Bitcoin

Cloud Security Mechanisms Seminar

Johannes Henning, Robin Schreiber

What is Bitcoin

- Digital virtual currency
  - Allows to perform exchange of monetary unions digitally, over a network
- Decentralized
  - No central, controlling entity that manages the network
  - Everyone agrees upon one protocol
- Anonymous (to some extent)
  - As long as the public key can not be attributed to a persons identity
- Secure

What's the challenge?

- Consensus Problem
    - Achieve overall system reliability in a distributed system in the presence of faulty processors
    - Byzantine Generals‘ problem:
        - To guarantee success, needs to attack the castle with at least ½ of the available other generals
        - Can only communicate by messaging
        - How to make sure that consensus is reached without spies invalidating the consensus?
- Bitcoin has to achieve Byzantine fault tolerance to be secure!

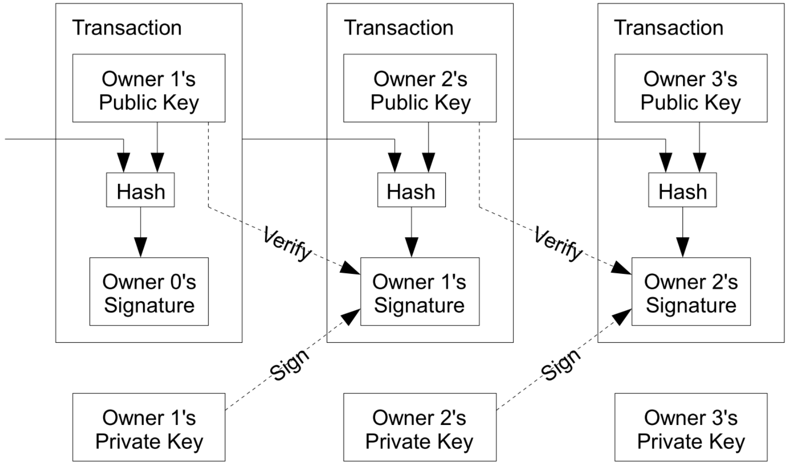
Concept

Implementation

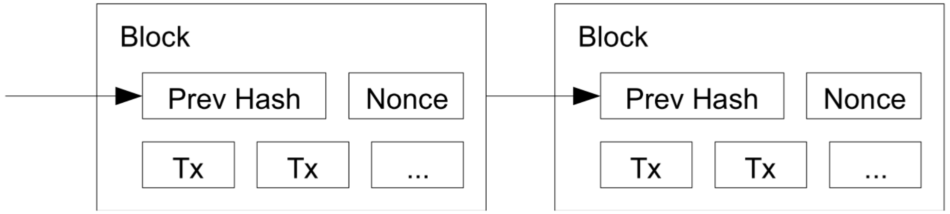
Discussion

Bitcoins three major Concepts

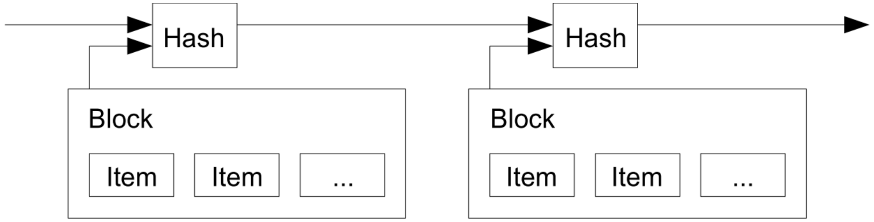
- Transactions



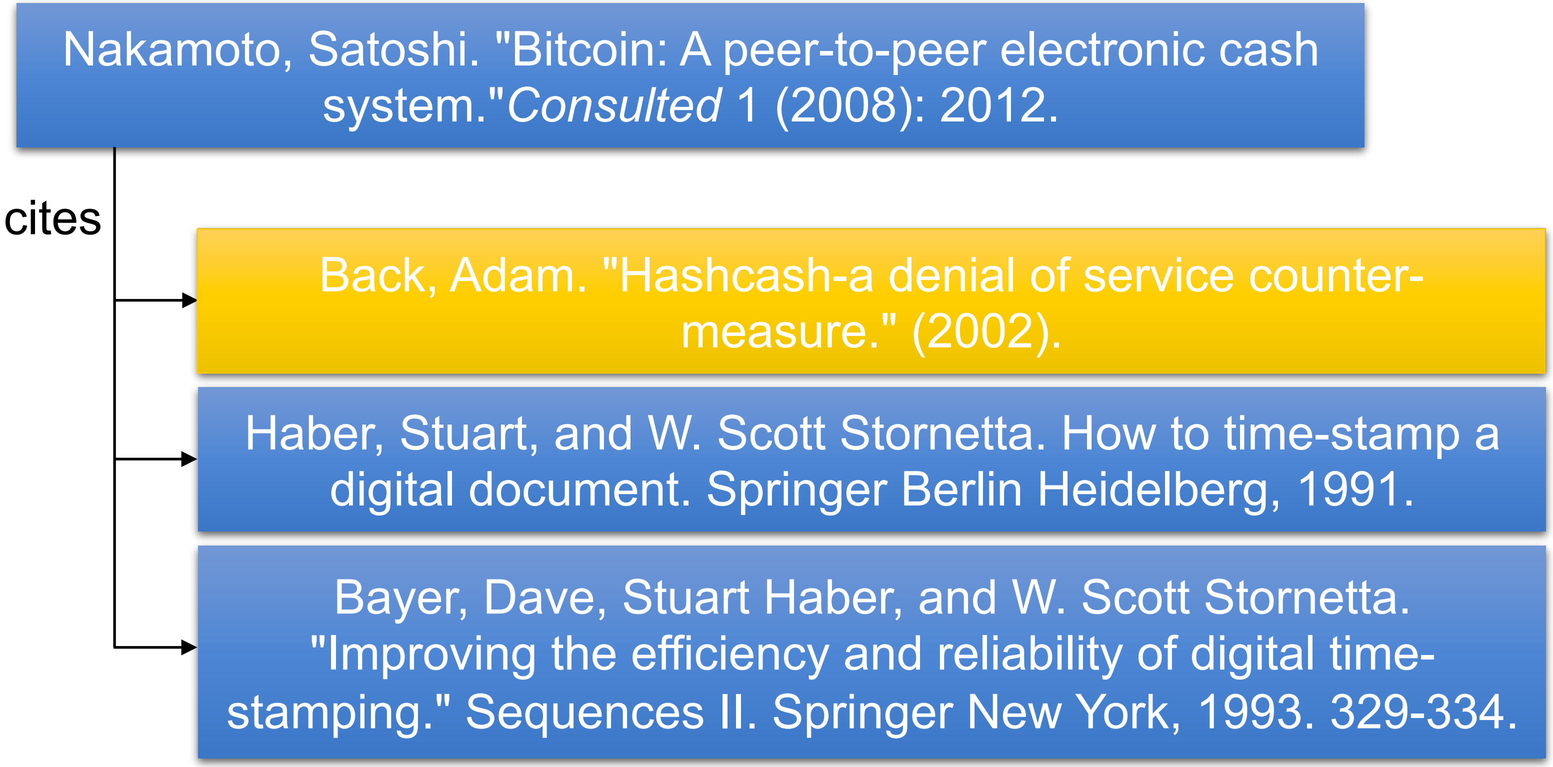
- Proof of work



- Timestamps



Proof of Work



Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system."*Consulted* 1 (2008): 2012.

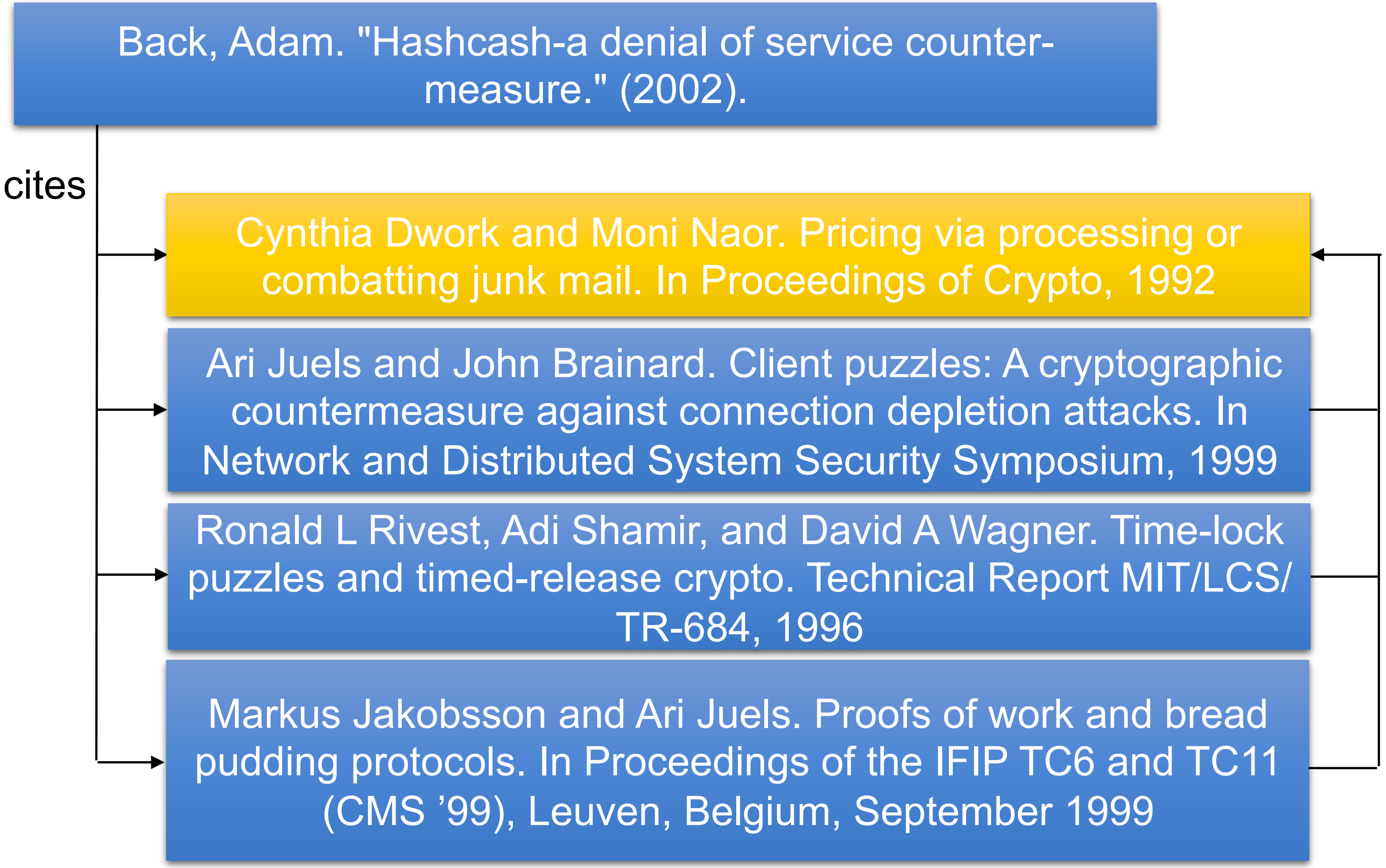
cites

Back, Adam. "Hashcash-a denial of service counter-measure." (2002).

Haber, Stuart, and W. Scott Stornetta. How to time-stamp a digital document. Springer Berlin Heidelberg, 1991.

Bayer, Dave, Stuart Haber, and W. Scott Stornetta. "Improving the efficiency and reliability of digital time-stamping." Sequences II. Springer New York, 1993. 329-334.

Proof of Work



Back, Adam. "Hashcash-a denial of service counter-measure." (2002).

cites

Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Proceedings of Crypto, 1992

Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In Network and Distributed System Security Symposium, 1999

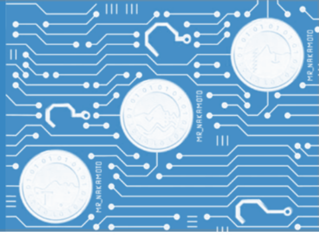
Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, 1996

Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Proceedings of the IFIP TC6 and TC11 (CMS '99), Leuven, Belgium, September 1999

Cynthia Dwork and Moni Naor. **Pricing via processing or combatting junk mail**. In Proceedings of Crypto, 1992

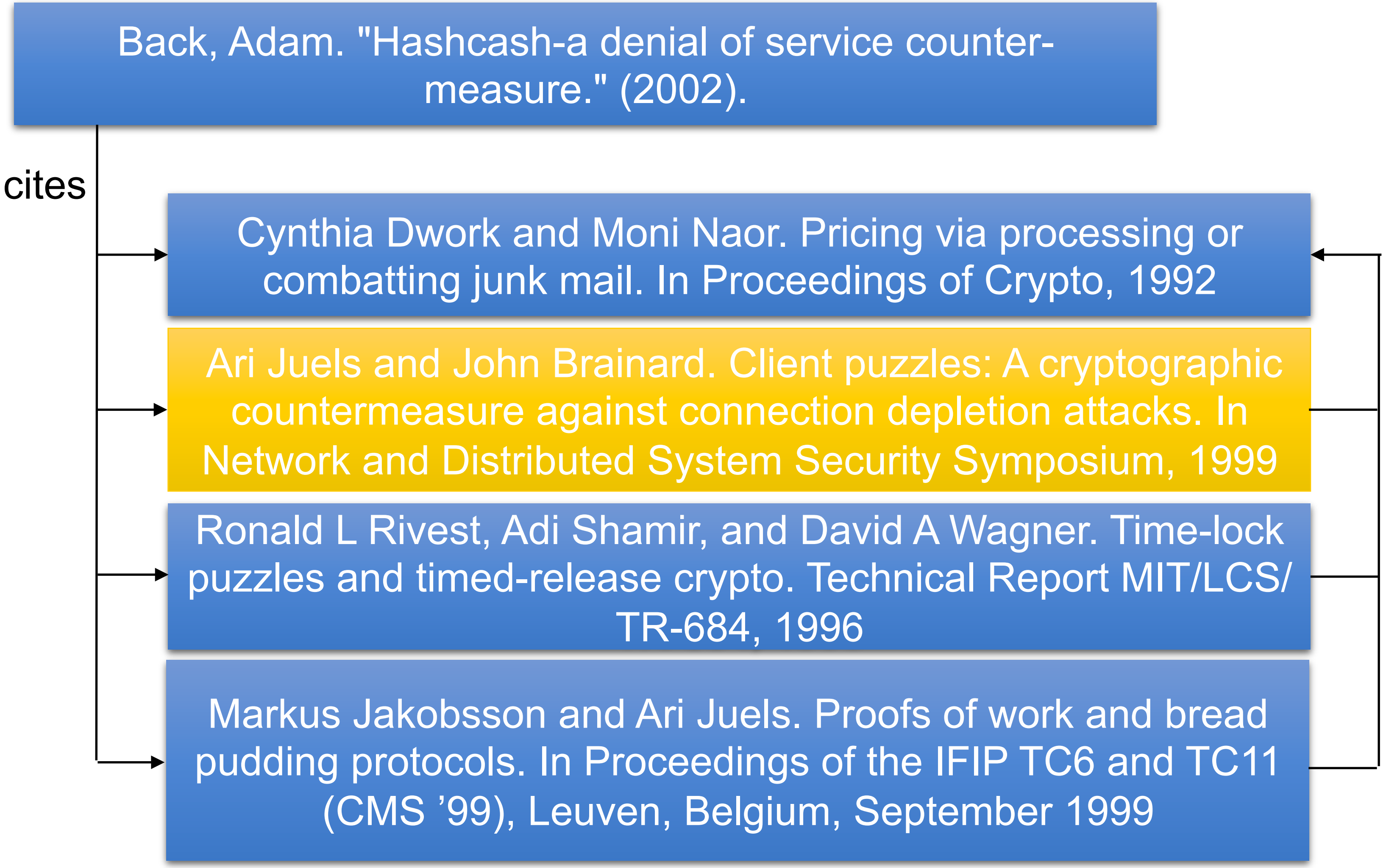
- Commonly acknowledged as the first idea of „Proof of work“ (POW) called pricing function
- Main Problem with email is, that it can be sent at practically no cost
  - Therefor more spam than with regular mail
  - Postage is needed
  - Real money cost for email does not seem practical
  - Computational cost as currency

Cynthia Dwork and Moni Naor. **Pricing via processing or combatting junk mail**. In Proceedings of Crypto, 1992



- Pricing function f
  - f is moderately easy to compute
  - f is not amendable to amortization
  - given x and y it is easy to compute y = f(x)
  - Suggested functions
    - Extracting Square roots module a prime number
    - Signature Schemes
  - trapdoor for trusted senders
  - non interactive
- Uselessness of computation identified as problem

Proof of Work



Back, Adam. "Hashcash-a denial of service counter-measure." (2002).

cites

Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Proceedings of Crypto, 1992

Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In Network and Distributed System Security Symposium, 1999

