Ethereum

Introduction

Leander Jehl

Ethereum

Overview

Ethereum is a Proof of Work blockchain that uses several of the improvements discussed previously.

- Ethereum uses 12 sec block delay.
- Different P2P network.
- Ethereum uses a different Proof of Work function to protect agains ASICs.
- Ethereum uses uncles.
- Ethereum uses the GHOST rule, instead of longest chain rule.

Similar to Bitcoin, Ethereum uses *hashes of a public key as address*, and signatures for authentication.

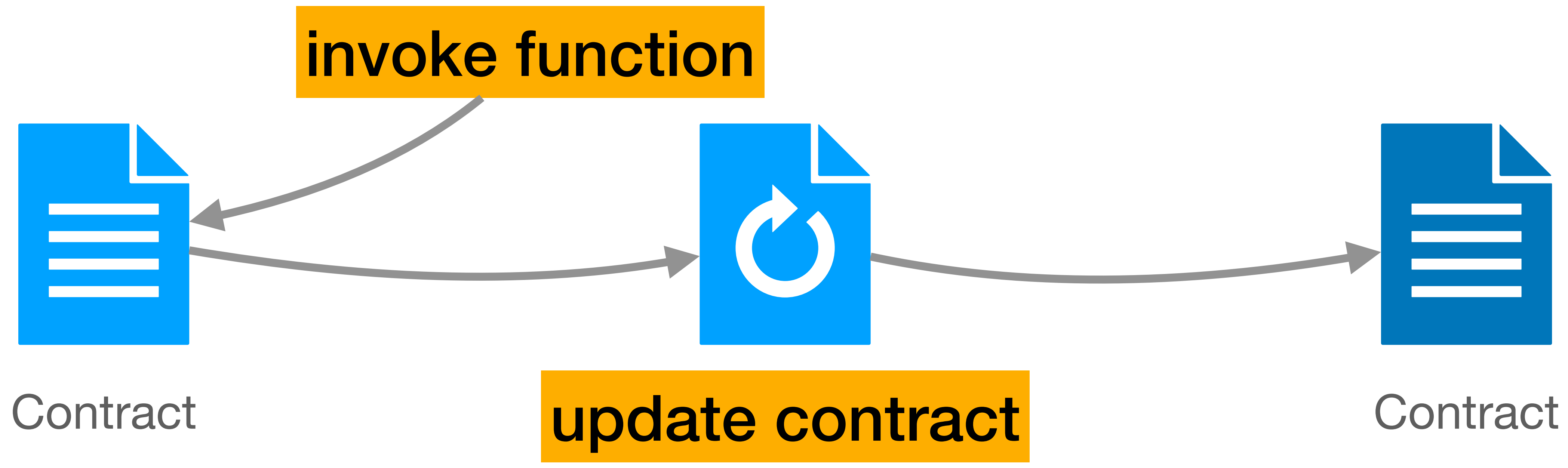
Ethereum has a cryptocurrency, Ether.

