LECTURE 04D

<EVM>

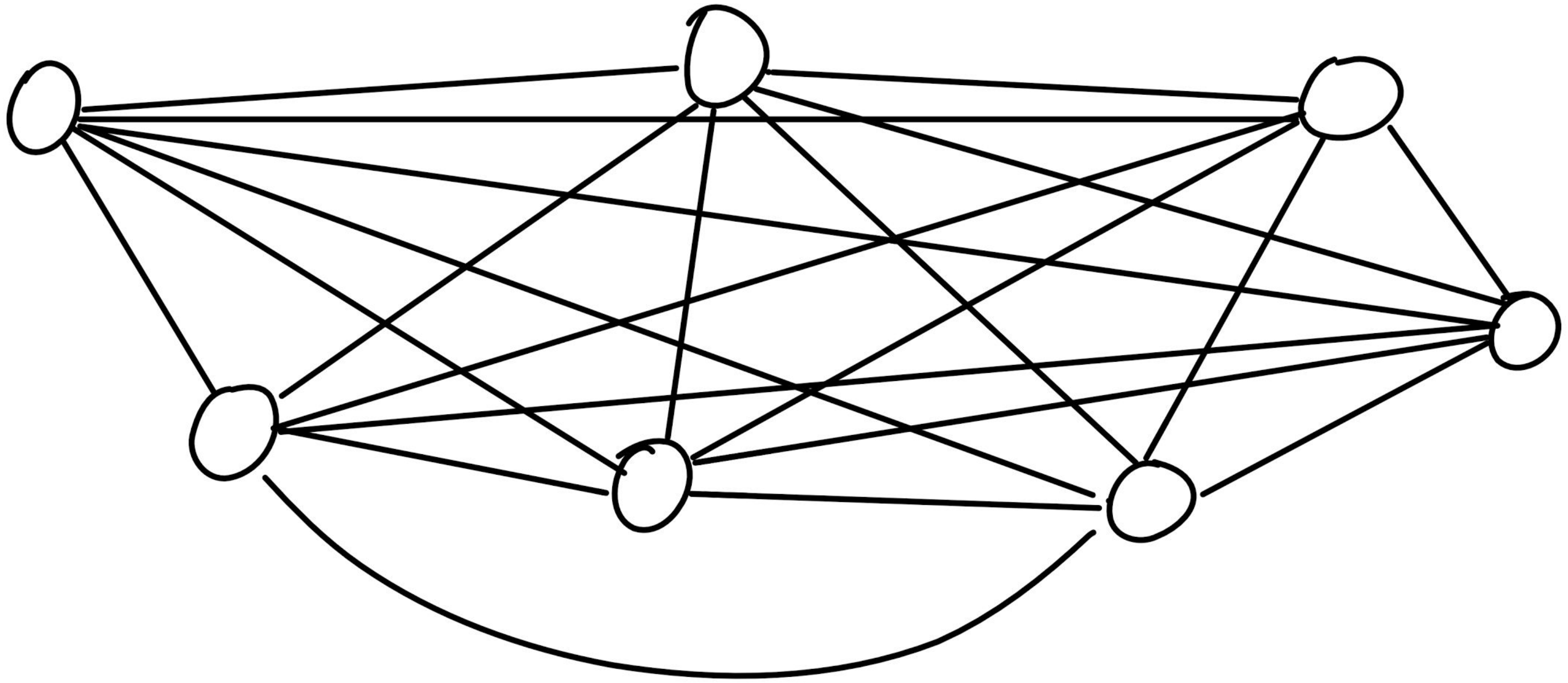
Nadir Akhtar

BLOCKCHAIN
AT BERKELEY

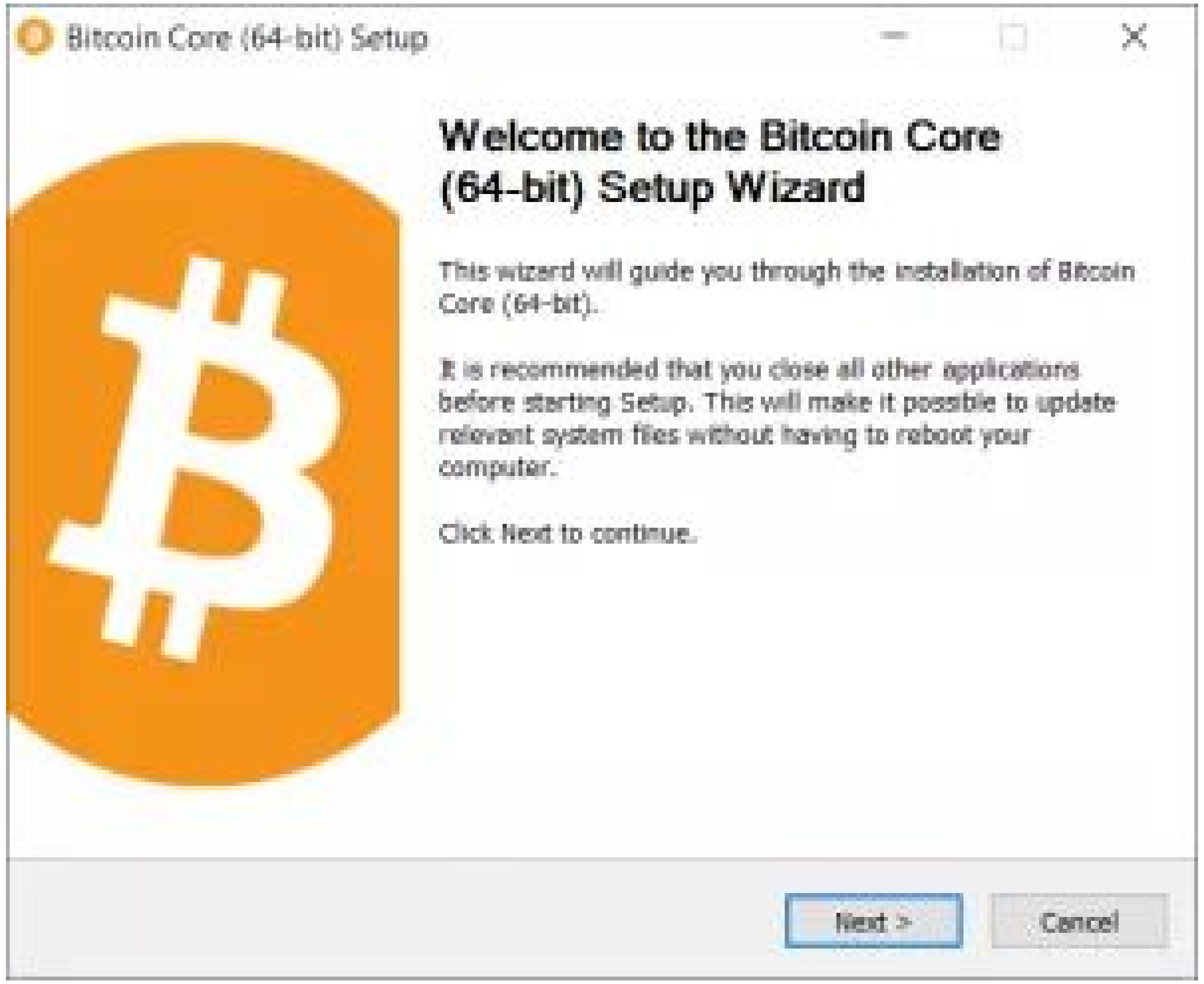
“WHO AM I?”

Heck if I know

- A blockchain network: →
- **Legend:**
  - ***Nodes:*** servers storing blockchain data
  - ***Edges:*** invisible connection over which peers send and receive data



"99% COMPLETE..."



Welcome to the Bitcoin Core (64-bit) Setup Wizard

This wizard will guide you through the installation of Bitcoin Core (64-bit).