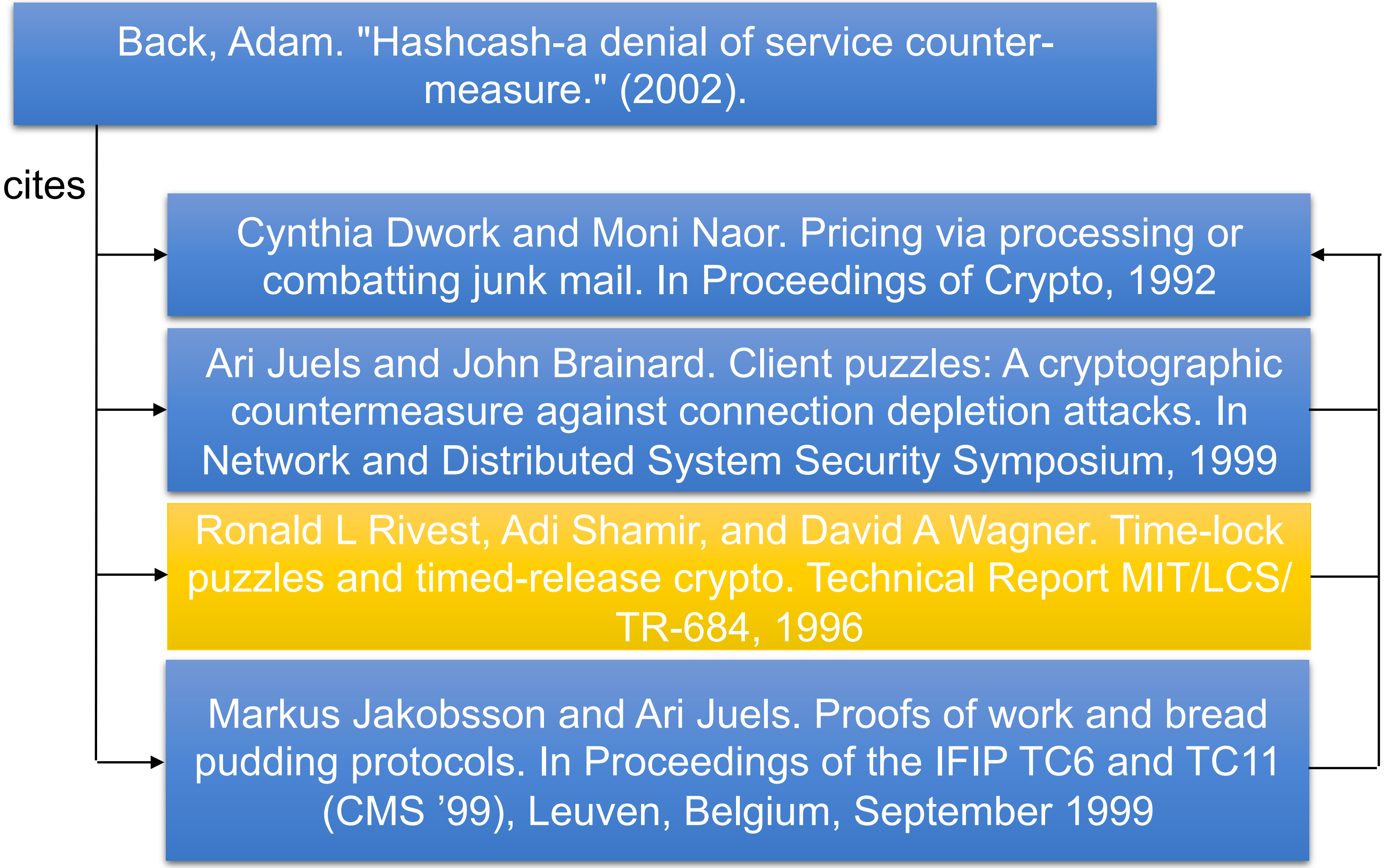
Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, 1996

Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Proceedings of the IFIP TC6 and TC11 (CMS ’99), Leuven, Belgium, September 1999

Ari Juels and John Brainard. **Client puzzles: A cryptographic countermeasure against connection depletion attacks.** In Network and Distributed System Security Symposium, 1999

- Use POW to create DOS Save protocols
- When the server load gets critical, distribute puzzles to the clients
- Client has to solve puzzle to have his request processed
- Puzzle consists of computing a partial hash collision
- This can be done stateless

Proof of Work



Back, Adam. "Hashcash-a denial of service counter-measure." (2002).

cites

Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Proceedings of Crypto, 1992

Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In Network and Distributed System Security Symposium, 1999

Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, 1996

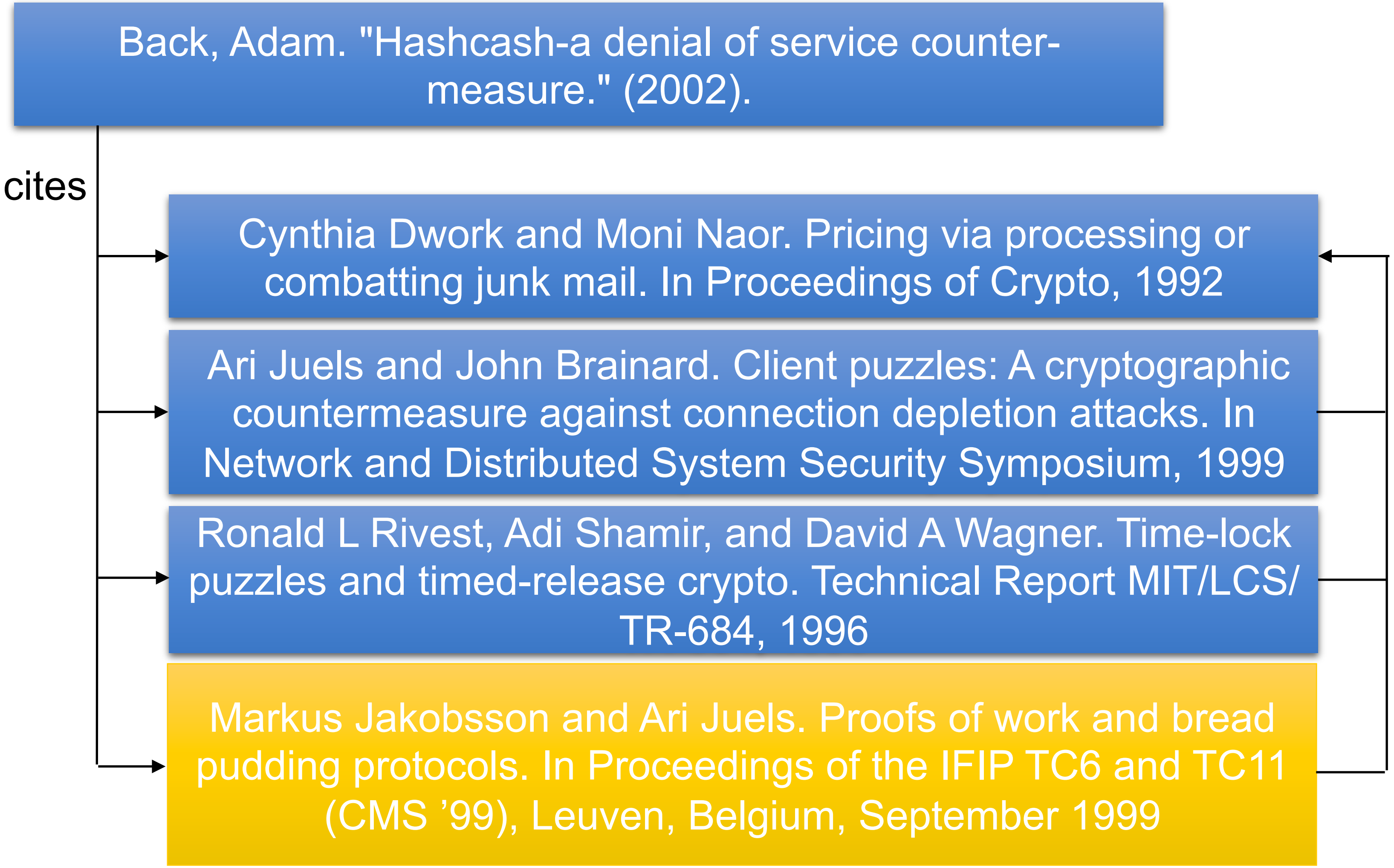
Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Proceedings of the IFIP TC6 and TC11 (CMS ’99), Leuven, Belgium, September 1999

Ronald L Rivest, Adi Shamir, and David A Wagner. **Time-lock puzzles and timed-release crypto**. Technical Report MIT/LCS/TR-684, 1996

- Different approach to POW:
  - Crypt information in a way that it cannot be decrypted before a certain amount of time has passed
- Real world approach: use trusted party
- Naive digital approach: Use hash collision based POW
  - Problem: Time necessary to compute solution is dependent on computing resources
  - Trivial to parallelize
- Crypt based on RC5
  - To solve it iterative squaring of a number is required

⇨ Not parallelizable

Proof of Work



Back, Adam. "Hashcash-a denial of service counter-measure." (2002).

cites

Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Proceedings of Crypto, 1992

Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In Network and Distributed System Security Symposium, 1999

Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, 1996

Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Proceedings of the IFIP TC6 and TC11 (CMS ’99), Leuven, Belgium, September 1999

Markus Jakobsson and Ari Juels. **Proofs of work and bread pudding protocols**. In Proceedings of the IFIP TC6 and TC11 (CMS ’99), Leuven, Belgium, September 1999

- In security algorithms proof protocols are essential
  - e. g. proofing you have a secret corresponding to authenticated public key
- POW aims to show that a certain amount of computation was performed in a particular interval of time
- Problem with POW is that the required computations far exceeds that of conventional cryptographic functions
- POW already proposed for:
  - access metering, time capsules, uncheatable benchmarks, spam protection, DOS

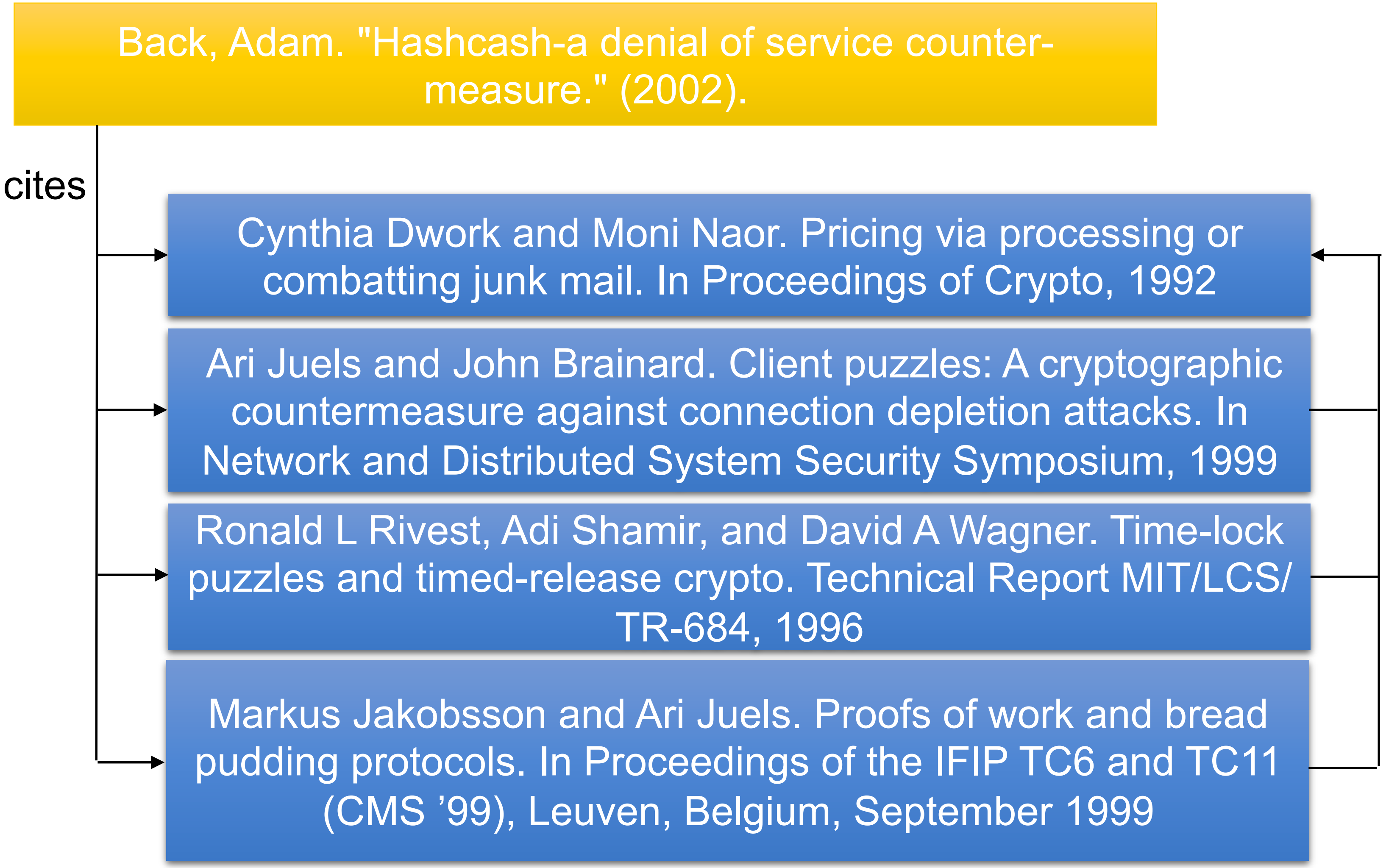
Markus Jakobsson and Ari Juels. **Proofs of work and bread pudding protocols**. In Proceedings of the IFIP TC6 and TC11 (CMS ’99), Leuven, Belgium, September 1999

- Improvements of Dwork and Naor approach
  - infeasible precomputation
  - loose complexity categorization
  - interleaving
  - interactive protocols
- Bread pudding Protocol
  - Partition POW into smaller POWs
  - Clients themselves distribute small POWs to their clients

Other related ideas

- Puzzle Auctions
  - Tune the difficulty of the puzzles depending on the capabilities of the clients and workload of the server
- Offline karma
  - Guarantee fairness in a P2P network
  - Karma proposed central managed currency for Gnutella or Napster
  - Offline karma is a decentralized version of karma
    - Main difficulty is to avoid double spending, done by reminting
- Memory bound functions
  - When POW is dependent on CPU speed, fairness might suffer
  - Especially considering emails, spammer might be able to access far more computing power than regular users
  - See also „ ‚Prove of work‘ Proves Not to Work“

Proof of Work



Back, Adam. "Hashcash-a denial of service counter-measure." (2002).

cites

Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Proceedings of Crypto, 1992

Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In Network and Distributed System Security Symposium, 1999

Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, 1996

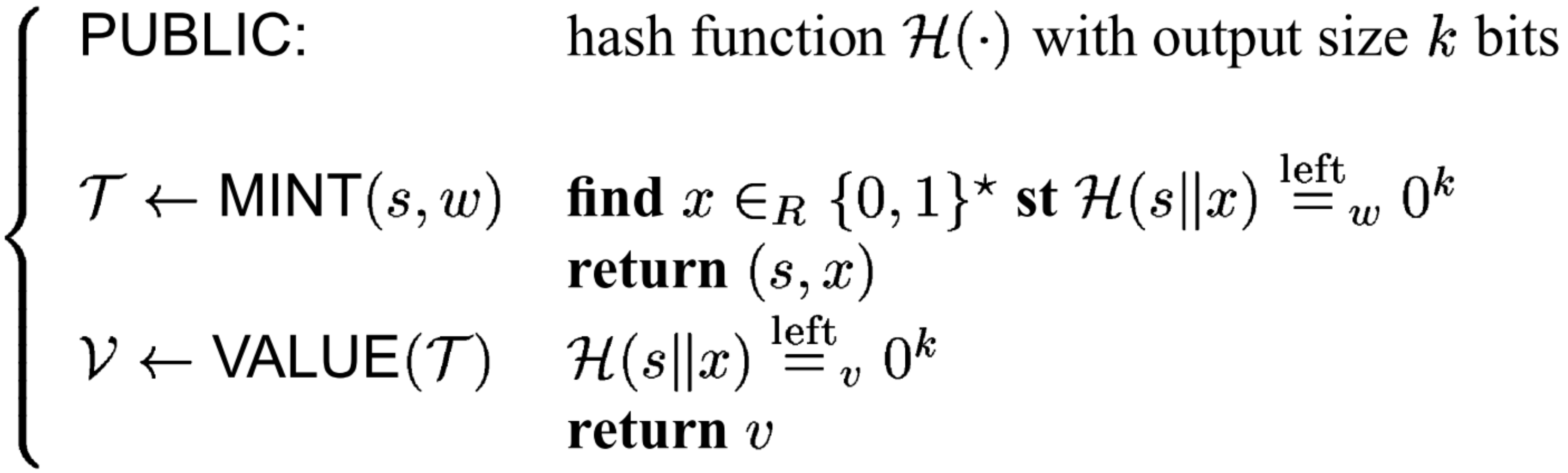
Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Proceedings of the IFIP TC6 and TC11 (CMS '99), Leuven, Belgium, September 1999

