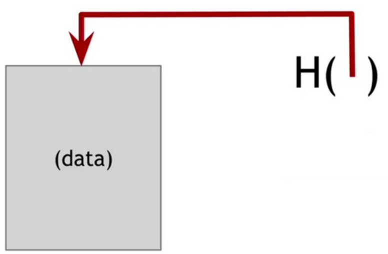
Decentralized Systems

Bitcoin (cnt)

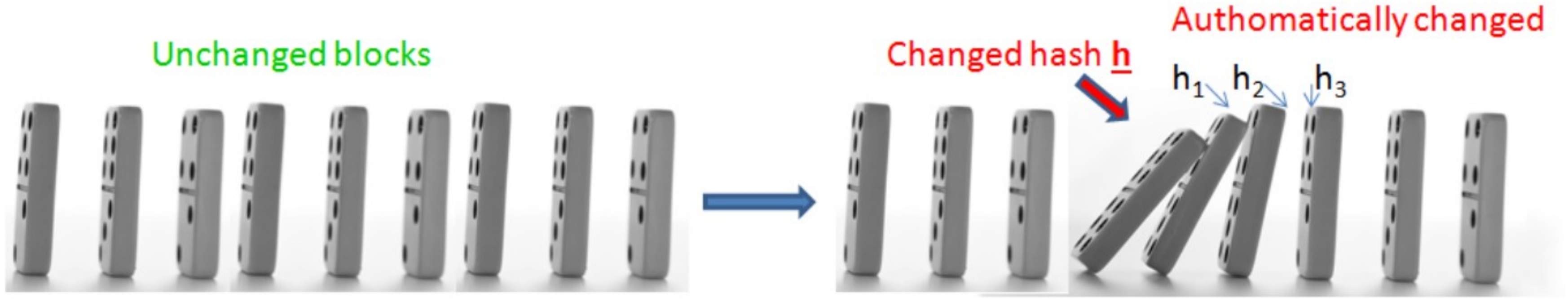
Hash pointers and Merkle trees

Hash pointer

- A **hash pointer** is a pointer to where some information is stored together with a cryptographic hash of the information



- This data structure makes the blockchain immutable



Merkle tree

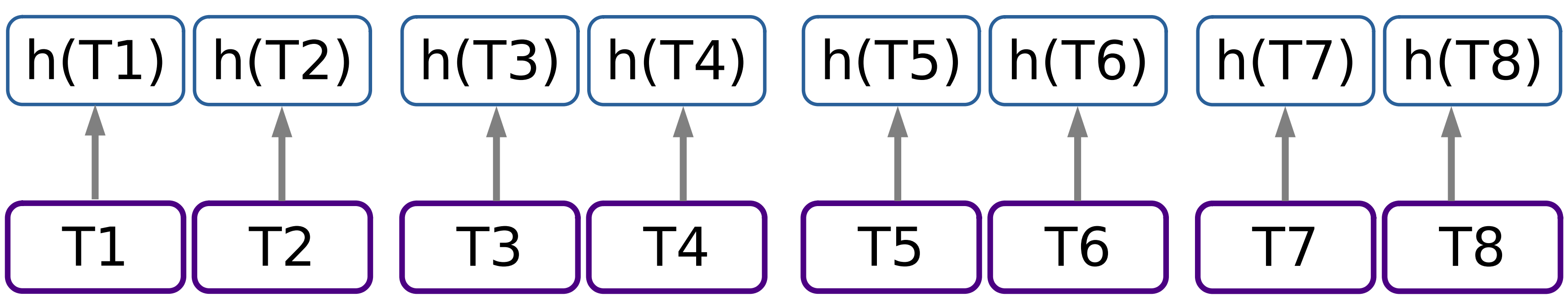
- A Merkle tree is a **binary tree with hash pointers**
- Consider the **Bitcoin transactions** of a block
  - they can be stored using a Merkle tree
- Transactions are grouped into pairs, if the number of transactions is odd, the last transaction is duplicated

Merkle tree

- A Merkle tree is a **binary tree with hash pointers**
- Consider the **Bitcoin transactions** of a block
  - they can be stored using a Merkle tree
- Transactions are grouped into pairs, if the number of transactions is odd, the last transaction is duplicated

