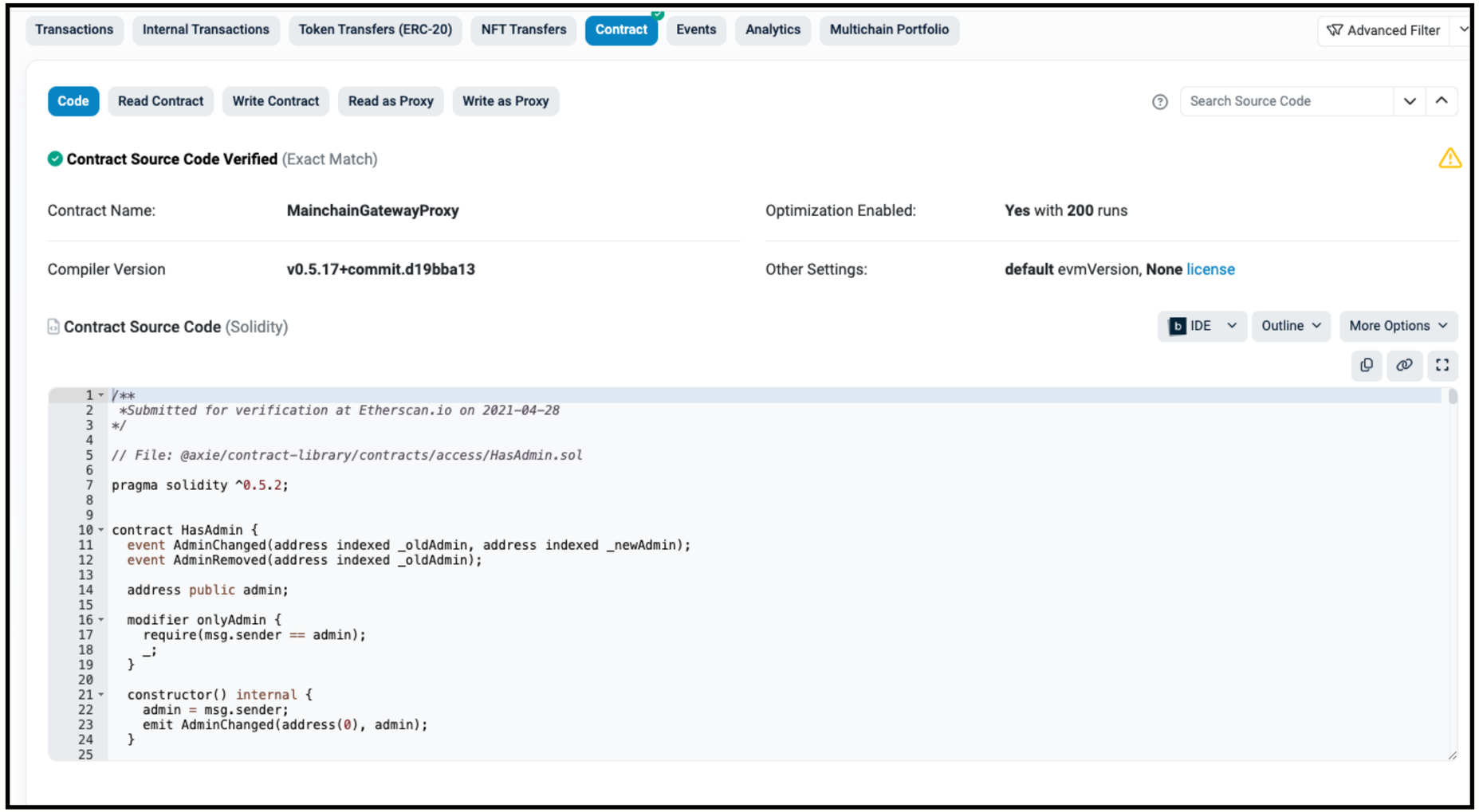
Let’s Look at a Transition

Contract: 0x1a2a1c938ce3ec39b6d47113c7955baa9dd454f2

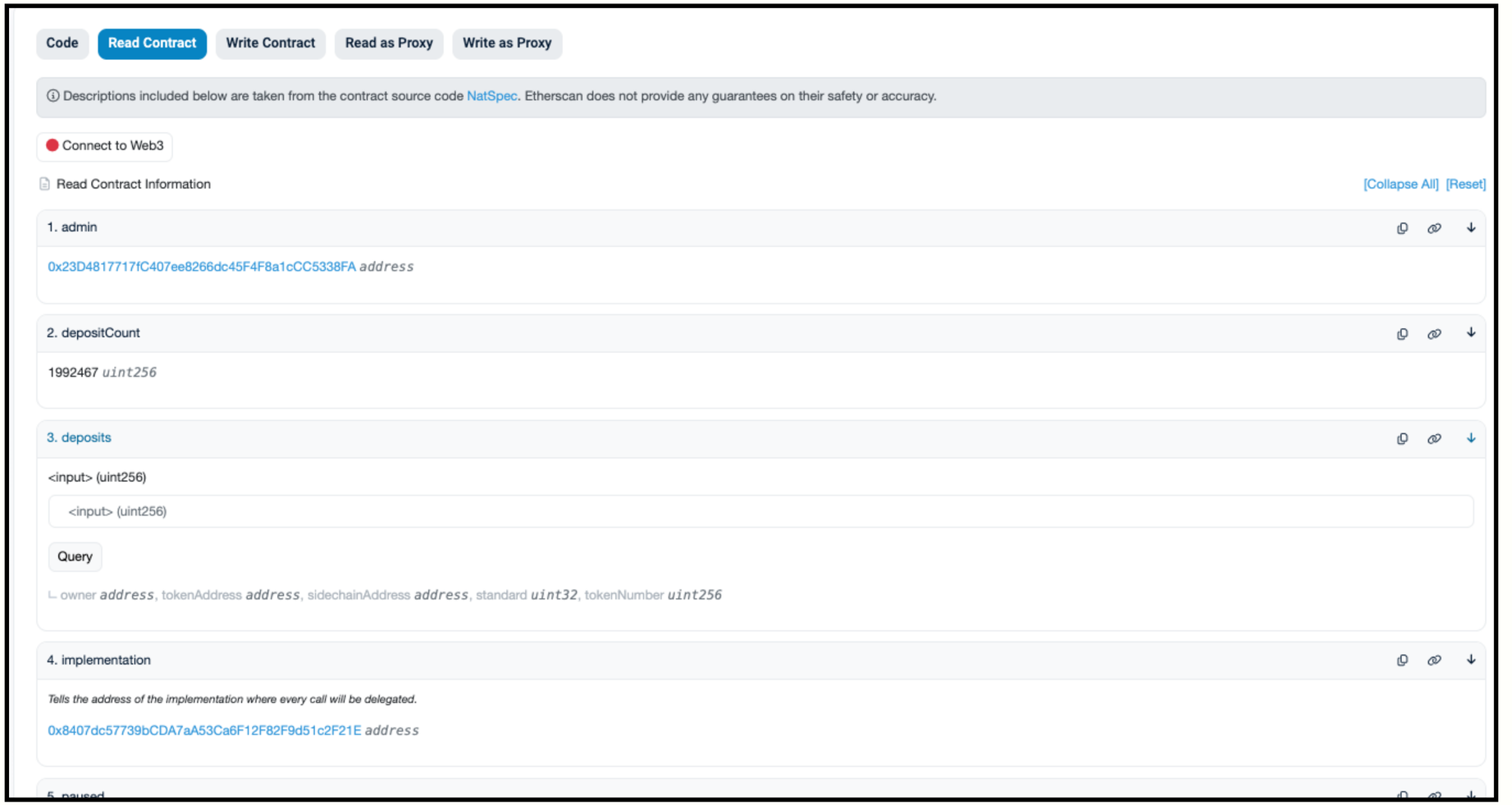
- (Axie Infinity: Ronin Bridge)



Let’s Look at a Transition



Let’s Look at a Transition



4

Discussion Session

How to raise funds?