Cassiopeia
House of Computer Science

www.cs.aau.dk

The blockchain

Michele Albano
Distributed and Embedded Systems Group
Aalborg University

AALBORG UNIVERSITET

Outline

- Crypto tools: hash, signatures, asymmetric crypto, Merkle trees
- Blockchain basics: hash pointers, consensus, proof of work
- Blockchain details: transactions, Bitcoin, what a user is, miners
- Basics on Ethereum smart contracts

Cryptographic hash function

H: X = $\{0,1\}^*$ -> Y = $\{0,1\}^L$

- with L fixed
  - Examples for L: 128/160/256/512 bits

- Informal property:
  - a small change in the input produces a completely different output

Cryptographic hash function

H: X = {0,1}* -> Y = $\{0,1\}^L$

- with L fixed
  - Examples for L: 128/160/256/512 bits

- Informal property:
  - a small change in the input produces a completely different output

- Security properties (collisions exist, but are hard to find):
  - Pre-image resistance: $\forall\, y \in Y$, it is hard to find $x \in X$ such that $\mathrm{H(x)} = \mathrm{y}$
  - Second pre-image resistance: given a particular $M \in X$ and thus $\mathrm{h} = H(M)$, it is hard to find M' with $\mathrm{H(M')} = \mathrm{h}$
  - Collision-resistance: it is hard to find $x_1 \in X$ and $x_2 \in X$, with $x_1 \neq x_2$ and $H(x_1) = H(x_2)$

Case study: Rock-paper-scissors

- Let us play over the internet
  - You play your rock / paper / scissors move
  - I play my move
  - One wins
- Ideally, we could play at the same time but my cam is not working, and there is some jitter on the audio
  - Or the rock-paper-scissors championship is today, and I have no 4g connection, so we play via email
- Well, tell me your move, and you can trust I will tell you mine!!!
- Or can you do something using hash functions?

Let us build a solution 1/3

- M – your move
- H – Hash function

- You tell me H(M)
- I will then tell you my move

- Does this work?

Let us build a solution 2/3

- M – your move
- H – Hash function
- S – random string computed by you
- || - concatenation

- You tell me H(M||S)
- I will then tell you my move

- Does this work?
- What if I don’t trust you?
  - Maybe you found a collision, after looking for it for 10 years.

Let us build a solution 3/3

- M – your move
- H – Hash function
- S – random string computed by you
- || - concatenation
- T – a nonce / a timestamp / a string I sent you

- You tell me H(T||M||S)
- I will then tell you my move

SHA: Secure Hash Algorithm

- The Secure Hash Algorithm is a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS)

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Rounds	Bitwise operations	Security (bits)
SHA-1 **FIPS 180**		160	160	512	80	and, or, add, xor, rot	Theoretical attack (2^{61})
SHA-2 **FIPS 180**	SHA-224	224	256 (8×32)	512	64	and, or, xor, shr, rot, add	112
	SHA-256 Bitcoin	256					128
	SHA-384	384	512 (8×64)	1024	80	and, or, xor, shr, rot, add	192
	SHA-512	512					256
	SHA-512/224	224					112
	SHA-512/256	256					128
SHA-3 **FIPS 202**	SHA3-224	224	1600 ($5 \times 5 \times 64$)	1152	24	and, xor, rot, not	112
	SHA3-256 Ethereum (Keccak 256)	256		1088			128
	SHA3-384	384		832			192
	SHA3-512	512		576			256

https://en.wikipedia.org/wiki/Secure_Hash_Algorithm

AALBORG UNIVERSITET

Let’s use hashes!

- I want to use hash functions to sign a large data structure
  - I compute the hash h
  - I send you the hash using a secure/expensive channel
  - I send you data using normal channel
  - If your H(data) is not h, you know it is bad!
- You would have to ask for the whole data once again

Let's use hashes!

- I want to use hash functions to sign a large data structure
  - I compute the hash h
  - I send you the hash using a secure/expensive channel
  - I send you data using normal channel
  - If your H(data) is not h, you know it is bad!
- You would have to ask for the whole data once again
- Ok, let's split data in chunks $c_1 \ldots c_k$
  - I compute the hashes $h_1 = \mathrm{H}(c_1) \ldots h_k = H(c_k)$
  - I send you the hashes using a secure/expensive channel
  - I send you data using normal channel
  - If your H(c_i) is not h_i, you know it is bad, and I just ask for c_i
- All cool, but sending all the hashes is heavy
  - One per chunk!

Let's chain the hashes!

- Let's split data in chunks $c_1 \ldots c_k$
  - I compute the hashes $h_1 = \mathrm{H}(c_1), h_2 = \mathrm{H}(c_2||h_1) \ldots h_k = H(c_k||h_{k-1})$
  - From time to time, you can ask me a hash, for example h_{10} to verify the first 10 chunks (blocks). Later on, you ask me h_{20} to verify the chunks 11 to 20

Digital signatures

Three algorithms: KeyGen, Sign, Verify

- `(sk, pk) := generateKeys(keysize)`
  - sk: secret signing key, pk: public verification key
- `sig := sign(sk, message)`
  - Computes the signature of the message using the secret key
- `isValid := verify(pk, message, sig)`
  - Decrypts the signature using the public key and compare the result with the message

For sake of Performance

- Algorithms using private/public keys are very slow
- Usually, the message is hashed with a known algorithm
  - known = I publicly declare which algorithm I am using
- Then, I compute the signature on the hash
- The receiver will decrypt the signature using the public key, compute the hash of the message, and compare

Try it yourself: key generation

www.cs.aau.dk

Elliptic encryption

```
from cryptography.hazmat.primitives.asymmetric import ec

private_key = ec.generate_private_key(ec.SECP256K1())

public_key = private_key.public_key()
```

The algorithm used for bitcoin transactions

Try it yourself: sign

```
from cryptography.hazmat.primitives import hashes
data = b"A message I want to sign"



signature = private_key.sign(
    data,ec.ECDSA(hashes.SHA256())
)
```

The hashing used for bitcoin

Elliptic encryption

Try it yourself: verify

```
public_key = private_key.public_key()

public_key.verify(signature, data, ec.ECDSA(hashes.SHA256()))
```

All the parameters must match the `sign` function's parameters

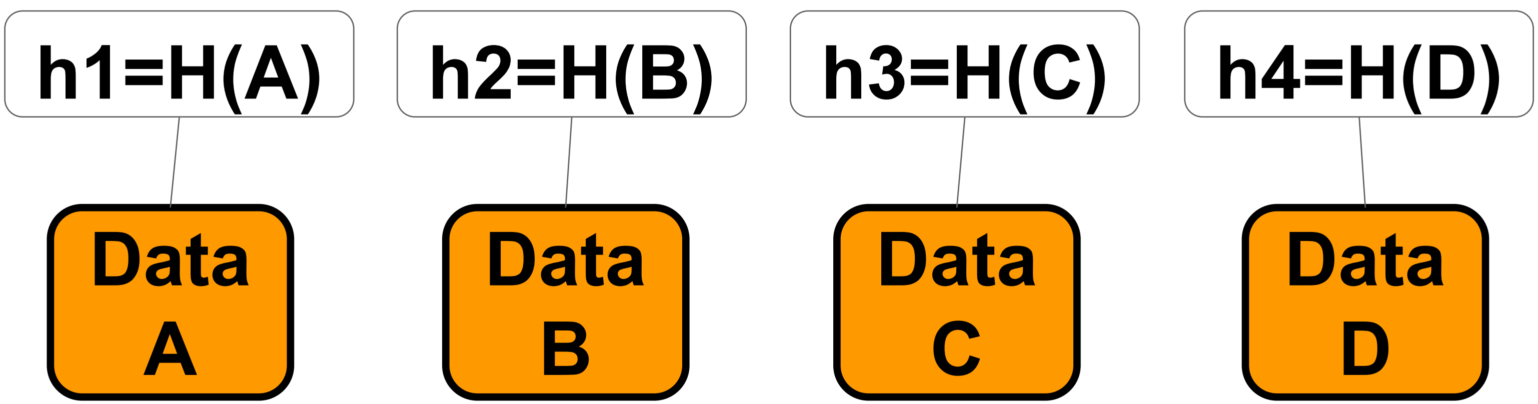
This does not return a value
It will raise a cryptography.exceptions.InvalidSignature exception if the signature is not valid

Merkle Tree (Hash tree)

- Hash trees or Merkle trees
  - a data structure summarizing information about a big quantity of data with the goal to check the content
- Introduced by Ralph Merkle in 1979
  - Combines hash functions with binary tree structure
- Main ingredient: a complete binary tree built starting from an initial set of symbols
  - exploits a hash function H (SHA1, MD5)
  - leaves: H applied to the initial symbols
  - internal nodes: H applied to the sons of a node

