Smart contracts

Self-enforcing and executing contracts

What are smart contracts?

“A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs.”

-Szabo 1994

“Smart contracts are computer protocols that facilitate, verify, or enforce the negotiation or performance of a contract, or that make a contractual clause unnecessary.”

-Wikipedia, April 2017

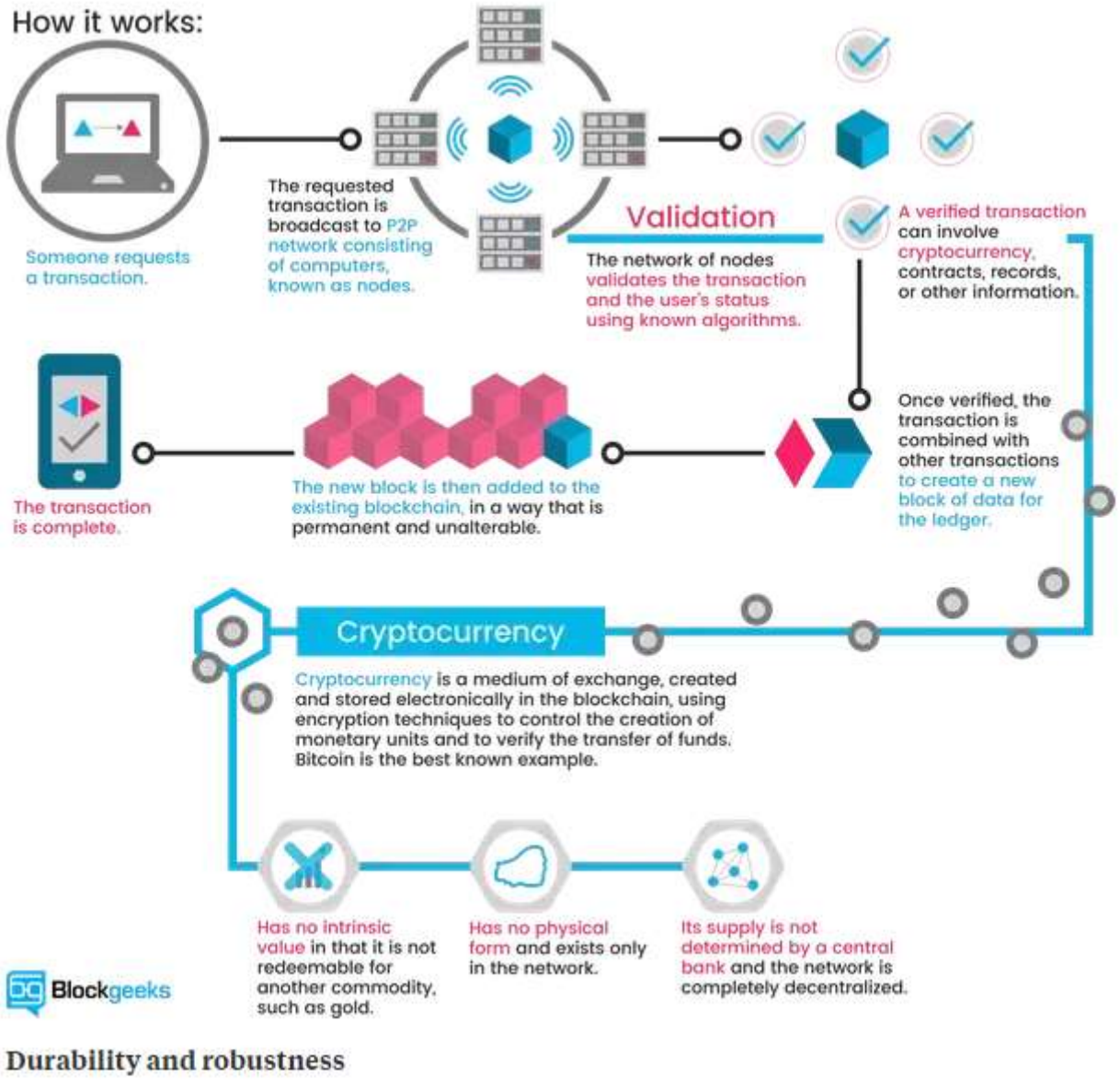
What are smart contracts?