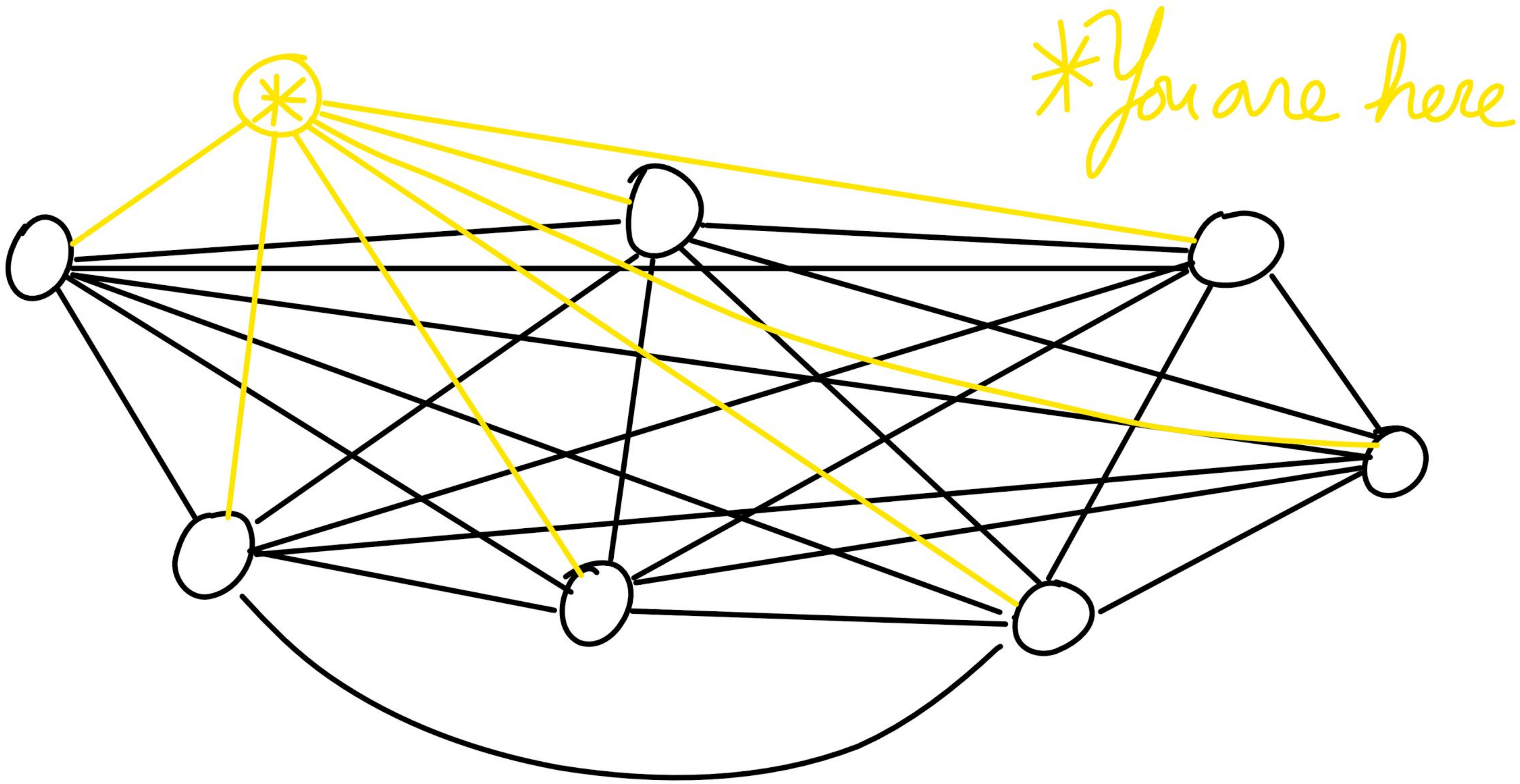
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

*low-res aesthetic intentional

"WHO AM I?"