Back, Adam. "" (2002). **Hashcash-a denial of service counter-measure.**

- Originally to throttle abuse of un-metered internet resources like email
- Digital currency is achieved by computing a cost function which is called minting (prägen)
  - functions are parameterized by the amount of work the user has to do
  - interactive/non-interactive
  - trapdoor free
  - publicly auditable
  - unbounded probabilistic cost

Back, Adam. "" (2002). **Hashcash-a denial of service counter-measure.**

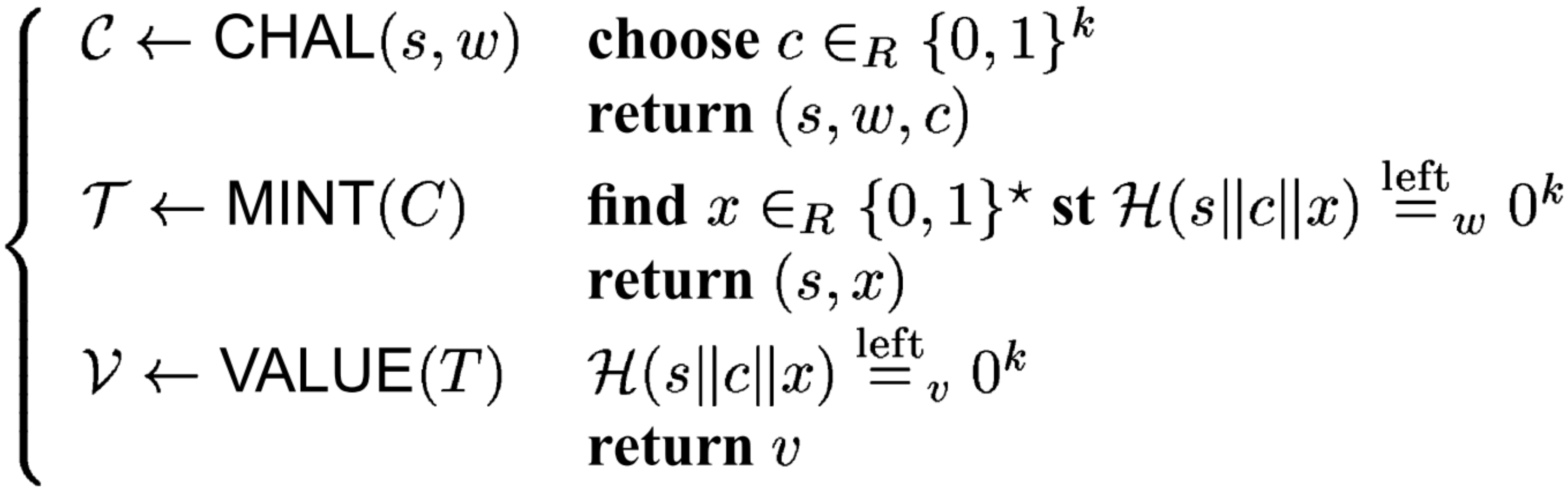
- The original Hashcash function



$$\left\{\begin{array}{ll}
\textsf{PUBLIC:} & \text{hash function } \mathcal{H}(\cdot) \text{ with output size } k \text{ bits} \\
 & \\
\mathcal{T} \leftarrow \textsf{MINT}(s, w) & \textbf{find } x \in_R \{0,1\}^\star \textbf{ st } \mathcal{H}(s \| x) \overset{\text{left}}{=}_w 0^k \\
 & \textbf{return } (s, x) \\
\mathcal{V} \leftarrow \textsf{VALUE}(\mathcal{T}) & \mathcal{H}(s \| x) \overset{\text{left}}{=}_v 0^k \\
 & \textbf{return } v
\end{array}\right.$$

Back, Adam. "" (2002). **Hashcash-a denial of service counter-measure.**

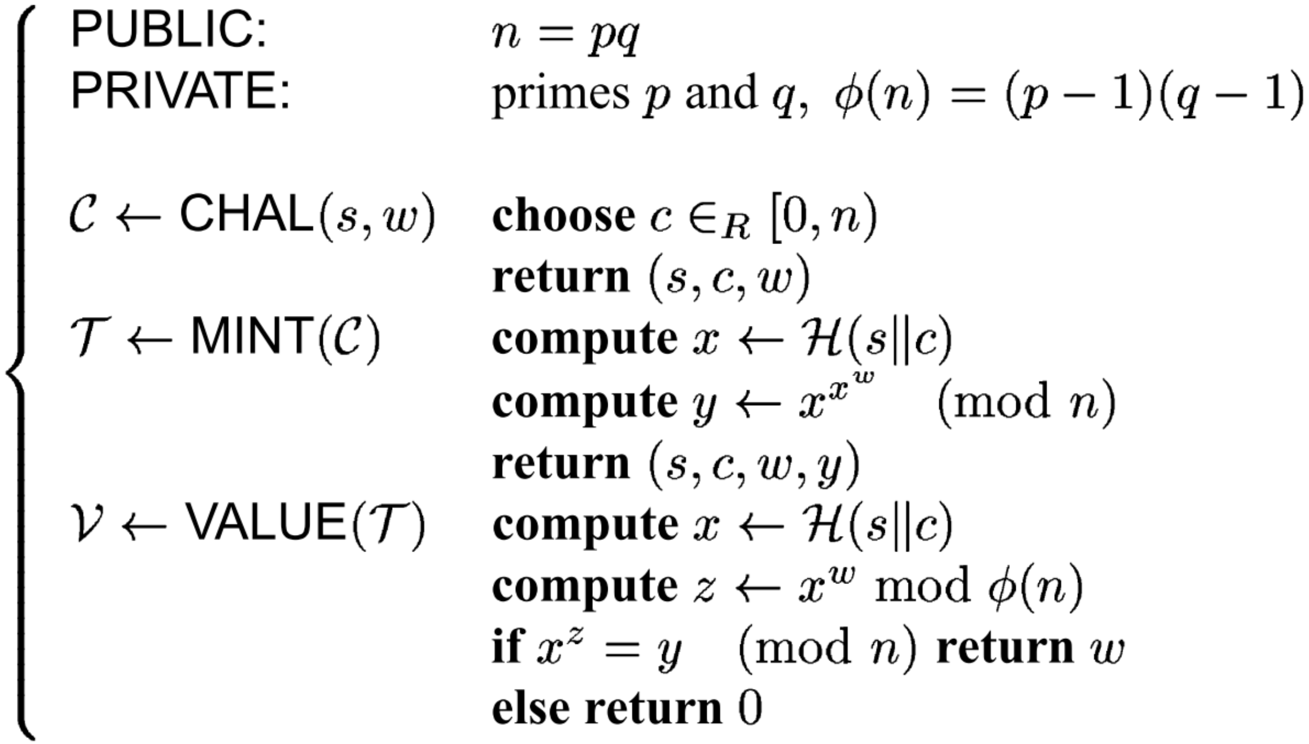
- The interactive Hashcash function



$$\left\{\begin{array}{ll}
\mathcal{C} \leftarrow \mathsf{CHAL}(s,w) & \textbf{choose } c \in_R \{0,1\}^k \\
 & \textbf{return } (s,w,c) \\
\mathcal{T} \leftarrow \mathsf{MINT}(C) & \textbf{find } x \in_R \{0,1\}^\star \textbf{ st } \mathcal{H}(s\|c\|x) \overset{\text{left}}{=}_w 0^k \\
 & \textbf{return } (s,x) \\
\mathcal{V} \leftarrow \mathsf{VALUE}(T) & \mathcal{H}(s\|c\|x) \overset{\text{left}}{=}_v 0^k \\
 & \textbf{return } v
\end{array}\right.$$

Back, Adam. "" (2002). **Hashcash-a denial of service counter-measure.**

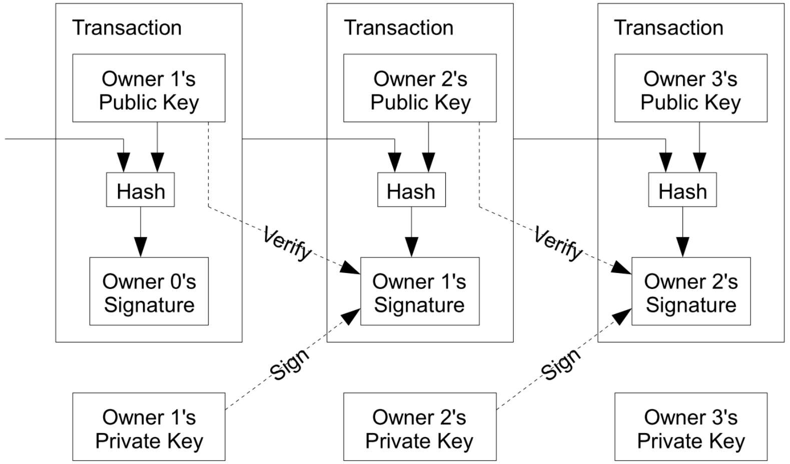
- The non parallel low variance Hashcash function



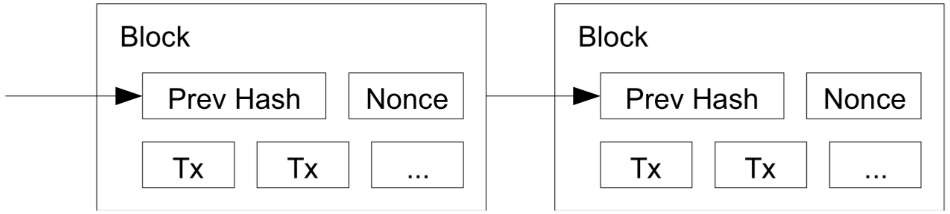
$$\left\{\begin{array}{ll}
\text{PUBLIC:} & n = pq \\
\text{PRIVATE:} & \text{primes } p \text{ and } q,\ \ \phi(n) = (p-1)(q-1) \\
 & \\
\mathcal{C} \leftarrow \text{CHAL}(s,w) & \textbf{choose } c \in_R [0,n) \\
 & \textbf{return } (s,c,w) \\
\mathcal{T} \leftarrow \text{MINT}(\mathcal{C}) & \textbf{compute } x \leftarrow \mathcal{H}(s\|c) \\
 & \textbf{compute } y \leftarrow x^{x^w} \pmod{n} \\
 & \textbf{return } (s,c,w,y) \\
\mathcal{V} \leftarrow \text{VALUE}(\mathcal{T}) & \textbf{compute } x \leftarrow \mathcal{H}(s\|c) \\
 & \textbf{compute } z \leftarrow x^w \bmod \phi(n) \\
 & \textbf{if } x^z = y \pmod{n} \textbf{ return } w \\
 & \textbf{else return } 0
\end{array}\right.$$

Bitcoins three major Concepts

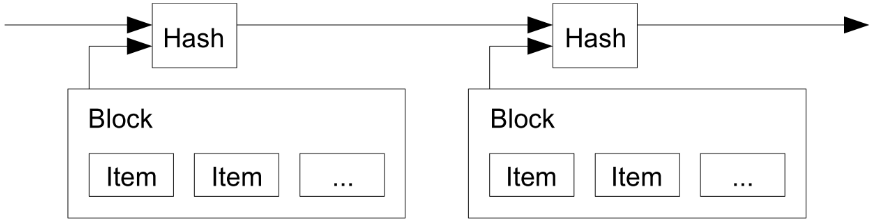
- Transactions



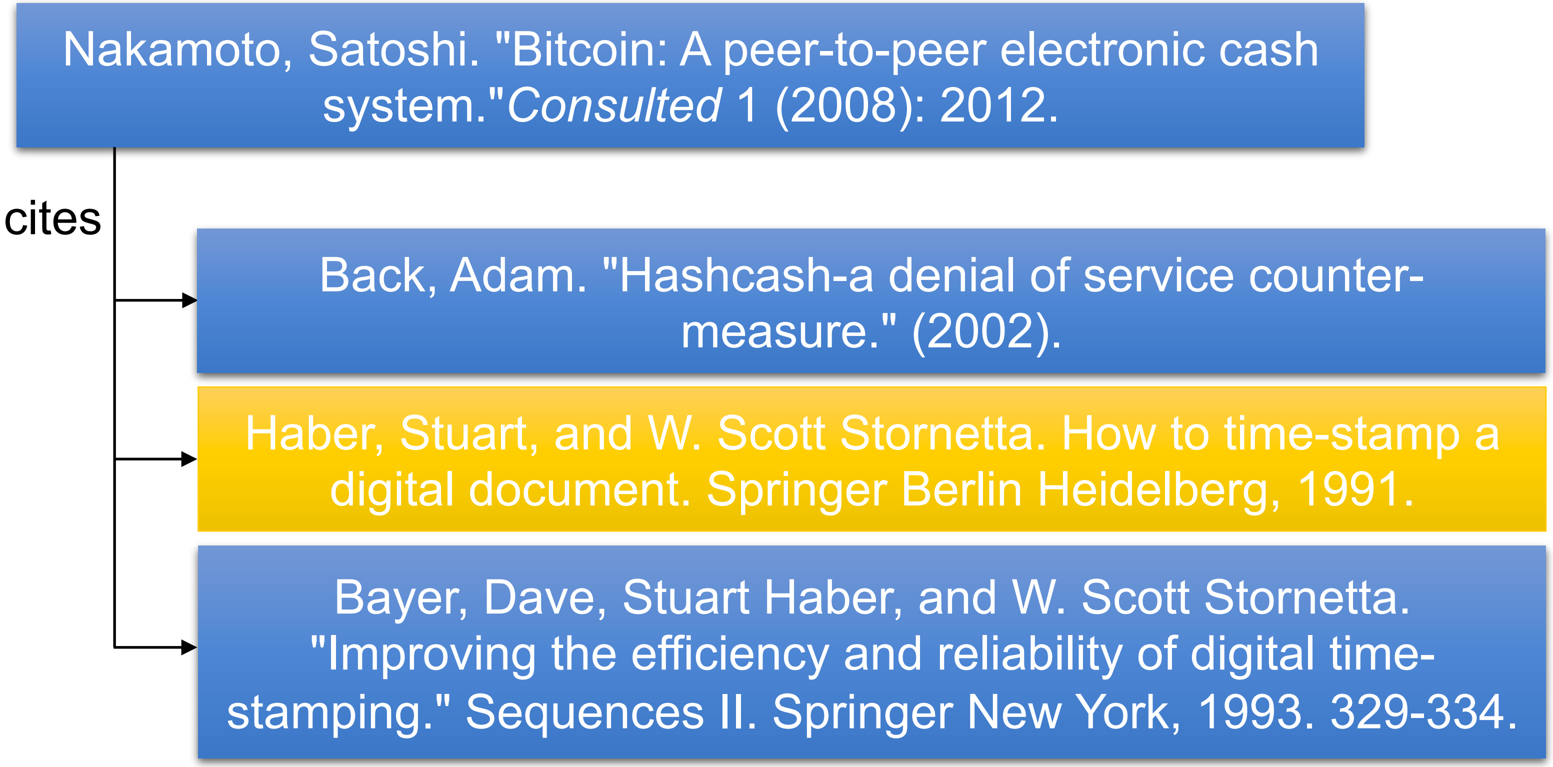
- Proof of work



- Timestamps



Timestamps



Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system."*Consulted* 1 (2008): 2012.

cites

Back, Adam. "Hashcash-a denial of service counter-measure." (2002).

Haber, Stuart, and W. Scott Stornetta. How to time-stamp a digital document. Springer Berlin Heidelberg, 1991.

Bayer, Dave, Stuart Haber, and W. Scott Stornetta. "Improving the efficiency and reliability of digital time-stamping." Sequences II. Springer New York, 1993. 329-334.

Haber, Stuart, and W. Scott Stornetta. **How to time-stamp a digital document**. Springer Berlin Heidelberg, 1991.

- Digital documents are becoming more important
  - are even being introduced into the legal domain
- Analog documents can be dated by an authority and are hard to tamper with
- Digital documents cannot be reliably timestamped because they are not bound to a physical medium and are easy to manipulate
- Naive Solution: TSS (Trusted Time-Stamping Service) stores the document
  - Problems:
    - privacy, bandwidth, incompetence, trust

Haber, Stuart, and W. Scott Stornetta. **How to time-stamp a digital document**. Springer Berlin Heidelberg, 1991.

- Proposed solution
  - Hash the document
    - Hash function is easy to compute
    - Infeasible to fine x, x‘ so that h(x) = h(x‘)
  - Digital signature
    - Parties can validate timestamps
  - Linking & Distribution

Haber, Stuart, and W. Scott Stornetta. **How to time-stamp a digital document**. Springer Berlin Heidelberg, 1991.