- Pre-written logic (code)
- The logic is stored and replicated on a blockchain
- The state is stored and replicated on a blockchain
- Executed by a network of computers
- May result in ledger updates

In essence

- Blockchains give us trustworthy storage
- Smart contracts give us trustworthy calculations

How is the storage trustworthy?



Durability and robustness

1. Cannot be controlled by a single entity

2. No single point of failure

3. Data is embedded, by definition public*

4. It cannot be practically corrupted

* There are blockchains that provide features of anonymity and encrypted data. Zcash, Monero and Ethereum provides a “view key” functionality for this.

How is the logic trustworthy?

- The formal description of the logic is known to at least the participants of the contract
- The logic is deterministic and the consensus is powerful
  - Any deviance, be it to hardware errors or attempts to manipulate the chain, will be quickly outvoted
- The logic is stored on the blockchain, which makes the logic unalterable
- The state the logic is applied to is observable
- The integrity of the initial and subsequent states are guaranteed by the storage trustworthiness

Some of the potentials

- Automating manual tasks
  - Smart contracts are software. Software is typically faster than manual work
- Accuracy
  - Automated transactions are not only faster than manual work, but also less error prone
- Mitigates execution risk
  - Decentralised execution makes manipulation much harder, if not practically impossible
- Fewer intermediaries
  - Smart contracts reduces the need for third-party services
- Lower cost
  - Faster, less errors, less risk and fewer middle men translates to lower cost

What do they currently look like?

- Typically quite simple, objective logic
  - If current date equals payment date, and the payment has not been made, pay bond-holders coupon
  - Bond-holder determined by DLT, integrity of information is guaranteed by DLT
  - Interest component and notional part of the state are recorded in the DLT
  - All information is available to the participants
- For example, smart bonds by UBS
  - Built on Ethereum
  - Recreated the bond's issuance, interest calculation, bond payments and maturation process
  - No pre- and post-trade intermediaries were necessary

So what does the logic look like?

```
contract MyToken {
    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;

    /* Initializes contract with initial supply tokens to the creator of the contract */
    function MyToken(
        uint256 initialSupply
        ) {
        balanceOf[msg.sender] = initialSupply;              // Give the creator all initial tokens
    }

    /* Send coins */
    function transfer(address _to, uint256 _value) {
        if (balanceOf[msg.sender] < _value) throw;           // Check if the sender has enough
        if (balanceOf[_to] + _value < balanceOf[_to]) throw; // Check for overflows
        balanceOf[msg.sender] -= _value;                     // Subtract from the sender
        balanceOf[_to] += _value;                            // Add the same to the recipient
    }
}
```

https://www.ethereum.org/token

How can it be made simpler?

- Domain Specific Languages will come up, for example describing payments on a bond
- Likely similar to DSLs found in many institutions, e.g.

```
notional = 100000 USD;
coupon = 4.5%;
start_date = 2017/01/01;
end_date = 2027/01/01;
bond_holders = extract_bond_holders_from_blockchain(current_date());
if current_date() == end_date:
   pay bond_holders notional;
schedule = from start_date + 6M to end_date every 6M on calendar NY;
for date in schedule:
   if current_date() == date:
      pay bond_holders notional * coupon * 0.5 on date;
```

- Still not prose, but a lot better than regular code

Partial visibility

- Sometimes there is a need for keeping participants in a contract on a strict need-to-know basis
- Some blockchains allow for view keys – a participant may share this with another participant, and it unlocks the content of part of the contract
  - Any party to a trade will likely not want what positions he or she is taking in plain view all over the block chain.
  - Can reduce risk or perception of unfair treatment.

Difficulties, present and future

- What happens if there is a bug in the implementation of a contract?
- Who can be considered, and is willing, to be an “Oracle” for supplying reference data to a blockchain?
- Frequency of blockchain solutions is far slower than what we are used to
- When present, anonymity is a double-edged sword – how can we ensure we are dealing with a reputable counterparty?