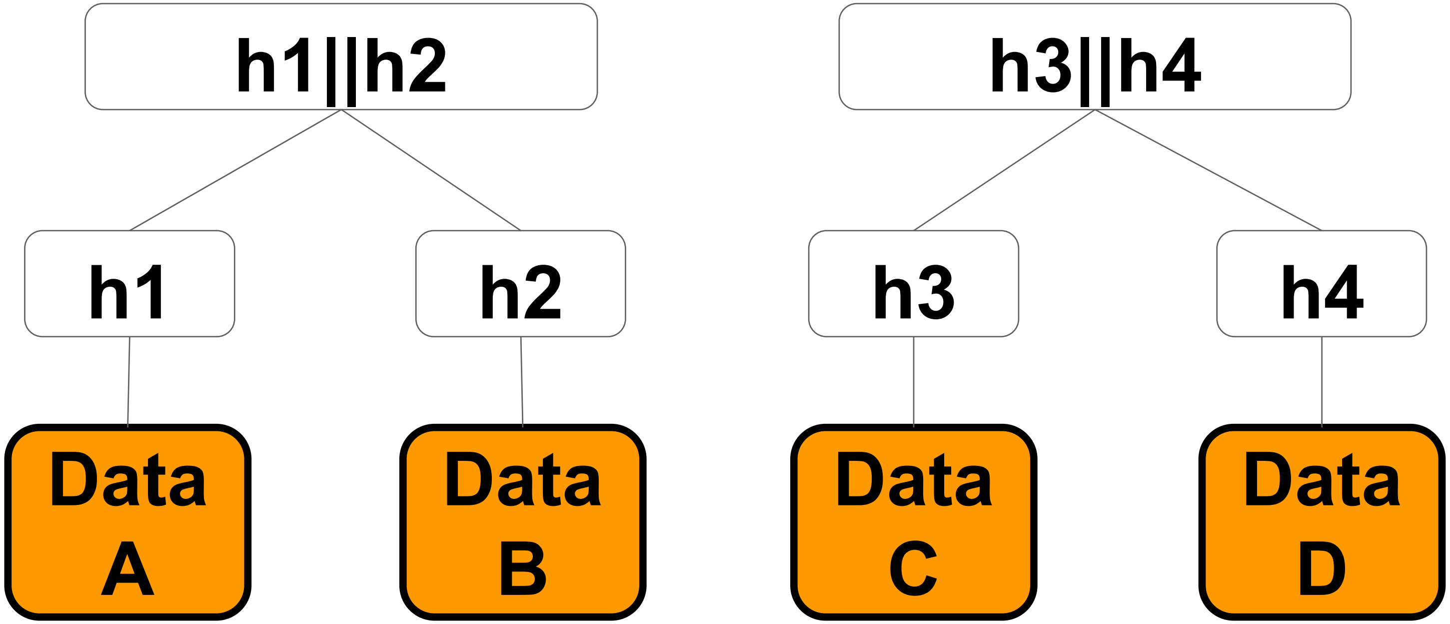
Merkle tree data structure

- You have 4 blocks?
- Hash them



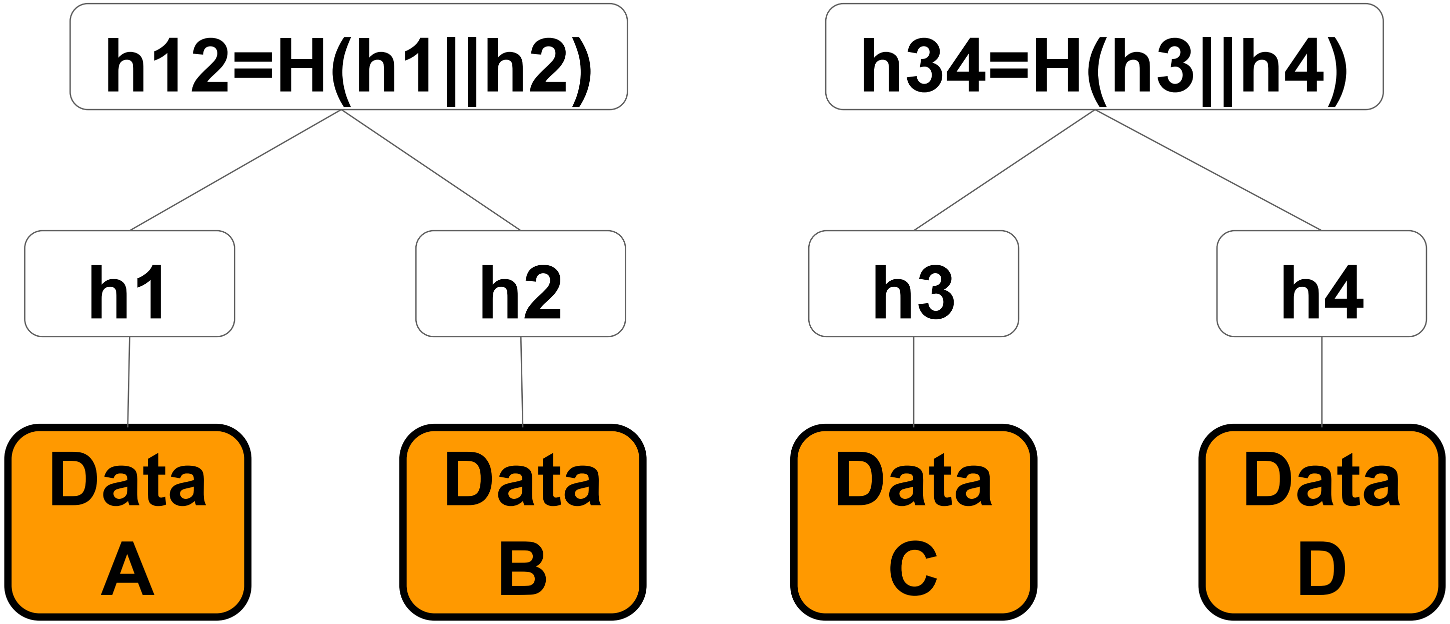
Merkle tree data structure

- Now concatenate them



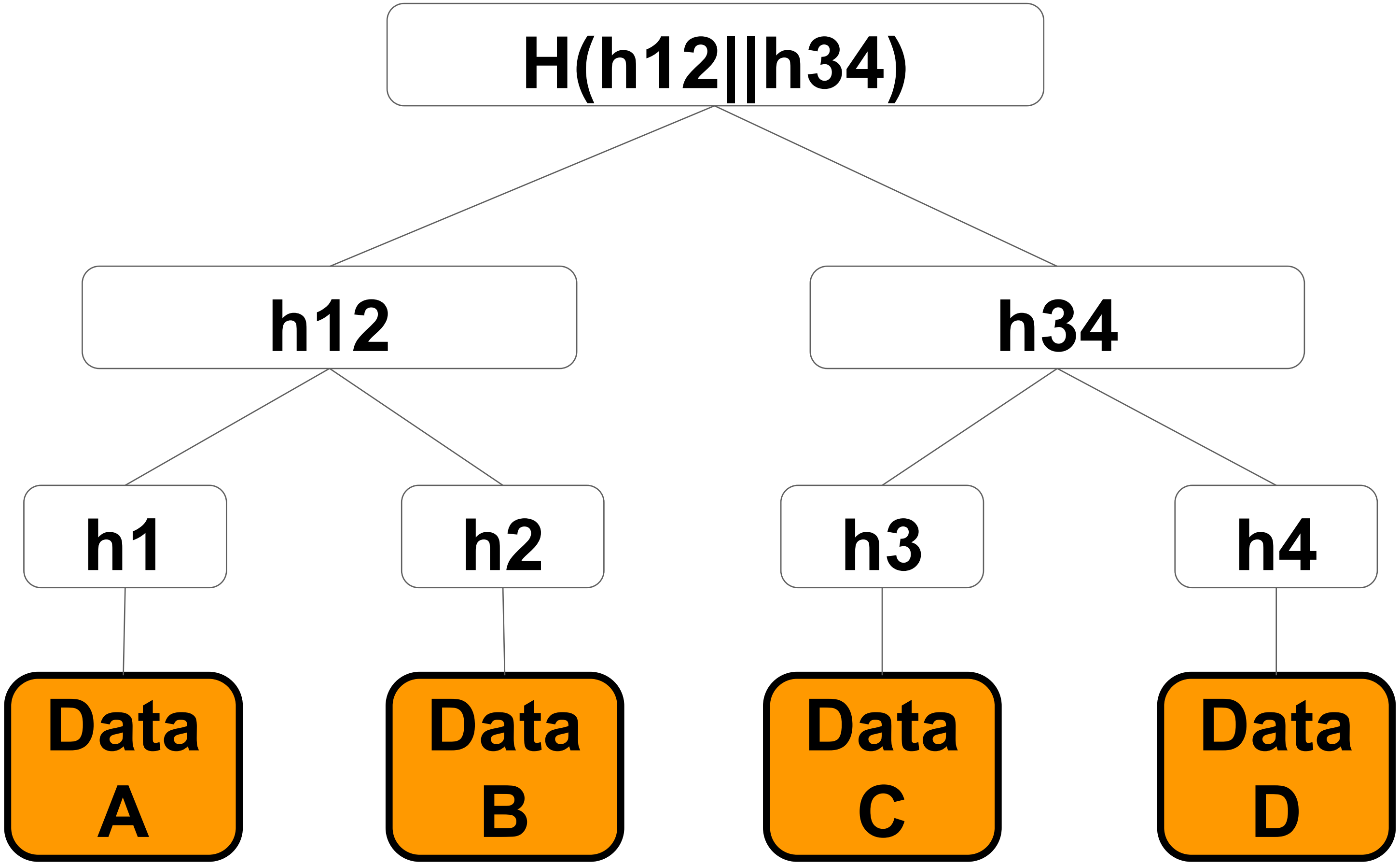
Merkle tree data structure

- Hash the concatenation of the hashes



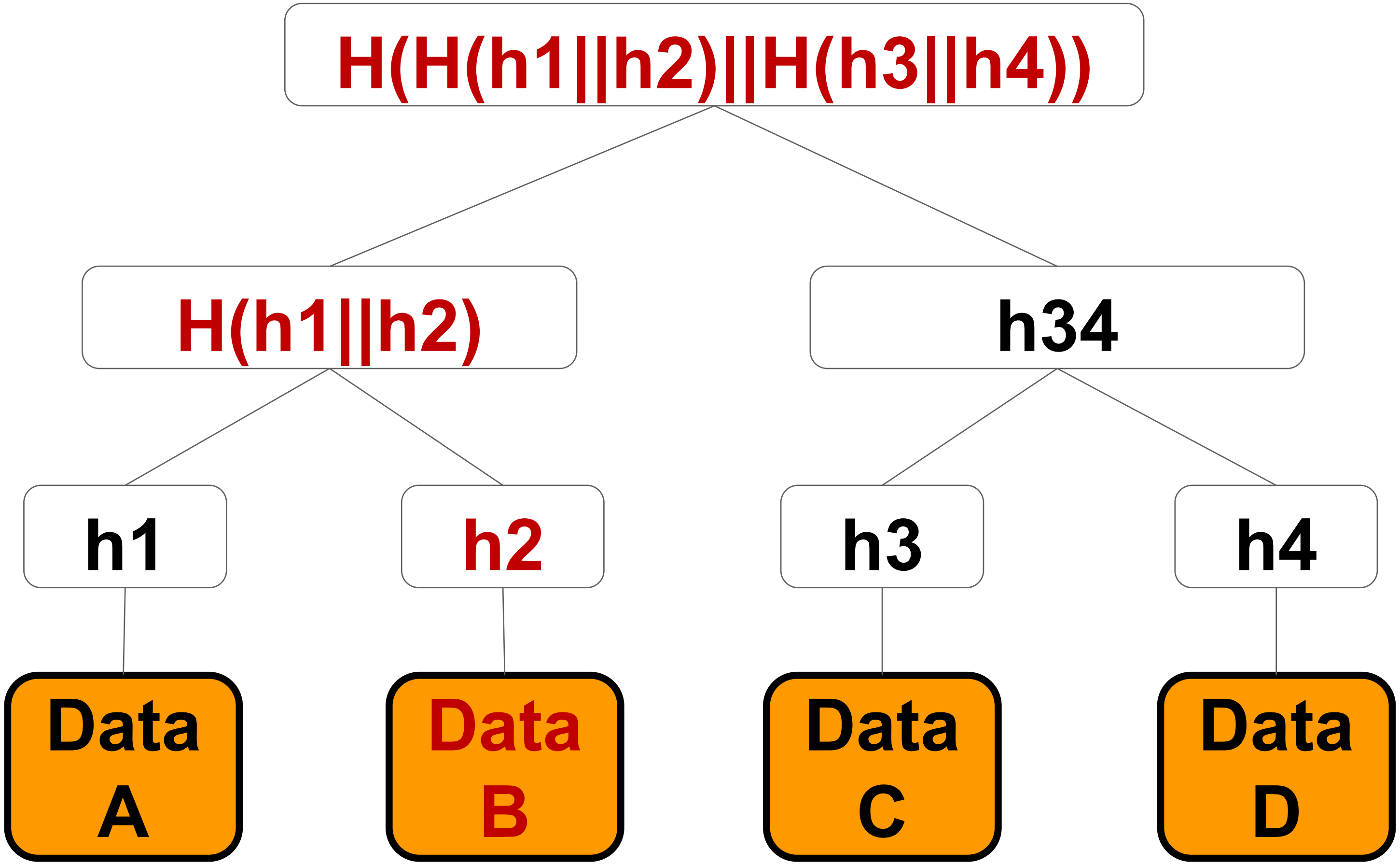
Merkle tree data structure

- And repeat, so on and so forth
- It is a tree of hash



Merkle tree data structure

- If data B is corrupted, it invalidates one whole branch of the tree

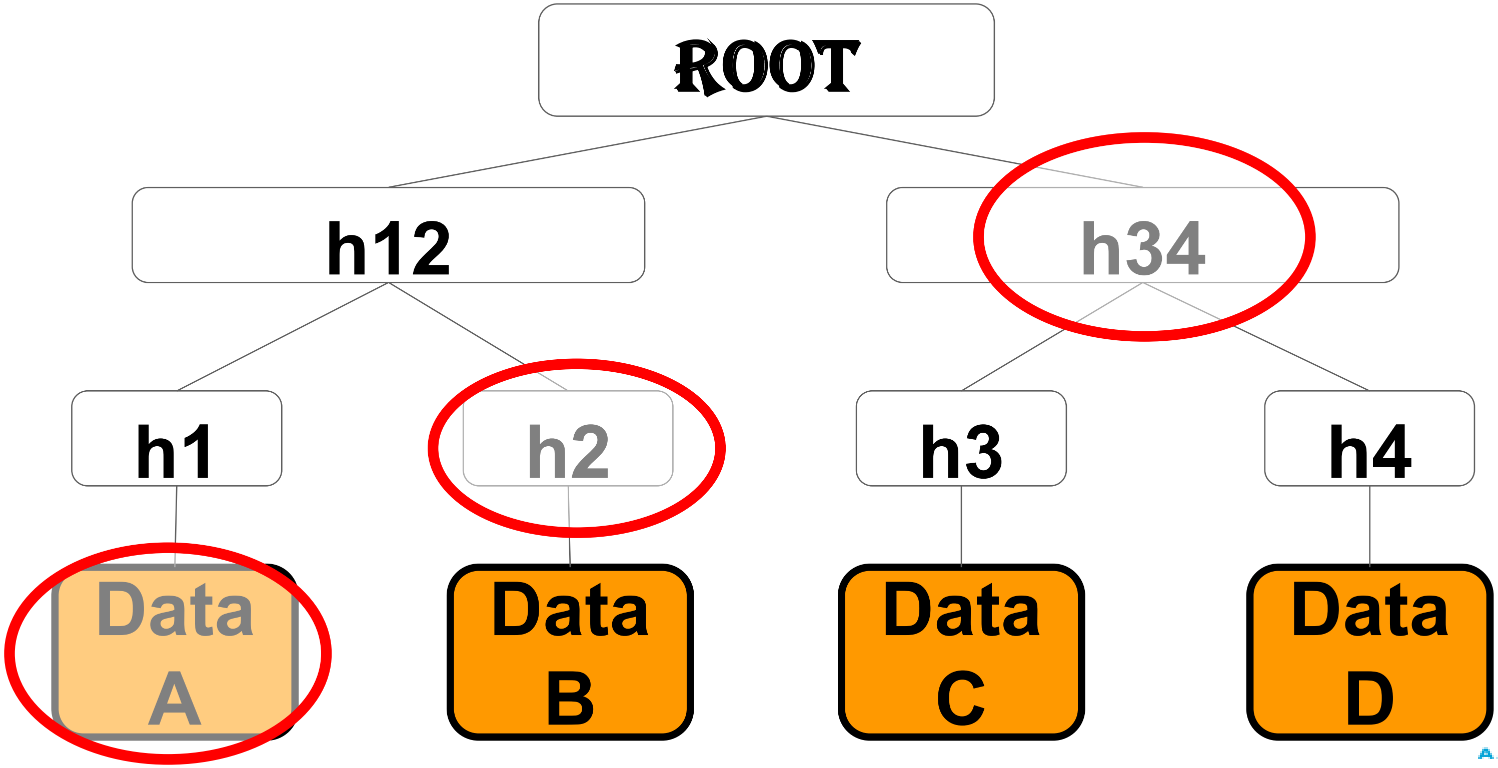


Building a Merkle tree

- Let us organize all data as leaves of the hash tree
  - Nodes at height h will depend on 2^h leaf values
  - Tree of height H has N= 2^H leaves
- Obtaining the root **P** requires calculating all N leaf values plus 2^H-1 more hash function evaluations
  - Is it too much work? It is **2N**

Benefits of a Merkle Tree

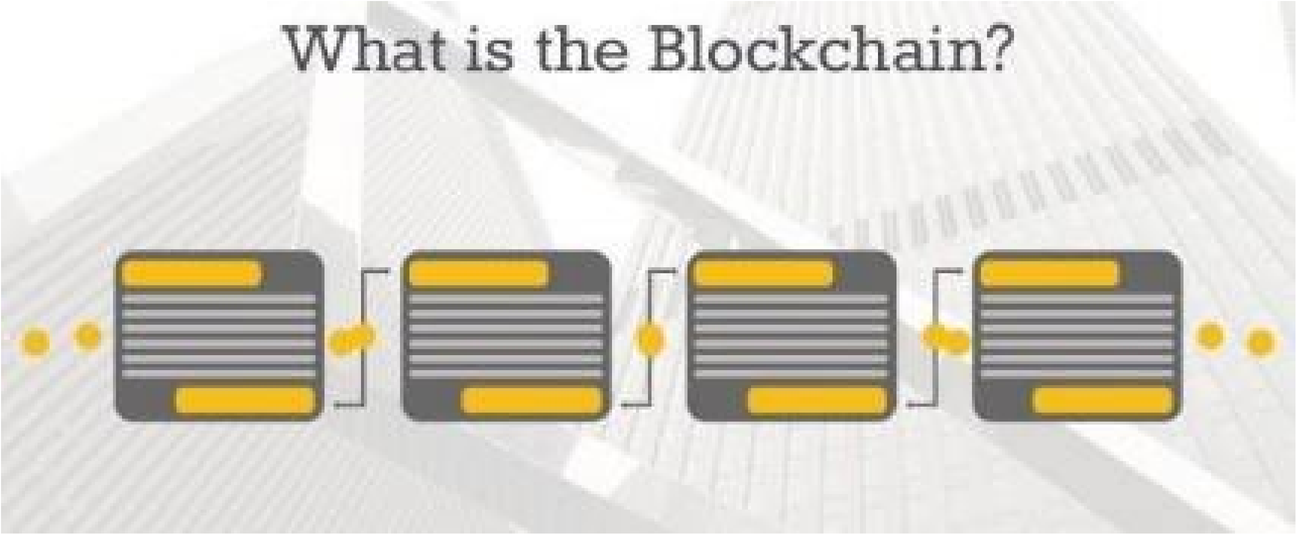
- Let us store the root of the hash tree somewhere!
  - If suspicious, you can ask me for all data and verify the root is right
- Compute the root from all the data
  - You can cache the hashes of the “witnesses” for performance
  - You can hash elements “as you go” if data are produced slowly