- Linking & Distribution

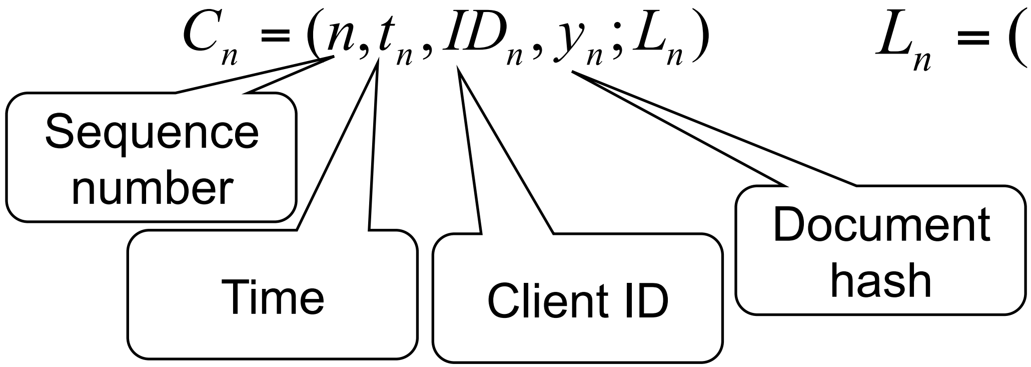
  > *„If an event was determined by certain earlier events, and determines certain subsequent events, then the event is sandwiched securely into its place in history“*

  - Signed certificates are chained into a linear list
    - infeasible to insert into or delete from

Haber, Stuart, and W. Scott Stornetta. **How to time-stamp a digital document**. Springer Berlin Heidelberg, 1991.

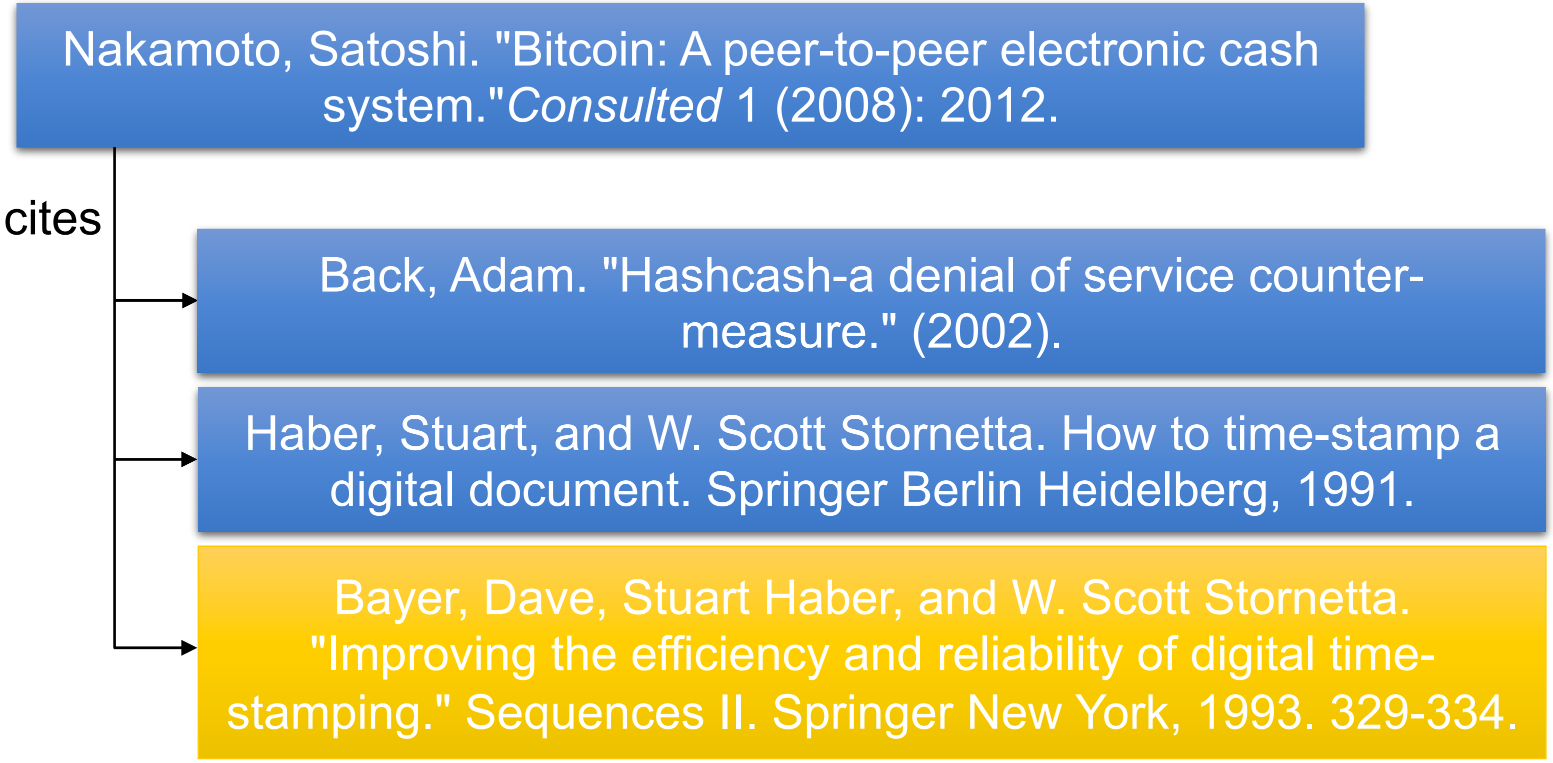
- Linking & Distrubution
  - TSS signs the client certificate c

$$C_n = (n, t_n, ID_n, y_n; L_n) \qquad L_n = (t_{n-1}, ID_{n-1}, y_{n-1}, H(L_{n-1}))$$



  - random-witness solution
    - several members of the client pool must sign the hash
    - members are determined at random (hash is the seed)
      - infeasible to choose clients at will/manipulate choosing

Timestamps



Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system."*Consulted* 1 (2008): 2012.

cites

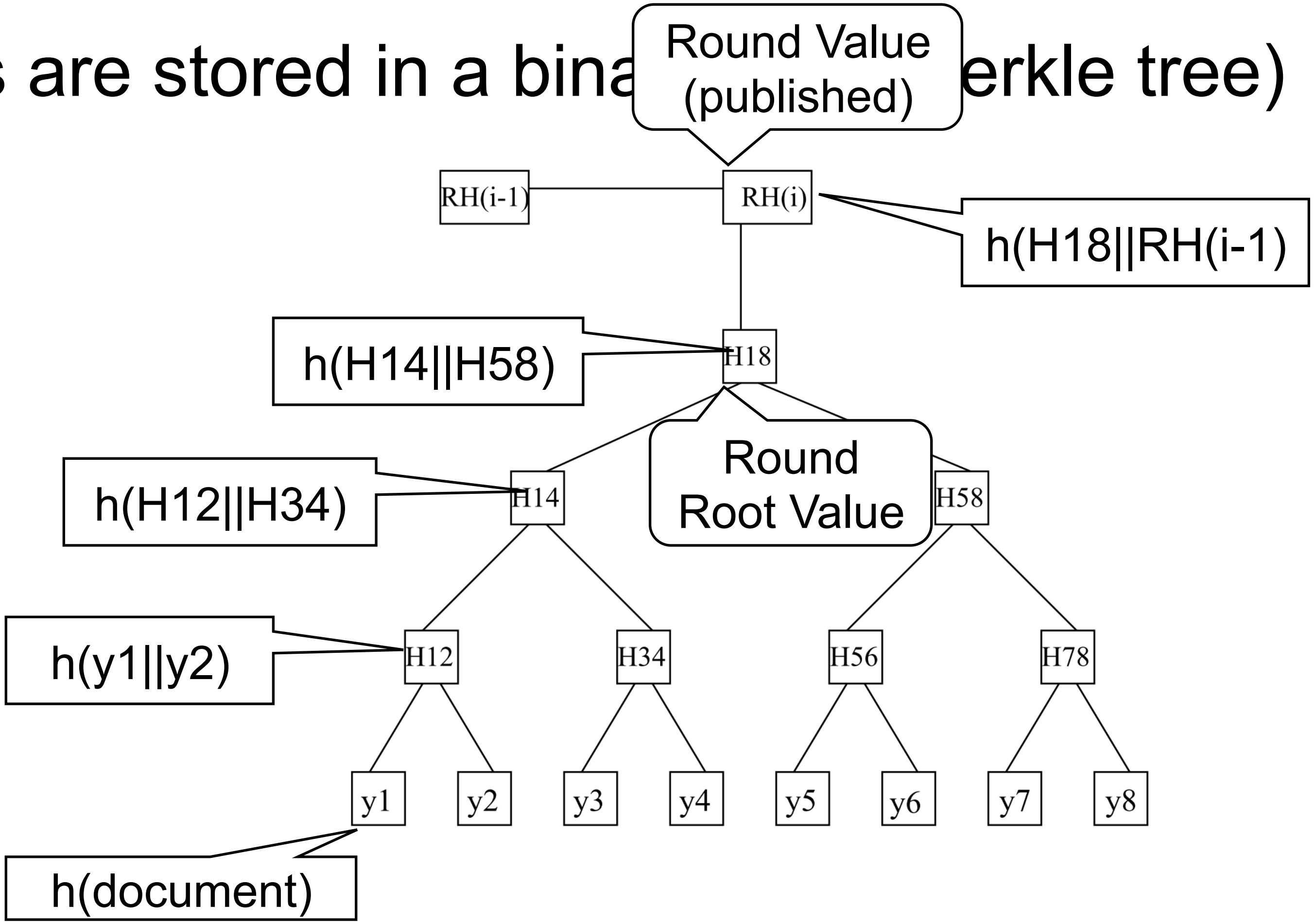
Back, Adam. "Hashcash-a denial of service counter-measure." (2002).

Haber, Stuart, and W. Scott Stornetta. How to time-stamp a digital document. Springer Berlin Heidelberg, 1991.

Bayer, Dave, Stuart Haber, and W. Scott Stornetta. "Improving the efficiency and reliability of digital time-stamping." Sequences II. Springer New York, 1993. 329-334.

Bayer, Dave, Stuart Haber, and W. Scott Stornetta. "**Improving the efficiency and reliability of digital time-stamping**." *Sequences II*. Springer New York, 1993

- Hashes are stored in a bina erkle tree)

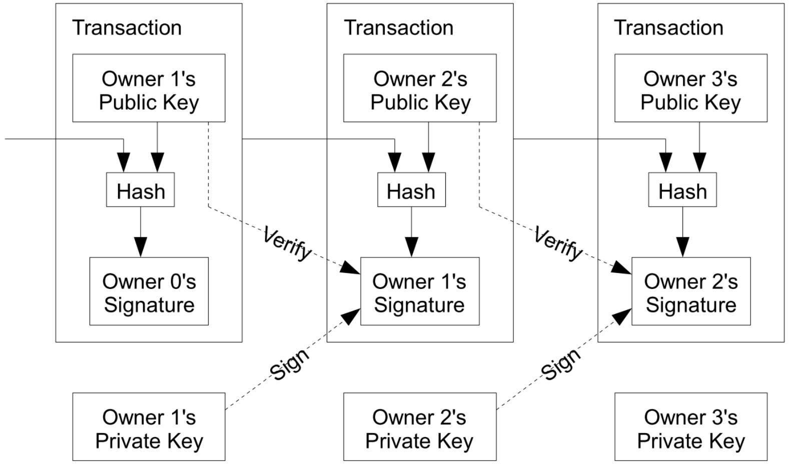


Bayer, Dave, Stuart Haber, and W. Scott Stornetta. "**Improving the efficiency and reliability of digital time-stamping**." *Sequences II*. Springer New York, 1993

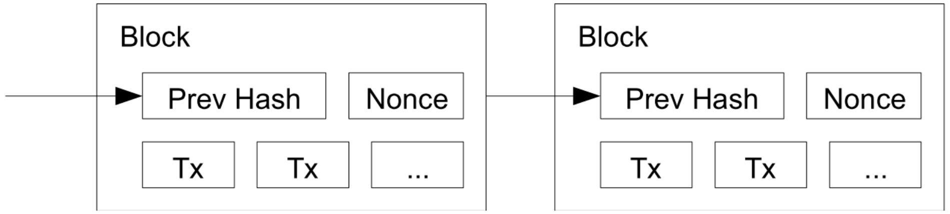
- Aging of hash functions/certificates
  - functions for encryption or hashing that are secure now, may become vulnerable in the future
  - re-timestamp/sign certificates using new functions, before the old one become exploitable
- linear list vs random witness vs tree
- Used as method for URNs

