(Almost)

Bringing secure Bitcoin transactions to your smartphone

Davide Frey$^1$, Marc X. Makkes$^2$, Pierre-Louis Roman$^3$, François Taïani$^3$, Spyros Voulgaris$^2$

$^1$ Inria Rennes
$^2$ Vrije Universiteit Amsterdam
$^3$ University of Rennes 1 – IRISA – ESIR
TL;DR blockchain

Bitcoin: peer-to-peer payment system, backed by a blockchain

Blockchain overview:

- Distributed database: full replica at every node
- Trustless: no authority, nodes verify everything locally
- Probabilistic immutability: very difficult to replace a block
- Resistant: incentives to have honest nodes
TL;DR blockchain

The blockchain data structure

Ledger of transactions
Transactions are encapsulated in blocks
Blocks provide probabilistic immutability
Reverse linked list of blocks
TL;DR blockchain

The blockchain data structure

Ledger of transactions
Transactions are encapsulated in blocks
Blocks provide probabilistic immutability
Reverse linked list of blocks

\[ H_{i-1} = 0x00bab10c \]
Difficulty
Timestamp
Tx Merkle tree root
Nonce

[ Tx\(_1\), Tx\(_2\), ..., Tx\(_n\) ]
TL;DR blockchain

The blockchain data structure

Ledger of transactions
Transactions are encapsulated in blocks
Blocks provide probabilistic immutability
Reverse linked list of blocks

\[
H_{i-1} = 0x00bab10c
\]

SHA-256

\[
H_{i-1} = 0x\text{deadbeef}
\]

Difficulty
Timestamp
Tx Merkle tree root
Nonce

\[
[ \ Tx_1, \ Tx_2, \ …, \ Tx_n \ ]
\]

Time
**TL;DR blockchain**

**The blockchain data structure**

Ledger of transactions

Transactions are encapsulated in blocks

Blocks provide probabilistic immutability

Reverse linked list of blocks

---

\[ H_{i-1} = \text{SHA-256}^2 \]

- Difficulty
- Timestamp
- Tx Merkle tree root
- Nonce

\[ [\text{Tx}_1, \text{Tx}_2, \ldots, \text{Tx}_n] \]
Making a transaction

Miners

Peers

Tony

10 BTC

B3

B3

B4

B2

B8

Lenny

0 BTC

Miners

Pierre-Louis Roman (U. Rennes)
Secure Bitcoin on smartphones
ARM 2016
Making a transaction

Transaction: 0xbaadc0de

Peers

Tony
10 BTC

Miners

0xbaadc0de

Miners

Peers

Lenny
0 BTC

Transaction: 0xbaadc0de
Making a transaction

Transaction: 0xbaadc0de

Peers

Tony

Miners

0xbaadc0de

0 BTC

Lenny

Making a transaction

Transaction: 0xbaadc0de

Peers

Tony

Miners

0xbaadc0de

Lenny

Tony

Peers

2 BTC

40 BTC

2 BTC

Lenny

0 BTC

8 BTC

Miners

Block

0xbaadc0de

[...]
Transactions

Valid transaction:
- Spend $\geq$ Receive
- The money exists, and is not spent yet

Money moves from transaction to transaction
## Transactions

**Valid transaction:**
- Spend $\geq$ Receive
- The money exists, and is not spent yet

**Money moves from transaction to transaction**

<table>
<thead>
<tr>
<th>List of inputs</th>
<th>List of outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 previous Tx + output index, proof ownership</td>
<td>0 value, ownership challenge</td>
</tr>
<tr>
<td>[...]</td>
<td>[...]</td>
</tr>
<tr>
<td>n-1 previous Tx + output index, proof ownership</td>
<td>m-1 value, ownership challenge</td>
</tr>
</tbody>
</table>
Verifying transactions

**UTxO set**
Set of Unspent Transaction Outputs
Verifying transactions

**UTxO set**

Set of Unspent Transaction Outputs

---

UTxO set = \{ 1, 2, 3 \}

---

Time

---

Metadata

1 → 4
2 → 5
Verifying transactions

**UTxO set**

Set of Unspent Transaction Outputs

Time

UTxO set = \{ 1, 2, 3, 4, 5 \}

Metadata

- 1 → 4
- 2 → 5

UTxO set = \{ 4, 2, 3, 4, 5 \}
Verifying transactions

**UTxO set**

Set of Unspent Transaction Outputs

UTxO set = \{ 3, 4, 5 \}
Verifying transactions

**UTxO set**
Set of Unspent Transaction Outputs

UTxO set = \{ 3, 4, 5, 6, 7, 8 \}
Verifying transactions

UTxO set

Set of Unspent Transaction Outputs

UTxO set = \{ 6, 7, 8 \}
Verifying transactions

UTxO set
Set of Unspent Transaction Outputs

UTxO set = \{ 6, 7, 8 \}
**Verifying transactions**

**UTxO set**
Set of Unspent Transaction Outputs

UTxO set = \{ 6, 7, 8 \}

**Problem 1: time**
Building the UTxO is long, can’t skip-ahead
Problem 2: resources

Building the UTxO is bandwidth and storage heavy
Problem 2: resources

Building the UTxO is bandwidth and storage heavy
Problem and solution

Fully secured node = long bootstrap, high storage, high bandwidth
Problem and solution

**Fully secured node** = long bootstrap, high storage, high bandwidth

**Directly download the UTxO**

How do you trust the downloaded set?
Puts its hash in the blockchain
$H_{i-1} = 0x00bab10c$
Difficulty
Timestamp
Tx Merkle tree root
Nonce