What matters is that you know

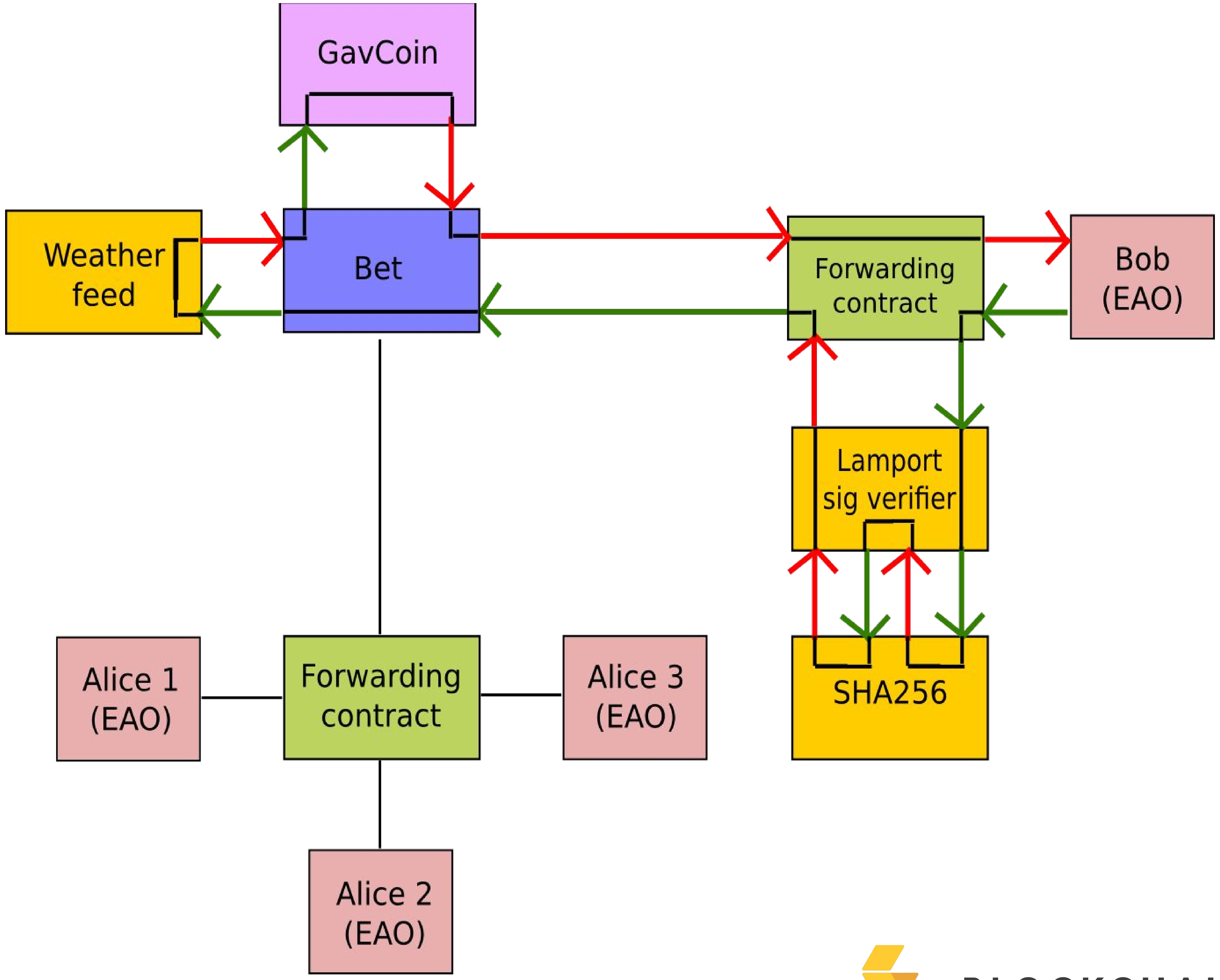


WHAT'S ETHEREUM?

AUTOMATING THE BLOCKCHAIN

- **Smart Contracts**
  - Another type of account; not a user account (aka externally owned account (EOA))
  - Code with addresses and balances living on the Ethereum blockchain
    - Cannot be edited once deployed
    - Requires no centralization for execution