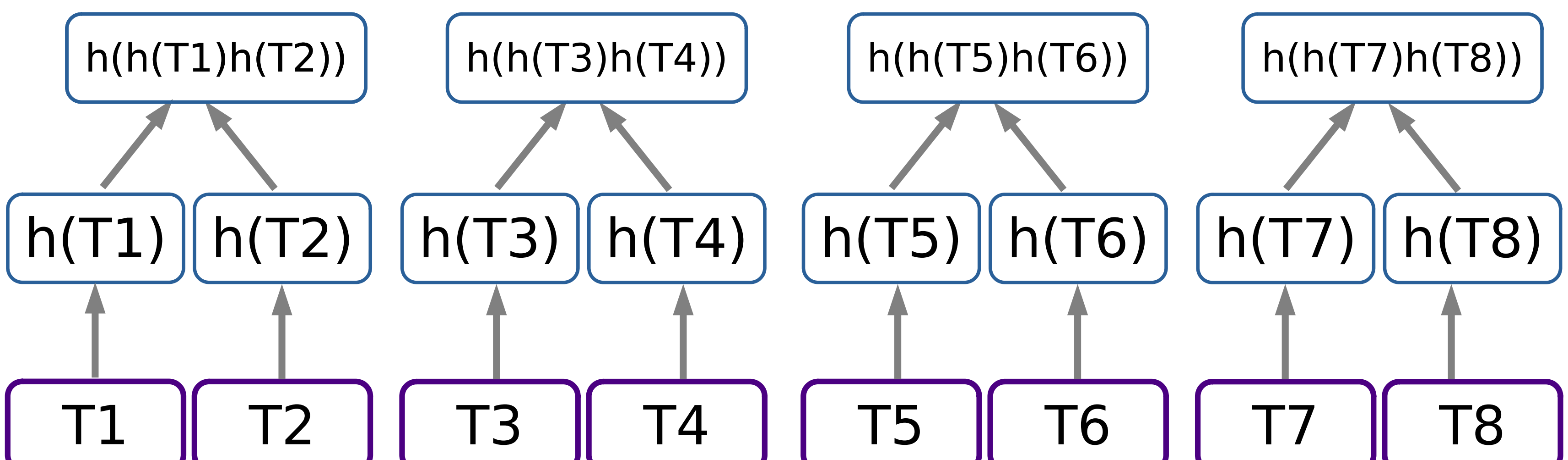
Merkle tree

- Compute the hash of each transaction

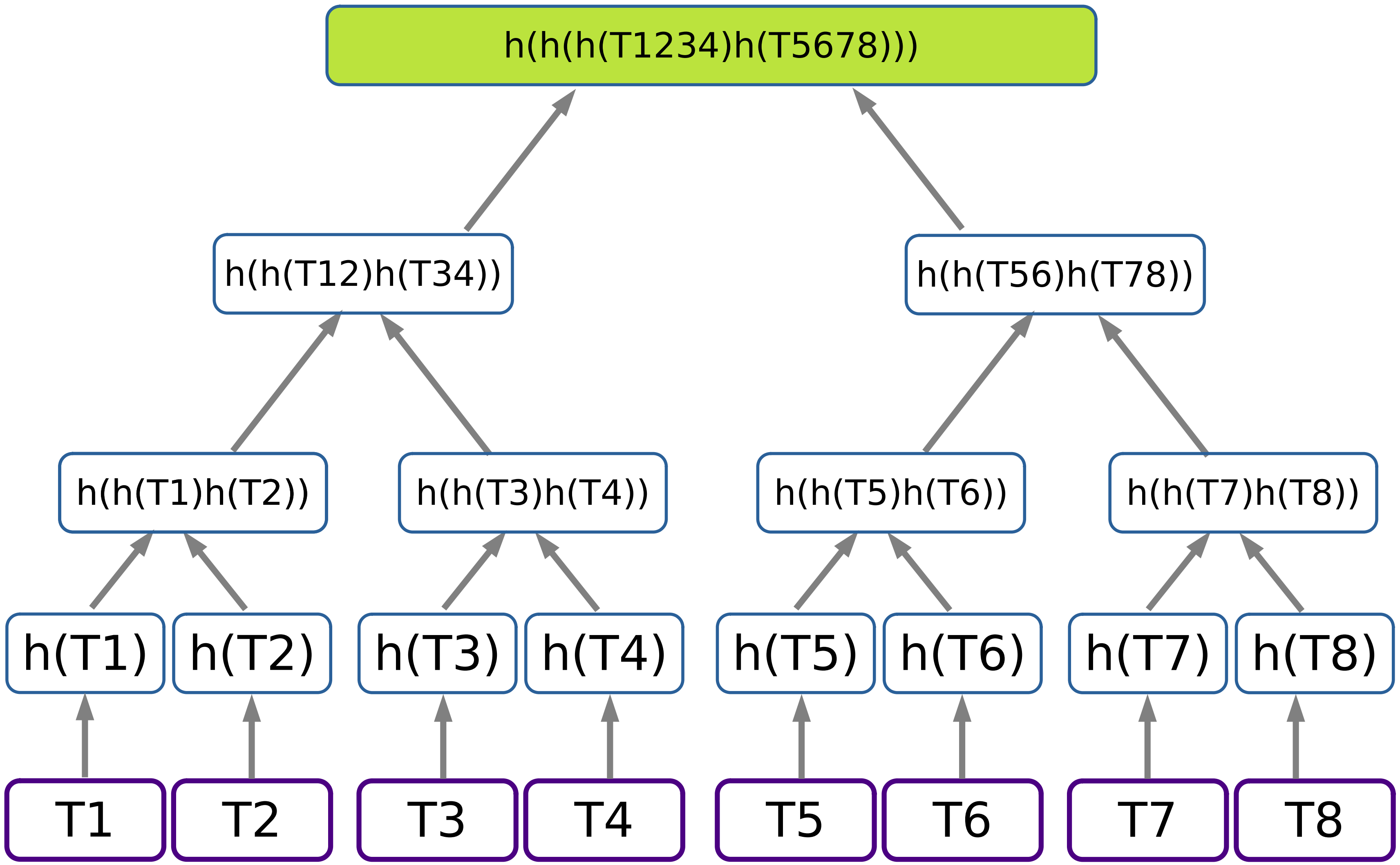


Merkle tree

- Compute the hash of each transaction
- Concatenate the digests and compute the hash
- Continue...

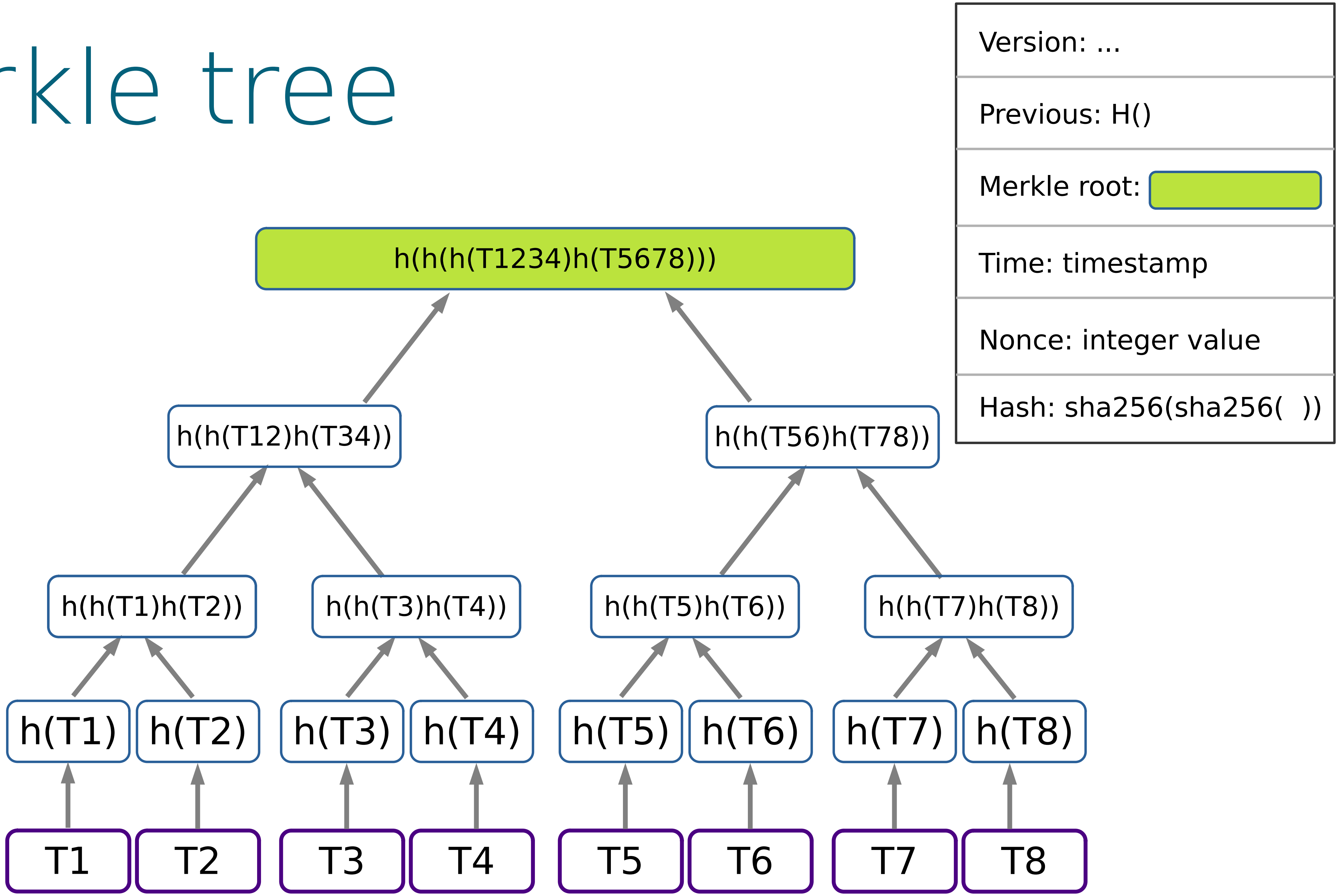


Merkle tree



h(h(h(T1234)h(T5678)))
h(h(T12)h(T34))
h(h(T56)h(T78))
h(h(T1)h(T2))
h(h(T3)h(T4))
h(h(T5)h(T6))
h(h(T7)h(T8))
h(T1)
h(T2)
h(T3)
h(T4)
h(T5)
h(T6)
h(T7)
h(T8)
T1
T2
T3
T4
T5
T6
T7
T8

