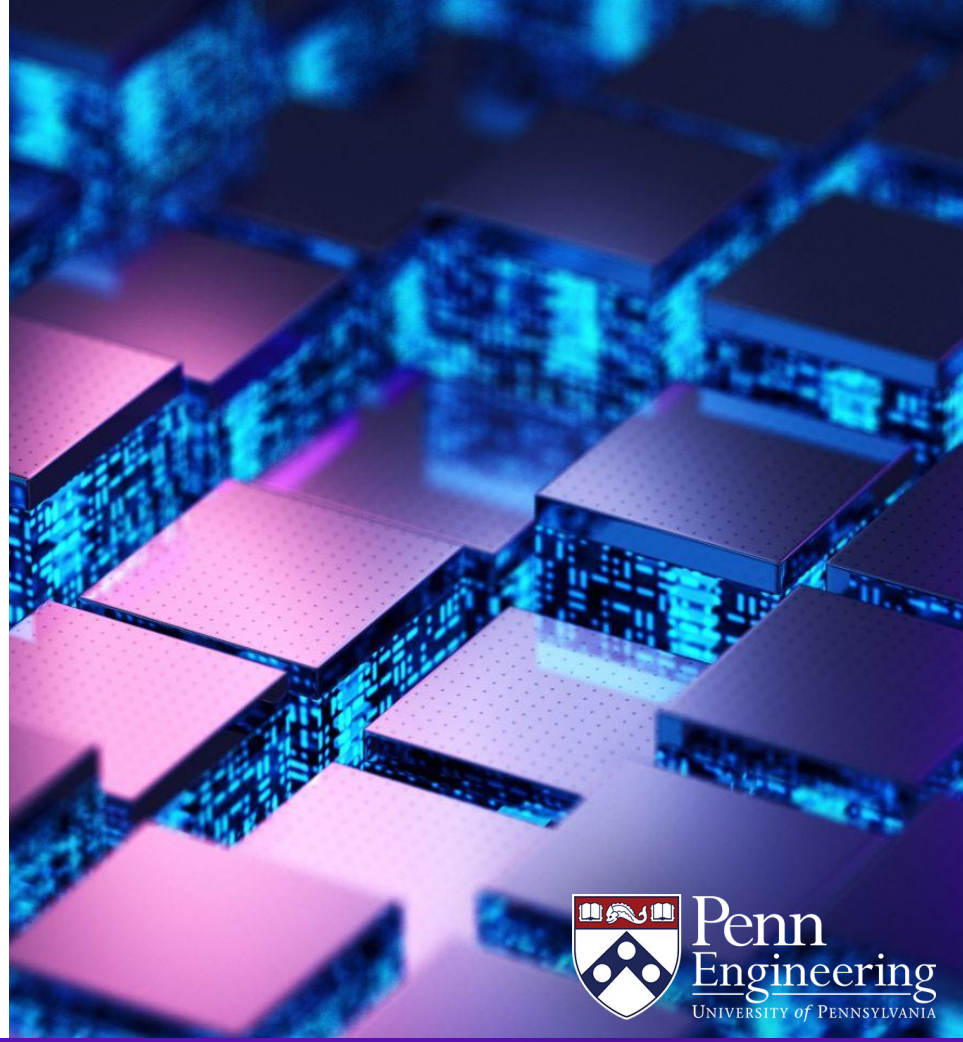
EAS 5830: BLOCKCHAINS

The EVM

Professor Brett Hemenway Falk



The Ethereum Virtual Machine

- The Ethereum Virtual Machine determines how programs (contracts) are executed

The Ethereum Virtual Machine

- The Ethereum Virtual Machine determines how programs (contracts) are executed
- The EVM is the blockchain's Operating System

The Ethereum Virtual Machine

- The Ethereum Virtual Machine determines how programs (contracts) are executed
- The EVM is the blockchain's Operating System
- The EVM is single threaded

