Blockchains from a Distributed Computing Perspective

Gabriela Vargas, Dalia Camacho

Contents

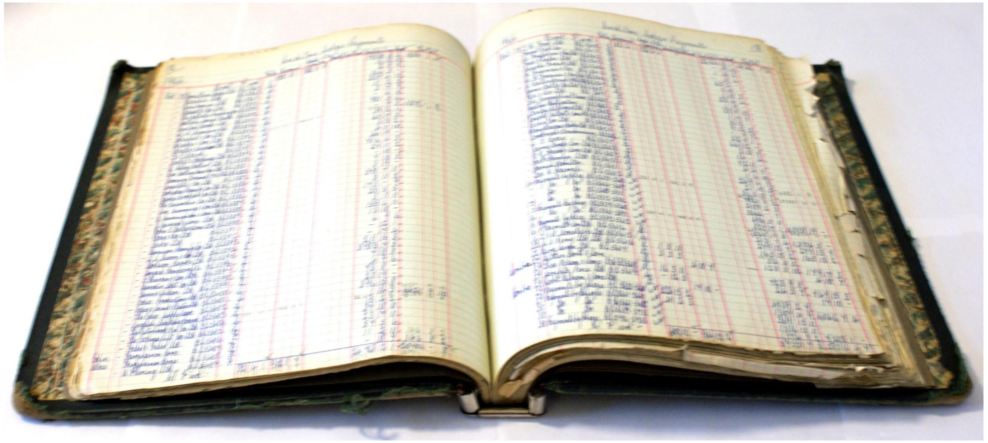
Contents

The Ledger Abstraction

The Ledger Abstraction

Log of transactions that is:

- Indelible
- Append-only
- Public
- Accessible to all parties
- Tamper-proof

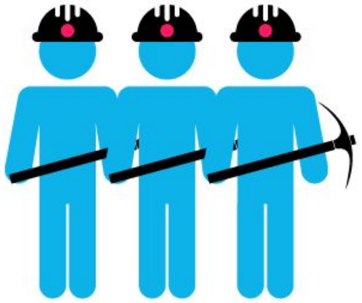


Blockchain ledger precursors

Blockchain Ledger precursors

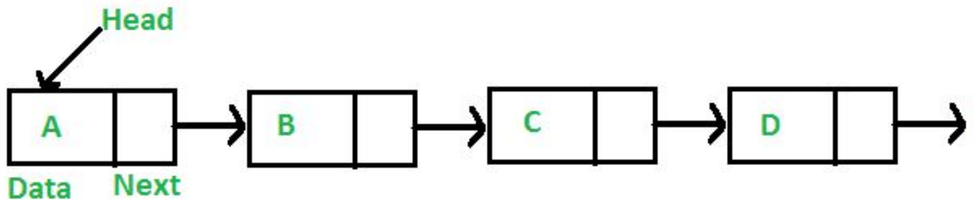
One document at a time

Pool of documents



Miners propose a document each to save in ledger

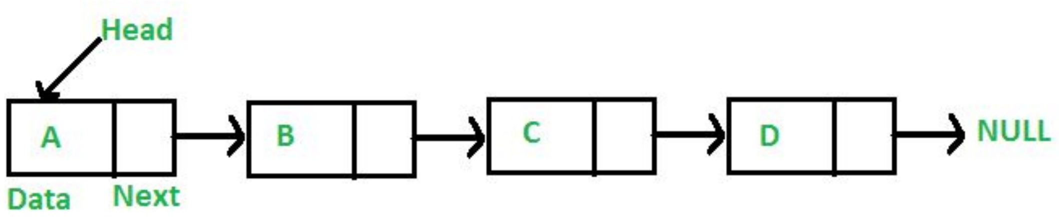
Document selected by consensus is stored in ledger