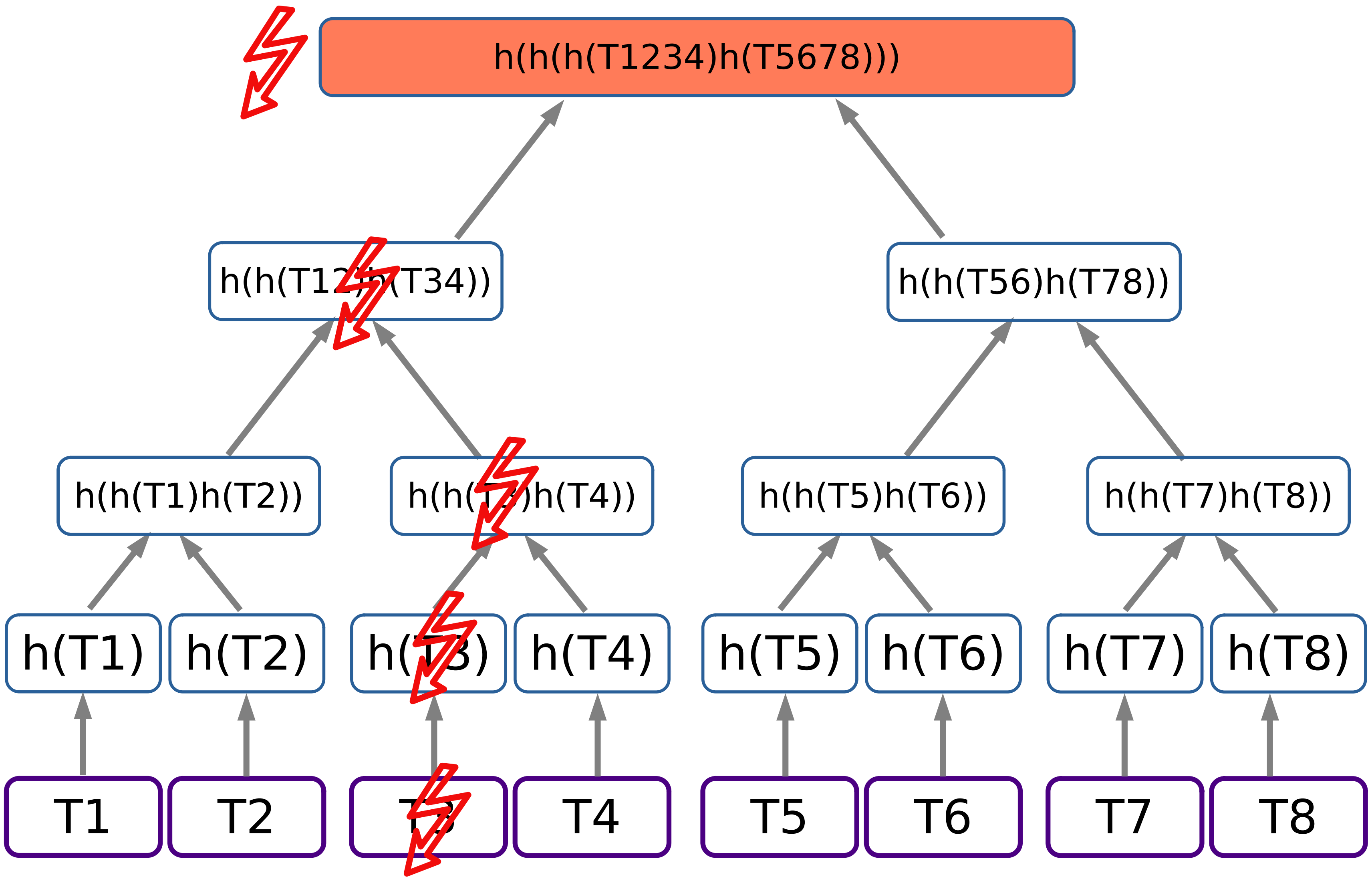
Merkle tree


Version: ...
Previous: H()
Merkle root:
Time: timestamp
Nonce: integer value
Hash: sha256(sha256( ))
h(h(h(T1234)h(T5678)))
h(h(T12)h(T34))
h(h(T56)h(T78))
h(h(T1)h(T2))
h(h(T3)h(T4))
h(h(T5)h(T6))
h(h(T7)h(T8))
h(T1)
h(T2)
h(T3)
h(T4)
h(T5)
h(T6)
h(T7)
h(T8)
T1
T2
T3
T4
T5
T6
T7
T8

Merkle tree

- The **root of the tree** is enough to check whether any transaction of the block has been modified
- If an attacker **tampers a transaction** or an intermediate leaf and the hash is cryptographic, then **the root changes**