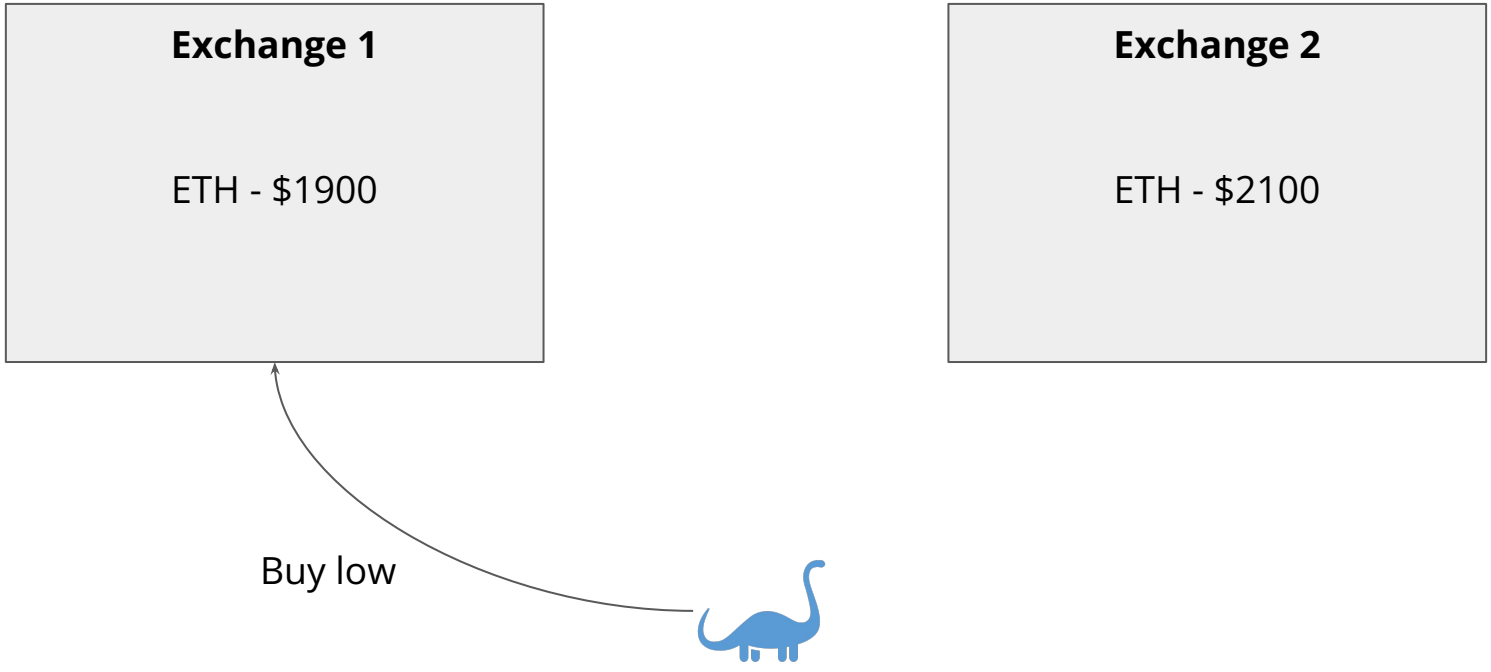
The Ethereum Virtual Machine

- The Ethereum Virtual Machine determines how programs (contracts) are executed
- The EVM is the blockchain's Operating System
- The EVM is single threaded
  - No parallelism