Outline

- Crypto tools: hash, signatures, asymmetric crypto, Merkle trees
- Blockchain basics: hash pointers, consensus, proof of work
- Blockchain details: transactions, Bitcoin, what a user is, miners
- Basics on Ethereum smart contracts

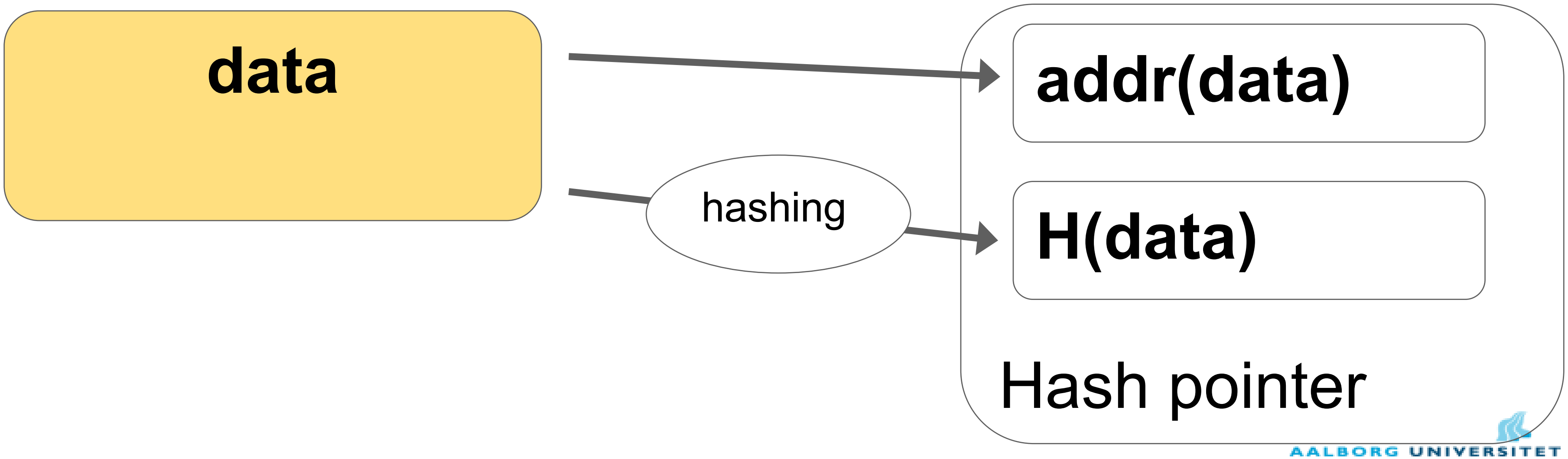
Blockchain



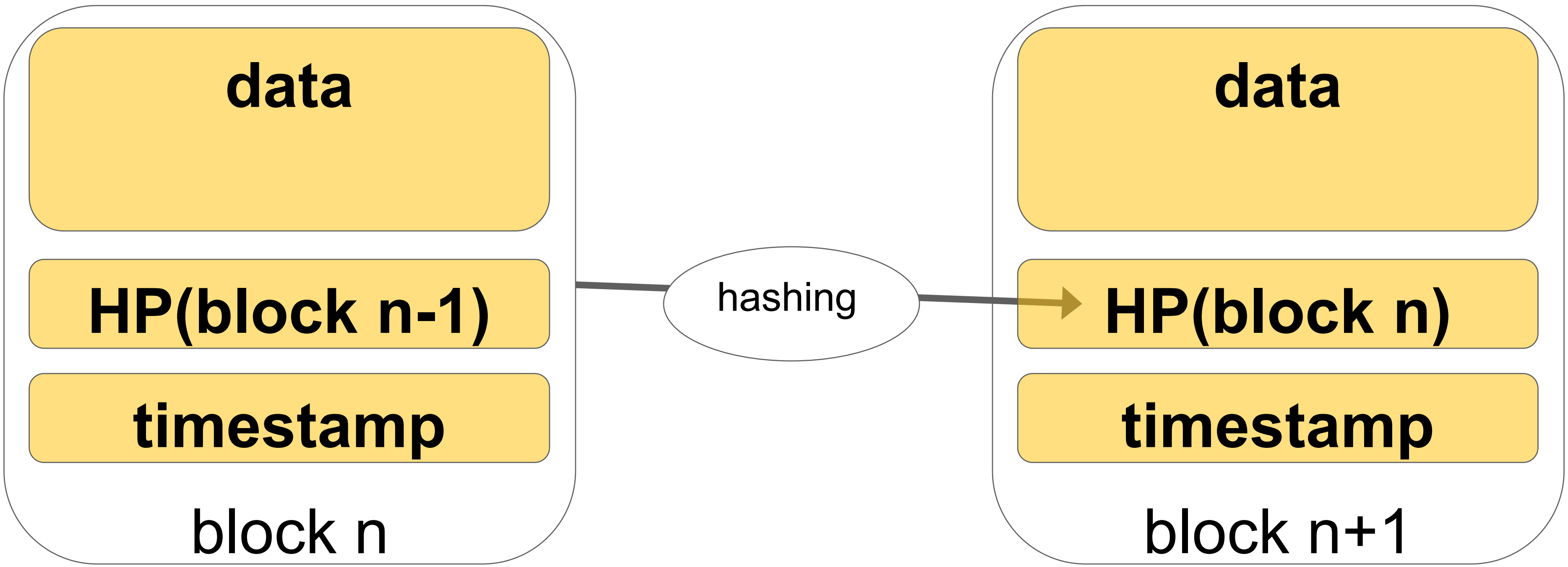
- *A blockchain is a digitized, decentralized, public ledger of all cryptocurrency transactions. Constantly growing as 'mined' blocks (the most recent transactions) are recorded and added to it in chronological order, it allows market participants to keep track of digital currency transactions without central recordkeeping. (merkle tree)*
- Wikipedia: "A distributed database that is used to maintain a continuously growing list of records, called *blocks*. Each block contains a timestamp and a link to a previous block"

Hash pointer

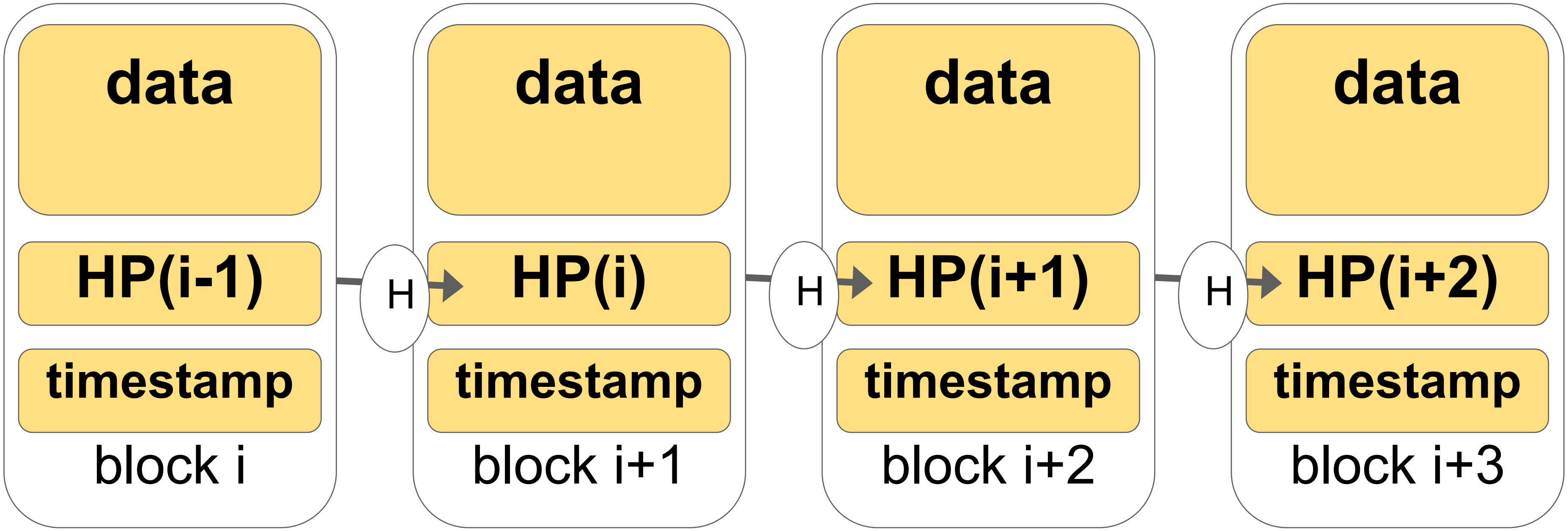
- Hash Pointer
  - a pointer to where some info is stored
  - a cryptographic hash of the info
- If we have a hash pointer, we can
  - ask to get the info back
  - verify that it hasn't changed
- Tamper-evident data pointer = Hash Pointer (HP)



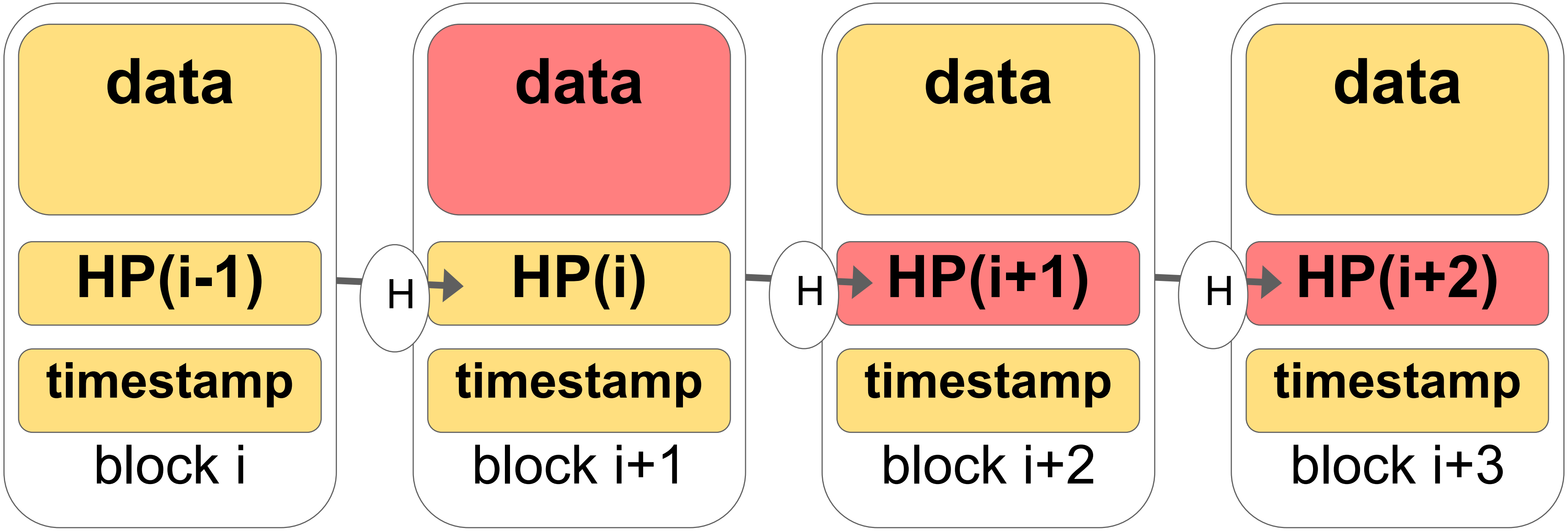
Blockchain!



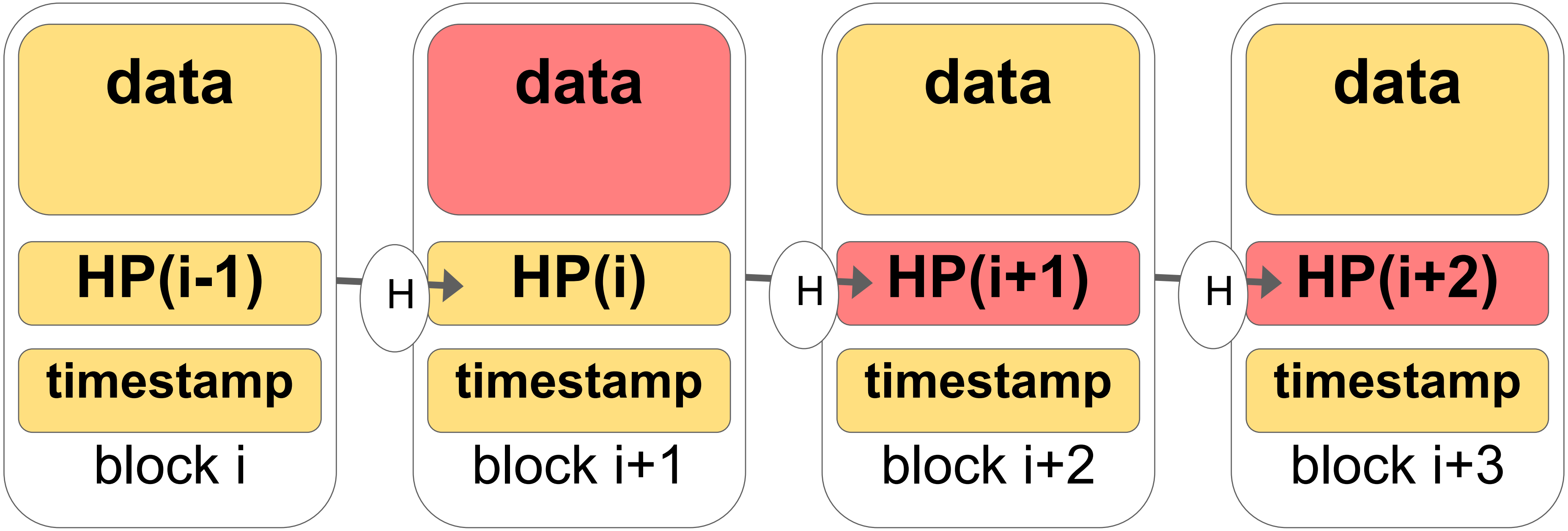
- Information organized into blocks
- Each block has a hash pointer (HP) to previous block
- To verify *block n*, hash it and compare to *HP(block n)*
  - Which is contained into your *block n+1*
- Tamper free block addition



- BECAUSE NAMING IT "TAMPER-PROOF LINKED LIST" WAS TOO LAME!



