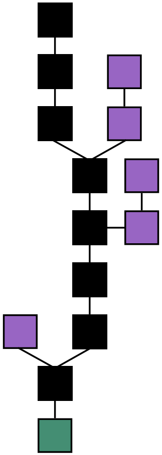
Bitcoin: Simplified Payment Verification

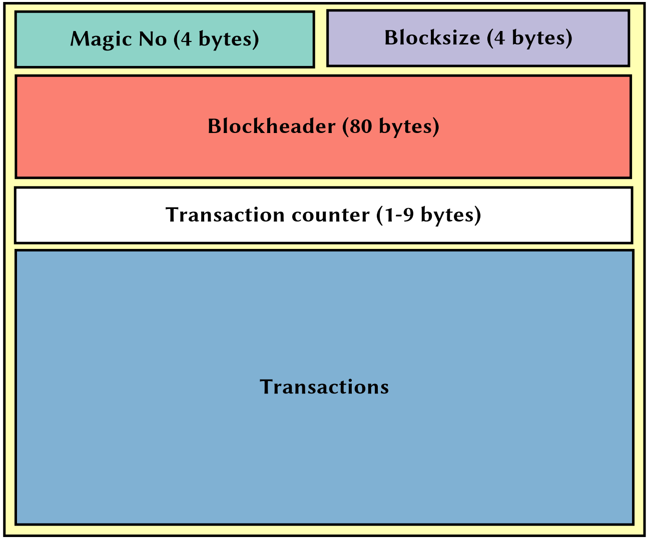
George Vlahavas

September 24, 2018

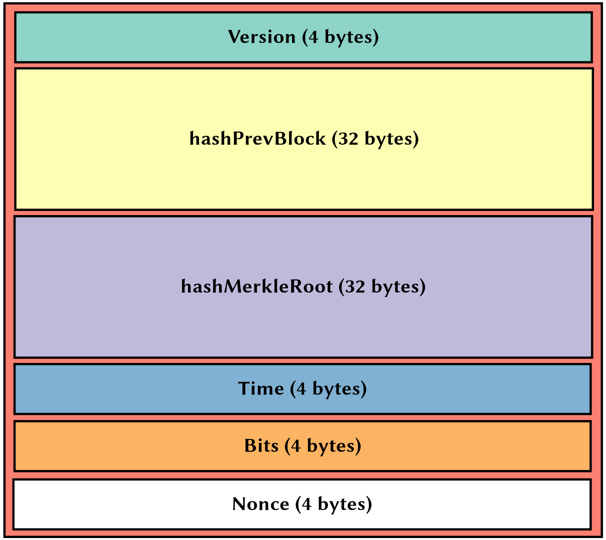
The blockchain



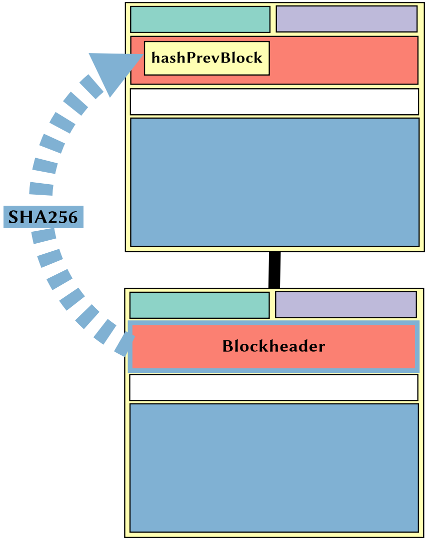
What's in a block?



What's in a block header?