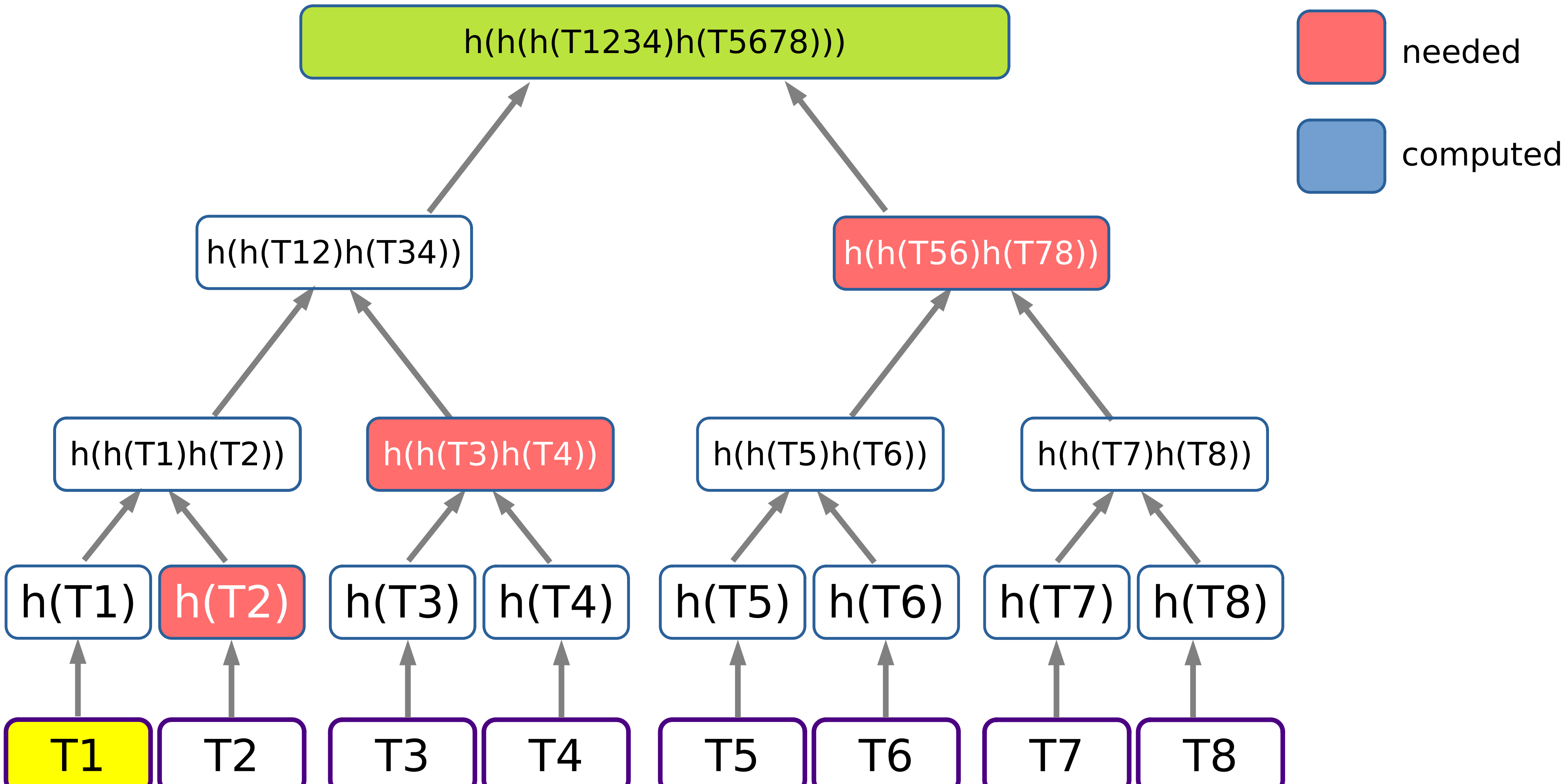
Tampering in a Merkle tree



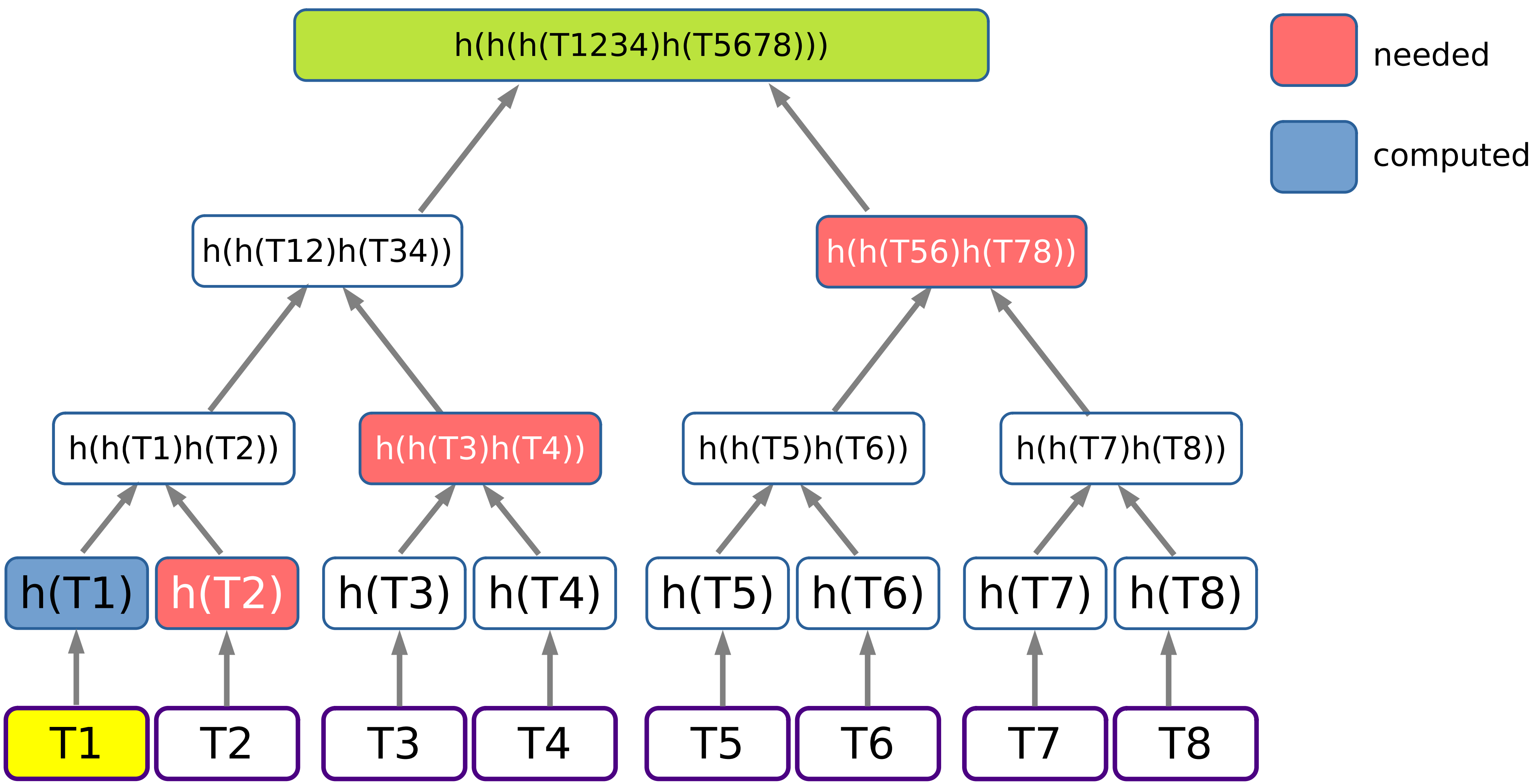
Proof of validity

- Problem
  - Bob wants to check if a certain Bitcoin transaction T is contained in a block and it has not been tampered
- Solution
  - He only needs to ask for the hashes of each sibling, he can recompute the root, and check if the two hashes match
  - The number of operations is $\log_2(n)$ where n is the number of transactions in the block