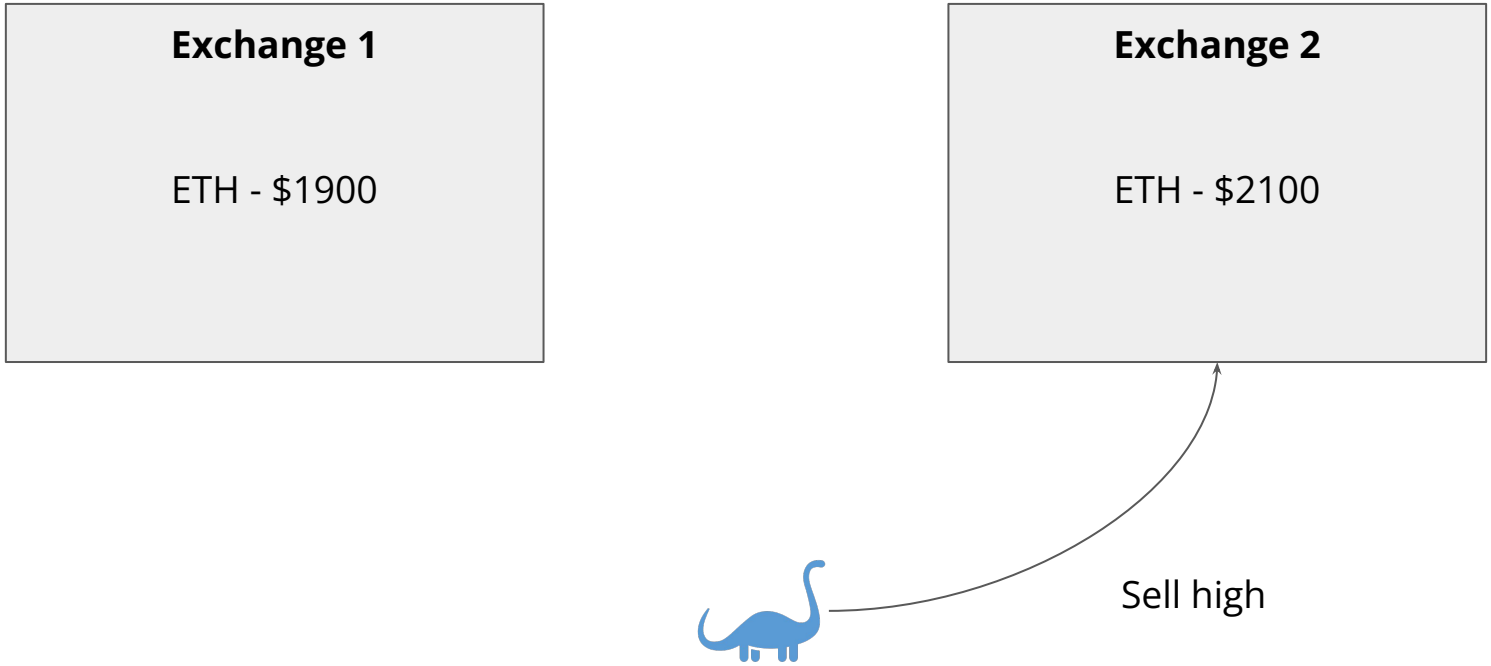
The Ethereum Virtual Machine

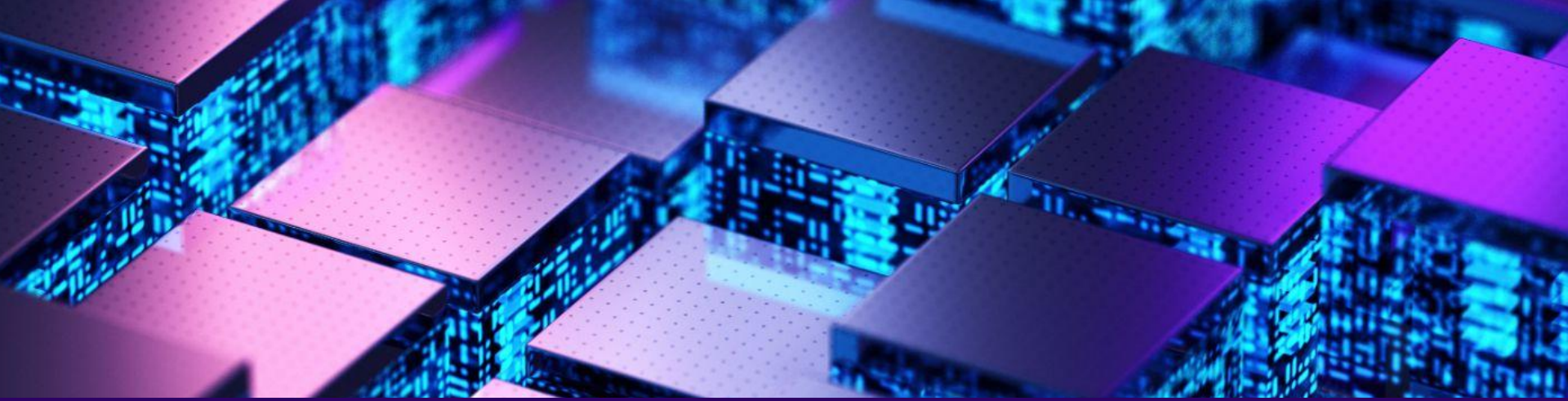
- The Ethereum Virtual Machine determines how programs (contracts) are executed
- The EVM is the blockchain's Operating System
- The EVM is single threaded
  - No parallelism
  - Transactions are executed one at a time and in order