Hashing the previous block



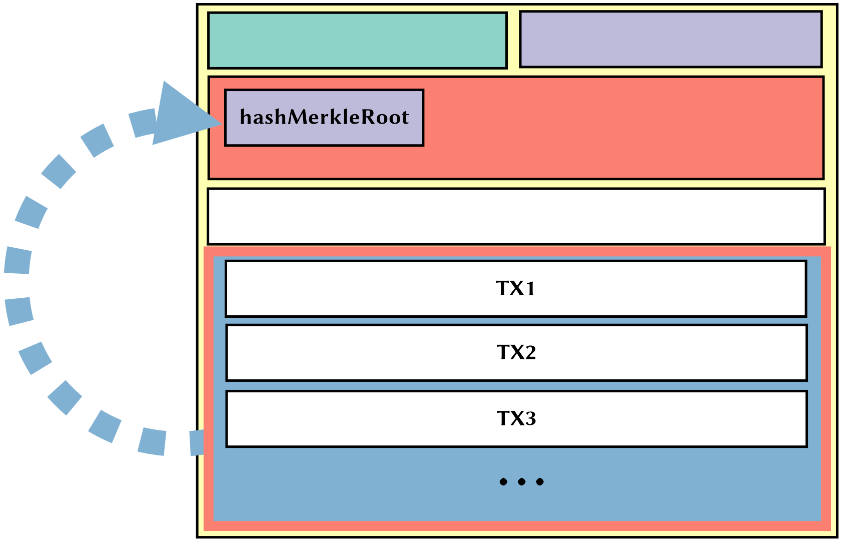
Transaction immutability

- The hash of the previous block is calculated only using the previous block's header
- But then how is it ensured that transactions in previous blocks won't be altered?
  - *The transactions themselves are hashed and their hash is stored in the current block's Merkle Root*

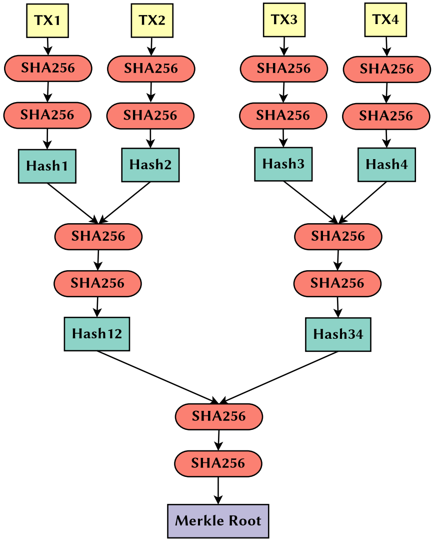
Transaction immutability

- So, if a transaction is altered, the Merkle Root would be changed.
- Since the Merkle Root is included in the block header, if the Merkle Root is changed, the block header contents would be changed.
- If the block header contents are changed the block header hash would be changed.

Calculating the Merkle Root



The Merkle Tree



But why?

Why do we go through all that trouble and not just hash the entire block, including the transactions and put that in the next block's hashPrevBlock?

- SPV nodes!

Full nodes

- Full nodes receive, validate and propagate every single transaction and every single block throughout the network.
- But you cannot run a full node on a smartphone or an embedded system. The Bitcoin blockchain is currently about 200GB!

SPV nodes

SPV: Simplified Payment Verification

- Lightweight clients
- Only download block headers
  - 1000 times smaller blockchain size
- Only filter/receive relevant transactions

SPV: Verifying transactions

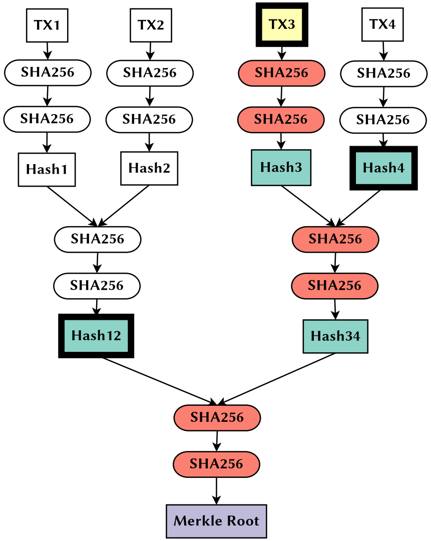
How does an SPV node verify that a transaction has been mined into a block?