- If you tamper with data in block i
- You **have** to tamper with hash pointer i+1, hash pointer i+2 etc etc
  - Since HP i+2 depends on HP i+1 etc etc



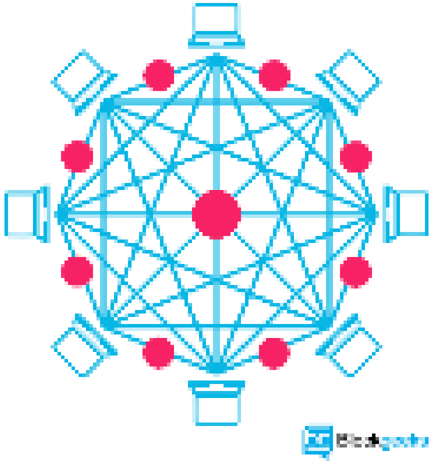
- If you tamper with data in block i
- You **have** to tamper with hash pointer i+1, hash pointer i+2 etc etc
    - Since HP i+2 depends on HP i+1 etc etc
- Requirement: to store the HP on the head somewhere safe
    - Where? **Everywhere!** … next topic to discuss

Blockchain as a decentralized database

- **Consistency:** Information held on a blockchain exists as a shared — and continually reconciled — distributed database
- **Robustness:** No centralized version of this information exists for a hacker to corrupt
- **Availability:** Data are stored by millions of computers simultaneously, and can be accessed and verified by anyone on the internet

Blockchain network architecture

- Node: A computer connected to the blockchain network using a client that performs the task of validating and relaying data, e.g.: transactions
- The node gets a copy of the blockchain upon joining the blockchain network
- Every node is an “administrator” of the blockchain (in this sense, the network is decentralized)



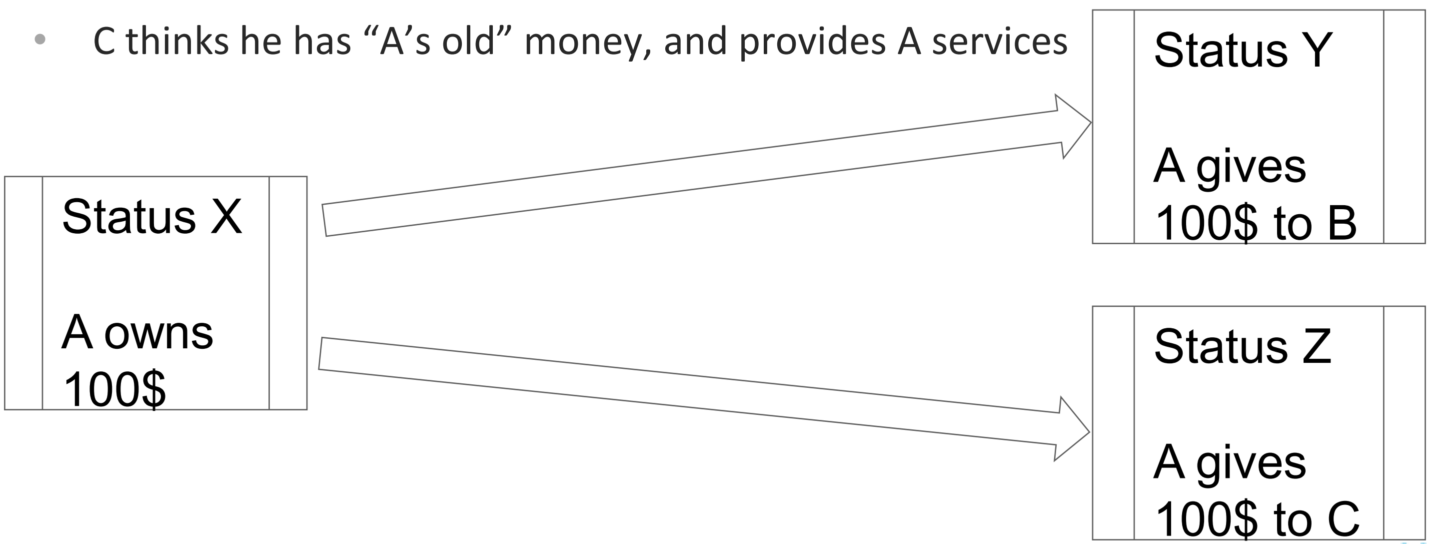
AALBORG UNIVERSITET

Challenge: Double spending conundrum 1/2

- Let us consider that information in ledger is transactions
  - A writes in the blockchain that it gave B $$$ in exchange for something
  - B can use the information in the block to prove ownership of “A’s old” money

Double spending conundrum:

- A takes status X of the blockchain, it writes on a block it gives B 100$, and send the blockchain around
  - B receives it from the network, B believe it owns “A’s old” money, and provides A services
- Then A takes status X of the blockchain and add another transaction to C
  - C thinks he has “A’s old” money, and provides A services



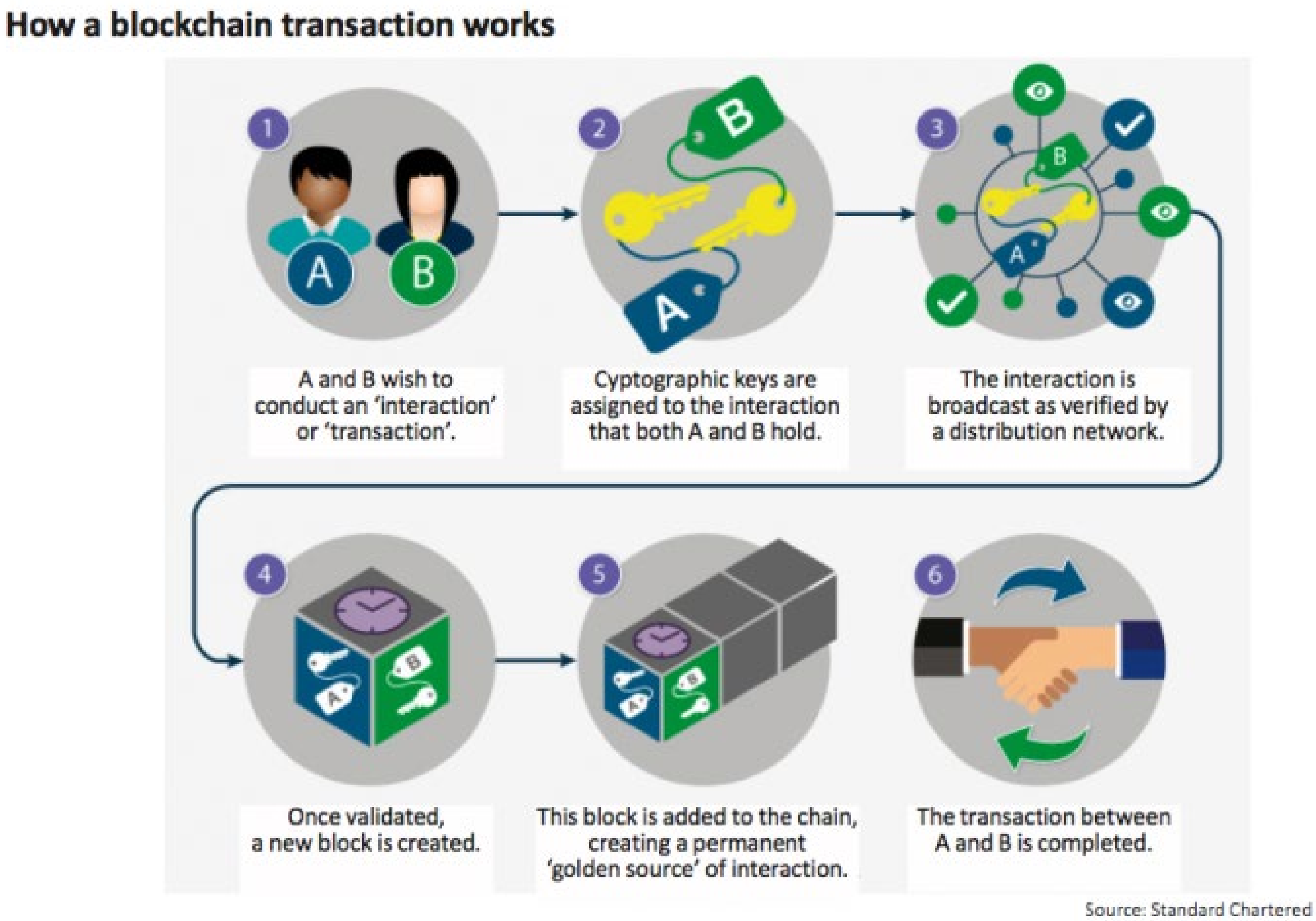
Challenge: Double spending conundrum 2/2

- Double-spending is the result of successfully spending digital money more than once
- Protection against double spending:
  - Verify each transaction added and ensure that the input for the transaction had not previously already been spent
  - Need to have just one valid blockchain, always growing with all accepted transactions, and validated by means of consensus
    - No Forks!!!

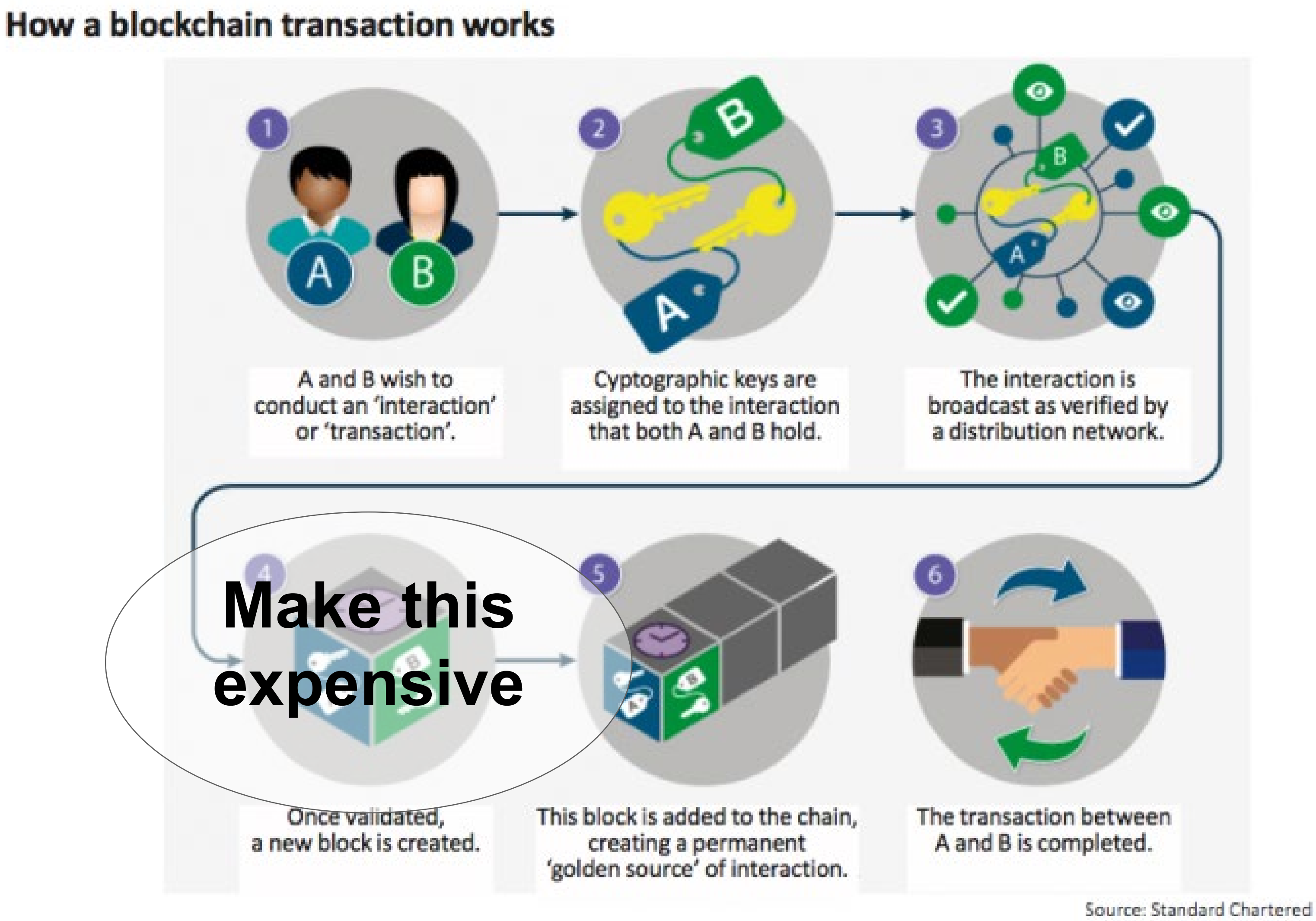
Consensus?