Ethereum

Smart Contract code

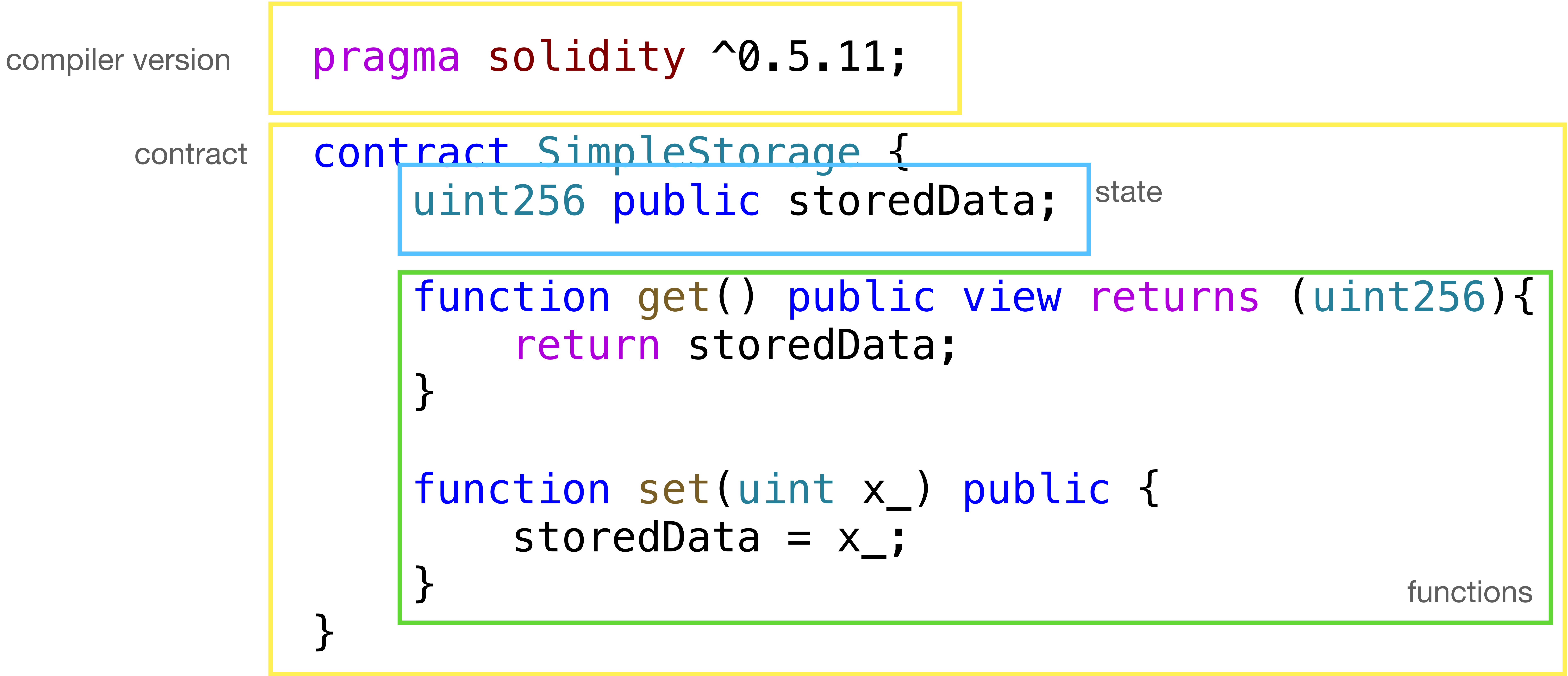
A contract is like a object from OOP, with fields and methods

- variables containing state (stored in account, mutable)
- functions



Ethereum

Example: Simple Storage



```
pragma solidity ^0.5.11;

contract SimpleStorage {
    uint256 public storedData;

    function get() public view returns (uint256){
        return storedData;
    }

    function set(uint x_) public {
        storedData = x_;
    }
}
```

Ethereum

Example: Simple Storage

Simple online IDE: https://remix.ethereum.org/

Fun tutorial: https://cryptozombies.io/

- Constructors
- Basic types and collections
- Visibility (private, public)
- Inheritance
- Modifiers (view, pure)

Ethereum

Example: Simple Storage

Who can invoke functions?

- any user

Who can view values?

- anyone

Who can change the code?

- noone

```
pragma solidity ^0.5.11;

contract SimpleStorage {
    uint256 public storedData;

    function get() public view returns (uint256){
        return storedData;
    }

    function set(uint x_) public {
        storedData = x_;
    }
}
```

Ethereum

Smart Contract code

Smart contract code is immutable and public

- Anyone can trust smart contract (if it is not too complex)
  - No need to trust the creator of the contract
- No one can fix bugs in the contract
- Anyone can find and exploit bugs in the contract

Ethereum

Smart Contract code

- Assembly for Ethereum Virtual machine (EVM)
- Compiled from higher level language (Solidity)
- Stored in account (codeHash)

Ethereum

Accounts

Ethereum uses accounts instead of UTXO.
Thus the state of Ethereum contains for every account:

- **address:** e.g. pub-key hash
- **balance**: amount of Ether the address owns
- **nonce**: sequence number of last transaction sent from this account
- **storage root**: *only for non-user accounts (contract account)*
- **code hash:** *only for non-user accounts (contract account)*

Ethereum

Accounts

Smart Contracts are also represented as accounts.
A contract account has:

- **address:** *e.g. hash from creator address & creation trancation nonce*
- **balance**: amount of Ether the address owns
- **nonce**: number of other contract created by this contract
- **storageRoot**: hash of data stored in this contract
- **codeHash:** hash of the code of this contract

Ethereum

Accounts

In a contract written in Solidity, you can access:

- The address of the current contract:

```
address contractaddress = address(this);
```

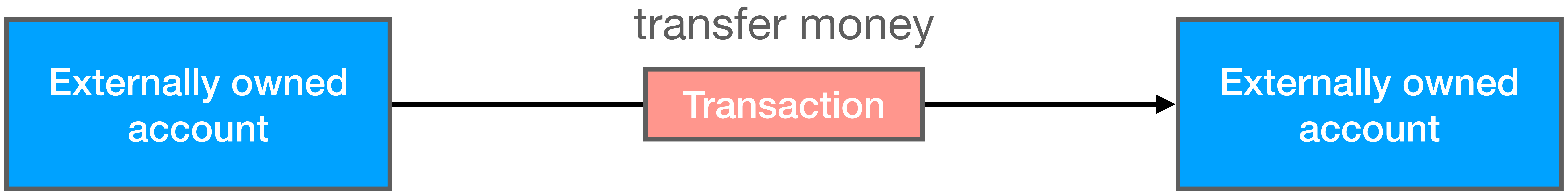
- The balance of the contract:

```
uint balance = contractaddress.balance;
```

Ethereum

Transactions and authenticaion

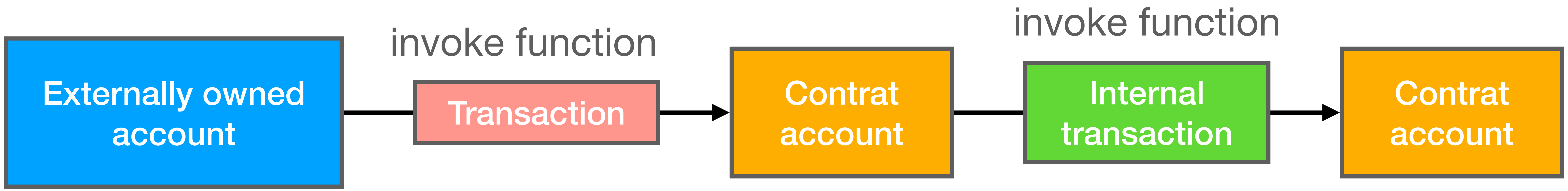
Transactions are used to transfer ether, invoke functions, and deploy new contracts.



Ethereum

Transactions and authenticaion

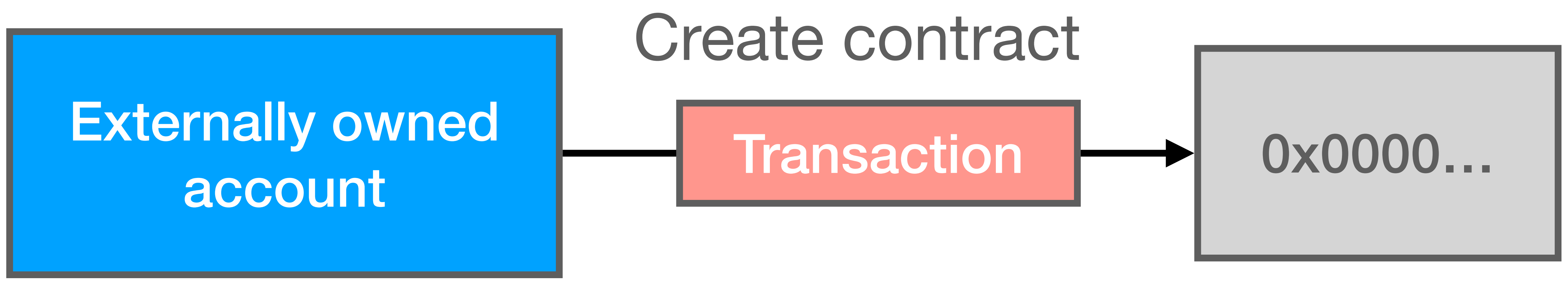
Transactions are used to transfer ether, invoke functions, and deploy new contracts.