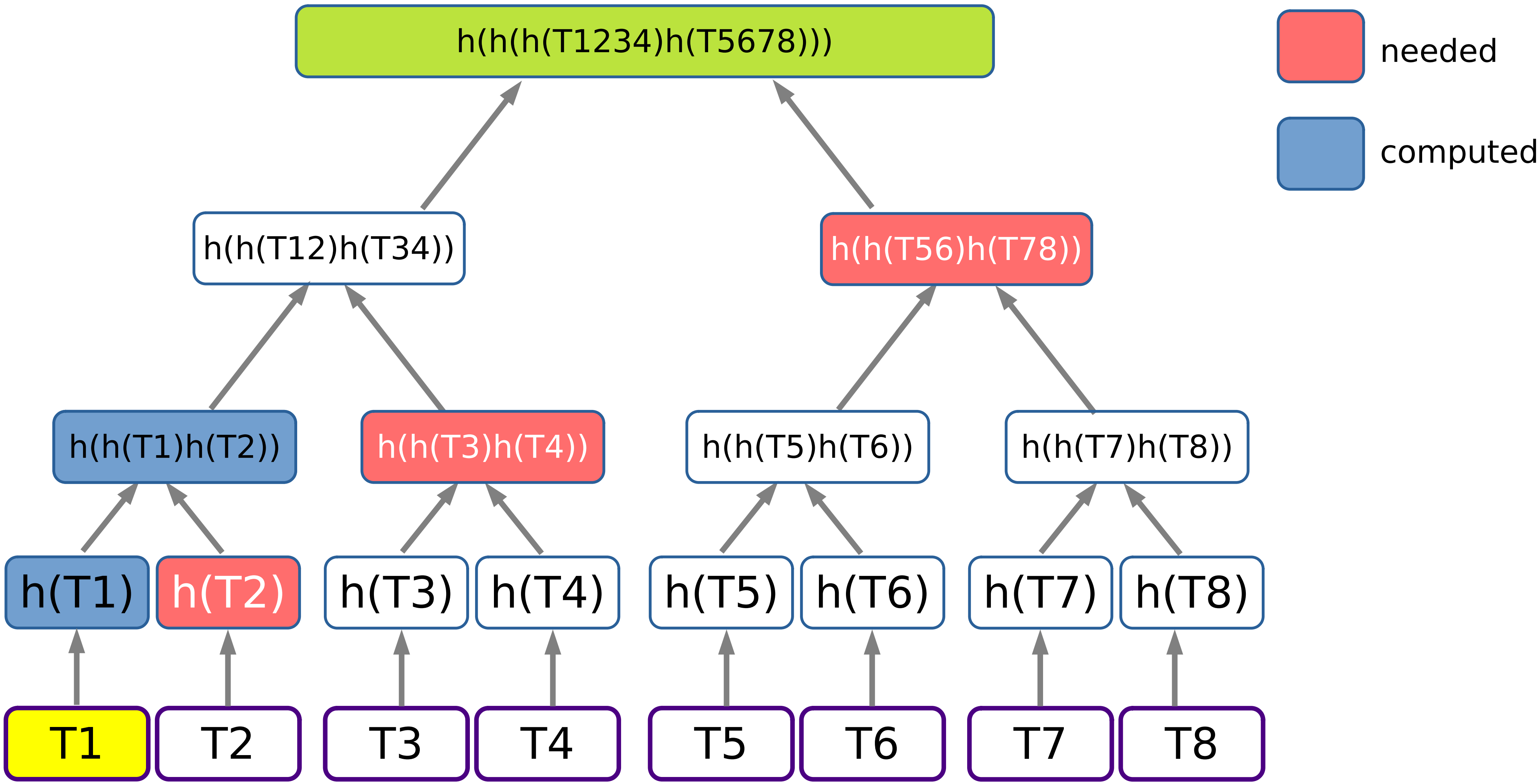
Proof of validity for T1



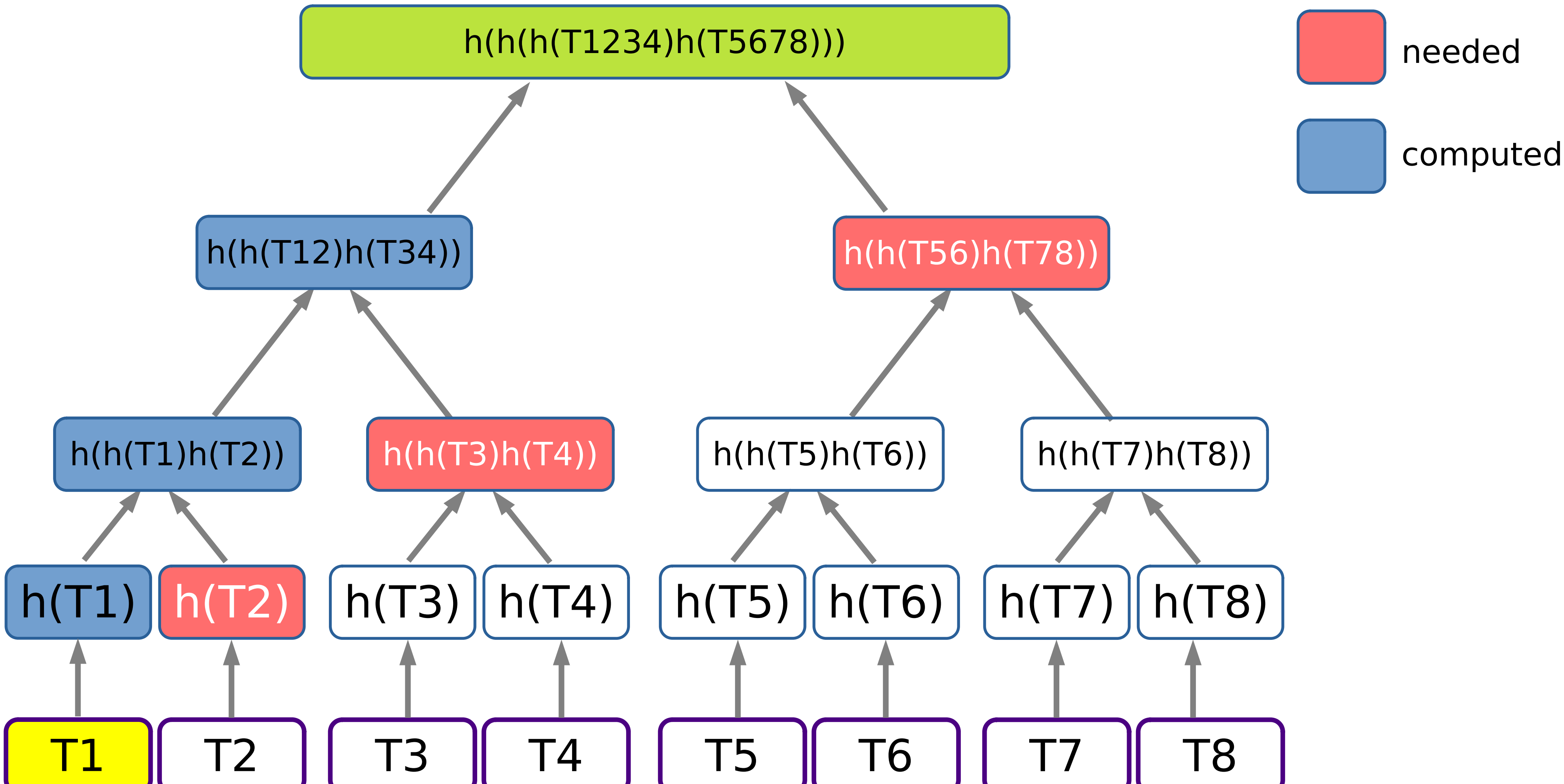
Proof of validity for T1



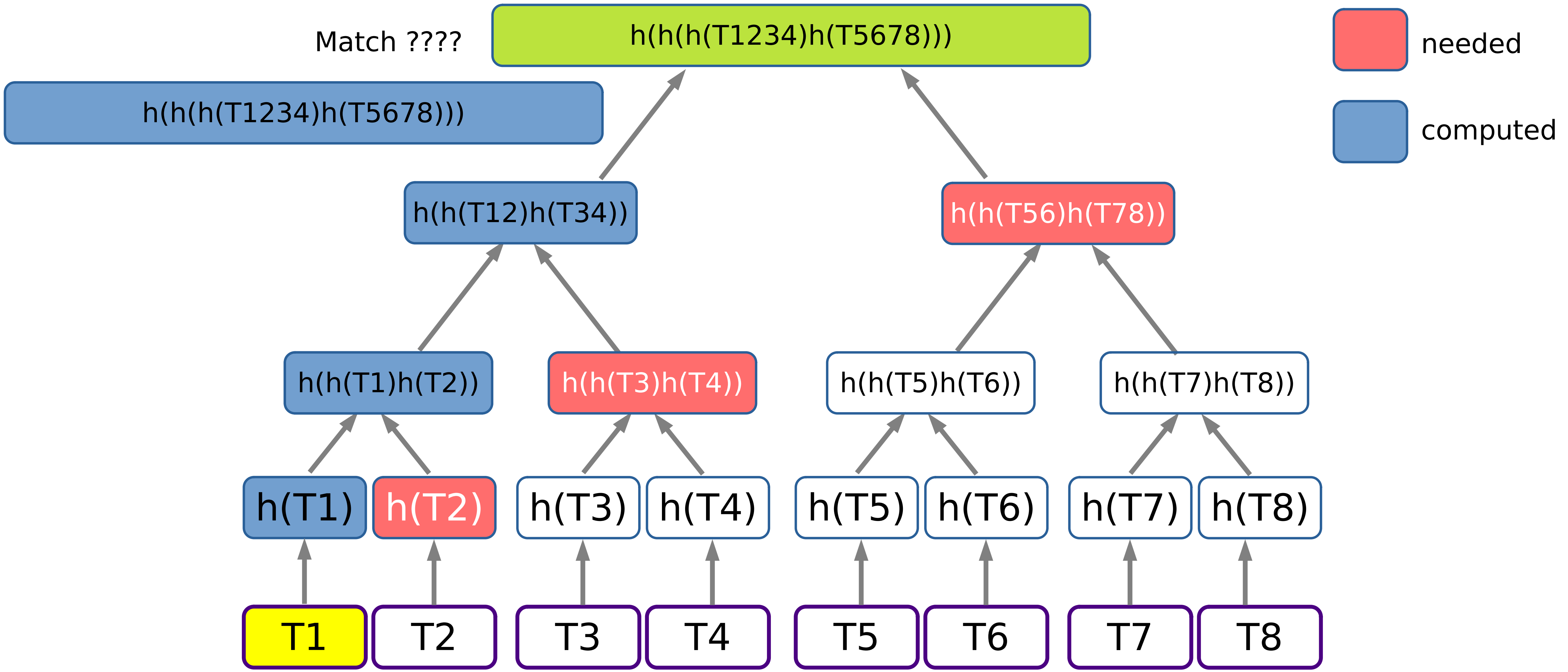
Proof of validity for T1



Proof of validity for T1



Proof of validity for T1



Coinbase extra-nonce

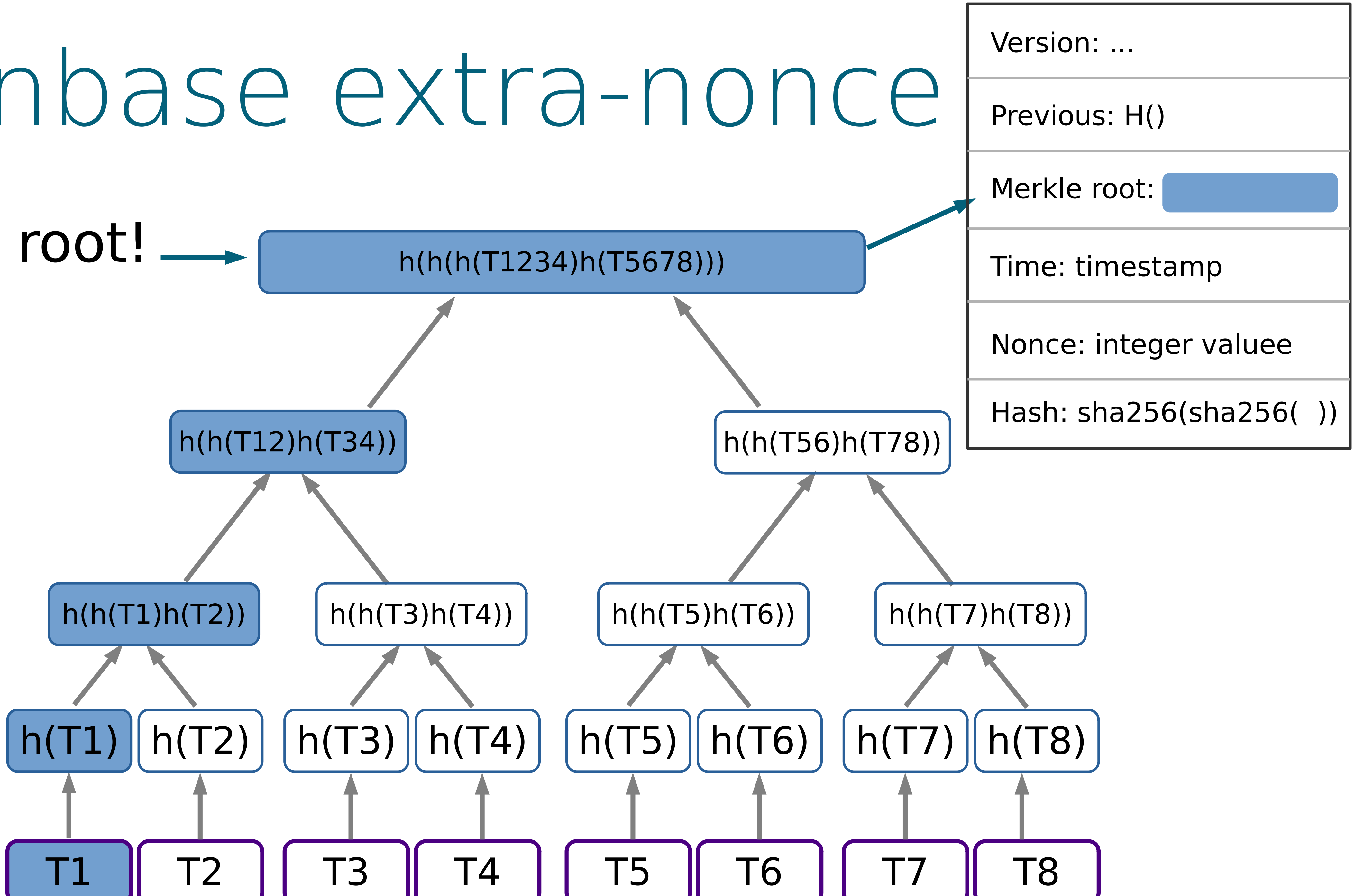
- The Coinbase transaction looks like a normal transaction but
  - generates brand-new coins, it has only one (invalid) input, a **null hash pointer**, and an **extra-nonce field**
  - the **reward** is sent to **one or more outputs** (miners can distribute the block reward to other addresses)
- For each block we are guaranteed to produce different hashes