Bitcoins three major Concepts

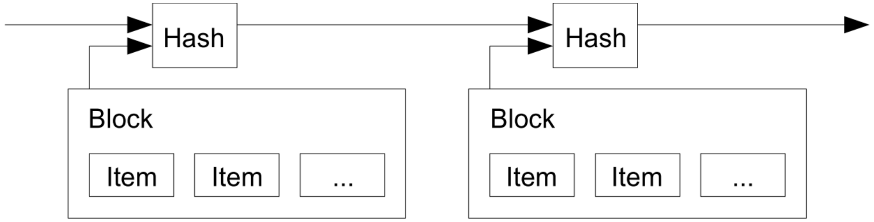
- Transactions



- Proof of work

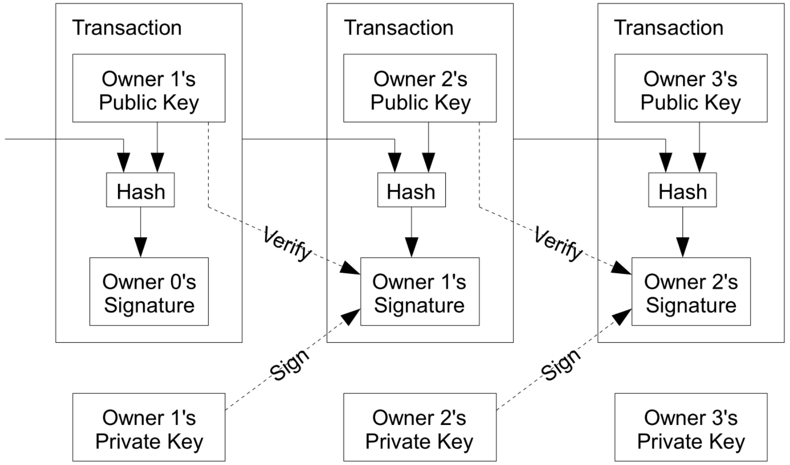


- Timestamps

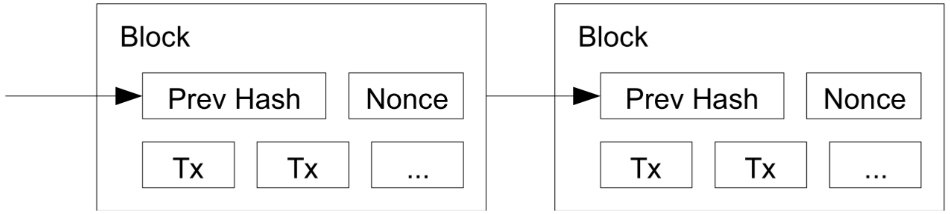


Bitcoins three major Concepts

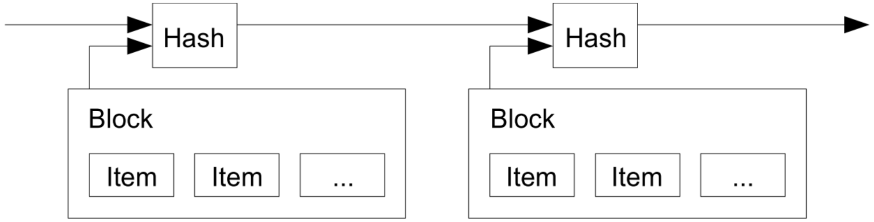
- Transactions



- Proof of work



- Timestamps



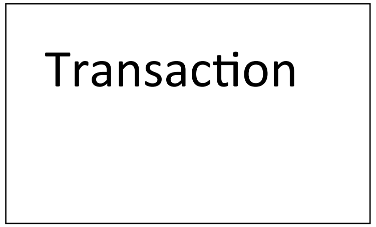
Concept

Implementation

Discussion

Basic Ingredients

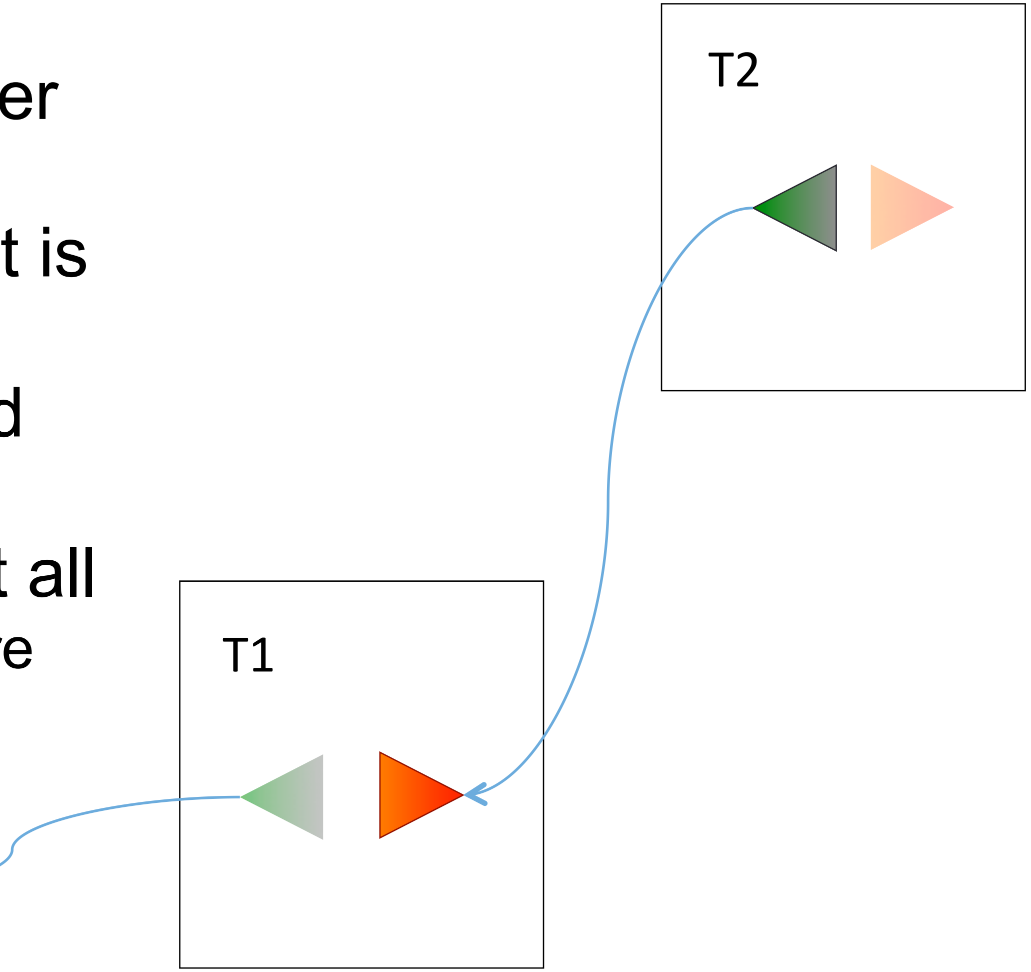
- **Account**
  - Is Public / Private Key Pair
  - Small keysizes (256 Bit) by employing Elliptical Curve Cryptography
  - **If such a pair is lost, all Bitcoins associated with it are gone as well!**
- **Wallet**
  - Is a collection of Accounts
- **Transaction**
  - An Object that represents the transfer of bitcoins between Accounts



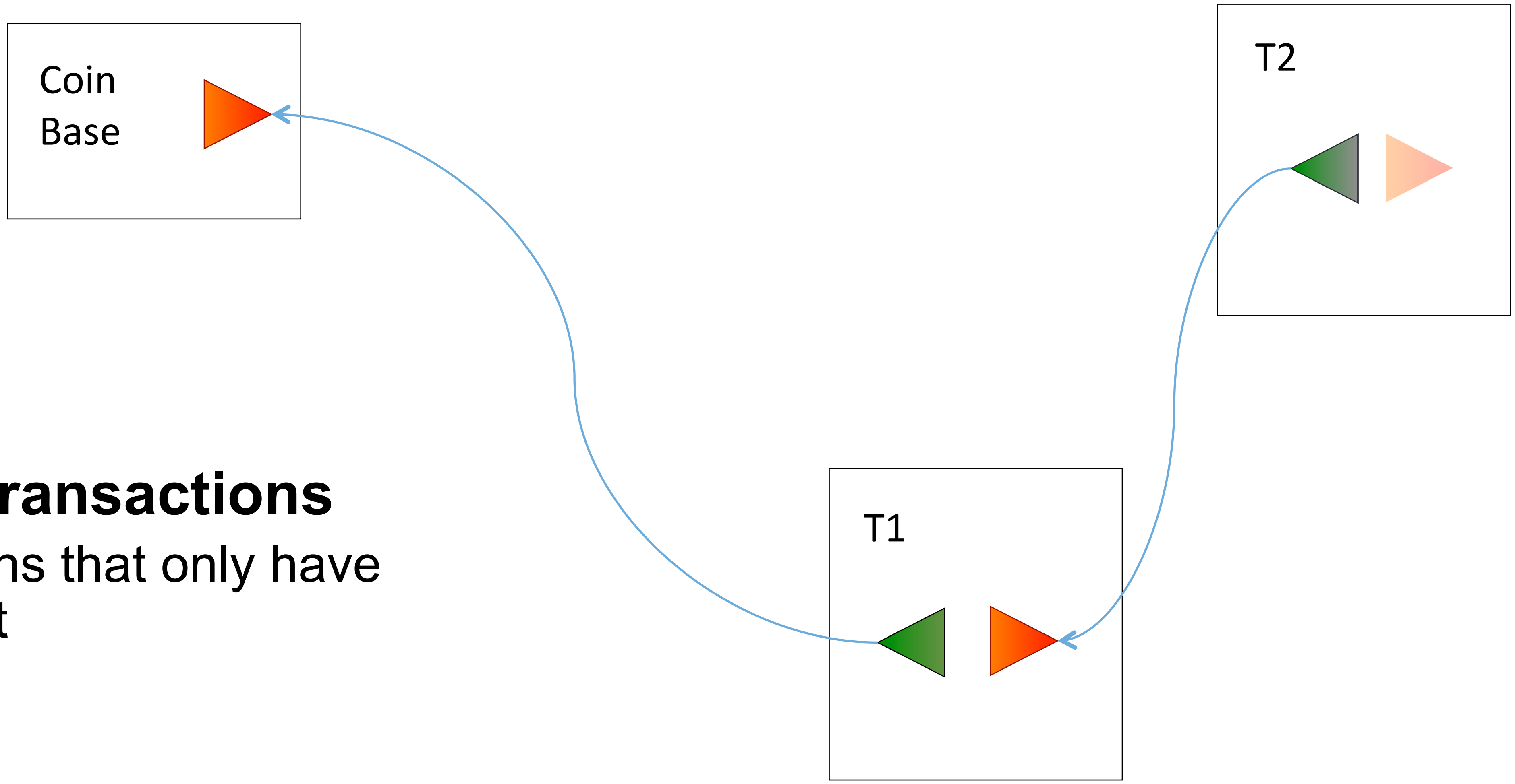
Inputs & Outputs

- **Input**: A reference to the Output of another transaction
- **Output**: A specific amount of bitcoins that is accessible to a specific account
- Transaction may have multiple inputs and outputs
- Outputs are either spent entirely or not at all
  - Sender has to create change Outputs that are adressed to one of his own accounts

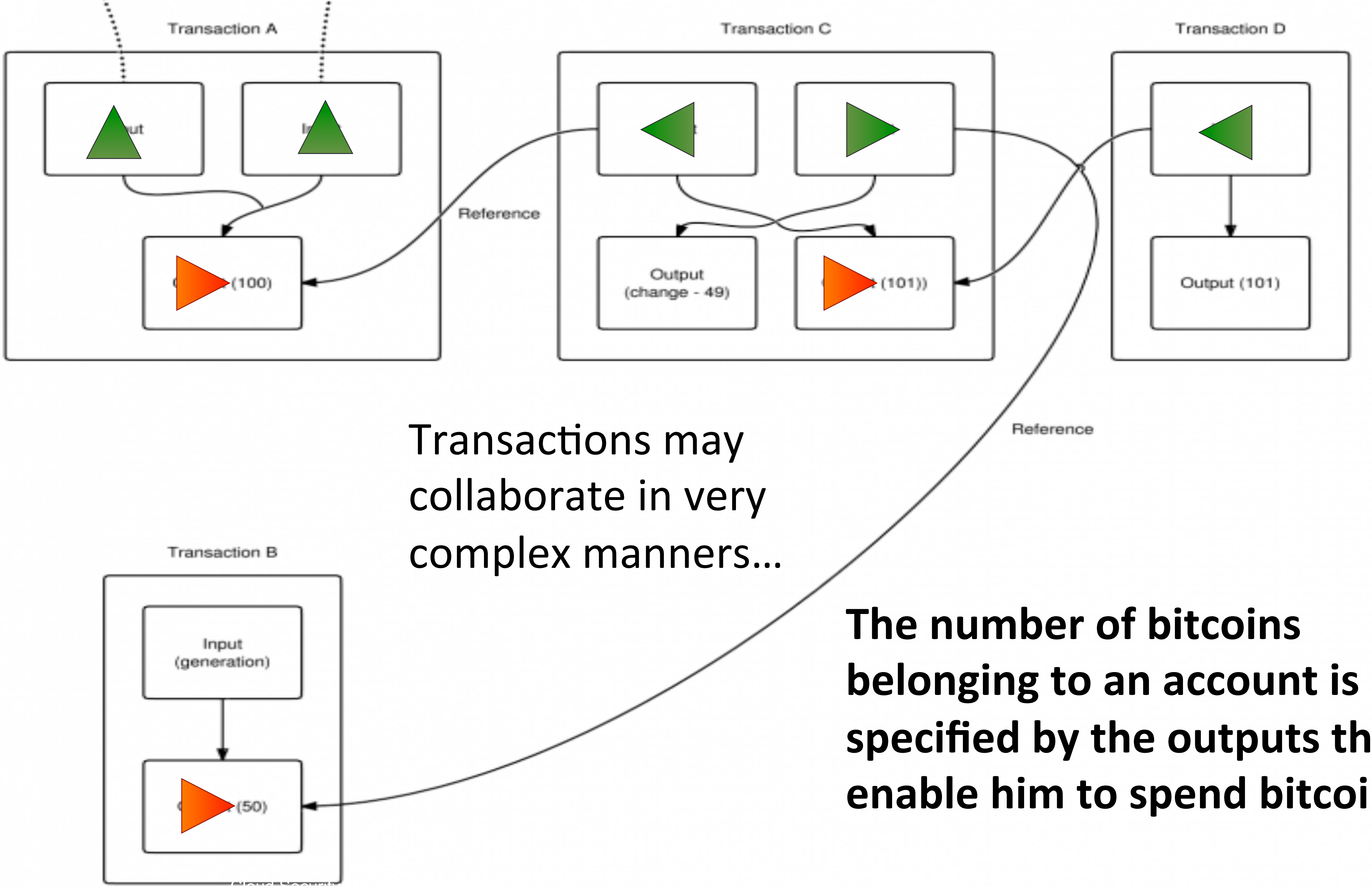
But where do Bitcoins come from?



Inputs & Outputs



- **Coinbase Transactions**
  - Transactions that only have one Output

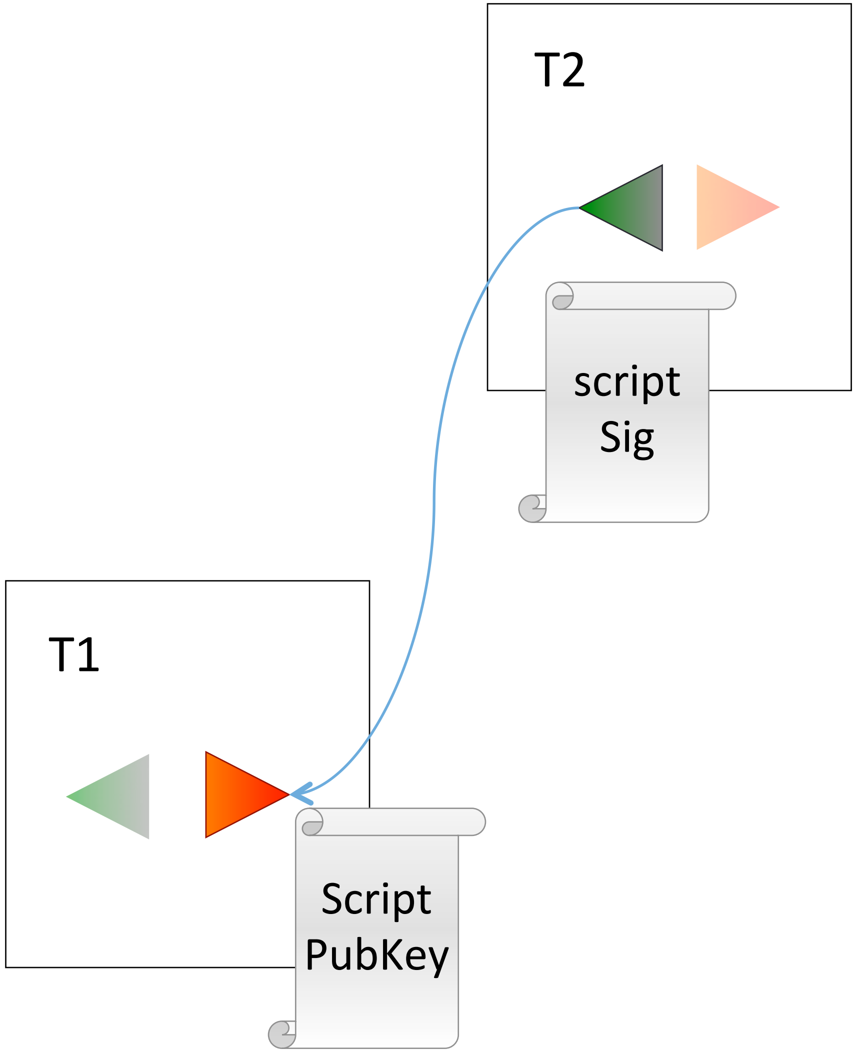


Transactions may collaborate in very complex manners…

The number of bitcoins belonging to an account is specified by the outputs that enable him to spend bitcoins

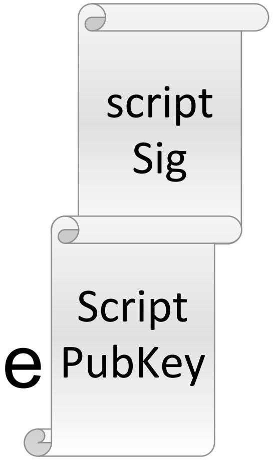
Transaction (verification)

- Transactions are secured by Scripts, which check that their outputs are used only by an authorized spender
- The Spender (Someone using a Transactions output) must provide:
  - A signature to show evidence of the private key corresponding to the public key in the Output
- Spender provides this information by a Script that simply places the arguments on the top of the stack



Transactions (Verification)

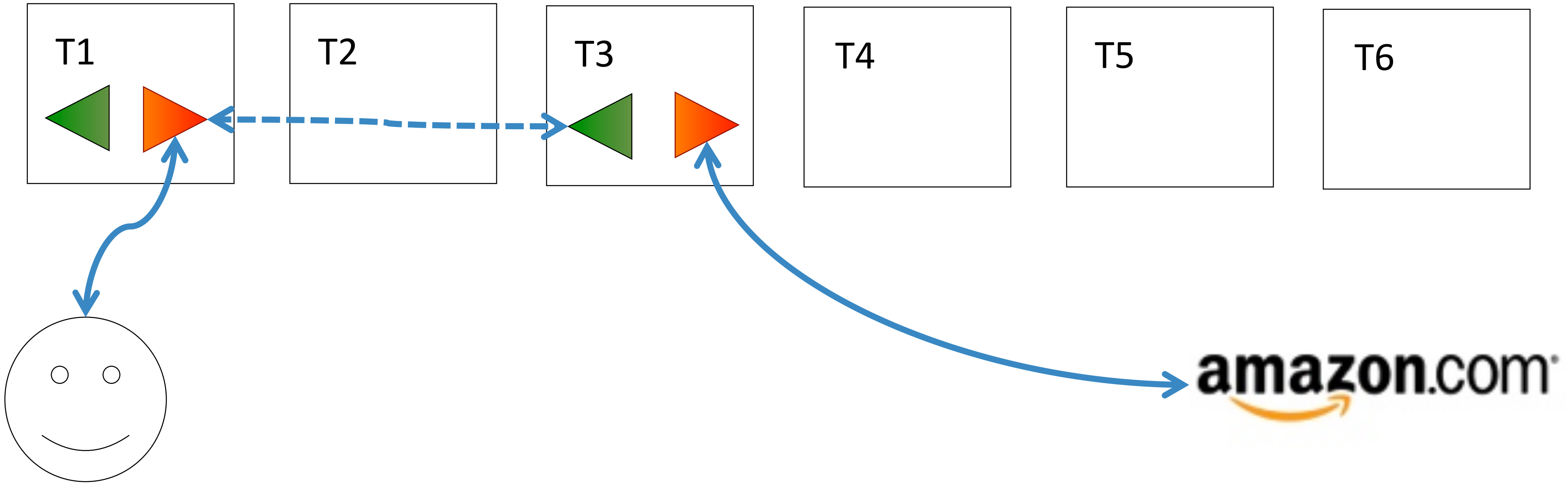
- Transactions are verified by combining these scripts:
  - ***ScriptSig*** of the **Input of the “spending” transaction**
  - ***ScriptPubKey*** of the **Output of the “available” transaction**
- Scripts are formulated in a Forth-like stack based language
  - Not turing complete (for security reasons)



```
scriptPubKey: <pubKey> OP_CHECKSIG
scriptSig: <sig>
```