- Can we use Paxos?
  - As soon as you provide me services, I create 1,000,000 fake ids (Sybil attack) and I distribute the second blockchain
  - It has consensus! I can double spend my $$$!!
  - The problem is that creating new identities is cheap
- Let us do something that is not so cheap

A Blockchain Transaction

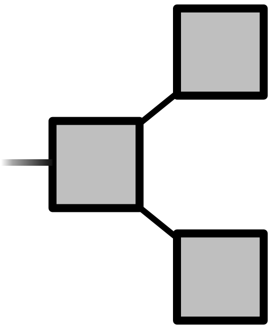


A Blockchain Transaction



Nakamoto consensus

- Let us decide that all the nodes send to each other the blockchain

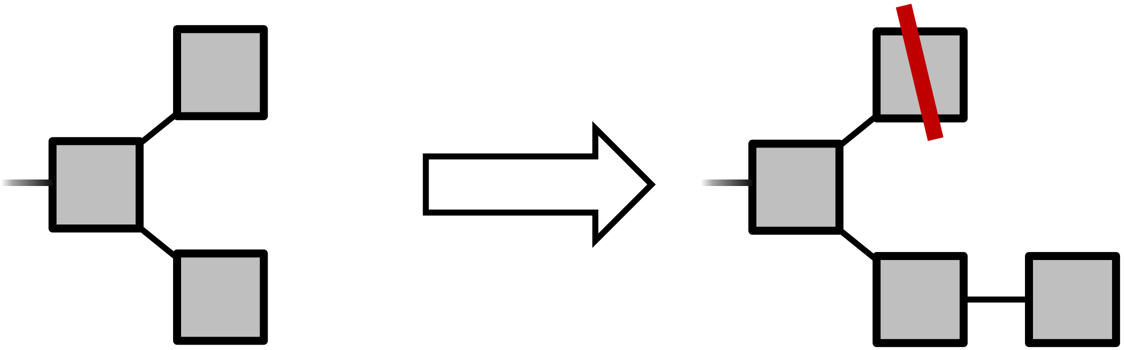


- But a fork is natural in distributed systems!

- **The rule: The longest blockchain has consensus!**

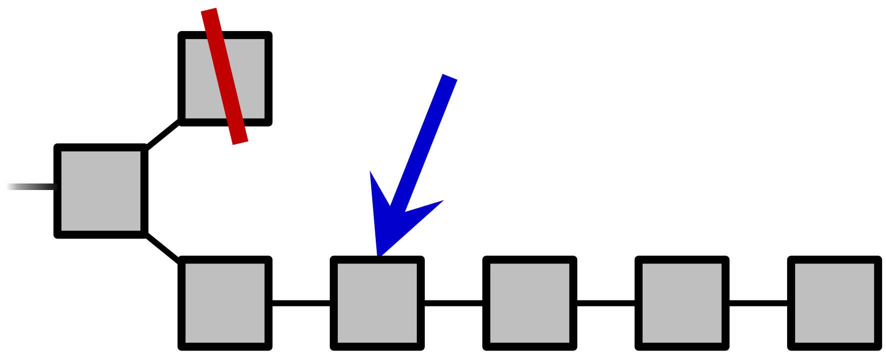
Nakamoto consensus

- I consider that
  - adding a block is computationally expensive
  - most nodes are honest
- If I receive a blockchain that is shorter than mine, I ignore it
  - It was created later than mine
  - It can be out of sync



Nakamoto consensus

- A transaction is "accepted" if it is buried deep enough



- If I see a longer blockchain, I embrace it, and try to add my blocks to it.
- Statistically, the accepted transaction will not be eaten by a newer (shorter) transaction
  - All transactions added in the blocks are immutable
  - e,.g.: no double spending

How do I make *adding a block* expensive?

Proof of Stake (For a later class :-P)

Proof of Work

- Provide a computational (hashing) puzzle that is
  - hard to solve when you want to add a valid block
  - easy to solve when you want to verify that a block is valid
- All the nodes working actively to support the blockchain (miners, more on this later) take e.g.: 7 seconds to add a block
  - You cannot start working on your fork before a transaction is *accepted,* which implies adding N blocks
  - Your malicious computers will not be able to redo the work on your fork (adding N+1 blocks) faster than all other miners add blocks on top of the *accepted* block

We still have to specify 2 things

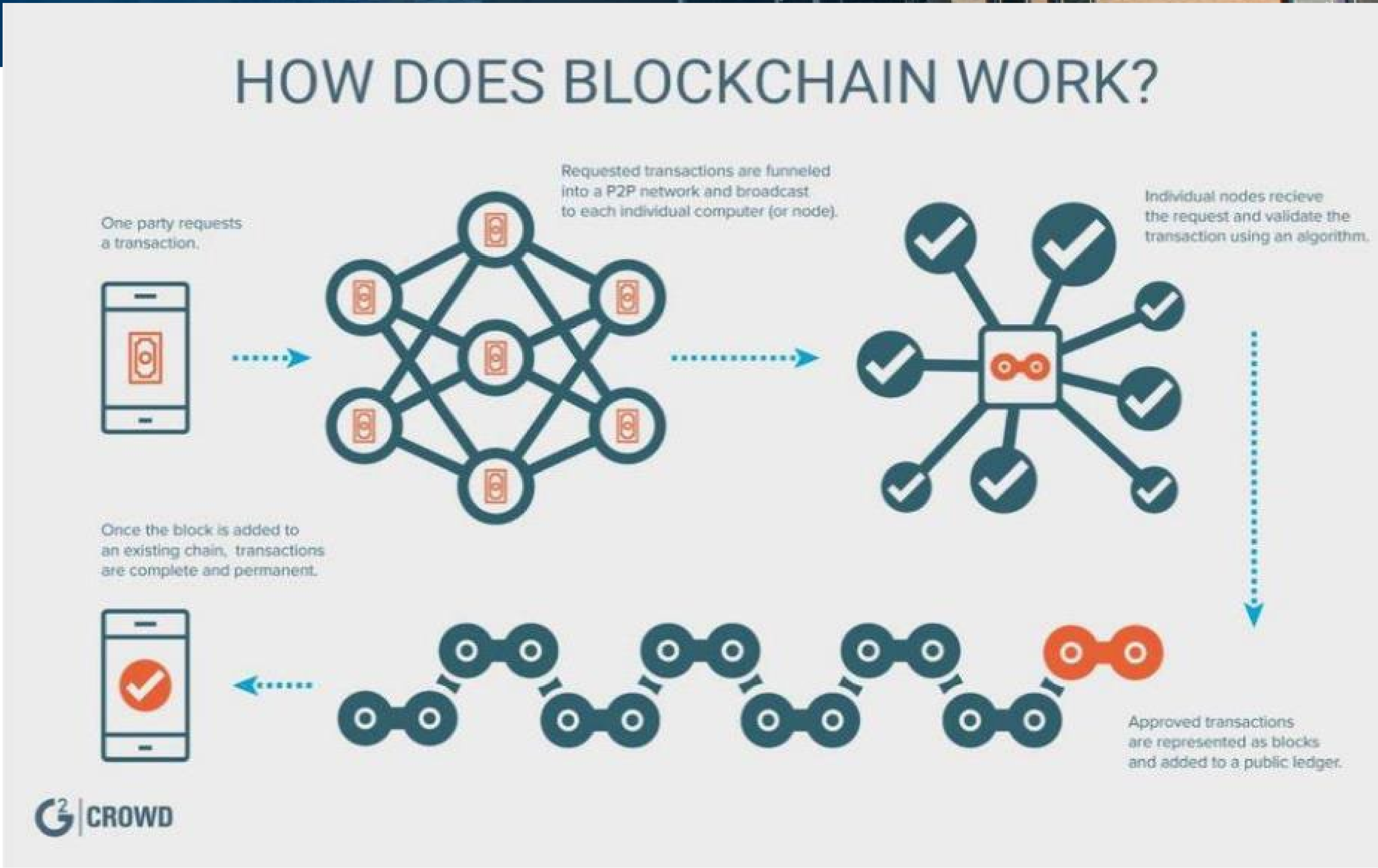
- The *proof of work* puzzle
- What is inside this *immutable transaction* data structure

The puzzle

- I insert a *nonce* in the header
  - It has to be hashed together with the rest of the header
- The *nonce* is good when H(header) starts with n zeros
  - Need to select nonces *at random* until one fits the puzzle
  - n defines how hard it is to add a new block
- We use SHA256, I give you challenge n = 36
  - Each *nonce* has prob 2^{-36} of success
  - When you succeed, only takes me one hash to check

The transaction

- “User P1 transfers 200$ to User P2”
- I can go back one (or more) blocks to verify that User P1 owned the money



AALBORG UNIVERSITET

Outline

- Crypto tools: hash, signatures, asymmetric crypto, Merkle trees
- Blockchain basics: hash pointers, consensus, proof of work
- Blockchain details: transactions, Bitcoin, what a user is, miners
- Basics on Ethereum smart contracts

Focus on transactions

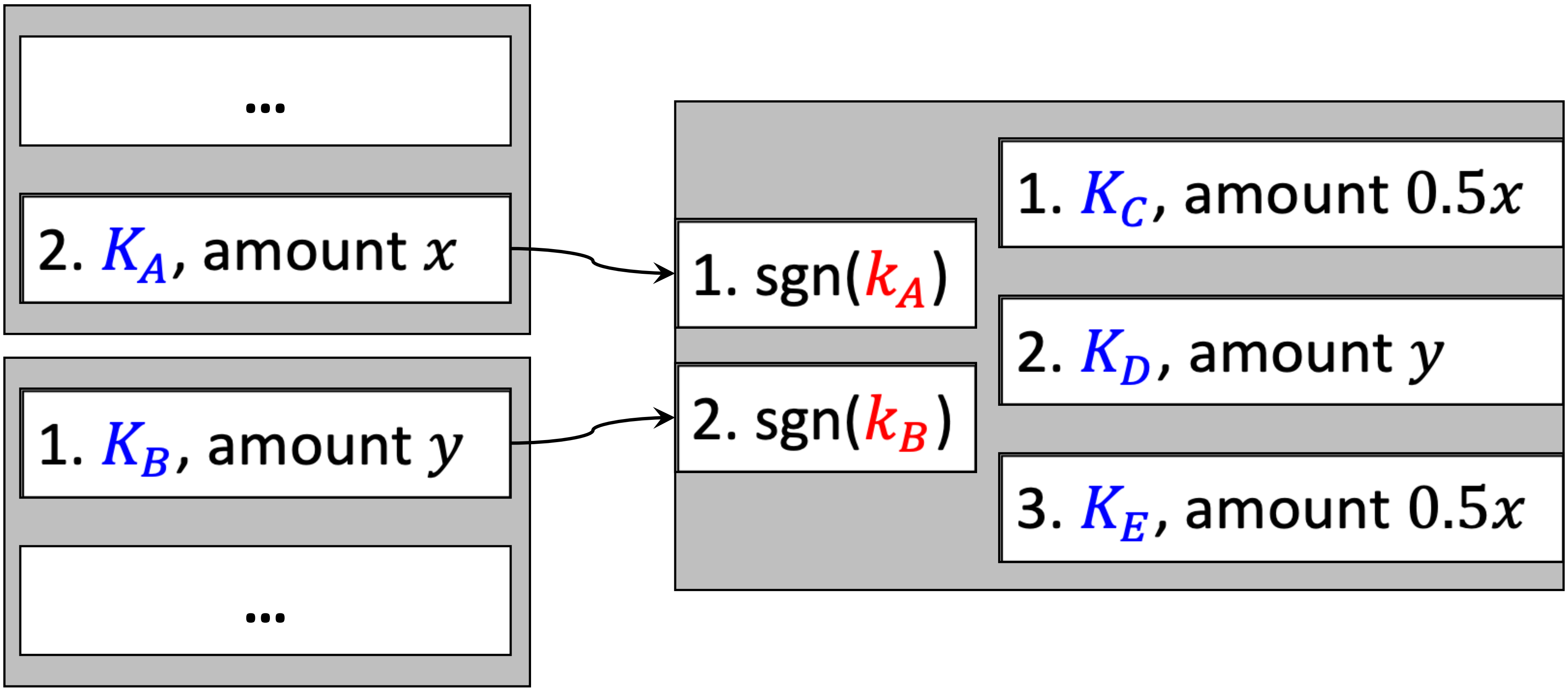
- Are you really happy with the description of the data structures and algorithms?
  - User P1?
  - Verification of transactions?

What is a user?

- A user is somebody who can transfer money
  - A wallet, a user's identity, a pair (sk : private key, pk : public key)

- Transaction: ([input transactions], [output identity pk, how much], signature)
  - The input transactions refer to previous transactions that transferred money to the user
  - It specifies to whom transfer each fraction of the money, by means of users' public keys
  - It signs the transaction with its private key