  - Can call each other (and itself, directly or indirectly)
  - Run with "gas" to prevent infinite loops
  - Use cases: Store and maintain data, create multisig wallet (BitGo), manage contract or user relationships, serve as software library, act as "forwarding contract"

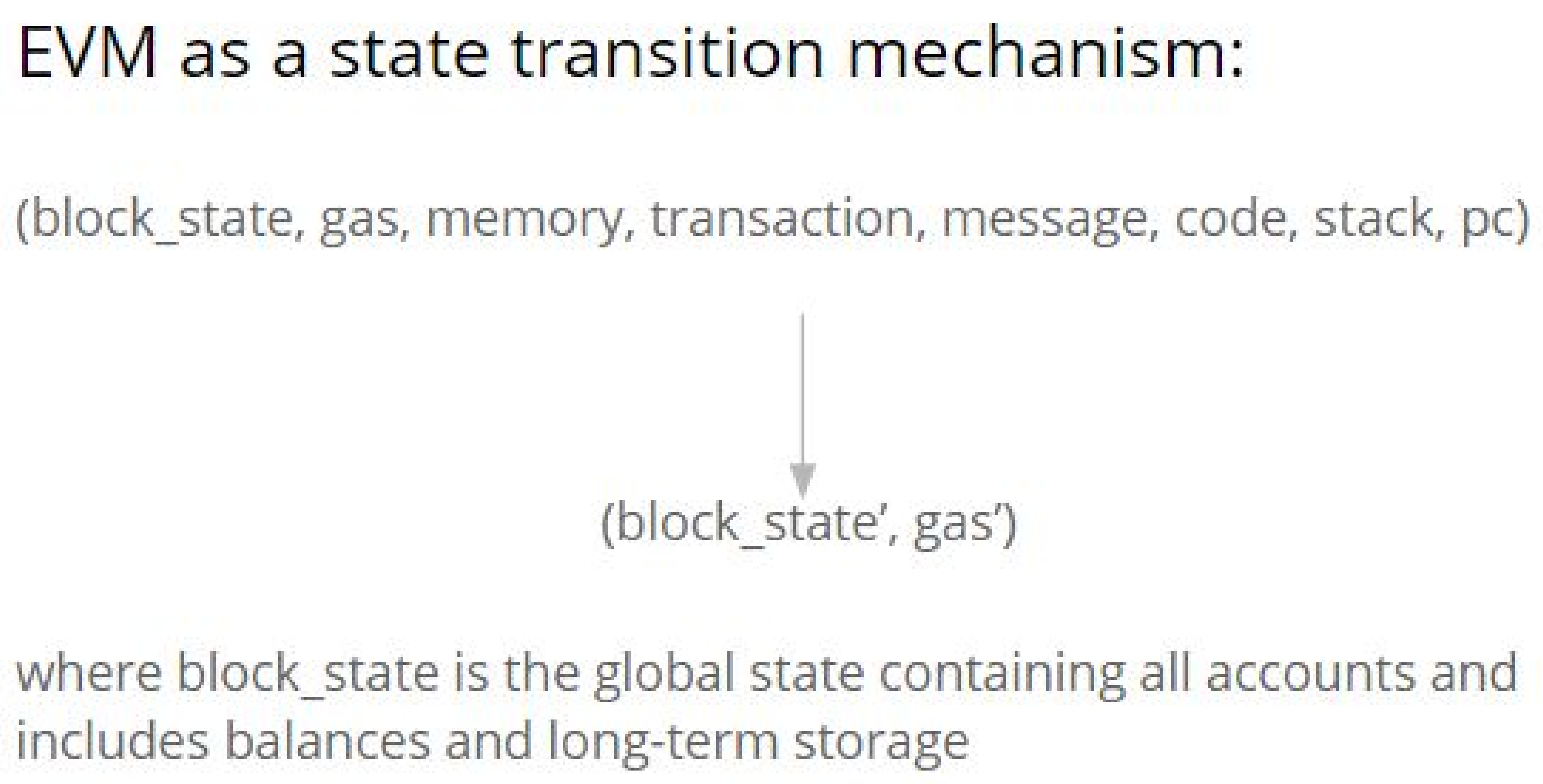
https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial



HOW DOES ETHEREUM RUN CODE?

“Is it magic?”

- **EVM (Ethereum Virtual Machine)**
  - “[A] large decentralized computer containing millions of objects, called ‘accounts’, which have the ability to maintain an internal database, execute code and talk to each other.”
  - Virtual machine: “a software implementation of a machine (i.e. a computer) that executes programs like a physical machine” (think Java VM) (http://www.cubrid.org/blog/dev-platform/understanding-jvm-internals/)
  - Runs bytecode with recursive message-sending functionality on every node on the network to verify blocks
  - ***memory:*** byte-array of infinite size
  - ***program counter:*** pointer to current instruction



EVM as a state transition mechanism:

(block_state, gas, memory, transaction, message, code, stack, pc)

↓

(block_state', gas')

where block_state is the global state containing all accounts and includes balances and long-term storage

```
PUSH1 0 CALLDATALOAD SLOAD NOT PUSH1 9 JUMPI STOP JUMPDEST PUSH1 32 CALLDATALOAD PUSH1 0 CALLDATALOAD SSTORE
```

EVM'S UPBRINGING

“Do we have to know all this?”

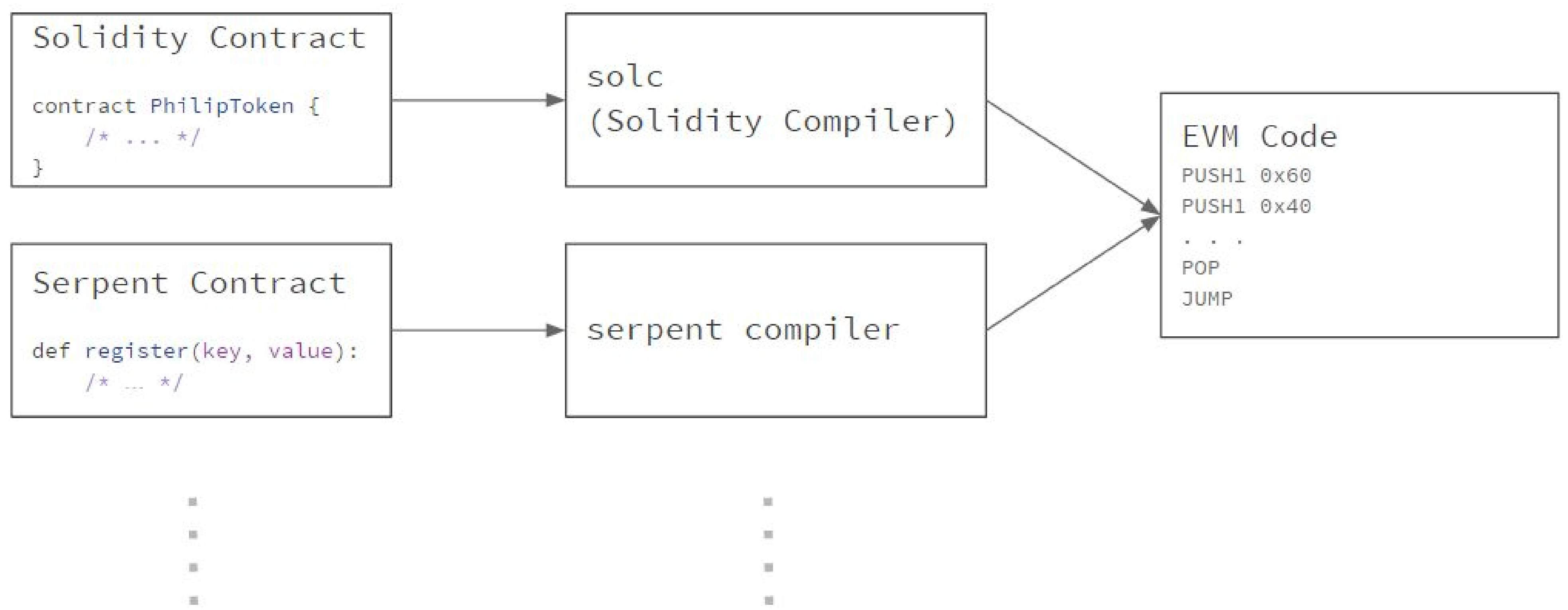
- **EVM Design Goals:**
  - ***Simplicity:*** op-codes should be as low-level as possible. The number of op-codes should be minimized.
  - ***Determinism:*** The execution of EVM code should be deterministic; the same input state should always yield the same output state.
  - ***Space Efficiency:*** EVM assembly should be as compact as possible
  - ***Optimization:*** Data sizes tend to be bigger; optimize for these larger addresses, read block and transaction data, interact with state, etc
    - easily handle 20-byte addresses (typical public keys) and custom cryptography with 32-byte values, modular arithmetic used in custom cryptography
  - ***Security:*** it should be easy to come up with a gas cost model for operations that makes the VM non-exploitable