- The SPV node connects to a full node
- The SPV node asks the full node to search for the transaction it is interested in
- The full node responds with:
  - the block header hash of the block the transaction is included in
  - the Merkle Path of the transaction
- The SPV node can identify the block using its block header hash; it already has that info
- The Merkle Root is also part of the block header, so the SPV node already has that too

Verifying the Merkle Path

- The SPV node needs to rehash the transaction and verify the Merkle Root that is stored in the block header
- It needs to do that as efficiently as possible
- Using the Merkle Path, it only downloads the relevant hashes from the full node
- The SPV node does not have to know anything about the other transactions in the block

Verifying the Merkle Path example



Verifying by height/depth

- Full nodes verify transactions by reference to their block *height*
  - It can verify that the UTXO remains unspent
- SPV nodes verify transactions by reference to their block *depth*
  - It cannot validate whether the UTXO is unspent, it just surmises that

SPV node security issues

- An SPV node can confirm that a transaction is actually included in a block
- An SPV node cannot be convinced that a transaction exists if the transaction does not really exist
- But, a transaction can be "hidden" from an SPV node
  - DoS attacks
  - Double spend attacks
- SPV nodes connect randomly to several different nodes
  - Still, they are vulnerable to network partitioning or Sybil attacks
- A full node is the most secure one can hope for, but for most practical purposes, a well-connected SPV node is secure enough
- Also possible to only allow connections from an SPV node to a known trusted full node

SPV node privacy issues

- SPV nodes request specific transactions from the full nodes they connect to
- A 3rd party monitoring the network could associate bitcoin addresses with users

Bloom filters

Bloom filters allow SPV nodes to receive a subset of the transactions without revealing precisely which addresses they are interested in, through a filtering mechanism that uses probabilities rather than fixed patterns.

- Essentially, SPV nodes do not request specific transactions, but rather transactions following a specific pattern.
- Search patterns can be:

  More specific: Download fewer transactions, increasing accuracy, at the cost of privacy.

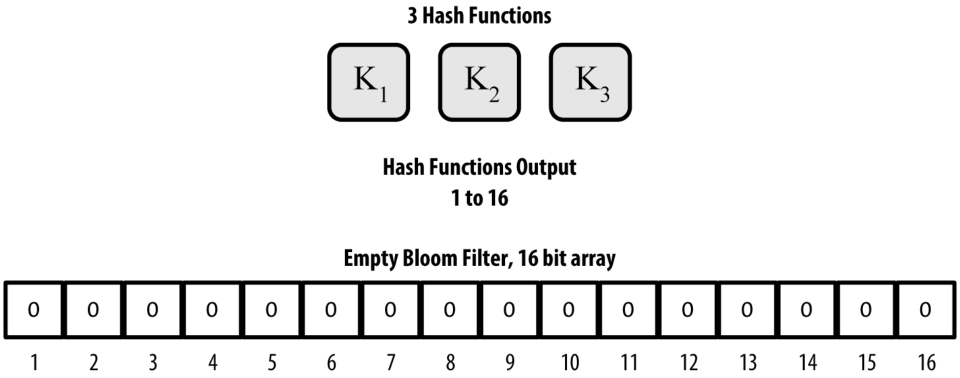
  Less specific: Download more transactions, most irrelevant, increases privacy.