Ledger

Blockchain Ledger precursors

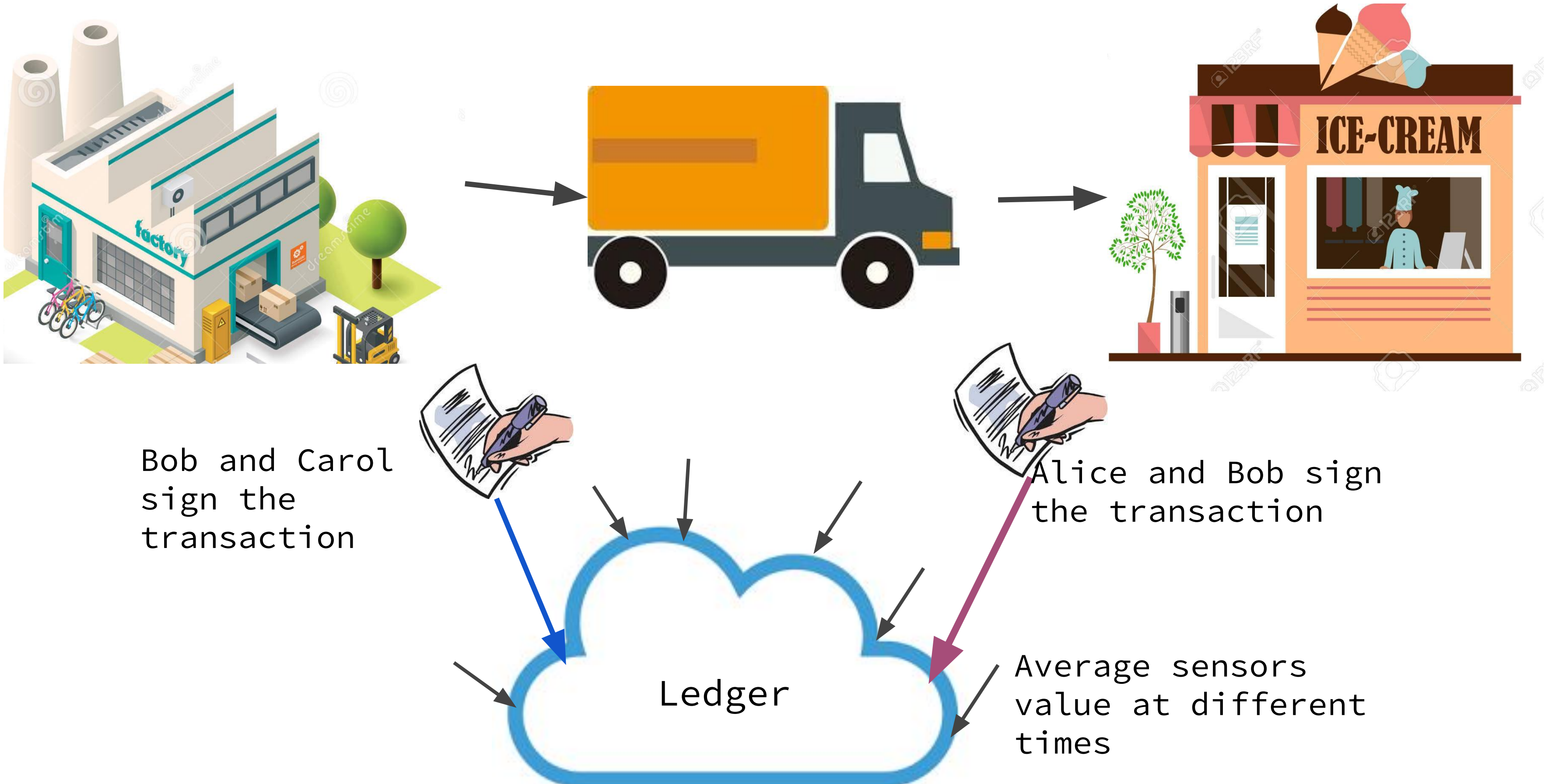
- Consider the ledger as a simple linked list



- Transactions to be written in the ledger are placed on a shared pool.
- Miners select the next transaction to be written into the ledger through a **consensus protocol**.
    - Each miner proposes which transaction of data to append to the layer and one of the proposed transactions is selected to append next onto the ledger.