Coinbase extra-nonce

- Miners change the nonce field in the block's header
- There are **2^{32} possible values for the nonce**
- Once tried all possible values, miners can **change the extra-nonce** of the Coinbase
- Computationally expensive, **the Merkle tree needs to reflect this change**

Coinbase extra-nonce

- New root!



Any **update** in the Coinbase
is **reflected up to the root**

Bitcoin network

Bitcoin Network

- The ecoystem of Bitcoin relies on a **non-structured P2P overlay network** with some similarities with Gnutella
- **Flooding** or **gossip protocols** are used for the propagation of the required information
- In this context, the information which is “stored” by the P2P network are **transactions** and **blocks**

Bitcoin Network

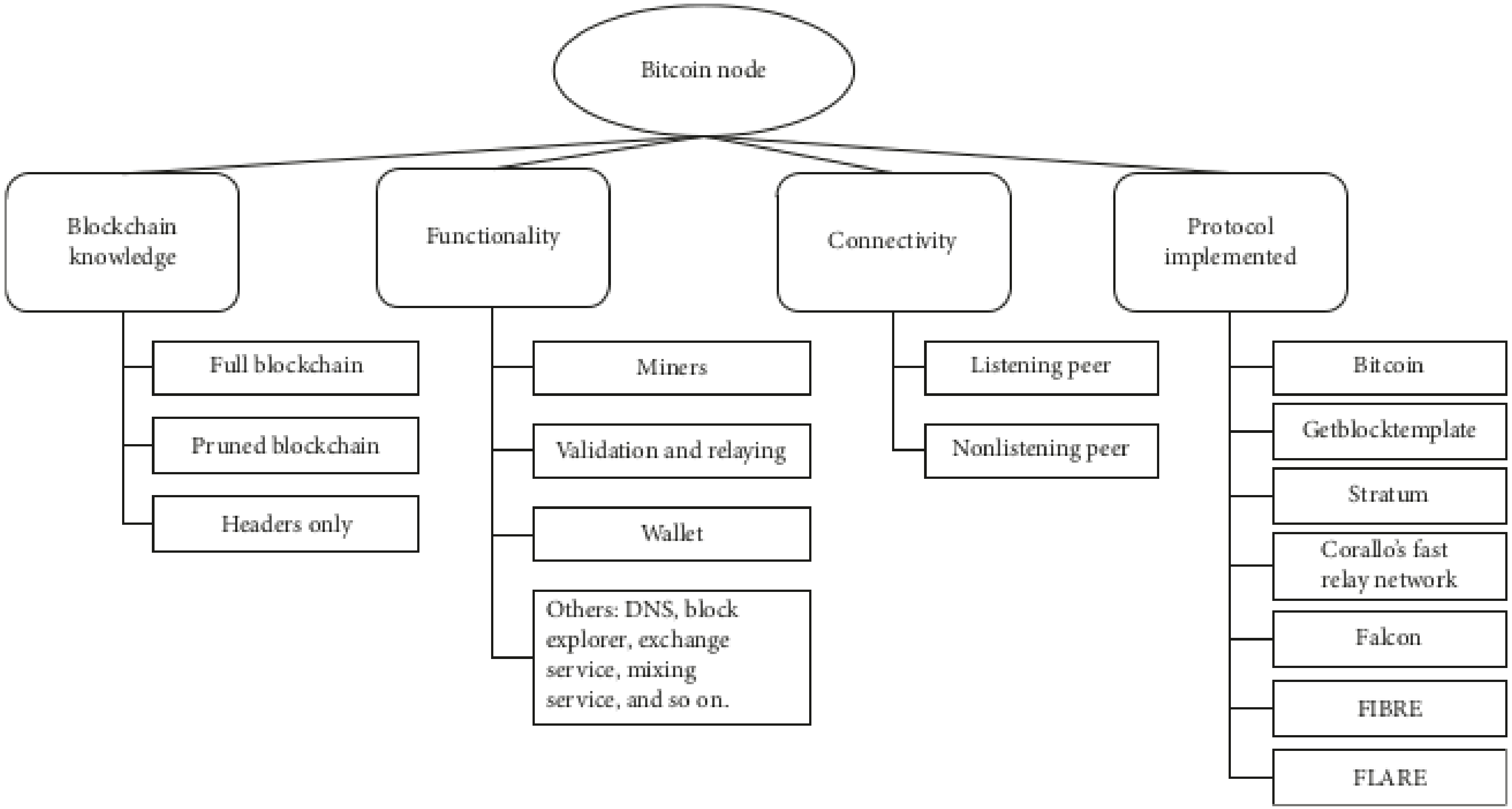


FIGURE 3: Bitcoin node classification.

Source: Cryptocurrency Networks: a New P2P Paradigm https://www.hindawi.com/journals/misy/2018/2159082/

Bitcoin Network

- **Users** are "identified" in the blockchain by their pairs of **public key/address**, **peers** are identified in the Bitcoin network by their **IP** or **Onion addresses**
- With respect to **connectivity**, they can be classified as
  - **listening** peers, e.g., nodes accepting incoming connections (Bitcoin daemons by default listen for incoming connections on TCP ports 8333 and 18333)
  - **nonlistening** peers, e.g., nodes not doing so