“DO I HAVE TO WRITE CODE WITH OPCODES?”

“That seems dumb.”

- **No! (unless you want to, I suppose)**
  - Solidity and Serpent: examples of higher level languages which compile down to same bytecode
- **Solidity**
  - Most similar to JavaScript
  - Designed specifically for EVM
  - Officially supported, unlike Serpent and LLL
  - “Statically typed, supports inheritance, libraries, and complex user-defined types” (https://solidity.readthedocs.io/en/develop/)
  - Most developed language and compiler
- **Compiles into EVM**
  - Sent transactions contain EVM bytecode

EVM Code Compilation



Solidity Contract

```
contract PhilipToken {
    /* ... */
}
```

Serpent Contract

```
def register(key, value):
    /* ... */
```

EVM Code

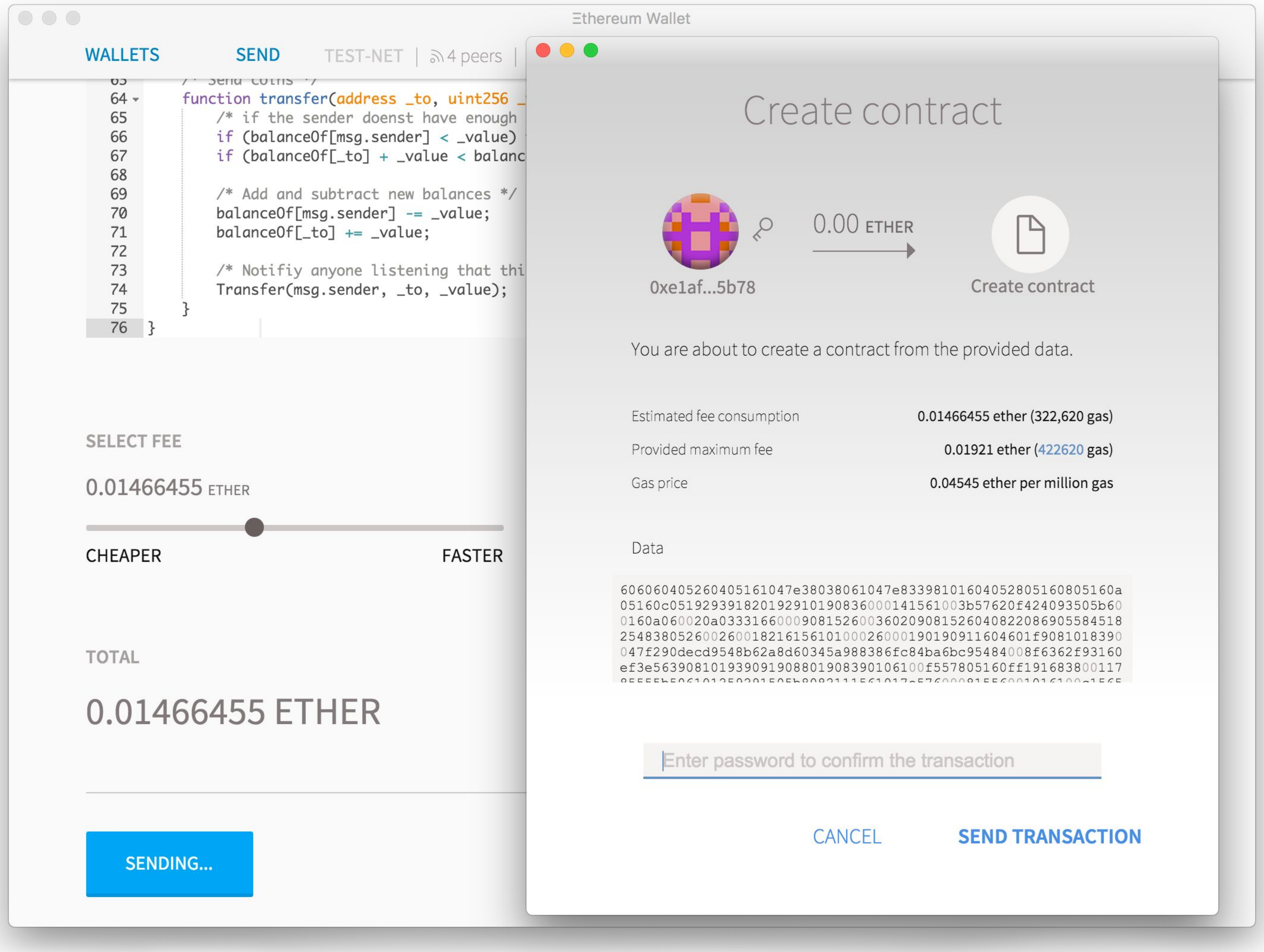
```
PUSH1 0x60
PUSH1 0x40
. . .
POP
JUMP
```