Private Blockchain Ledgers

Private Blockchain


factory
ICE-CREAM
Bob and Carol sign the transaction
Alice and Bob sign the transaction
Ledger
Average sensors value at different times

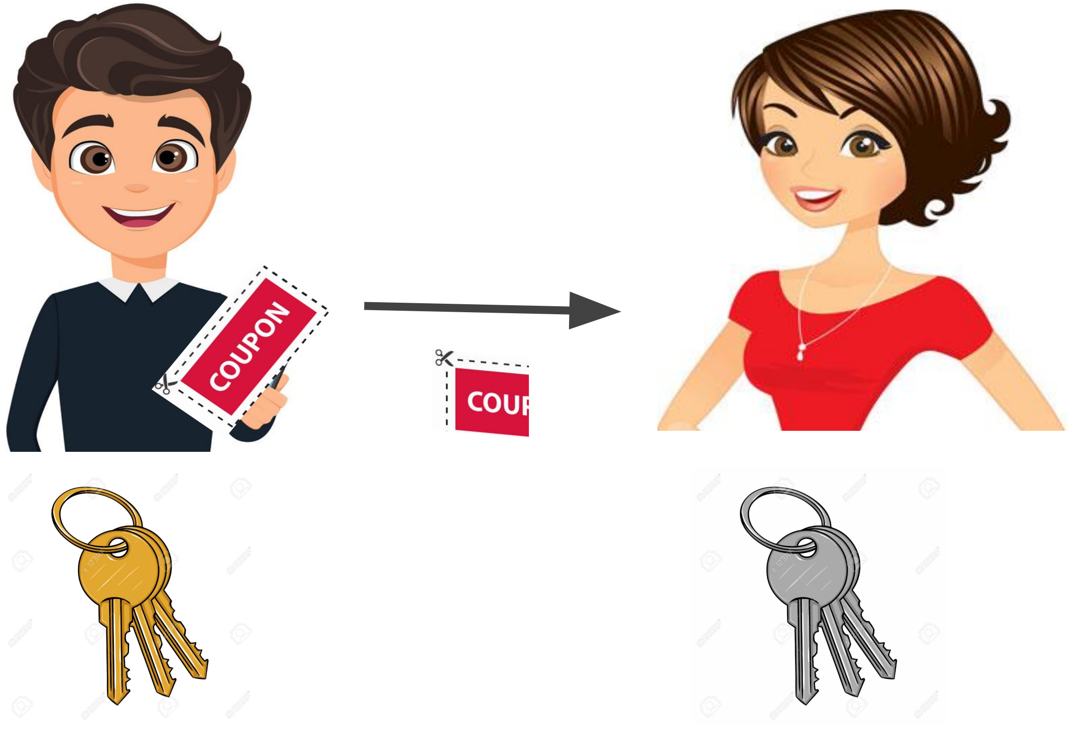
Private Blockchain

- Only the participants (including sensors) can write on the blockchain.
- Every participant is protected.
- When participants or sensors write on the blockchain the new blocks are timestamped by defining a hash that connects them to the previous block.