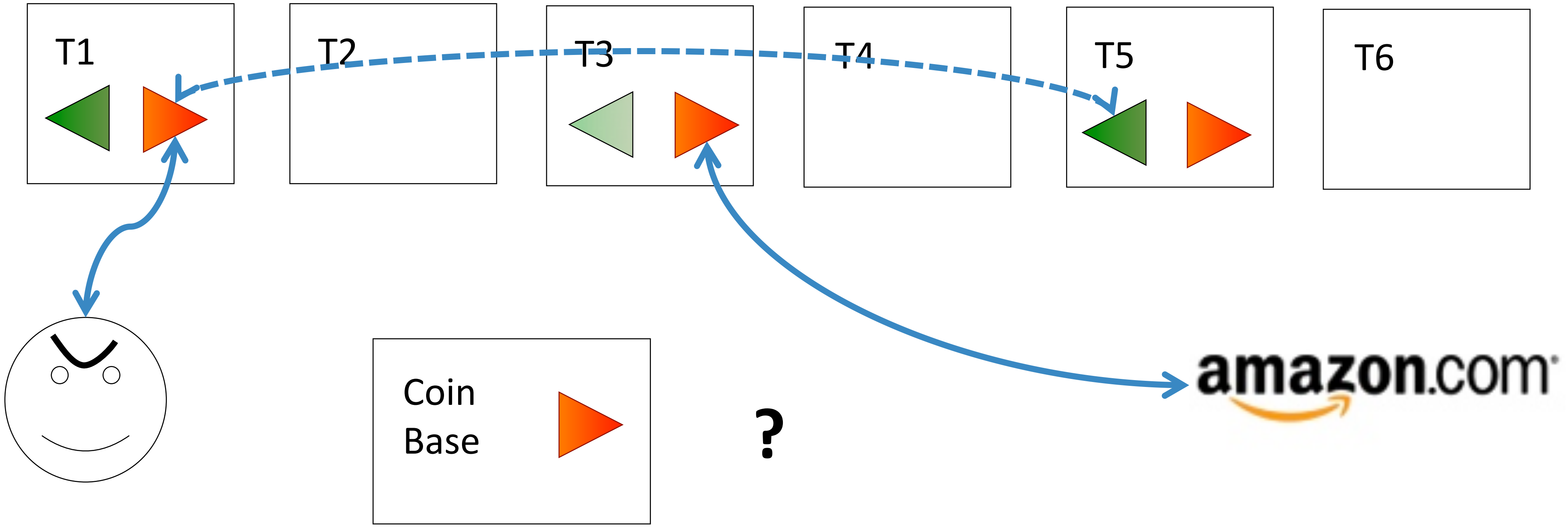
Stack	**Script Code**	**Semantics**
Empty.	<sig> <pubKey> OP_CHECKSIG	scriptSig and scriptPubKey are combined.
<sig> <pubKey>	OP_CHECKSIG	Constants are added to the stack.
TRUE	Empty.	Signature is checked for top two stack items.

Almost finished?

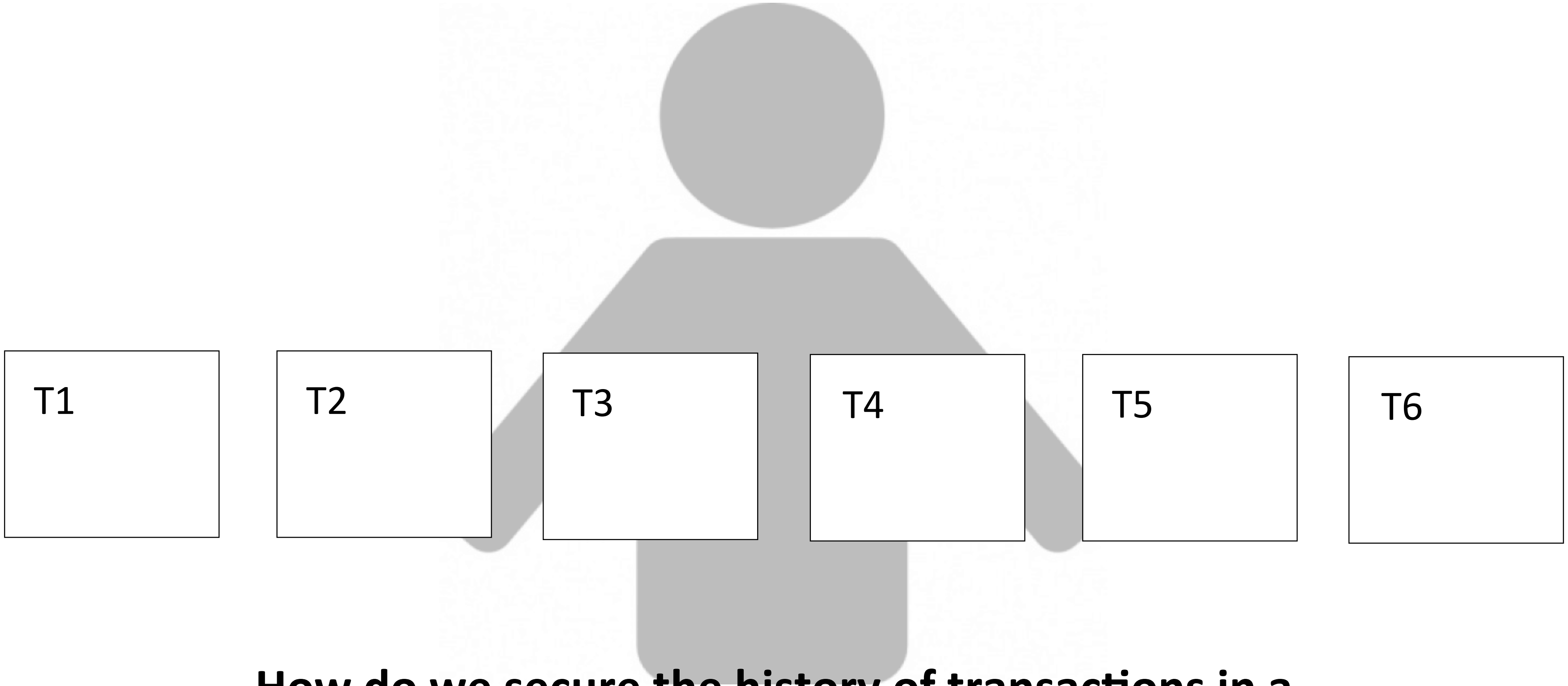


Almost finished? Not really …

If a Bitcoin is defined by its transactional history, its value is dependent on the immutability of this history!



Almost finished? Not really …



T1 T2 T3 T4 T5 T6

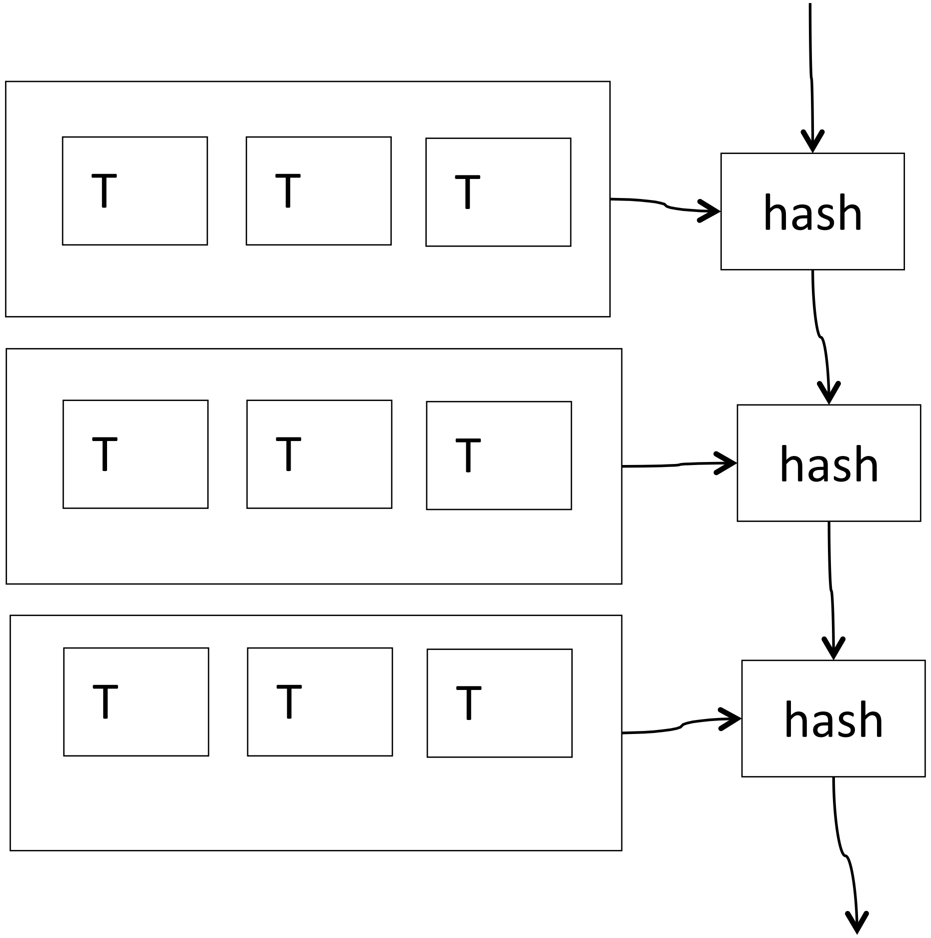
How do we secure the history of transactions in a distributed system, without a central authority?

Solution

Timestamping ⚭ Proof of work

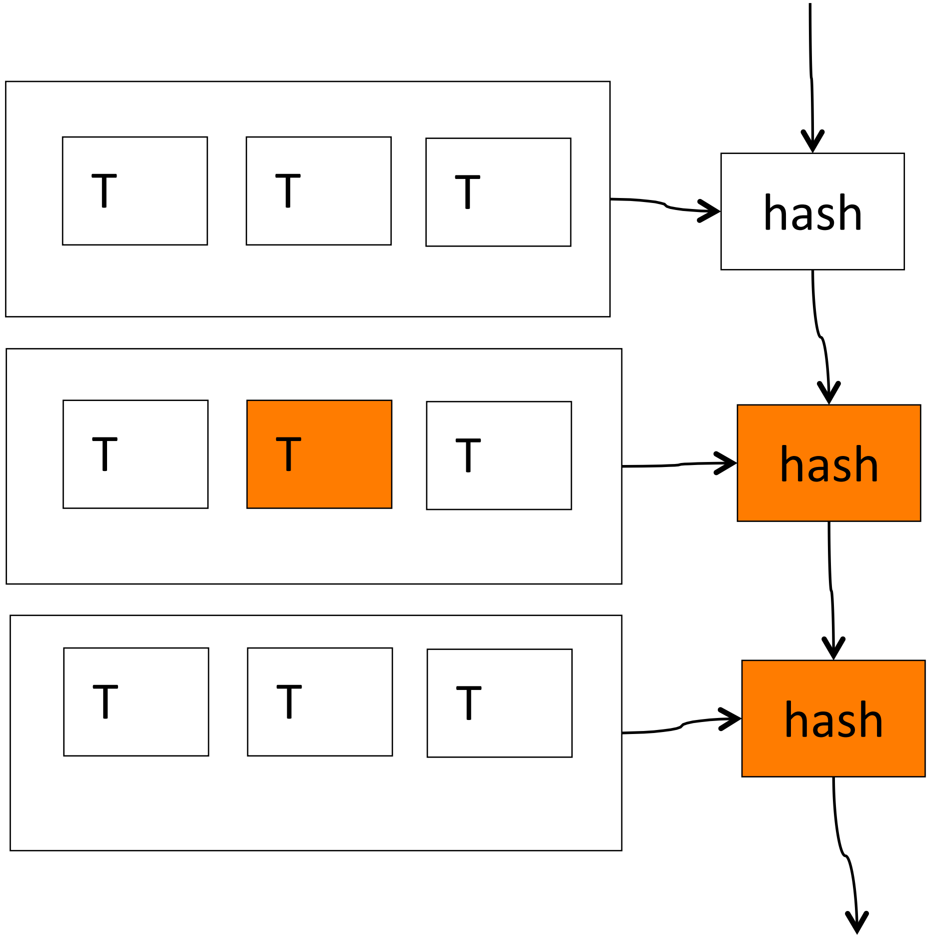
Timestamping

- Group transactions in chronological order into a hash-based chain, where each hash is dependent on the previous one
- Changes or removal of transactions is immediately noticed:
  - Block with specific transaction is hashed differently
  - Top hash of chain changes -> **Error**!



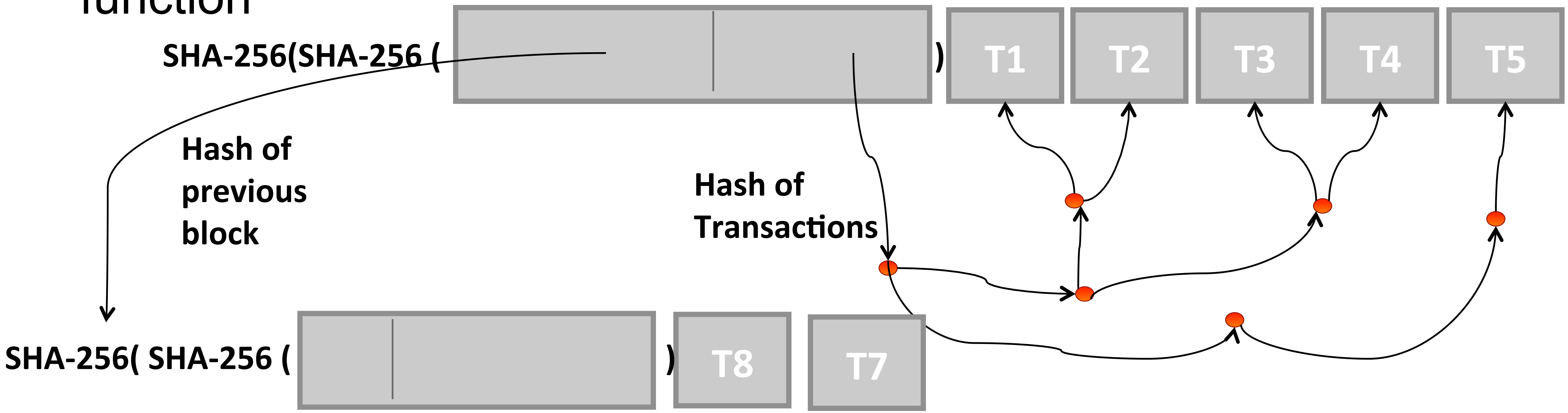
Timestamping

- Group transactions in chronological order into a hash-based chain, where each hash is dependent on the previous one
- Changes or removal of transactions is immediately noticed:
  - Block with specific transaction is hashed differently
  - Top hash of chain changes -> Error!