$[\text{Tx}_1, \text{Tx}_2, \ldots, \text{Tx}_n]$
$H_{i-1} = 0x00bab10c$
Difficulty
Timestamp
Tx Merkle tree root
Nonce

$H_{\text{blocks}}$

$[\text{Tx}_1, \text{Tx}_2, \ldots, \text{Tx}_n]$
\( H_{i-1} = 0x00bab10c \)

Difficulty
Timestamp
Tx Merkle tree root
Nonce

\( H_{\text{blocks}} \)

\( H_{\text{UTxO}} \)

\[ [ \text{Tx}_1, \text{Tx}_2, \ldots, \text{Tx}_n ] \]
$H_{i-1} = 0x00bab10c$

- Difficulty
- Timestamp
- Tx Merkle tree root
- Nonce

$H_{\text{blocks}}$

$H_{\text{UTxO}}$

$[\text{Tx}_1, \text{Tx}_2, \ldots, \text{Tx}_n ]$

$H_{\text{blocks}}$ to dig in the history

$H_{\text{UTxO}}$ to verify transactions

UTxO updates?

Need to "trust" the block
Reducing time and resource needs

Make the UTxO downloadable: enable skip-aheads!
Reducing time and resource needs

Make the UTxO downloadable queriable: skip-aheads and lower costs!

Broaden the spectrum of possible verification strategies
Adaptive strategies for different storage/bandwidth/security trade-offs
Reducing time and resource needs

Make the UTxO downloadable queriable: skip-aheads and lower costs!

Broaden the spectrum of possible verification strategies
Adaptive strategies for different storage/bandwidth/security trade-offs

With a partial UTxO, a client can:
- Verify new transactions
- Verify some transactions in a block
- Verify every transactions in a block (same as full node)
Verifying new transactions

"Show me the money"

Prevents:
- Spending fake coins

Possible attacks:
- Double spends
Verifying some transactions in a block

Verify transactions to you

Prevents:
- Spending fake coins
- Double spends
- Cheap attacks

Possible attacks:
- Blocks with valid transactions to you, invalid to others
Verifying every transactions in a block

Similar security level as full nodes

Prevents:
- Spending fake coins
- Double spends
- Blocks with valid transactions to you, invalid to others
- Basically everything

Vulnerable against:
- ∅
Partial UTxO

With a partial UTxO, a client can:

- Verify new transactions
- Verify some transactions in a block
- Verify every transactions in a block (same as full node)
Partial UTxO

With a partial UTxO, a client can:

- Verify new transactions
- Verify some transactions in a block
- Verify every transactions in a block (same as full node)

Splitting strategies:

- Sharding: query subsets of the UTxO (our proposal)
- Merkle tree: query individual coins in the UTxO
UTxO - Sharding

\[ H_{i-1} = 0x00bab10c \]
Difficulty
Timestamp
Tx Merkle tree root
Nonce

\[ H_{\text{blocks}} \]

\[ H_{\text{UTxO}} \]

\[ [ \text{Tx}_1, \text{Tx}_2, \ldots, \text{Tx}_n ] \]
UTxO - Sharding

UTxO set size = 1.5 GiB

Sharding policy: random, first k bits, e.g., k = 16

Number of shards = $2^{16}$

Shard size = $1.5 \text{ GiB} \div 2^{16} \approx 25 \text{ KiB}$

Server storage overhead: $2^{16} \times 32 \text{ B} = 2 \text{ MiB}$

Typical block verification:
- 2000 inputs minimum
- With random sharding, needs 2000 shards
- Client bandwidth consumption: $2000 \times 25 \text{ KiB} \approx 49 \text{ MiB} + 2 \text{ MiB}$

High bandwidth
\[ H_{i-1} = 0x00bab10c \]
Difficulty
Timestamp
Tx Merkle tree root
Nonce
\[ H_{\text{UTxO}} \]

[ \text{Tx}_1, \text{Tx}_2, \ldots, \text{Tx}_n ]

Author: Andrew Miller (UIUC)
UTxO - Merkle tree

Number of items in the UTxO = 15 M
Number of nodes in the tree = 2 × 15 M = 30 M
Number of hashes to reconstruct the merkle root: \(\log_2(30 \text{ M}) \approx 25\)

Server storage overhead: \(30 \text{ M} \times 32\text{ B} \approx 915\text{ MiB}\)

Typical block verification:
- 2000 inputs minimum
- Needs 2000 proof of existence in the tree
- Client bandwidth consumption: \(2000 \times 25 \times 32\text{ B} \approx 1.5\text{ MiB}\)

Balancing the tree, storage overhead on servers
Summary of solutions

<table>
<thead>
<tr>
<th></th>
<th>Client bandwidth</th>
<th>Server storage overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full UTxO</strong></td>
<td>1.5 GiB</td>
<td>0</td>
</tr>
<tr>
<td><strong>Sharded UTxO</strong></td>
<td>49+2 MiB</td>
<td>2 MiB</td>
</tr>
<tr>
<td><strong>Merkle tree UTxO</strong></td>
<td>1.5 MiB</td>
<td>915 MiB</td>
</tr>
</tbody>
</table>

Sharding is the most-balanced, but not ideal
The good, the bad and the unicorn

Decoupled storage
Quick client bootstrap
Light storage client
The good, the bad and the unicorn

Decoupled storage
Quick client bootstrap
Light storage client

High bandwidth clients?
Light clients are not as safe as full nodes
Low overhead for full nodes

Secure Bitcoin on smartphones
The good, the bad and the unicorn

Decoupled storage
Quick client bootstrap
Light storage client

High bandwidth clients?
Light clients are not as safe as full nodes
Low overhead for full nodes

Still looking for the optimal trade-off

Any questions?