Public Blockchain Ledgers

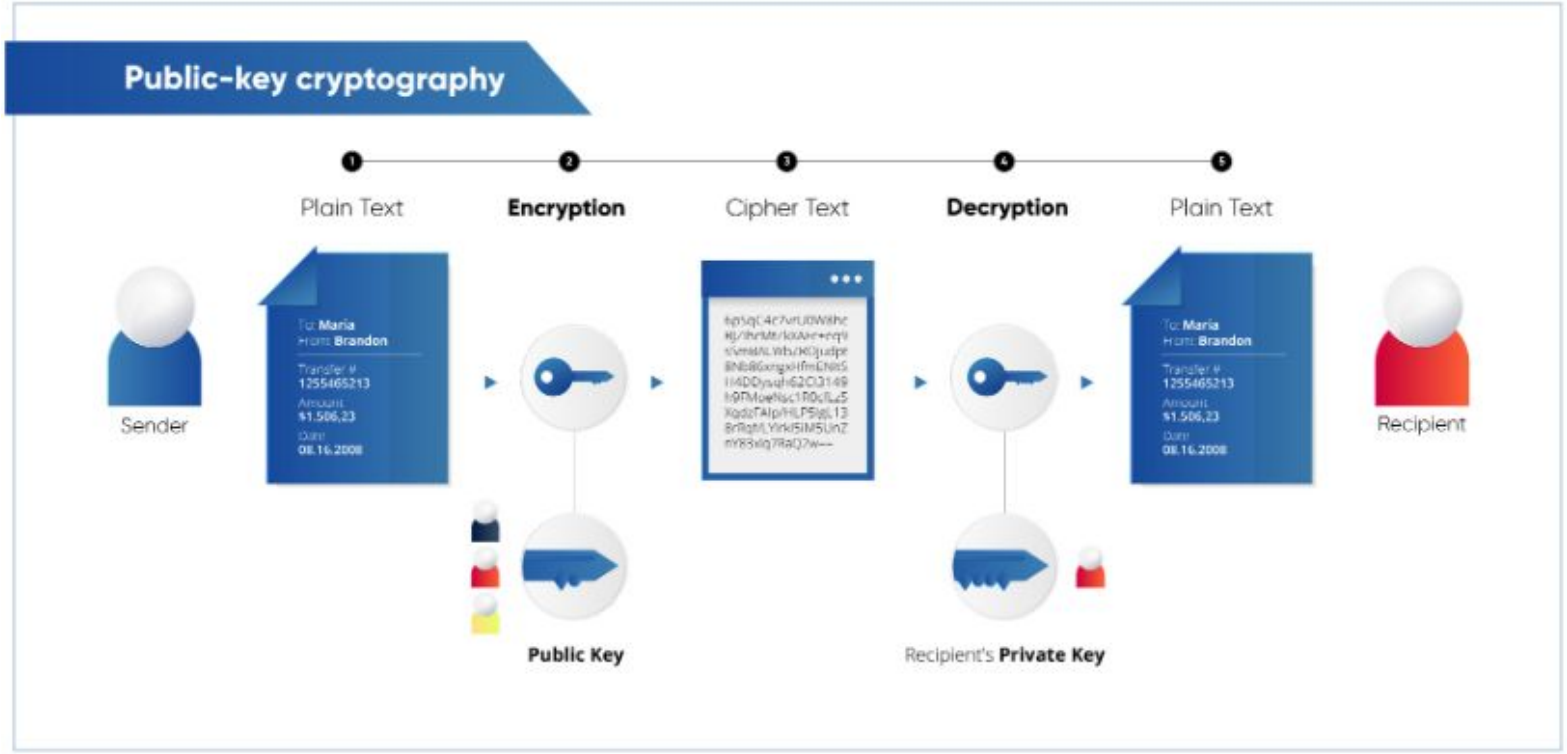
Public Blockchain Ledgers

Public Blockchain Ledgers



- **Private Key:** Confers ownership
- **Public Key:** Proofs ownership

Public and Private keys

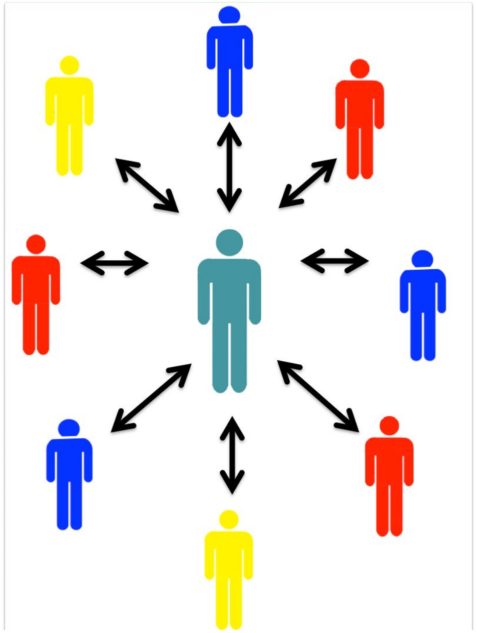


Two-tiered box analogy

Public Blockchain Ledgers

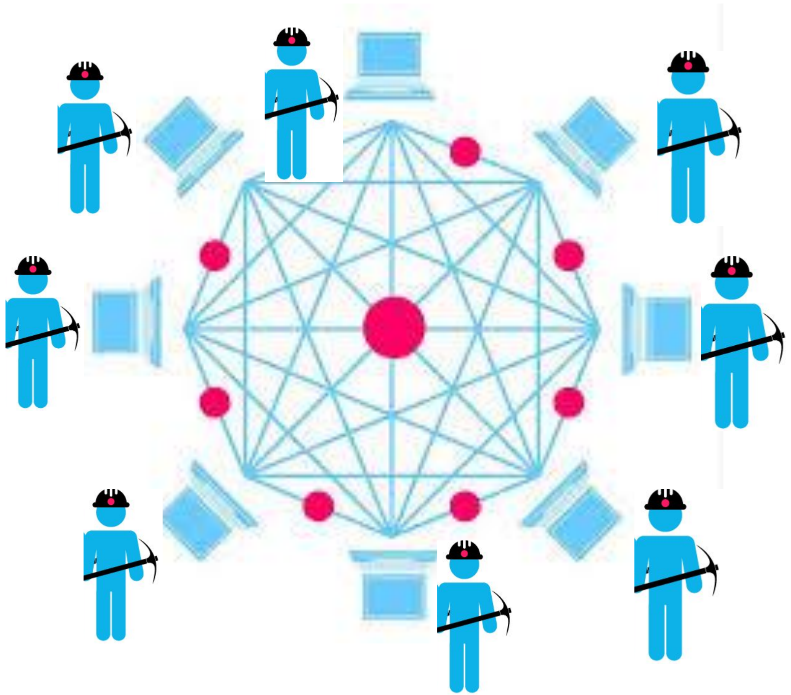
TRUST

Centralized



VS

Distributed



Public Blockchain Ledgers