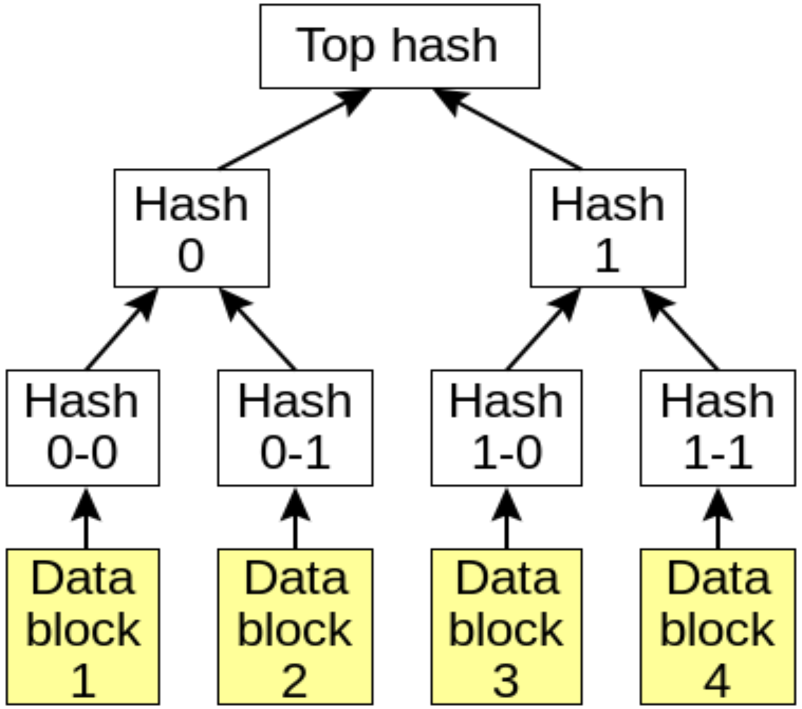
Timestamping

- Implement Timestamping by Combining two Hashes in a Blockheader
- Use **SHA-256(SHA-256(** block **))** as hash function

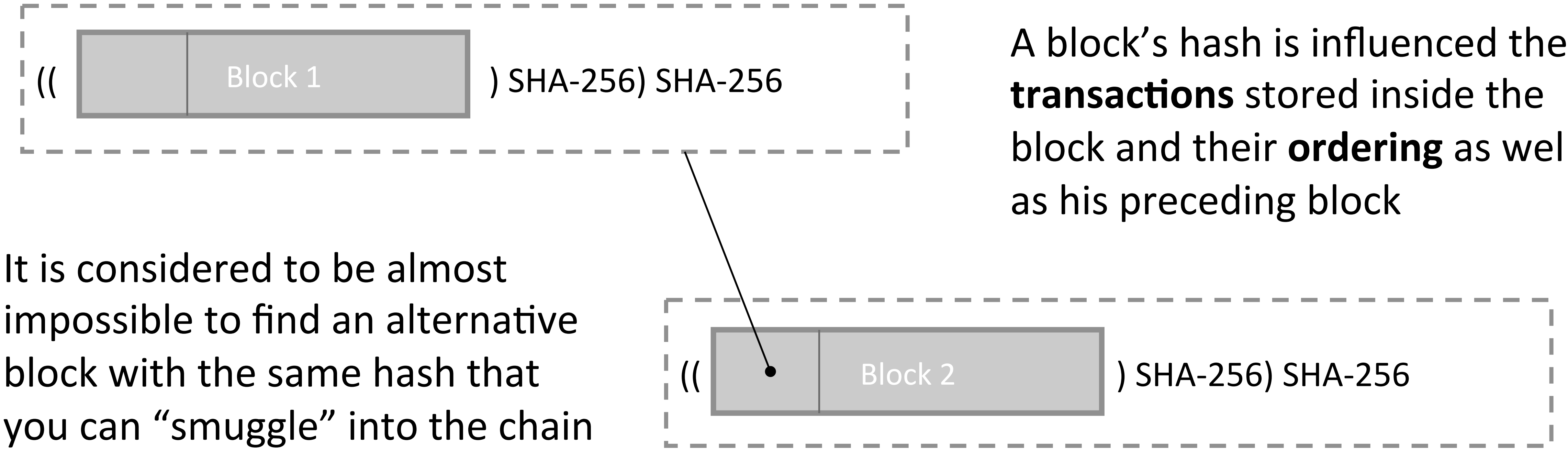


Timestamping– Merkle Tree

- Transactions are hashed into a “Merkle Tree”
- Top hash influenced by order and values of the transactions
- Benefits:
  - Only Root-hash contributes to the blocks hash, making a consistent amount of work
  - Shape of transactions irrelevant in case of **preimage attack** of SHA-256
  - To check that a transaction at a given index is part of the block, only its merkle-branch needs to be checked



Timestamping - Blockcomputation



A block's hash is influenced the **transactions** stored inside the block and their **ordering** as well as his preceding block

It is considered to be almost impossible to find an alternative block with the same hash that you can "smuggle" into the chain

Who to trust?

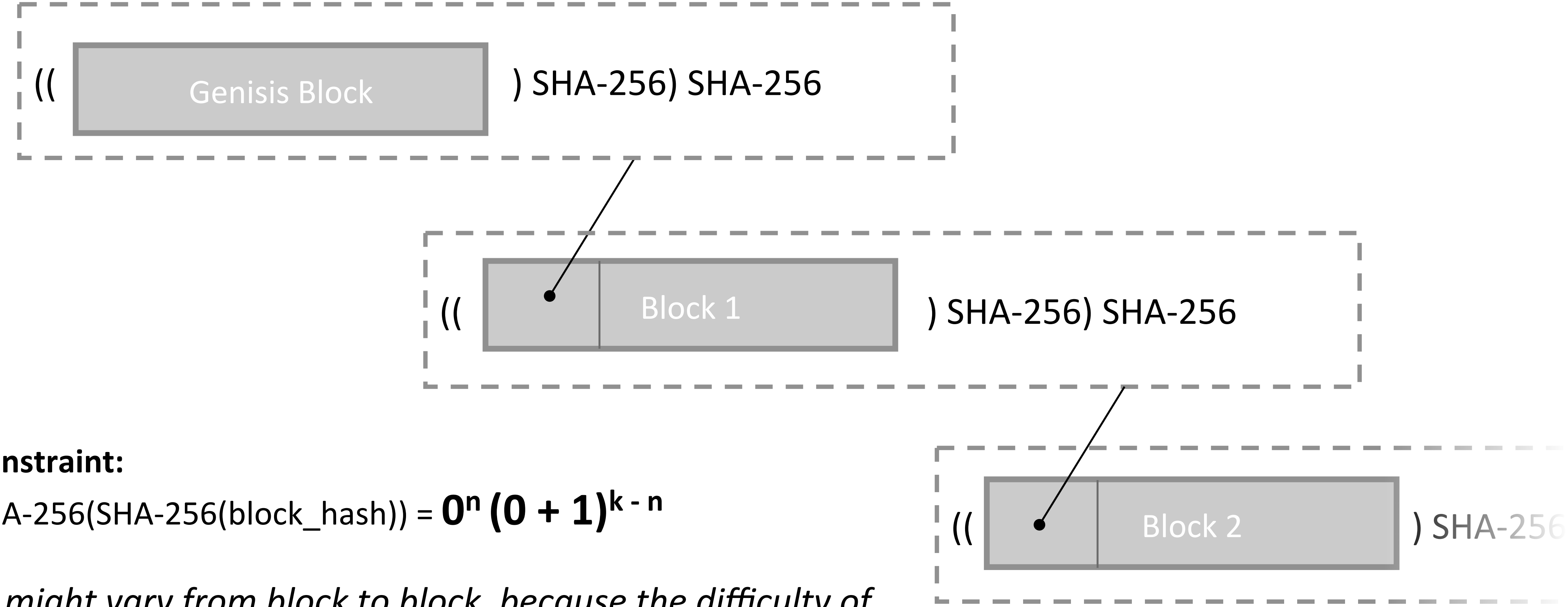
- **While the blockchain may be immutable, we still can not be sure how trustworthy the chain we have been supplied with is**
  - No central authority that manages the blockchain
  - A distributed network might have conflicting versions of transaction history

Assumption: The Majority of computing power is in the hands of people who desire a trustworthy currency

Idea: Proof of Work

1. Let the chain begin from a globally equal **initial block**
2. Make the creation of blocks **computationally expensive**
3. Only trust the chain who was the most expensive to compute

Proof of work- Blockcomputation



((Genisis Block) SHA-256) SHA-256
((Block 1) SHA-256) SHA-256
((Block 2) SHA-256

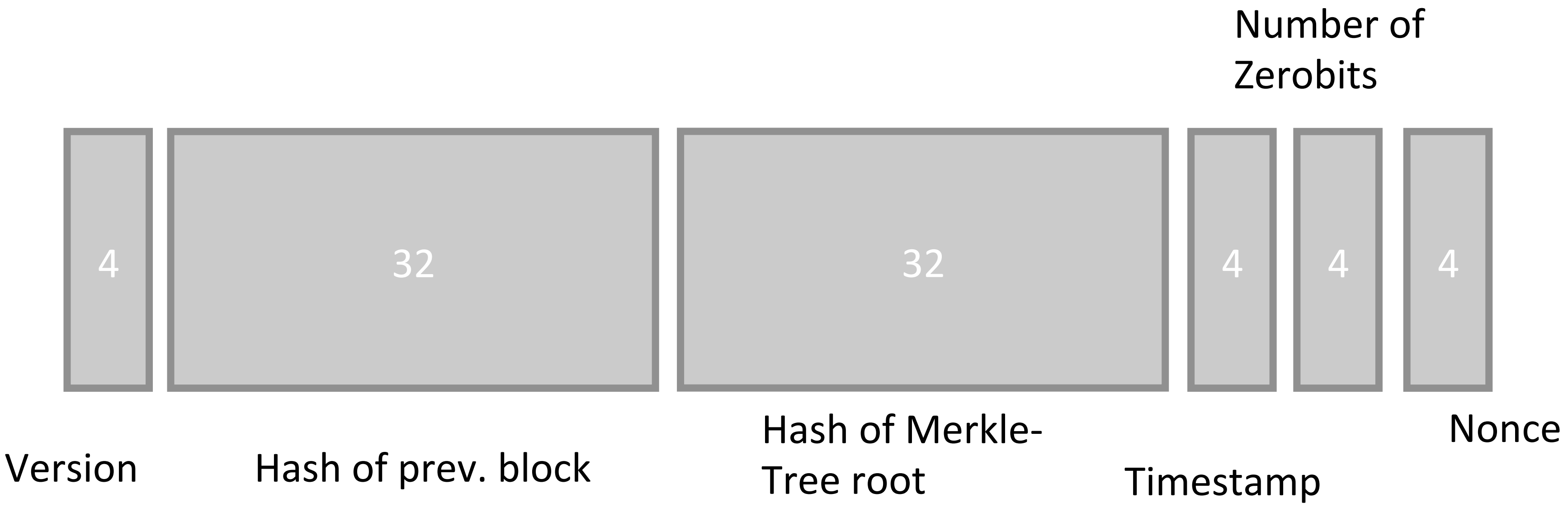
Constraint:

SHA-256(SHA-256(block_hash)) = $\mathbf{0^n (0 + 1)^{k - n}}$

(n might vary from block to block, because the difficulty of block computation is adjusted every 2016 blocks)

k = Length of the hash (currently 256 Bits)

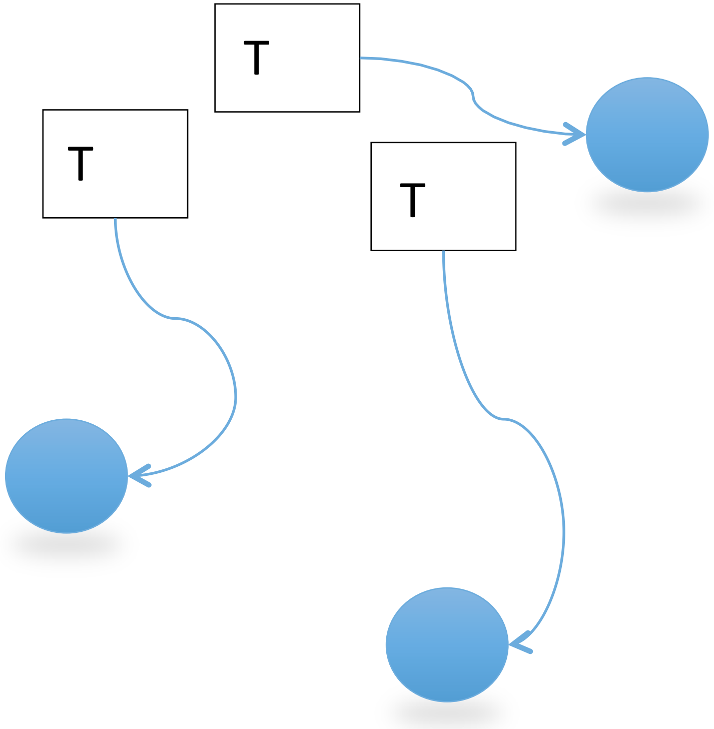
Proof of work - Blockheader



- Is the Part of the block that is actually hashed
- Contains all variation points to achieve the *"trailing zeroes property"*
  - *Nonce*
  - *Timestamp*
  - *Merkle Tree Root (changeable by reordering transactions)*

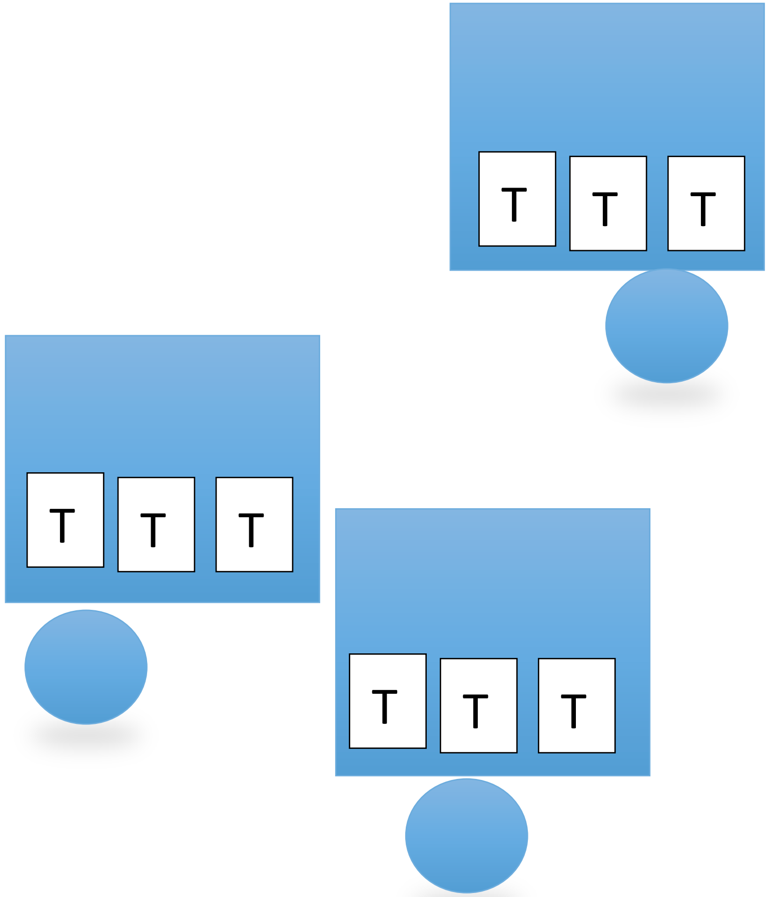
Network

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a solution for its block. **(trailing zeros)**
4. When a node finds a solution, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and do not reference outputs that have already been spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.



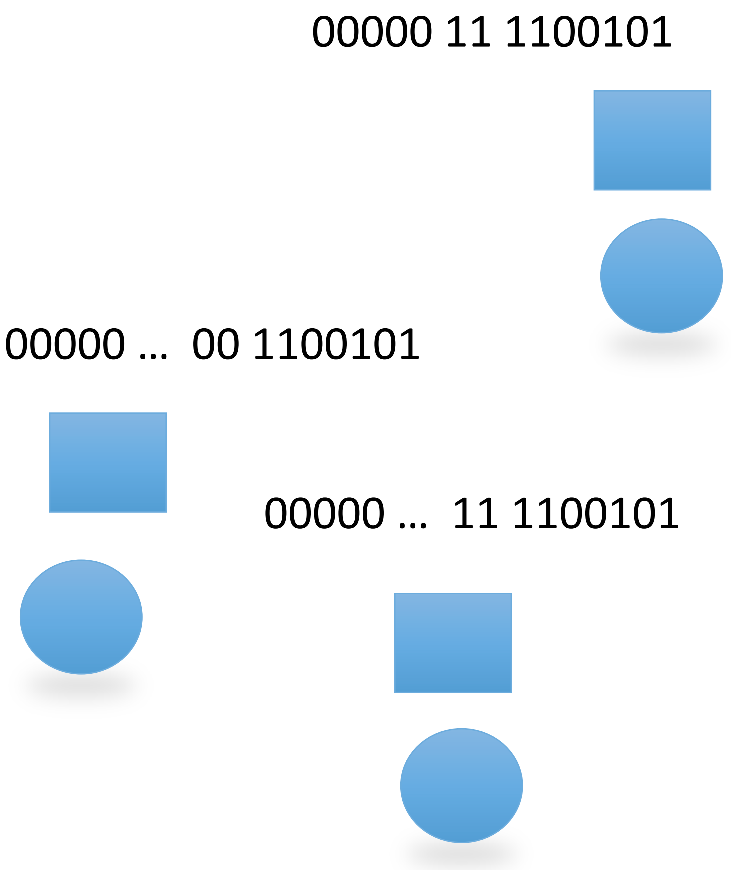
Network

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a solution for its block. **(trailing zeros)**
4. When a node finds a solution, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and do not reference outputs that have already been spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.