Transactions



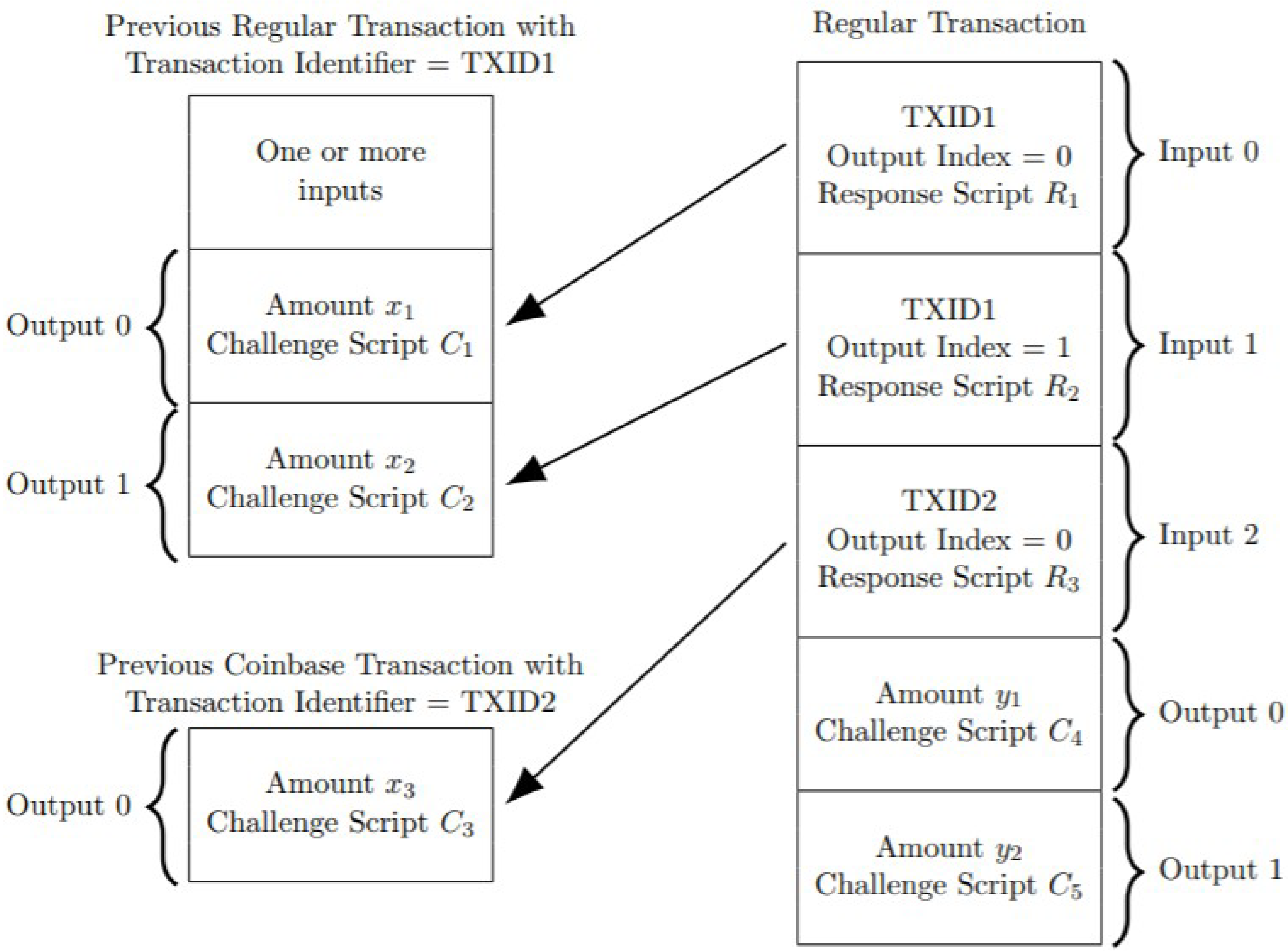
Characteristics of the transaction

- Since it must be signed with the sk, only the owner of the identity can transfer (spend) that money
  - If it points to input transactions from different identities p1 … pk, it will sign the transaction with all the s1 … sk to prove it owns the money

- All money from input transactions must be used
  - But part of the money can be transferred to the same user
  - Or another public key of the same user – for privacy

- Privacy: a real-life user can create one identity (sk,pk) each time it wants to receive money

More general case: scripts



More general output of a transaction

It contains a challenge script (also called locking script or scriptPubKey) with:

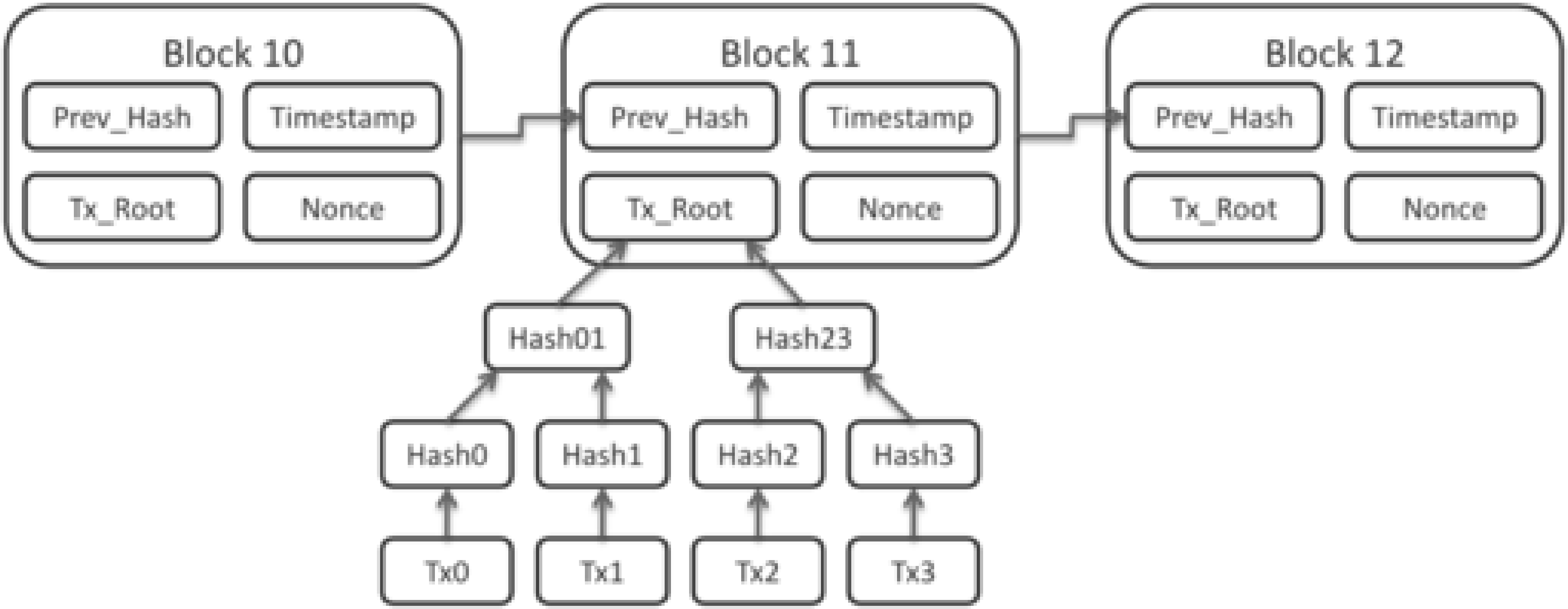
- the spending conditions under which the bitcoins associated can be spent

Usually, it requires just a valid signature. Other cases:

- It can be a script, encoded in a non-Turing complete language
  - To protect miners against DOS attacks (`while true {}`)
- A m-of-n multisignature challenge script
- Temporal parameters for automatic billing over time

How I save transactions in a block

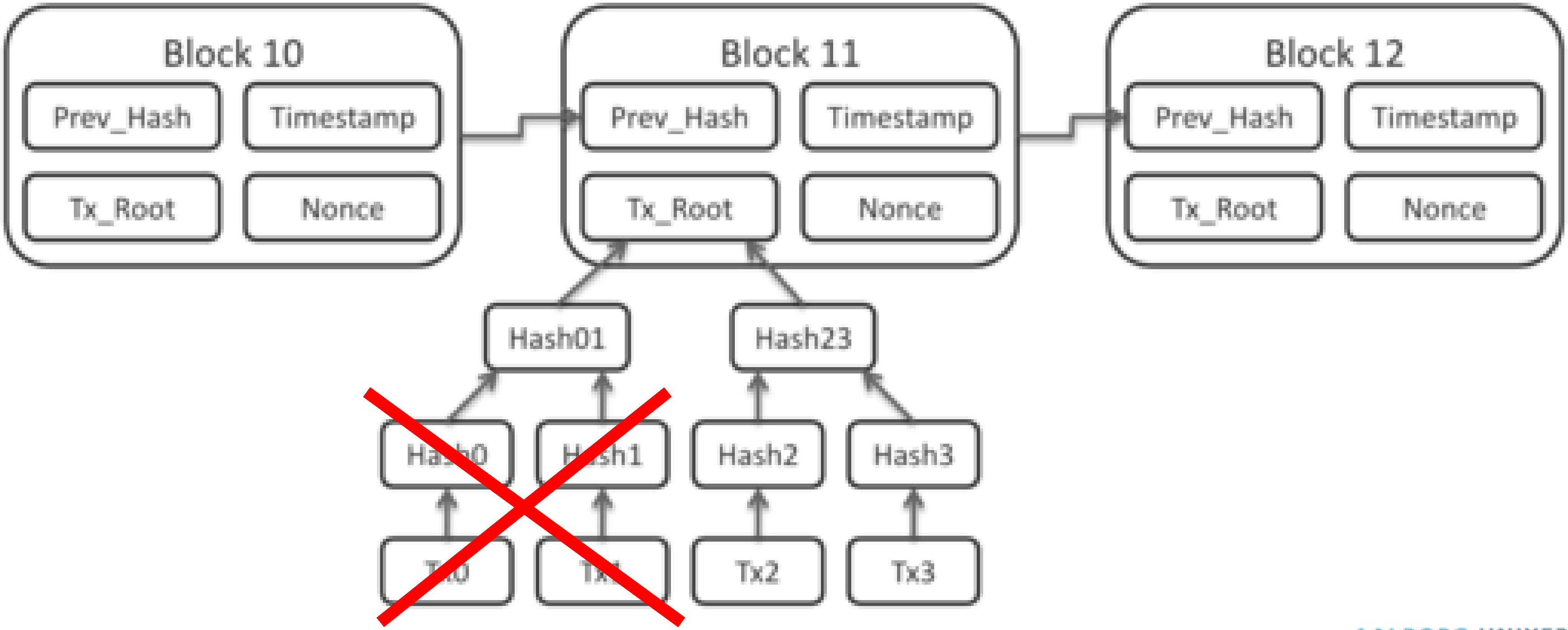
- Transactions are received continuously by nodes looking to solve the puzzle
- Organized into a Merkle tree
- Root hash is part of the new block header, thus it impacts next block's hash



AALBORG UNIVERSITET

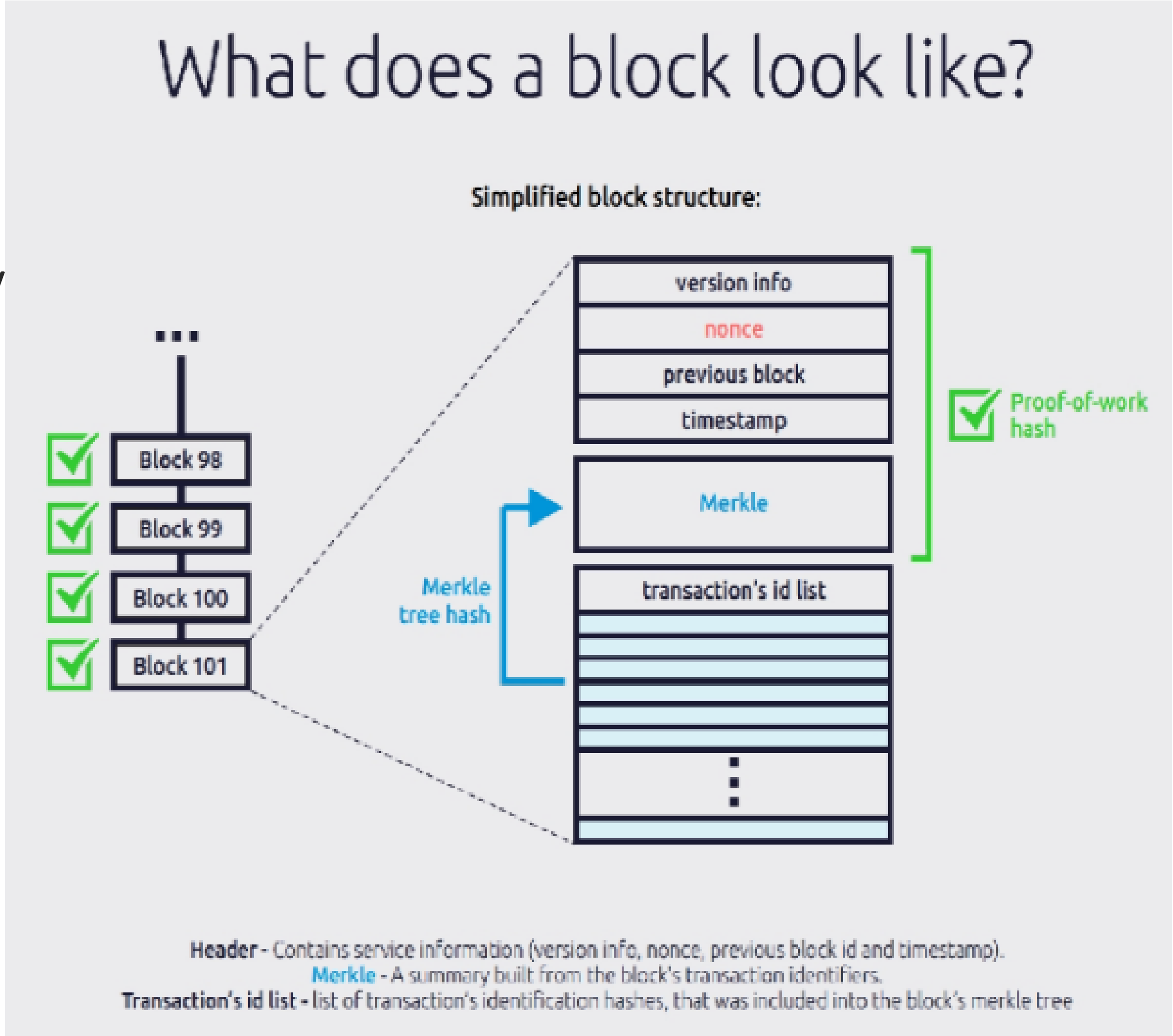
Why the Merkle tree?

- The Merkle tree can be computed as transactions are received
  - No need to recompute the hash of all the transactions after one is received
- To verify transactions, need to have the data of the block
- **However**, possible to discard branches of the Merkle tree when all its money is spent, and keep the hash only on the disk



How does everything add up?

- A miner will:
  - Verify all the transactions by looking that input transactions are covered and properly signed
  - Compute the Merkle root hash for the transactions
  - Solve the puzzle on the previous block, for immutability
  - Broadcast the new header
  - Go on collecting new transactions for next block



AALBORG UNIVERSITET

Why being a miner?