Riskless arbitrage



Riskless arbitrage





EVM Bytecode

EVM OpCodes

Instructions SHANGHAI

Search by Name Enter keyword...

OPCODE	NAME	MINIMUM GAS	STACK INPUT	STACK OUTPUT	DESCRIPTION
00	STOP	0			Halts execution
01	ADD	3	a b	a + b	Addition operation
02	MUL	5	a b	a * b	Multiplication operation
03	SUB	3	a b	a - b	Subtraction operation
04	DIV	5	a b	a // b	Integer division operation
05	SDIV	5	a b	a // b	Signed integer division operation (truncated)
06	MOD	5	a b	a % b	Modulo remainder operation
07	SMOD	5	a b	a % b	Signed modulo remainder operation
08	ADDMOD	8	a b N	(a + b) % N	Modulo addition operation
09	MULMOD	8	a b N	(a * b) % N	Modulo multiplication operation
0A	EXP	10	a exponent	a ** exponent	Exponential operation
0B	SIGNEXTEND	5	b x	y	Extend length of two's complement signed integer
10	LT	3	a b	a < b	Less-than comparison
11	GT	3	a b	a > b	Greater-than comparison

Expand

EVM Precompiles

Precompiled Contracts SHANGHAI

Search by Name Enter keyword...

ADDRESS	NAME	MINIMUM GAS	INPUT	OUTPUT	DESCRIPTION
0×01	ecRecover	3000	hash, v, r, s	publicAddress	Elliptic curve digital signature algorithm (ECDSA) public key recovery function
0×02	SHA2-256	60	data	hash	Hash function
0×03	RIPEMD-160	600	data	hash	Hash function
0×04	identity	15	data	data	Returns the input
0×05	modexp	200	Bsize, Esize, Msize, B, E, M	value	Arbitrary-precision exponentiation under modulo
0×06	ecAdd	150	x1, y1, x2, y2	x, y	Point addition (ADD) on the elliptic curve 'alt_bn128'
0×07	ecMul	6000	x1, y1, s	x, y	Scalar multiplication (MUL) on the elliptic curve 'alt_bn128'
0×08	ecPairing	45000	x1, y1, x2, y2, ..., xk, yk	success	Bilinear function on groups on the elliptic curve 'alt_bn128'
0×09	blake2f	0	rounds, h, m, t, f	h	Compression function F used in the BLAKE2 cryptographic hashing algorithm

Expand

Contract 0xBC4CA0EdA7647A8aB7C2061c2E118A18a936f13D

Featured: - Live Ethereum Argentina August 17th to 19th, Join us at **ethereumargentina.org**.

Bored Ape Yacht Club: BAYC Token | Source Code | Token Contract

Overview

ETH BALANCE

0.010201 ETH

ETH VALUE

$18.52 (@ $1,815.12/ETH)

TOKEN HOLDINGS

$299.76 (41 Tokens)

More Info

PRIVATE NAME TAGS

+ Add

CONTRACT CREATOR

Bored Ape Yacht Club: ... at txn 0x22199329b0aa1aa6...

TOKEN TRACKER

BoredApeYachtClub (BAYC)

Transactions | Internal Transactions | Token Transfers (ERC-20) | NFT Transfers | Contract | Events | Analytics | Comments

Contract 0xBC4CA0EdA7647A8aB7C2061c2E118A18a936f13D

Featured: - Live Ethereum Argentina August 17th to 19th, Join us at **ethereumargentina.org**.