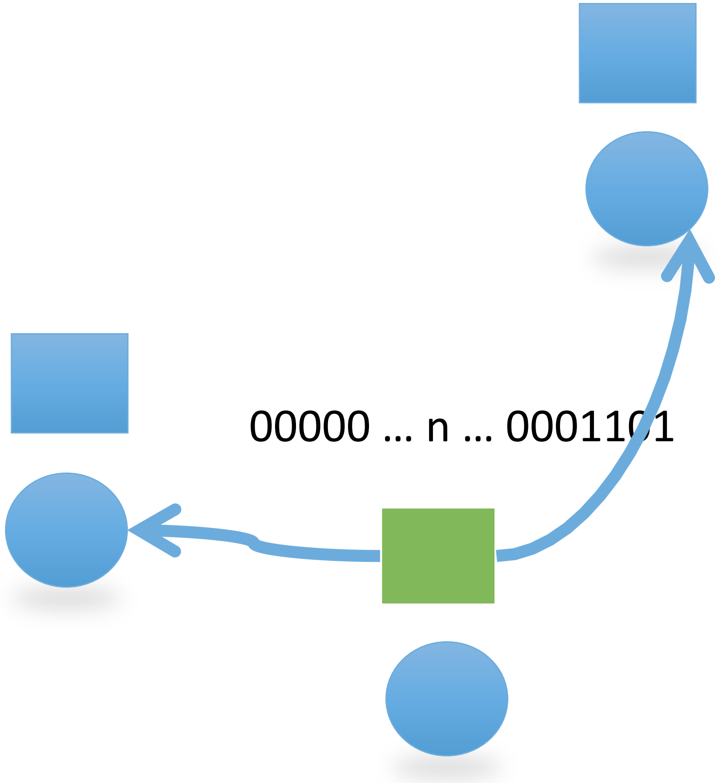
Network

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a solution for its block. **(trailing zeros)**
4. When a node finds a solution, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and do not reference outputs that have already been spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.



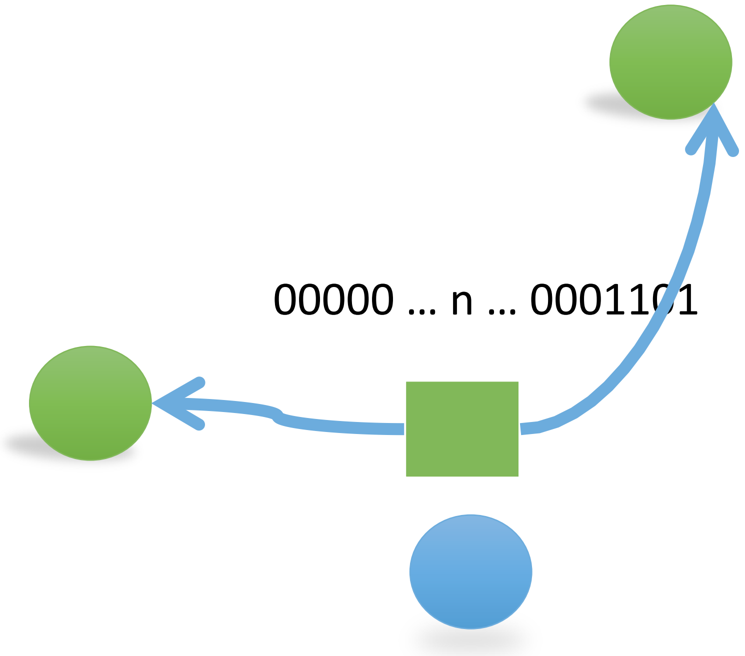
Network

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a solution for its block. **(trailing zeros)**
4. When a node finds a solution, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and do not reference outputs that have already been spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.



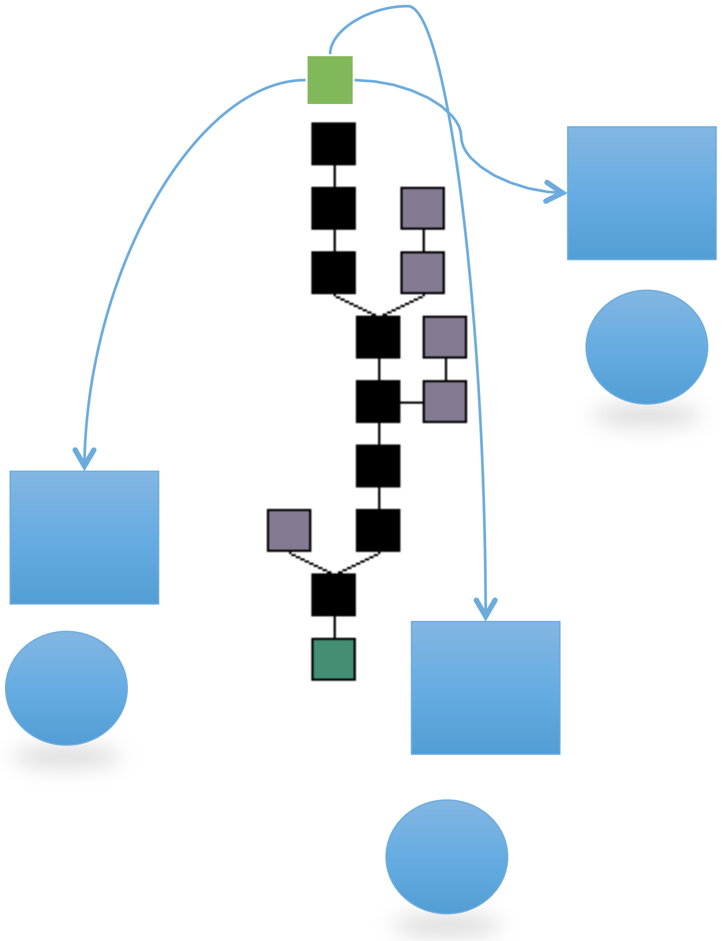
Network

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a solution for its block. **(trailing zeros)**
4. When a node finds a solution, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and do not reference outputs that have already been spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.



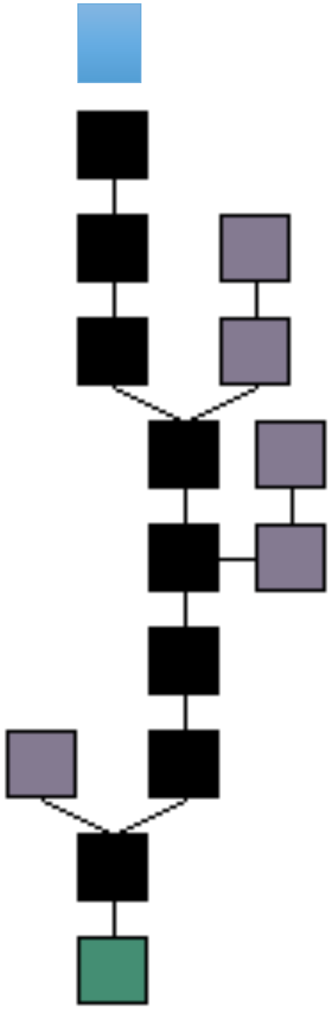
Network

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a solution for its block. **(trailing zeros)**
4. When a node finds a solution, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and do not reference outputs that have already been spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.



Network

- Blockchain expanded approx. every 10 Minutes
  - This means the network can serve all pending transactions at most every 10 minutes
  - Network automatically adjusts expected number of trailing zeros every 2016 Blocks so that the 10 minute bound is achieved
- Chain can fork into sub chains, **however**:
  - Only the chain, that is hardest to compute is considered as valid
  - It is very unlikely that the nodes in the network separate in such a way that a “racecondition” between two branches is created
    - Has happened though in the case of changes within the protocol



Blockchain - Mining

- Create incentive for participating nodes to mine blocks honestly (*e.g. not accepting controversial transactions*)
- New Bitcoins can be generated by the computation of new blocks
  - New Blocks contain a coinbase transaction at the beginning
  - Coinbase transaction “spawns” new bitcoins and assigns them to the miners account
  - These coins can only be spent, after the subchain that the block created has grown by more than 100 blocks *(Block maturing time)*
- The amount of Bitcoins that is spawned by coinbase transactions is halved every 210 000 Blocks
  - Will reach 0 by the time 21 Million Bitcoins are in circulation

Blockchain - Mining

- A Variety of specialized hardware exists, that is optimized for computing new blocks “at home”
  - Initially CPUs
  - GPUs improved mining speed by 50x – 100x
  - FPGAs improved power efficiency so that the cost / value relationship effectively trumped the GPUs
  - Newest rend are ASICs, which are specialized processors solely for the purpose of computing blocks

- Pooled Mining: Divide computation of a block into subparts and compute these in parallel on different machines
  - BitcoinCZ, 21bitcoin, slush’s …

Concept

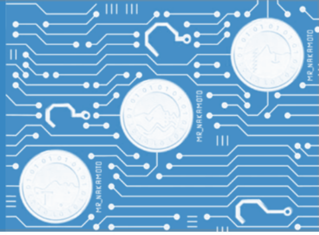
Implementation

Discussion

Using and extending Bitcoin

- The Idea behind Bitcoin solves the general problem of consensus in a distributed system without having a central instance
- Can therefore be applied to many problems that formally needed a central entity that organized communication:
  - Currencies: Bitcoin, **Zerocoin** …
  - DNS Servers: Namecoin
    - Blockchain contains Name/Value pairs that are attached to transactions
    - Names expire after 12000 blocks unless renewed with an update
  - Chat clients
    - Christian?

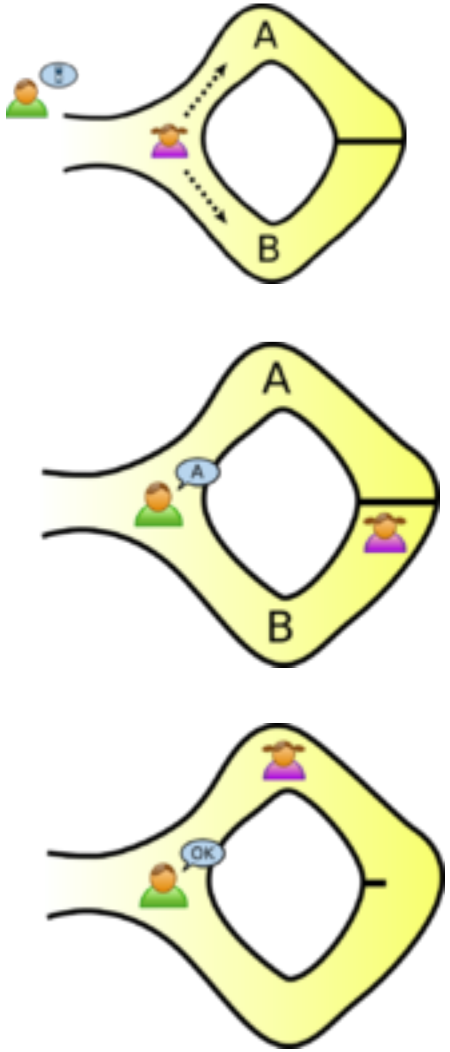
Anonymity



- While transactions in Bitcoin can be considered to be anonymous actually paying for real life services is not
- All anonymity dies with exchange for real money as well
- Solutions have been suggested but scarcely implemented
- One proposed solution: **Zerocoin**

Zerocoin

- Idea: Introduce new type of transactions that converts Bitcoins to Zerocoins
- Zerocoins can be transferred without being able to trace who send money to which party, while still preventing double spending
  - **Intuition**:
    - Moneyflow between transactions is hidden in a pool of coins, where money flows in and out
    - Participating Inputs prove their validity with a “zero knowledge proof”
    - Once the proof is performed, the pool creates an anonymous Output that contains the desired amount of coins to be spend

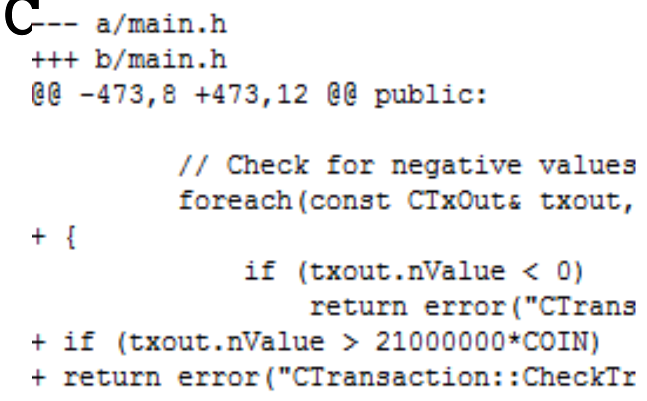


Legal implications

- What do you think?

Integer Overflow Exploit (08/2010)

- One basic validity test for transactions is that the sum of the outputs has to be equal to or smaller than the sum of inputs
- A specially prepared transaction exploited integer overflow to trick the system into thinking a transaction was valid
- 184 Billion BTC were transferred
- Exploit was discovered within 1,5 hours
- First patch was made available after 4 hours
- The “good” chain overtook the malicious one after ~9 hours



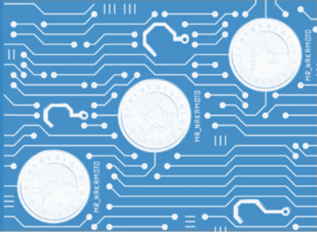
```
--- a/main.h
+++ b/main.h
@@ -473,8 +473,12 @@ public:

            // Check for negative values
            foreach(const CTxOut& txout,
+ {
                if (txout.nValue < 0)
                    return error("CTrans
+ if (txout.nValue > 21000000*COIN)
+ return error("CTransaction::CheckTr
```

Concept

Implementation

Discussion

Sources



- Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system."*Consulted* 1 (2008): 2012.
- Garcia, Flavio D., and Jaap-Henk Hoepman. "Off-line karma: A decentralized currency for peer-to-peer and grid applications." *Applied Cryptography and Network Security*. Springer Berlin Heidelberg, 2005.
- Vishnumurthy, Vivek, Sangeeth Chandrakumar, and Emin Gun Sirer. "Karma: A secure economic framework for peer-to-peer resource sharing." *Workshop on Economics of Peer-to-Peer Systems*. 2003.
- Back, Adam. "Hashcash-a denial of service counter-measure." (2002).
- Dwork, Cynthia, and Moni Naor. "Pricing via processing or combatting junk mail." *Advances in Cryptology—CRYPTO'92*. Springer Berlin Heidelberg, 1993.
- Wang, XiaoFeng, and Michael K. Reiter. "Defending against denial-of-service attacks with puzzle auctions." *Security and Privacy, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003.
- Rivest, Ronald L., Adi Shamir, and David A. Wagner. "Time-lock puzzles and timed-release crypto." (1996).
- Jakobsson, Markus, and Ari Juels. "Proofs of work and bread pudding protocols." *Secure Information Networks*. Springer US, 1999. 258-272.
- Laurie, Ben, and Richard Clayton. "Proof-of-work proves not to work."*Workshop on Economics and Information Security*. Vol. 2004. 2004.
- Liu, Debin, and L. Jean Camp. "Proof of work can work." *Fifth Workshop on the Economics of Information Security*. 2006.
- Abadi, Martin, et al. "Moderately hard, memory-bound functions." *ACM Transactions on Internet Technology (TOIT)* 5.2 (2005): 299-327.
- Dwork, Cynthia, Andrew Goldberg, and Moni Naor. "On memory-bound functions for fighting spam." *Advances in Cryptology-Crypto 2003*. Springer Berlin Heidelberg, 2003. 426-444.
- Haber, Stuart, and W. Scott Stornetta. *How to time-stamp a digital document*. Springer Berlin Heidelberg, 1991.
- Bayer, Dave, Stuart Haber, and W. Scott Stornetta. "Improving the efficiency and reliability of digital time-stamping." *Sequences II*. Springer New York, 1993. 329-334.
- Massias, H., X. Serret Avila, and J-J. Quisquater. "Design of a secure timestamping service with minimal trust requirement." *the 20th Symposium on Information Theory in the Benelux*. 1999.
- Haber, Stuart, and W. Scott Stornetta. "Secure names for bit-strings."*Proceedings of the 4th ACM Conference on Computer and Communications Security*. ACM, 1997.

The root problem with conventional currency is all the trust thats required to make it work. The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust. Banks must be trusted to hold our money and transfer it electronically, but they lend it out in waves of credit bubbles with barely a fraction in reserve. We have to trust them with our privacy, trust them not to let identity thieves drain our accounts. Their massive overhead costs make micropayments impossible.

A generation ago, multi-user time-sharing computer systems had a similar problem. Before strong encryption, users had to rely on password protection to secure their files, placing trust in the system administrator to keep their information private. Privacy could always be overridden by the admin based on his judgment call weighing the principle of privacy against other concerns, or at the behest of his superiors. Then strong encryption became available to the masses, and trust was no longer required. Data could be secured in a way that was physically impossible for others to access, no matter for what reason, no matter how good the excuse, no matter what.

Its time we had the same thing for money. With e-currency based on cryptographic proof, without the need to trust a third party middleman, money can be secure and transactions effortless.

Satoshi Nakamoto (http://p2pfoundation.net/Bitcoin)