EVM GAS

No, nothing to do with bowel movements.

- **Gas: Ethereum’s first defense against attacks**
  - Infinite loops are dangerous
  - Mathematically impossible to predict infinite loops
  - Solution: make contract calls expensive; opcodes eat up money
    - Transaction specifies `startgas` and `gasprice`
      - Miners tend to confirm higher gas prices, similar to Bitcoin TX fees
  - Contracts, when calling other contracts, **pass gas** remaining from initial call
- **“Where does the gas go?”**
  - Consumed gas goes to miners
  - Case a) contract successfully executes:
    - Remaining gas is returned to sender
  - Case b) contract runs out of gas before completion or fails to complete
    - Network state reverts, gas is **not refunded**
  - Computationally complex code only allowed for high rollers

https://blog.ethereum.org/2015/12/03/how-to-build-your-own-cryptocurrency/



```
    function transfer(address _to, uint256 _
        /* if the sender doesnt have enough
        if (balanceOf[msg.sender] < _value)
        if (balanceOf[_to] + _value < balanc

        /* Add and subtract new balances */
        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;

        /* Notifiy anyone listening that thi
        Transfer(msg.sender, _to, _value);
    }
}
```

SELECT FEE

0.01466455 ETHER

TOTAL

0.01466455 ETHER

Create contract

You are about to create a contract from the provided data.

Estimated fee consumption	0.01466455 ether (322,620 gas)
Provided maximum fee	0.01921 ether (422620 gas)
Gas price	0.04545 ether per million gas