Ethereum

Transactions and authenticaion

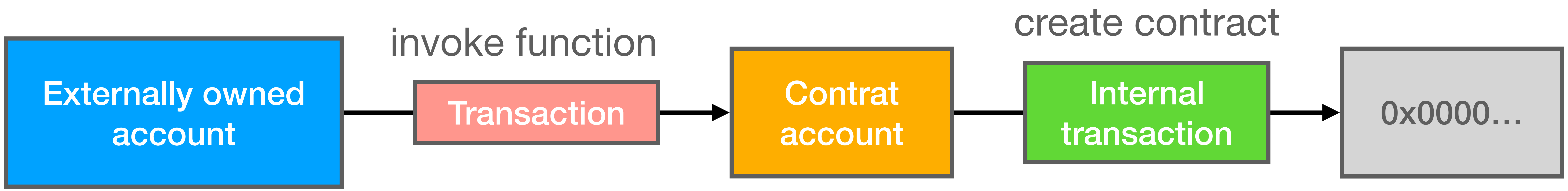
Transactions are used to transfer ether, invoke functions, and deploy new contracts.



Ethereum

Transactions and authenticaion

Transactions are used to transfer ether, invoke functions, and deploy new contracts.



Ethereum

Transactions and authenticaion

Transactions contain:

- *Nonce:* next sequence number for sender account
- *Gas price:* later
- *max gas:* later
- *Recipient:* destination Ethereum address
- *Value:* Amount of ether send to destination
- *Data:* Payload binay, e.g. function identifier and arguments
- *Signature:* Signature from sender, including his public key

Ethereum

Transaction validation

Transaction validation includes the following checks

- *Nonce:* is next sequence number for sender account
- Sender has sufficient balance to pay value and fees
- Transaction is correctly signed

When autheticating users in smart contract, we can rely on transaction validation!
Use *msg.sender* to access address invoking transaction.

Ethereum

Solidity example

```
contract SimpleBank {
    mapping(address => uint) private balances;
    address public owner;

    // function SimpleBank() deprecated syntax for
    constructor() public {
        owner = msg.sender;
    }

    function deposit() public payable returns(uint) {
        balances[msg.sender] += msg.value;
        return balances[msg.sender];
    }

    function withdraw(uint withdrawAmount) public returns (uint remainingBal){
        if (balances[msg.sender] >= withdrawAmount){
            balances[msg.sender] -= withdrawAmount;
            // this throws an error if fails.
            msg.sender.transfer(withdrawAmount);
        }
        return balances[msg.sender];
    }

    function balance() view public returns (uint) {
        return balances[msg.sender];
    }
}
```

Ethereum

Solidity example

What happens if data is empty?

- Money transfered to account. Default function run.

What is *msg.sender* for internal transactions?

- address of sending contract
    - a contract can have money in our bank!

Ethereum

Solidity exceptions

If a smart contract throws an exception, or error, state is reverted.

Ethereum

Gas

How to pay transaction fees in Ethereum?

- all bytecode instructions have a cost specified in Gas
- transaction has fixed cost in Gas
- especially: storing values is expensive

Transactions specify *Gas price* and *Gas limit*

- *Gas price* is ether given per gas
- *Gas limit* is how much the transaction may spend at most

Ethereum

Gas

Why specific gas per instruction:

- An infinite loop will cost infinitely much gas -> avoid denial of service

What happens if you hit the *Gas limit*?

- Exception is thrown and transaction reverted.
- Gas is still payed!

Which transactions are included?

- Miners will include transactions offering the highest gas price.