How bloom filters work

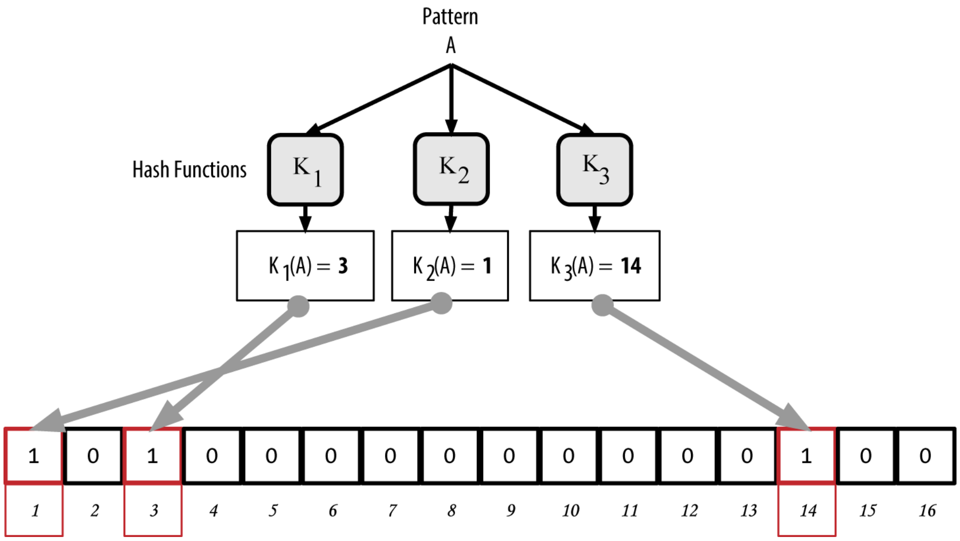
- M hash functions (max: 50)
- a bit field of size N (max size: 36000 bytes)
- Hash functions are designed to always produce an output that is between 1 and N
- Hash functions are generated deterministically, so that any node implementing a bloom filter will always use the same hash functions and get the same results for a specific input
- Bloom filters can be tuned by choosing different N and M values