- **Incentives**: every new block that gets accepted provides a number of bitcoins by default to the miner who found the nonce

- **Fees**: for each transaction, some difference between input and output, and the difference is paid to the miner finding the nonce

- Anyway, energy waste!
  - Bitcoin blockchain network's miners are attempting 450 thousand trillion solutions per second to validate transactions and **especially** find a nonce
  - Vast amounts of energy necessary to process and store transactions
  - Wasted resources: energy for $15million/day

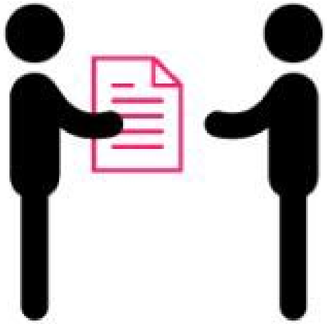
Outline

- Crypto tools: hash, signatures, asymmetric crypto, Merkle trees
- Blockchain basics: hash pointers, consensus, proof of work
- Blockchain details: transactions, Bitcoin, what a user is, miners
- Basics on Ethereum smart contracts

Beyond Bitcoin

- The challenge scripts could do much more than in Bitcoin
- Smart Contracts: Computer protocols that facilitate, verify, or enforce the negotiation or performance of a contract, or that make a contractual clause unnecessary
- Define the rules and penalties around an agreement in the same way that a traditional contract does, but also automatically enforce those obligations (code is law)
- Let’s talk about Ethereum

1

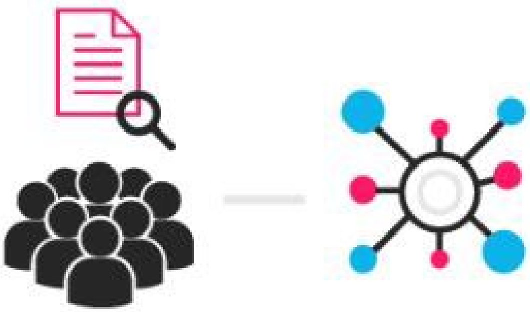


An option contact between parties is written as code into the blockchain. The individuals involved are anonymous, but the contact is the public ledger.

2

A triggering event like an expiration date and strike price is hit and the contract executes itself according to the coded terms.

3



Regulators can use the blockchain to understand the activity in the market while maintaining the privacy of individual actors' positions

Turing complete contracts on a blockchain

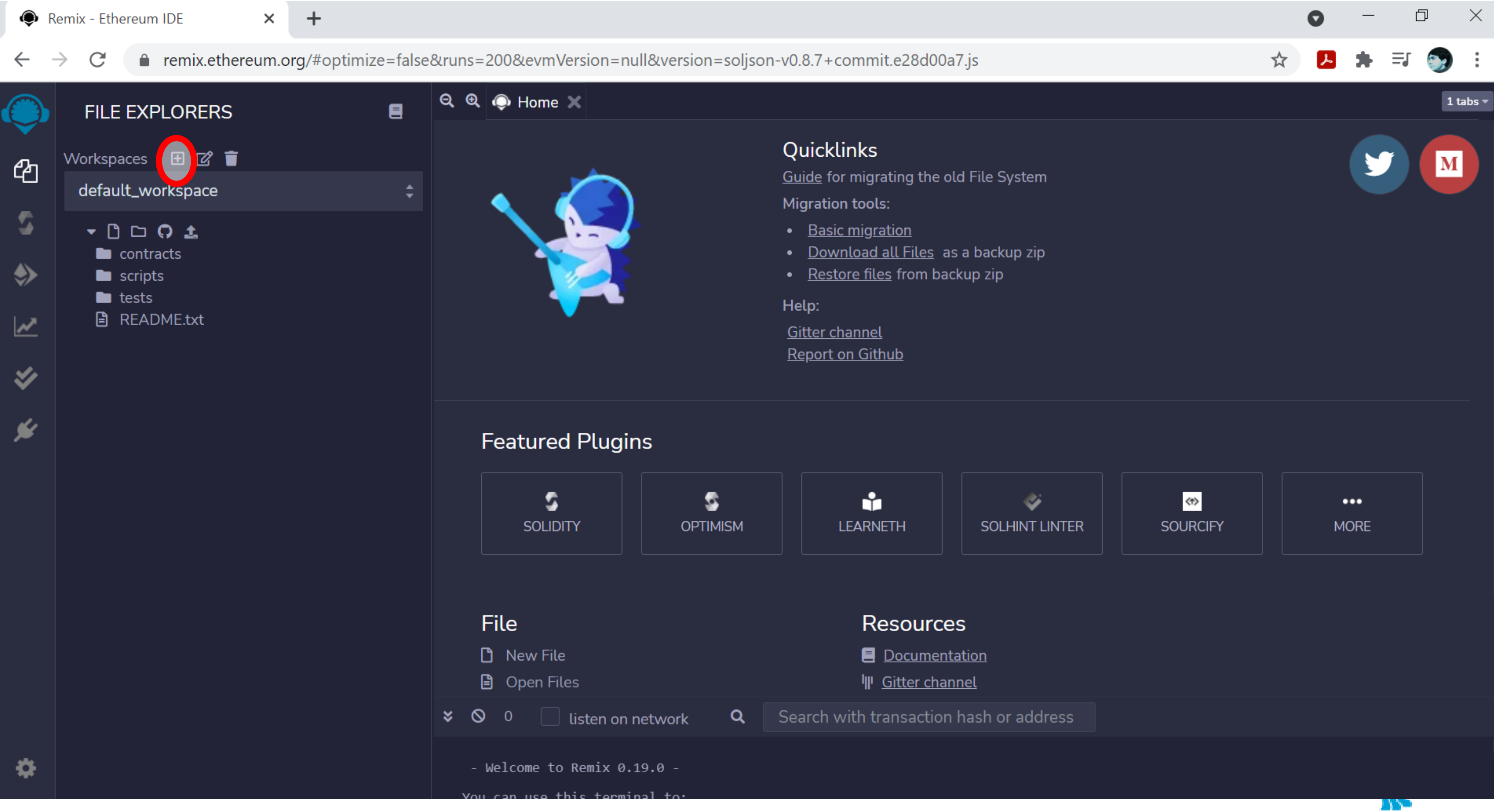
- Contracts are the main building blocks of Ethereum, the second most popular blockchain
- A contract is a computer program that lives inside the distributed Ethereum network and has its own ether balance of Ether (Ethereum's cryptocurrency / cryptofuel), memory and code.
- Every time you send a transaction to a contract, it executes its code, which can store data on the blockchain, issue transactions, and interact with other contracts.

Like a Jukebox

- Can be activated and run, by funding it with some Ether
  - to run, create a transaction sending a payment of ETH to the contract, and possibly supplying some other input information
  - the contract runs at most for a time dependent on how much gas (1 ETH = many many gas units) you paid
  - ETH fees are for the winning miner
- Each miner runs the smart contract, and produces same output
  - winning miner will publish the block to the rest of the network
  - other miners validate that they get the same result

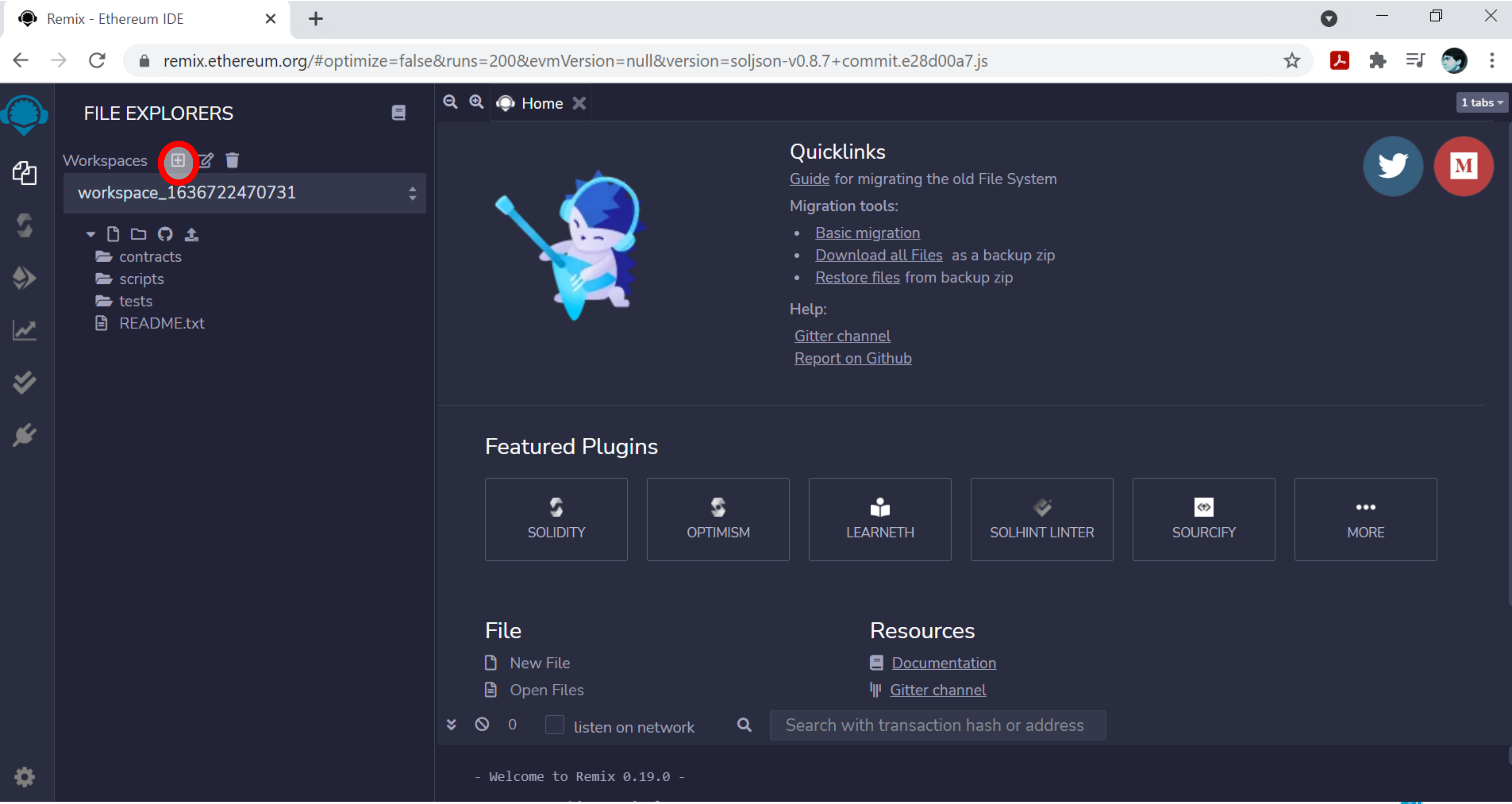
Let us write a **smart** contract

- Go to https://remix.ethereum.org and create a workspace



Add your contract

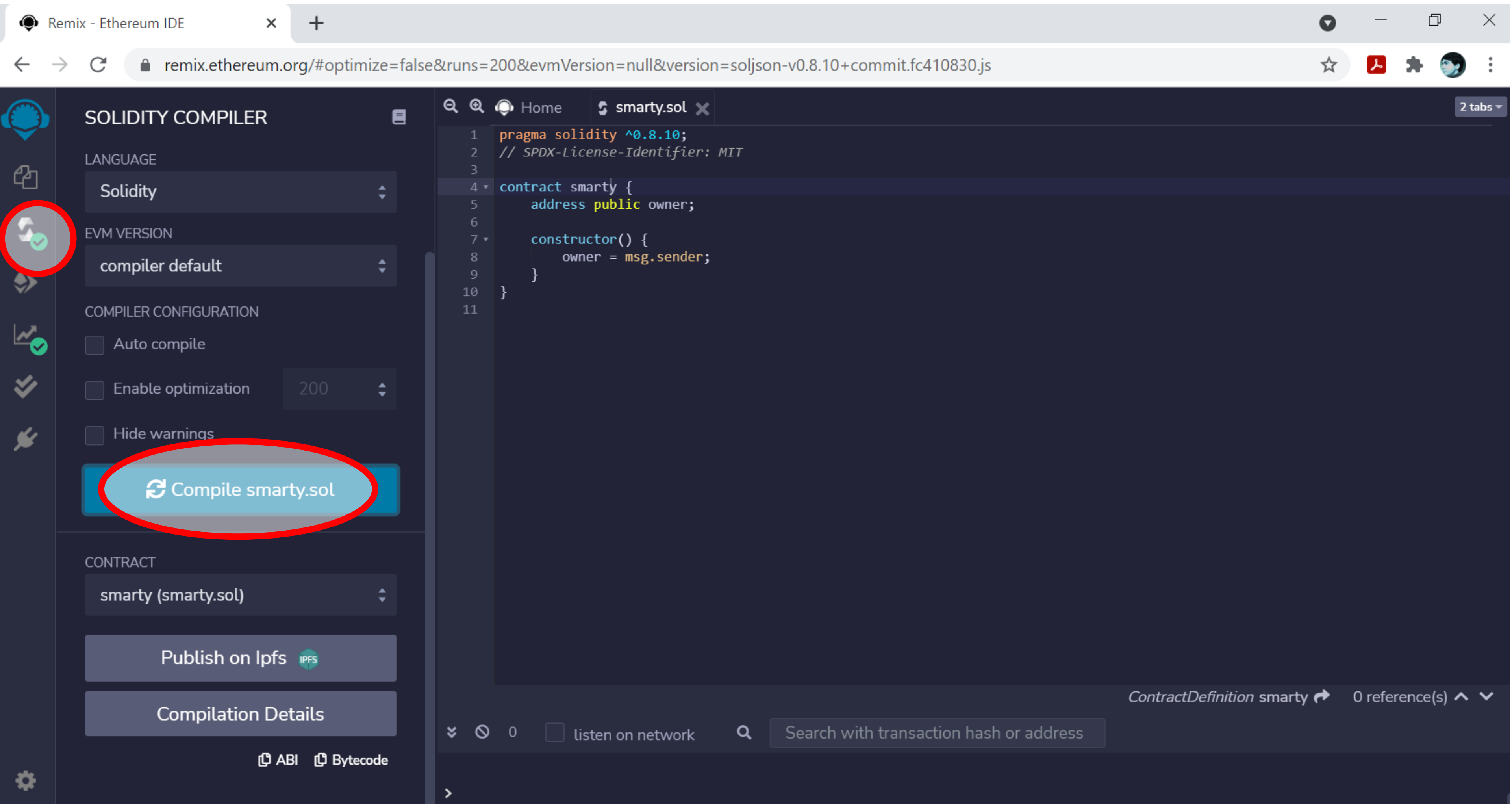
- Choose a **smart** name for your contract (“smarty.sol”?)