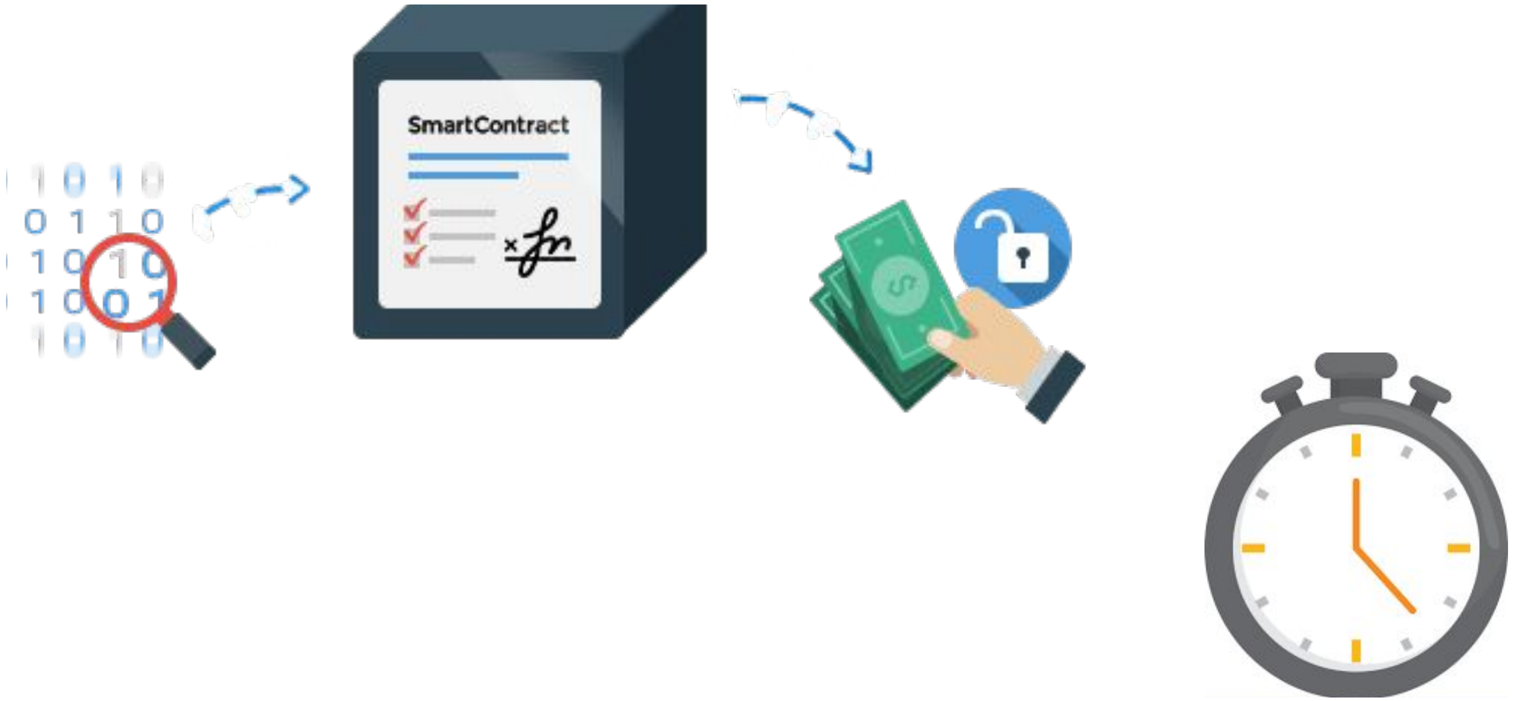
Proof of work

Smart Contracts

Functionality

- Smart contracts add functionality to blockchain ledgers.
- A hashlock h prevents an asset from being transferred until the contract receives a matching secret *s*, where *h=H(s)*, for *H* a cryptographic hash function.
- Timelock *t* prevents an asset from being transferred until a specified future time *t*.