Bored Ape Yacht Club: BAYC Token | Source Code | Token Contract

Overview

ETH BALANCE

0.010201 ETH

ETH VALUE

$18.52 (@ $1,815.12/ETH)

TOKEN HOLDINGS

$299.76 (41 Tokens)

More Info

PRIVATE NAME TAGS

+ Add

CONTRACT CREATOR

Bored Ape Yacht Club: ... at txn 0x22199329b0aa1aa6...

TOKEN TRACKER

BoredApeYachtClub (BAYC)

Transactions | Internal Transactions | Token Transfers (ERC-20) | NFT Transfers | Contract | Events | Analytics | Comments

Contract Source Code Verified (Exact Match)

Contract Name:	**BoredApeYachtClub**	Optimization Enabled:
Compiler Version	**v0.7.0+commit.9e61f92b**	Other Settings:

Contract Source Code (Solidity)

```
 *Submitted for verification at Etherscan.io on 2021-04-22
*/

// File: @openzeppelin/contracts/utils/Context.sol

// SPDX-License-Identifier: MIT

pragma solidity >=0.6.0 <0.8.0;

/*
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with GSN meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
```

forge verify-contract

NAME

forge-verify-contract - Verify smart contracts on a chosen verification provider.

SYNOPSIS

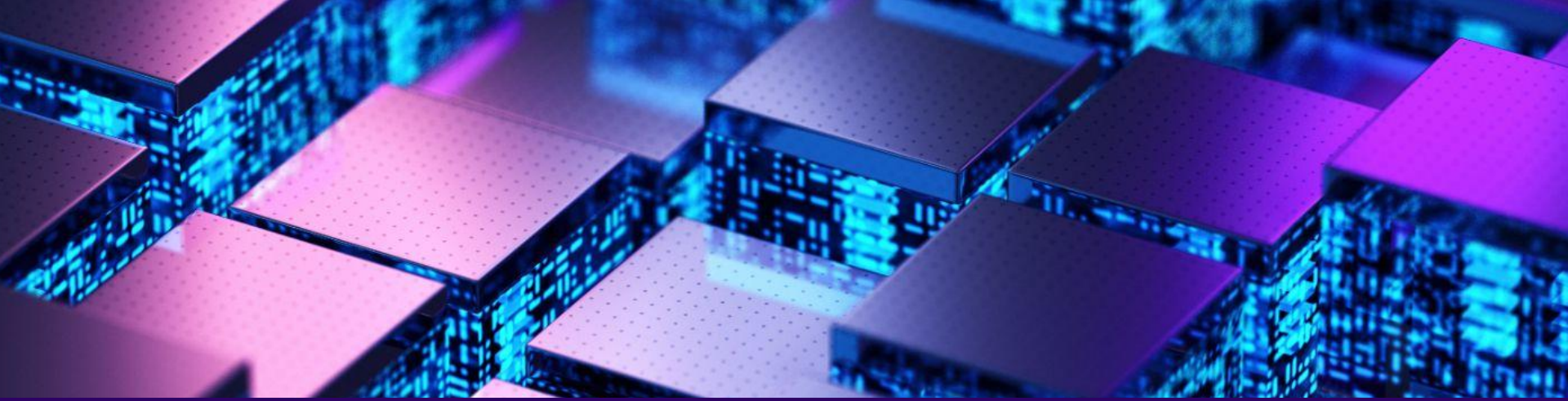
`forge verify-contract` [*options*] *address contract*

DESCRIPTION

Verifies a smart contract on a chosen verification provider.

You must provide:

- The contract address
- The contract name or the path to the contract (read below) In case of Etherscan verification, you must also provide:
- Your Etherscan API key, either by passing it as an argument or setting `ETHERSCAN_API_KEY`



Accounts

Accounts

- Address is 20 bytes

Accounts

- Address is 20 bytes
  - 40 hex digits

Accounts

- Address is 20 bytes
  - 40 hex digits
- Two types of accounts

Accounts

- Address is 20 bytes
  - 40 hex digits
- Two types of accounts
  - Contracts

Accounts

- Address is 20 bytes
  - 40 hex digits
- Two types of accounts
  - Contracts
    - No private key

Accounts

- Address is 20 bytes
  - 40 hex digits
- Two types of accounts
  - Contracts
    - No private key
    - Cannot initiate transactions