Bitcoin Network

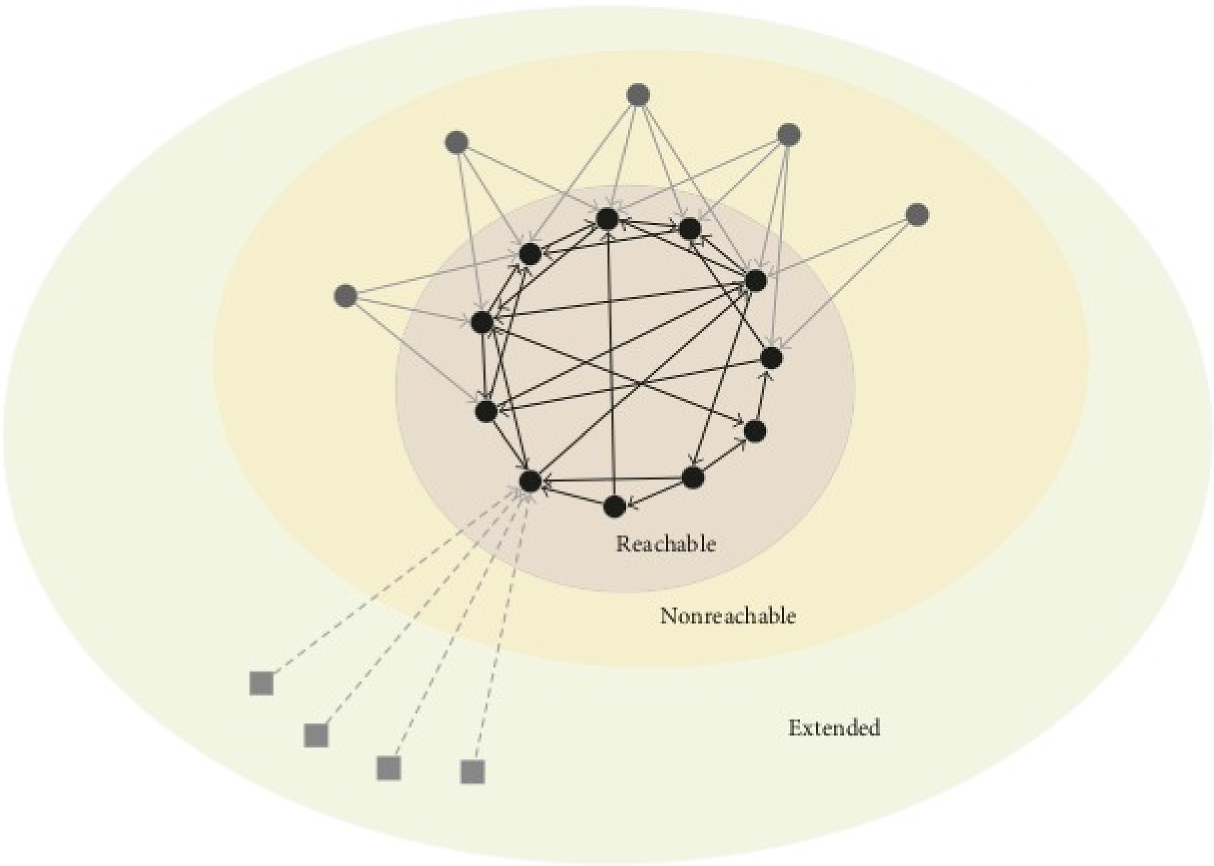
- The P2P network is composed of
  - The **reachable** network, formed by **all listening nodes** that talk the Bitcoin protocol (a sort of “servers”)
  - The **nonreachable** network, formed by nodes that talk the Bitcoin protocol but do not listen for incoming connections (a sort of “clients”, e.g., peers behind NAT or firewall)
  - The **extended** network formed by all nodes in the ecosystem (DNS, exchanges, explorers)

Bitcoin Network

- By default all peers maintain **up to 125 connections** with other peers
  - Each node, when joins the network **tries to connect to 8 other** peers (**outgoing** links)
  - Each node **can accept up to 117** connections from potential peers (**incoming** links)

Bitcoin Network

- See https://bitnodes.io/



Source: Cryptocurrency Networks: a New P2P Paradigm https://www.hindawi.com/journals/misy/2018/2159082/

Bitcoin Network

- No central authority
- Designed having in mind
  - **Reliability** (by design, every peer stores all relevant information, and this is highly inefficient from the storage point of view)
  - **Security** (properties of transactions and blocks can prevent flooding attacks, tampering of the data, and others)

Bitcoin Network

- **Flooding/DoS attacks**
  - Transaction flooding is prevented by **not relaying invalid transactions** and by requiring fees for valid transactions
  - Block flooding is prevented by **only relaying valid blocks** which must contain a **valid nonce** for the PoW
  - **Reputation-based** mechanism in which each node keeps a penalty score for every connection. The penalty increases with the number of malformed messages sent on the connection. Misbehaving peers whose penalty scores reach a given value are banned for 24 hours

Bitcoin Network

- **Join**
  - Many Bitcoin nodes usually operate behind a NAT or firewall
  - When a node **joins the network** it must **discover its own public IP address**
  - It **can send a GET request** to two hard-coded websites which reply with the address
  - It is also possible to **manually configure** the node with the public IP address and the port in the node's settings (e.g., via bitcoin.conf)

Bitcoin Network

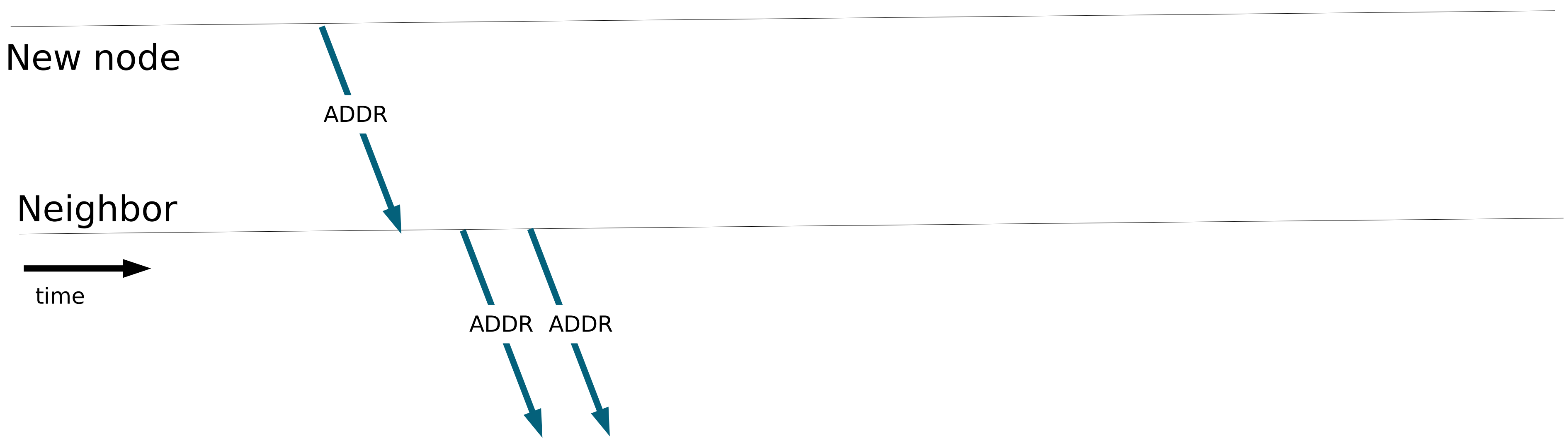
- **Join**
  - After joining the network, the node may **discover other IP addresses** from the peers it connects with
  - Each peer **keeps a list of IP addresses** associated with its connections into a local database
    - **tried** table: **IP + timestamp** of other peers known from past connections
    - **new** table: new **IP + timestamp**, populated by addresses learned by **DNS seeders** or by querying **neighbors**

Bitcoin Network

- **Address propagation**
  - Peers can request addresses sending **GETADDR** messages to their neighbors
  - They also receive unsolicited **ADDR** messages which contain IP addresses
  - Each node can decide to forward them or not

Bitcoin Network

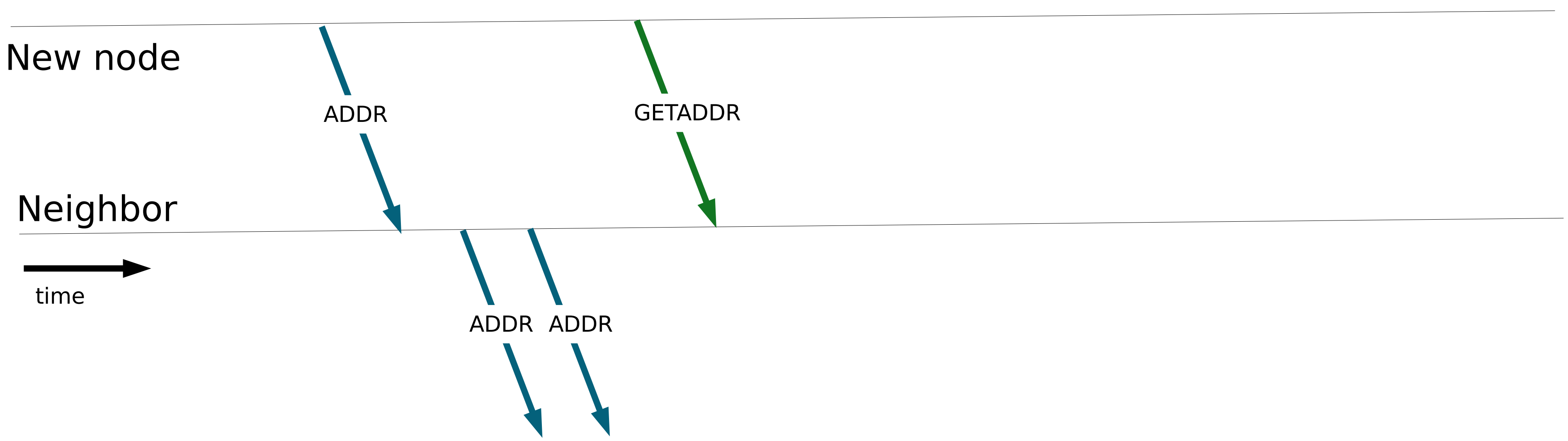
Bitcoin Network



The neighbor receiving the **ADDR message** from the new node will **store the received IP** in the local database and, in turn, will **forward it to its neighbors**