Data

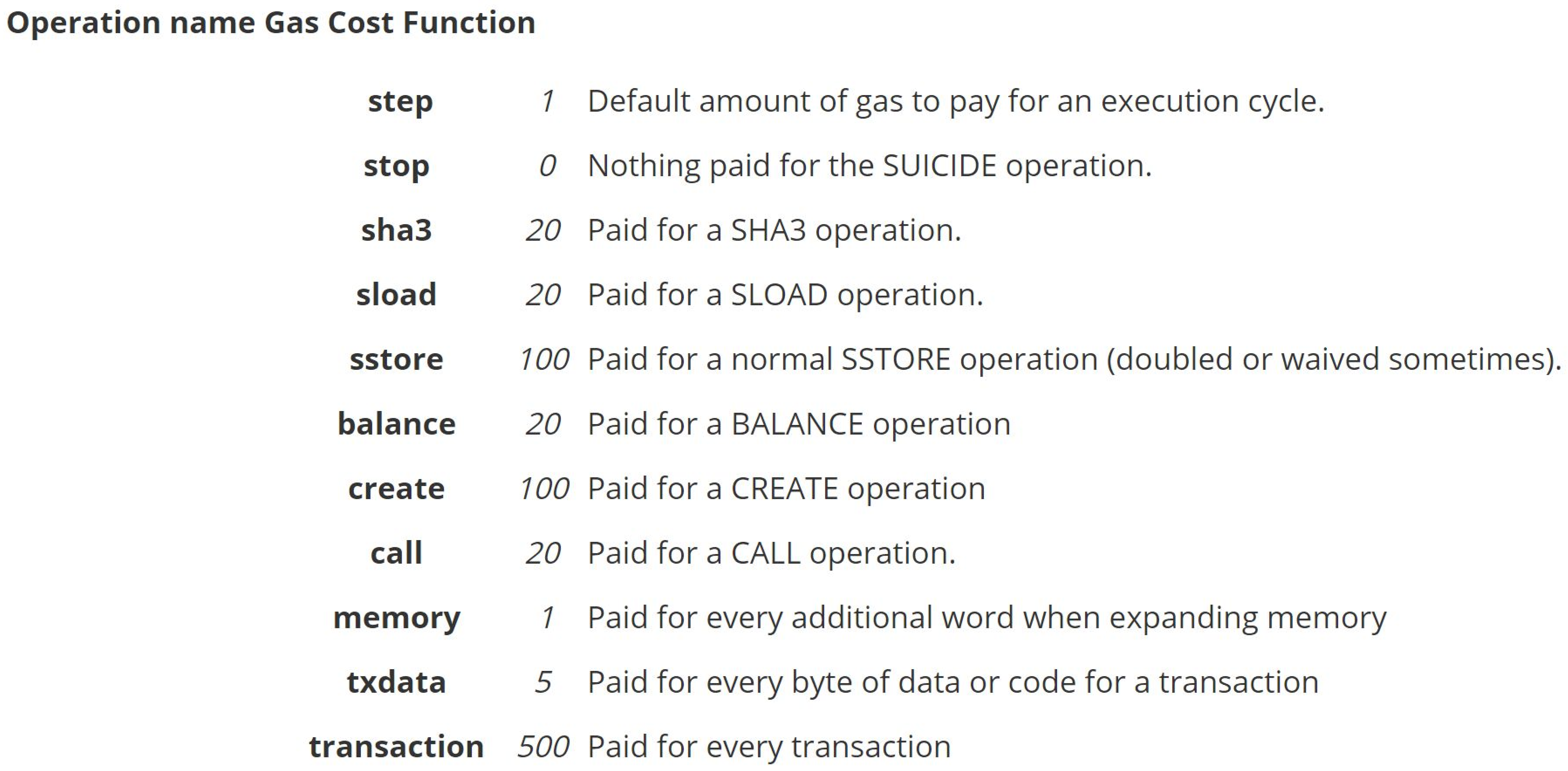
```
60606040526040516 1047e38038061047e83398101604052805160805160a
05160c051929391820192910190836000141561003b57620f424093505b60
0160a06002 0a0333166000908152600360209081526040822086905584518
2548380526002600182161561010002600019019091160460 1f9081018390
047f290decd9548b62a8d60345a988386fc84ba6bc95484008f6362f93160
ef3e563908101939091908801908390106100f557805160ff191683800117
```

Enter password to confirm the transaction

OPCODE GAS USAGE

EXAMPLES

https://www.cryptocompare.com/coins/guides/what-is-the-gas-in-ethereum/



Operation name	Gas Cost	Function
step	*1*	Default amount of gas to pay for an execution cycle.
stop	*0*	Nothing paid for the SUICIDE operation.
sha3	*20*	Paid for a SHA3 operation.
sload	*20*	Paid for a SLOAD operation.
sstore	*100*	Paid for a normal SSTORE operation (doubled or waived sometimes).
balance	*20*	Paid for a BALANCE operation
create	*100*	Paid for a CREATE operation
call	*20*	Paid for a CALL operation.
memory	*1*	Paid for every additional word when expanding memory
txdata	*5*	Paid for every byte of data or code for a transaction
transaction	*500*	Paid for every transaction

```
PUSH1 0 CALLDATALOAD SLOAD NOT PUSH1 9 JUMPI STOP JUMPDEST PUSH1 32 CALLDATALOAD PUSH1 0 CALLDATALOAD SSTORE
```

CENTRALIZED VS DECENTRALIZED COMPUTING

IT’S NOT ABOUT THE EFFICIENCY

- **Centralized:**
  - Cheaper.
    - No need to do the same operation on thousands of nodes.
  - Faster.
    - Information updates much more quickly than any distributed network.
  - Single point of failure.
    - Not as open to public regulation or verification, or as robust against attacks or manipulation.
- **Decentralized:**
  - Expensive.
    - Smart contract calls no longer of negligible expense
    - Much more time consumed for entire network to update
  - Secure.
    - Validated by a whole community.
  - Transparent.
    - Anyone can view or participate - including you!

LECTURE 04D

</EVM>

Nadir Akhtar

BLOCKCHAIN

AT BERKELEY