Functionality

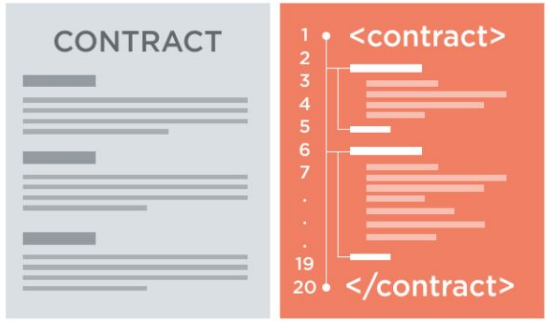


Smart Contracts as Objects

Components

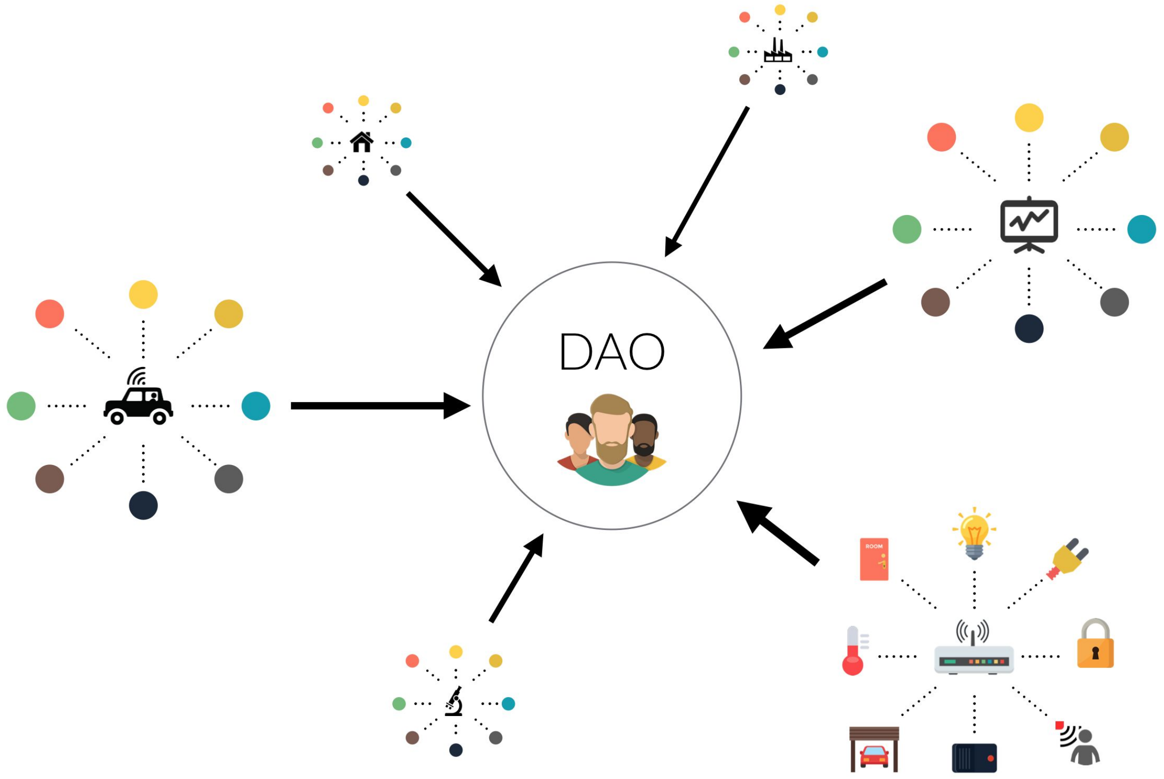
States

Constructors



Functions

Smart contracts as monitors



Smart contracts as monitors

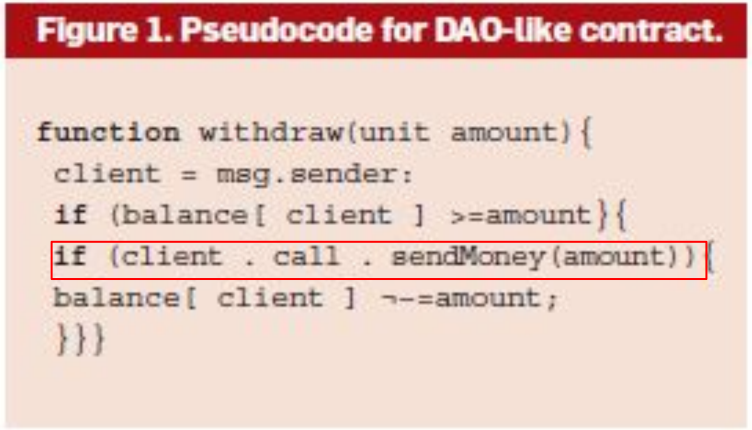


Figure 1. Pseudocode for DAO-like contract.

```
function withdraw(unit amount){
 client = msg.sender:
 if (balance[ client ] >=amount}{
 if (client . call . sendMoney(amount)){
 balance[ client ] ¬-=amount;
 }}}
```

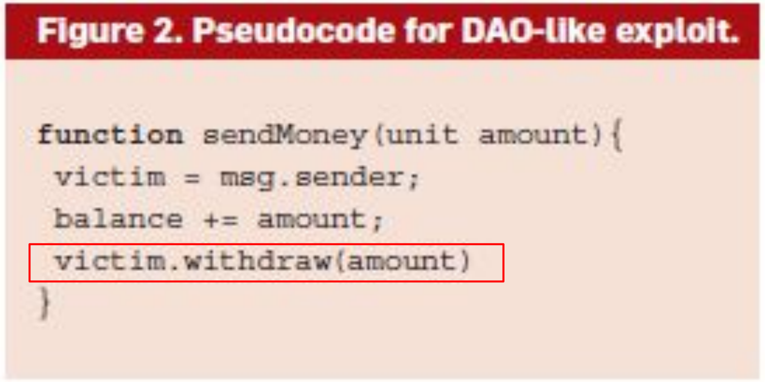


Figure 2. Pseudocode for DAO-like exploit.

```
function sendMoney(unit amount){
 victim = msg.sender;
 balance += amount;
 victim.withdraw(amount)
}
```

Smart Contracts as read-modify-write-operations

Smart Contracts as read-modify-write-operations

By using smart contracts one can launch a new token by **initial coin offerings** under the ERC20 standard.

The functions that correspond to an ERC20 contract are:

- **approve** in which the maximum amount of tokens someone can get is established.
- **allowance** is used to see the amount of tokens someone has.
- **transferFrom** it transfers an amount of tokens from one user to another

Smart Contracts as read-modify-write-operations

If function modifiers are not used correctly users may act maliciously.

This often occurs if a function is defined as public or external instead of making it private. If it is public anyone can alter the amount of tokens one can receive.