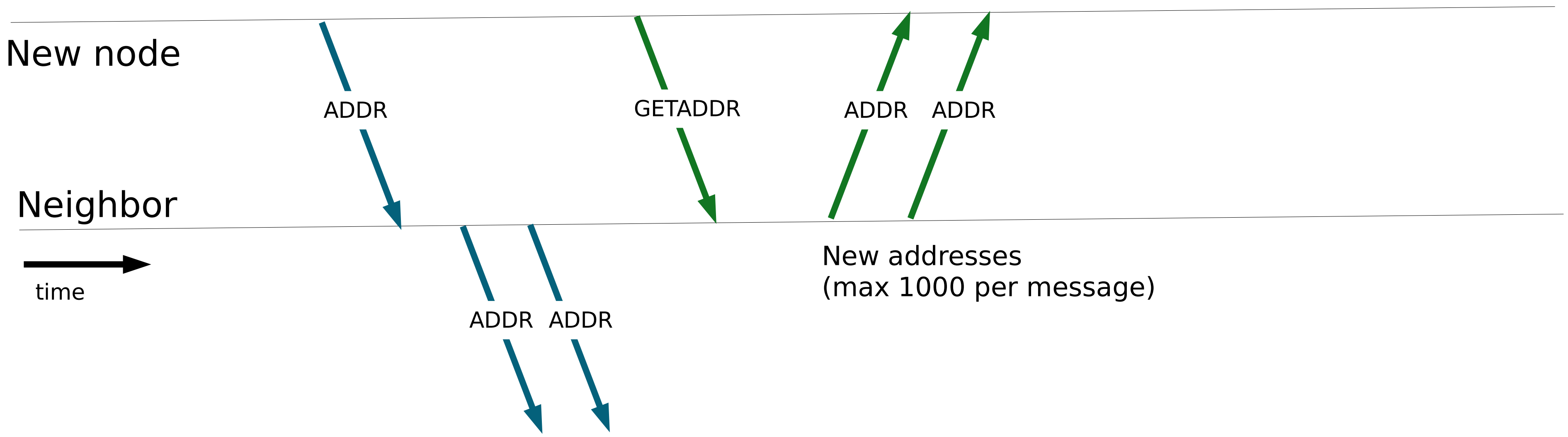
This ensures that a newly connected node is well known by the network

Bitcoin Network



Additionally, a **GETADDR** message can be sent by the new node asking its neighbors to return a list of IP addresses of other peers

Bitcoin Network



Bitcoin Network

- Peers try to always maintaining their 8 outgoing connections, selecting new nodes when their neighbors leave the network
- Neighbors are selected from the local database following a **pseudo random procedure** that gives the network **high dynamism** and keeps its **overall structure unknown**

Bitcoin Network

- Peers exchange also data structures
  - **Transactions** are the data structures **most usually seen** (see https://www.blockchain.com/charts)
    - Every single node can take part in a transaction by simply using a wallet, no matter of its type
  - **Blocks are less frequent**, on average 6 blocks per hour

Transaction propagation

- Bitcoin's method for transactions is **flooding-based**
- When a node **creates** a transaction
  - **serializes** it in hexadecimal format
  - **announces** it using an **INVENTORY** message

```
Message header:
    Command: "inv"
    Payload length: N

Variable-length payload:
    Count (4-byte unsigned integer): The number of inventory entries in the message.
    Inventory entries (variable length):
        Type (1-byte unsigned integer): The type of object that the inventory entry identifies.
        Hash (32-byte hash): The hash of the object.
```

0	Block
1	Transaction
2	Filtered block
3	Witness data
4	Compact block

Transaction propagation

- Bitcoin’s method for transactions is **flooding-based**
- When a node **receives** an **INVENTORY** message
  - requests the transaction using a **GETDATA** message
  - the sender node sends the full transaction data to the requesting nodes using the **TX** message
  - the receiving nodes validate the transaction and **add it to their mempool** (temporary store for unconfirmed transactions)

Transaction propagation

- Nodes keeps **valid transactions** in the **mempool** and answer requests for them
- Some Bitcoin clients uses **Bloom filter** encoding the transaction IDs in their mempool, thus only missing transactions can be exchanged
- Nodes periodically **exchange Bloom filters** to avoid forwarding transactions to those neighbors who already have them

Block propagation

- When a **miner finds a nonce**, it creates a block with
  - the block header (previous block hash, timestamp, nonce, etc.)
  - the list of transactions
  - the Merkle root (the cryptographic hash of all the transactions)
- It **advertizes** the new block to its neighbors with an **INVENTORY** message containing the hash of the block

Block propagation

- When neighbors **receive** the INVENTORY message, they check if they already have the block
- If they do not have it, they respond with a **GETDATA** request, asking for the full block
- The **miner** (peers) **sends** the block with a **NEWBLOCK** message
- The neighbors **verify the new block** and, if valid, they **forward** its to their neighbors

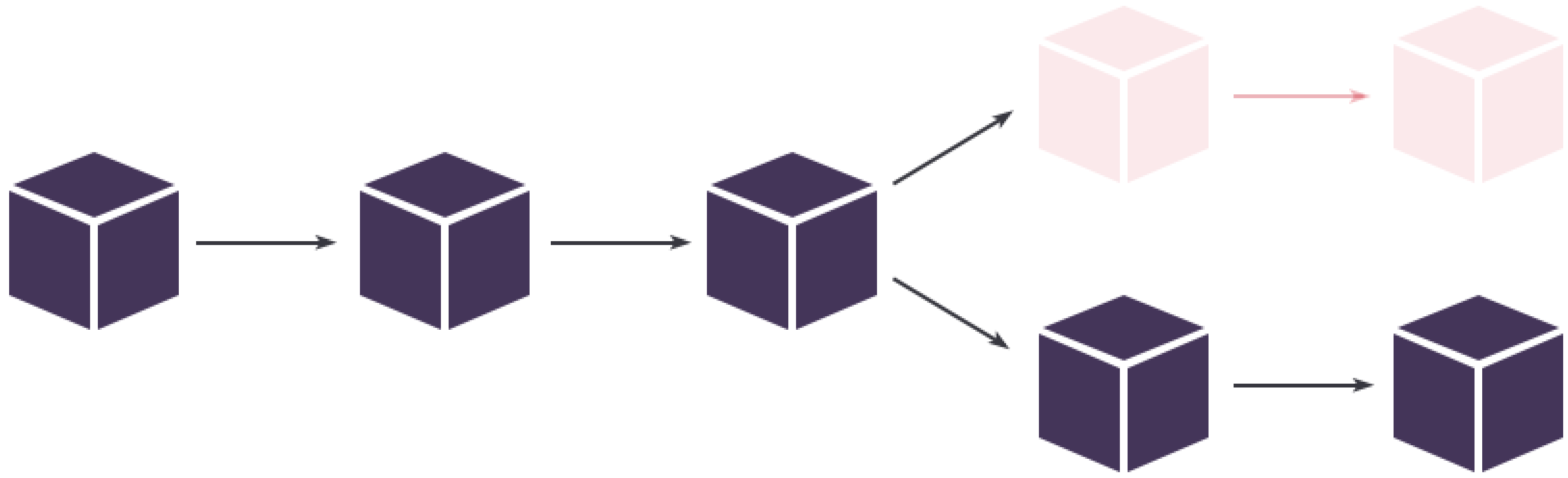
Block propagation

- Since blocks are larger and more critical for maintaining consensus, the Bitcoin network uses several optimizations to improve the speed and efficiency of block propagation

- The protocol specification is rather complex, if interested see https://en.bitcoin.it/wiki/Network (optional)

Block propagation

- Of course, delayed blocks may lead to **forks**
- Forks should be avoided as they are symptomatic for inconsistencies among the replicas in the network



Bitcoin network summary

- By design
  - **high level of reliability** thanks to its **redundancy**: the availability of a single node in the network contains the information to keep the system alive
  - **high inefficiency** in terms of **storage space**
  - **slow**
  - **computationally intensive (energy-intensive)**