Bloom filter example

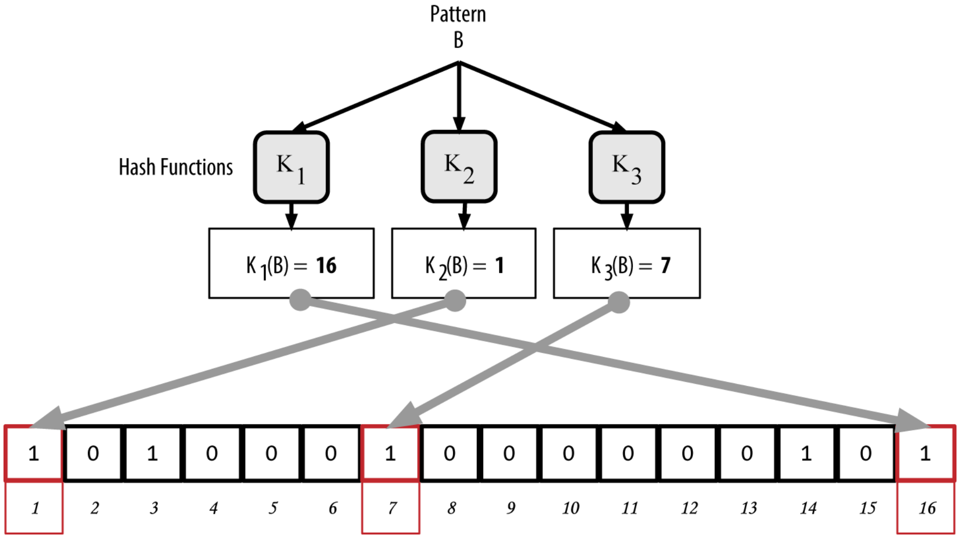
- $M = 3$
- $N = 16$



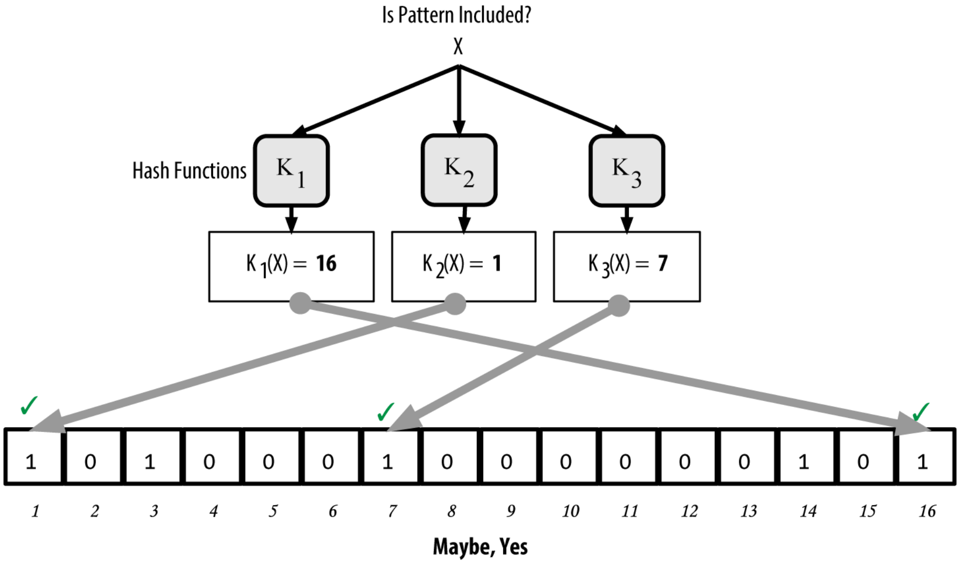
Bloom filters: Adding a pattern



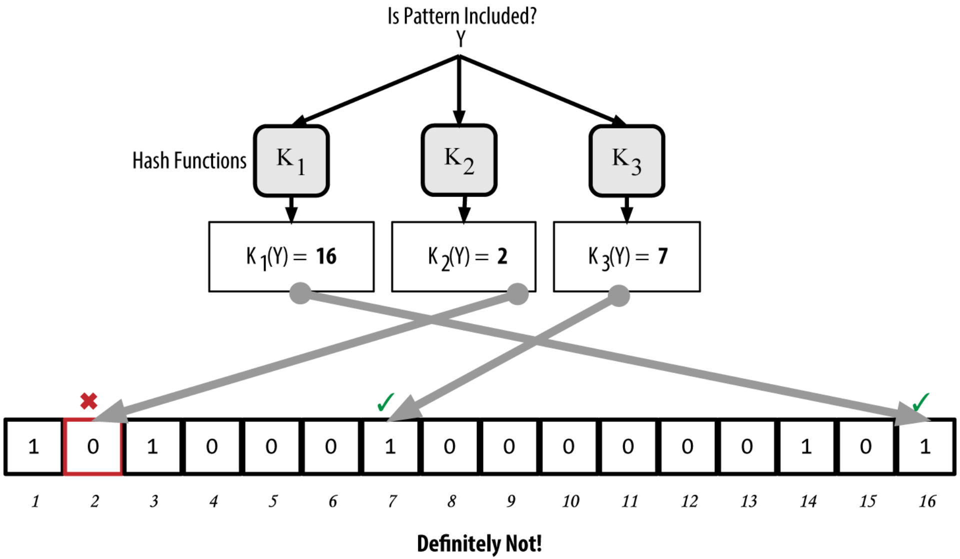
Bloom filters: Adding another pattern



Bloom filters: finding a pattern



Bloom filters: not finding a pattern



Bloom filter editing

It is only possible to:

- Clear bloom filters
- Add patterns to bloom filters

Not possible to remove patterns from a bloom filter.

Bloom filter search results

Searching for a pattern in a bloom filter will yield either:

- *Maybe* it is included, or
- It's *definitely* not included

How SPV nodes use bloom filters

1. The SPV node creates an empty bloom filter
2. The SPV node makes a list of everything it is interested in:
   - addresses
   - keys
   - hashes
3. The SPV node adds all of these to the bloom filter
4. The SPV node sends the bloom filter to the full node
5. The full node checks the bloom filter against the blockchain:
   - public key hashes
   - scripts
   - OP_RETURN values
   - public keys in signatures
6. Only transactions that match the bloom filter are sent back to the SPV node

How SPV nodes use bloom filters

7. For each matching transaction, the full node sends back:
    - Block headers
    - Merkle Paths
    - TX messages
8. The SPV node discards all irrelevant info
9. The SPV node uses the matched transactions to update its UTXO set and wallet balance
10. The SPV node modifies the bloom filter to match any future transactions referencing the UTXO it just found

Summary

SPV nodes...

- only download block headers
- nodes only receive relevant transactions
- verify transaction by reference to block depth
- are not as secure as full nodes, but secure enough for most simple usecases
- use Bloom filters to mitigate privacy issues

Contact

George Vlahavas

vlahavas.com
george@vlahavas.com
github.com/gapan