smarty.sol

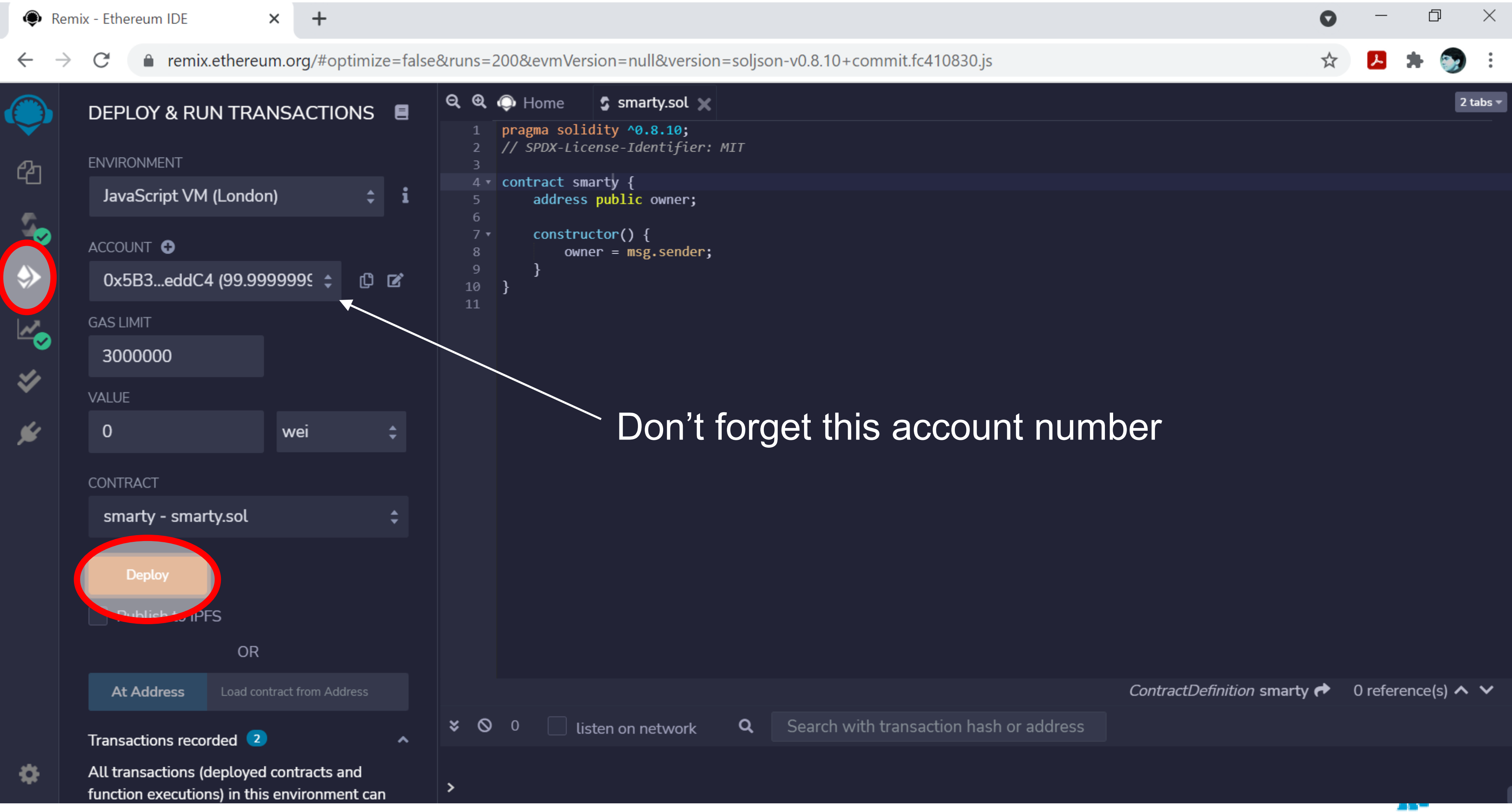

```
pragma solidity ^0.8.10;
// SPDX-License-Identifier: MIT

contract smarty {
    address public owner;

    constructor() {
        owner = msg.sender;
    }
}
```

Compile it

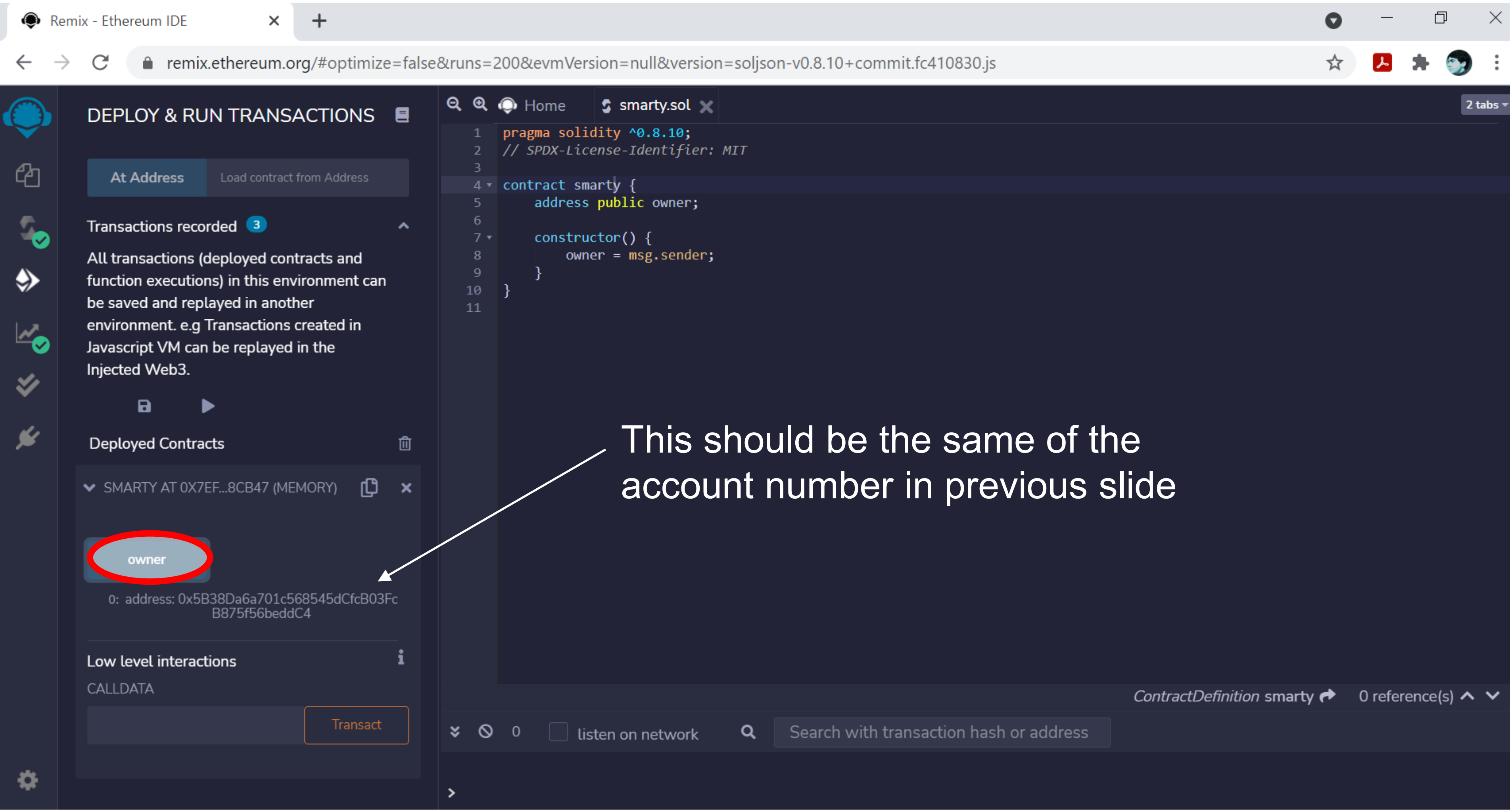

Remix - Ethereum IDE
remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.10+commit.fc410830.js
SOLIDITY COMPILER
LANGUAGE
Solidity
EVM VERSION
compiler default
COMPILER CONFIGURATION
Auto compile
Enable optimization
200
Hide warnings
Compile smarty.sol
CONTRACT
smarty (smarty.sol)
Publish on Ipfs
Compilation Details
ABI
Bytecode
Home
smarty.sol
2 tabs
pragma solidity ^0.8.10;
// SPDX-License-Identifier: MIT
contract smarty {
address public owner;
constructor() {
owner = msg.sender;
}
}
ContractDefinition smarty
0 reference(s)
0
listen on network
Search with transaction hash or address

Deploy it

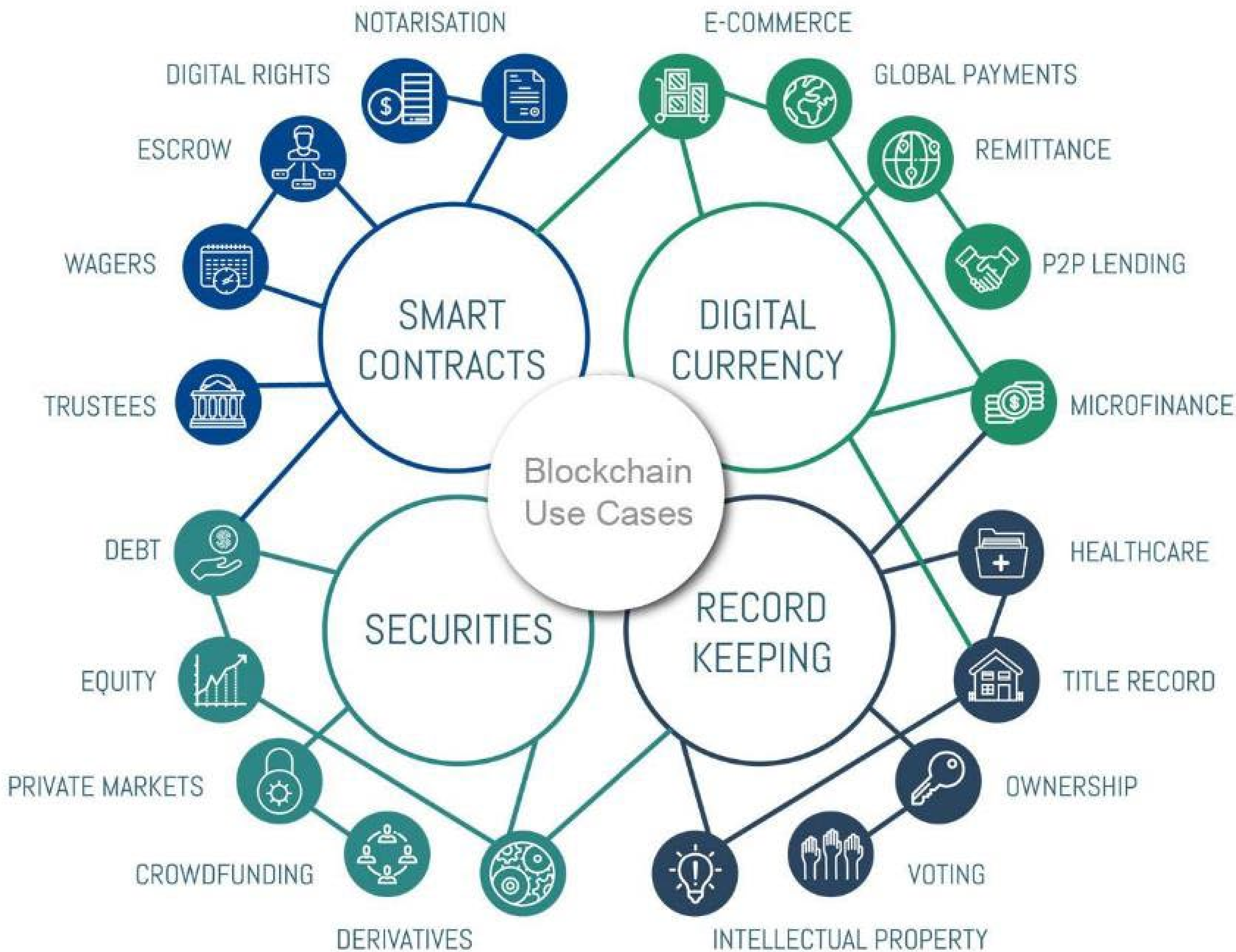
- … on a Javascript sandbox blockchain



Ask who the owner is



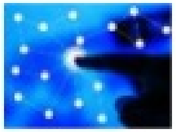
This should be the same of the account number in previous slide


NOTARISATION
E-COMMERCE
DIGITAL RIGHTS
GLOBAL PAYMENTS
ESCROW
REMITTANCE
WAGERS
P2P LENDING
SMART CONTRACTS
DIGITAL CURRENCY
TRUSTEES
MICROFINANCE
Blockchain Use Cases
DEBT
HEALTHCARE
SECURITIES
RECORD KEEPING
EQUITY
TITLE RECORD
PRIVATE MARKETS
OWNERSHIP
CROWDFUNDING
VOTING
DERIVATIVES
INTELLECTUAL PROPERTY

BLOCKCHAIN CHARACTERISTICS & BENEFITS

Public ledger system that records & validates	Secure & reliable, transparent
Transactions authorized my miners	Immutable, hacking-resistant
Removes need for intermediaries	Real-time settlement, operational cost savings
Distributed	Reduced risk

Easier for law enforcement to trace than cash, gold or diamonds



Blockchains aren't for everyone and they aren't for every solution. Here is a five point test.

1	Are there multiple parties in this ecosystem?	• Blockchains get more secure with more parties in the network. One participant networks are not especially secure.
2	Is establishing trust between all the parties an issue?	• Blockchains improve trust between participants by having multiple points of verification
3	Is it critical to have a tamper-proof permanent record of transactions?	• Blockchains create permanent records that cannot edited or deleted.
4	Are we securing the ownership or management of a finite resource?	• Core logic in the system is designed to prevent double counting of assets and record ownership and transfers.
5	Does this ecosystem benefit from improved transparency?	• Blockchains are transparent by design – where ownership or control of assets is public and transparent by design.

Cassiopeia
House of Computer Science

www.cs.aau.dk

END

AALBORG UNIVERSITET