Accounts

- Address is 20 bytes
  - 40 hex digits
- Two types of accounts
  - Contracts
    - No private key
    - Cannot initiate transactions
    - Address is last 20 bytes of (Keccak) hash of sender address + nonce

Accounts

- Address is 20 bytes
  - 40 hex digits
- Two types of accounts
  - Contracts
    - No private key
    - Cannot initiate transactions
    - Address is last 20 bytes of (Keccak) hash of sender address + nonce
  - Externally Owned Accounts (EOAs)

Accounts

- Address is 20 bytes
  - 40 hex digits
- Two types of accounts
  - Contracts
    - No private key
    - Cannot initiate transactions
    - Address is last 20 bytes of (Keccak) hash of sender address + nonce
  - Externally Owned Accounts (EOAs)
    - Address is last 20 bytes of (Keccak) hash of public key

Distinguishing accounts

- It’s hard to distinguish whether an account is an EOA or contract

Distinguishing accounts

- It's hard to distinguish whether an account is an EOA or contract
- if web3.eth.get_code returns nonzero, then it's a contract

Distinguishing accounts

- It's hard to distinguish whether an account is an EOA or contract
- if web3.eth.get_code returns nonzero, then it's a contract
  - if it returns 0, it may still be a contract

Distinguishing accounts

- It's hard to distinguish whether an account is an EOA or contract
- if web3.eth.get_code returns nonzero, then it's a contract
  - if it returns 0, it may still be a contract
- If you can find a public key corresponding to the account, then it's an EOA

Distinguishing accounts

- It's hard to distinguish whether an account is an EOA or contract
- if web3.eth.get_code returns nonzero, then it's a contract
  - if it returns 0, it may still be a contract
- If you can find a public key corresponding to the account, then it's an EOA
  - Transactions originating from the account will have a signature

Distinguishing accounts

- It's hard to distinguish whether an account is an EOA or contract
- if web3.eth.get_code returns nonzero, then it's a contract
  - if it returns 0, it may still be a contract
- If you can find a public key corresponding to the account, then it's an EOA
  - Transactions originating from the account will have a signature
- Some accounts cannot be distinguished:

Distinguishing accounts

- It's hard to distinguish whether an account is an EOA or contract
- if web3.eth.get_code returns nonzero, then it's a contract
  - if it returns 0, it may still be a contract
- If you can find a public key corresponding to the account, then it's an EOA
  - Transactions originating from the account will have a signature
- Some accounts cannot be distinguished:
  - Find an address where you *could* deploy a contract

Distinguishing accounts

- It's hard to distinguish whether an account is an EOA or contract
- if web3.eth.get_code returns nonzero, then it's a contract
  - if it returns 0, it may still be a contract
- If you can find a public key corresponding to the account, then it's an EOA
  - Transactions originating from the account will have a signature
- Some accounts cannot be distinguished:
  - Find an address where you *could* deploy a contract
  - Send ETH to that address (that address will be created)

Distinguishing accounts

- It's hard to distinguish whether an account is an EOA or contract
- if web3.eth.get_code returns nonzero, then it's a contract
  - if it returns 0, it may still be a contract
- If you can find a public key corresponding to the account, then it's an EOA
  - Transactions originating from the account will have a signature
- Some accounts cannot be distinguished:
  - Find an address where you *could* deploy a contract
  - Send ETH to that address (that address will be created)
  - At that point, it looks like an EOA (no TXs originating from the address)

Distinguishing accounts

- It's hard to distinguish whether an account is an EOA or contract
- if web3.eth.get_code returns nonzero, then it's a contract
  - if it returns 0, it may still be a contract
- If you can find a public key corresponding to the account, then it's an EOA
  - Transactions originating from the account will have a signature
- Some accounts cannot be distinguished:
  - Find an address where you *could* deploy a contract
  - Send ETH to that address (that address will be created)
  - At that point, it looks like an EOA (no TXs originating from the address)
  - You can later deploy a contract to that address

The EVM is not Ethereum

- The EVM is completely separate from the consensus mechanism

EVM chains

Penn Engineering
UNIVERSITY of PENNSYLVANIA