Applying Private Information Retrieval to Lightweight Bitcoin Clients

Kaihua Qin, Henryk Hadass, Arthur Gervais and Joel Reardon
Bitcoin SPV Clients
Simple Payment Verification (SPV) Clients

Blockchain Block
Simple Payment Verification (SPV) Clients

Header

Blockchain Block
Simple Payment Verification (SPV) Clients

Blockchain Block

- Header
- Transactions
Simple Payment Verification (SPV) Clients

Blockchain Block

Transactions

Header

02000000 .......................... Block version: 2
06ff0bfba482862a30ca40d46d9e8
9100336eb86ad9c000000000000000  Hash of previous block's header
9d10aa3f29b94386ca9385656f04de2
70d0a2b1080cad13ba9b0488aab31471  ... Merkle root
24085a5d .......................... Unix time: 1451239992
30c1b1b8 .......................... Target: 0xbcc3c0 * 256**(0x18-3)
fe9f8064 .......................... Nonce
Simple Payment Verification (SPV) Clients

Blockchain Block

Header

Transactions

Hash

Transaction

Hash

Transaction

Hash

Transaction

Hash

Transaction
## Simple Payment Verification (SPV) Clients

### Blockchain Block

<table>
<thead>
<tr>
<th>Header</th>
<th>Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>02000000</td>
<td></td>
</tr>
<tr>
<td>bdf0f1b1d80a2862a30c4a4d34f03d0e8</td>
<td></td>
</tr>
<tr>
<td>9103d3e6b48688c5000000000000000</td>
<td>Hash of previous block's header</td>
</tr>
<tr>
<td>9d1da35c2e94926e6a9385955f006e02</td>
<td></td>
</tr>
<tr>
<td>70d6a2081086cd3db3b9b0468ab31471</td>
<td>Merkle root</td>
</tr>
<tr>
<td>36df5a58</td>
<td></td>
</tr>
<tr>
<td>30c31b1b8</td>
<td>Unix time: 1615239972</td>
</tr>
<tr>
<td>fo0f0864</td>
<td>Target: 0x1bc380 * 256**(0x18-3)</td>
</tr>
<tr>
<td></td>
<td>None</td>
</tr>
</tbody>
</table>

### Transactions

```
Hash

Transaction

Transaction

Transaction

Transaction
```
Simple Payment Verification (SPV) Clients

Transactions

Blockchain Block

Header

<table>
<thead>
<tr>
<th>Header</th>
<th>Transactions</th>
<th>Blockchain Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>02000000 .......................... Block version: 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b6f0b1d6b02b2662a30ca460d46d9e8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>310d334bea8b0c00000000000000000</td>
<td>Hash of previous block’s header</td>
<td></td>
</tr>
<tr>
<td>9d10aa32e9493b66ac93556550f60e2e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70d8a20b0ddd12eb3b048e8ab31471</td>
<td>Merkle root</td>
<td></td>
</tr>
<tr>
<td>24985a4a</td>
<td>.................. Unix time: 1615239972</td>
<td></td>
</tr>
<tr>
<td>70c31b18</td>
<td>.................. Target: 0x1bc030 * 256**(0x18-3)</td>
<td></td>
</tr>
<tr>
<td>fe9f0864</td>
<td>.................. Nonce</td>
<td></td>
</tr>
</tbody>
</table>

Merkle Root

Hash

Transaction

Transaction

Transaction

Transaction

Transaction
Simple Payment Verification (SPV) Clients

SPV client

Full Bitcoin node
Simple Payment Verification (SPV) Clients

SPV client

Did I receive new coins?

Full Bitcoin node
Simple Payment Verification (SPV) Clients

SPV client → Did I receive new coins? → Relevant transactions → Full Bitcoin node
Simple Payment Verification (SPV) Clients

SPV client

Did I receive new coins?

Relevant transactions

Full Bitcoin node

Privacy!
<table>
<thead>
<tr>
<th>Bloom filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
Bloom Filter

Insertion

\{ @_1, @_2, @_3 \}

Bloom filter

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Bloom Filter

Insertion

{ @1, @2, @3 }

Bloom filter

\[ \begin{array}{cccccccc}
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{array} \]
Bloom Filter

Insertion

Bloom filter

\{ \@_1, \@_2, \@_3 \}
Bloom Filter

Insertion

Bloom filter

{ @₁, @₂, @₃ }

0 1 0 0 1 1 0
Bloom Filter

Insertion

Bloom filter

{ @1, @2, @3 }

Membership test

{ @1, @4, @5 }
Bloom Filter

Insertion

Bloom filter

Membership test
Bloom Filter

Insertion

Bloom filter

Membership test
Bloom Filter

Insertion

Bloom filter

Membership test

@_4 False positive

target False Positive Rate (FPR)
Bloom Filter

Insertion

Bloom filter

Membership test

@4 False positive

target False Positive Rate (FPR)
Bloom Filter

Insertion

Bloom filter

Membership test

@4 False positive
@5 True negative

target False Positive Rate (FPR)

BIP-37 (Bloom Filters)

- SPV client
- Full Bitcoin node
- Full Bitcoin node
BIP-37 (Bloom Filters)

SPV client

Full Bitcoin node

Full Bitcoin node

Bloom filter
BIP-37 (Bloom Filters)
BIP-37 (Bloom Filters)

SPV client

Connection

Bloom filter

Full Bitcoin node

transactions

Full Bitcoin node

Is transaction relevant for Bloom filter?
BIP-37 (Bloom Filters)

SPV client

Connection

Bloom filter

Relevant transactions

Full Bitcoin node

@1
@2
@3

Bloom filter

0100110

Is transaction relevant for Bloom filter?

transactions

Full Bitcoin node
BIP-37 (Bloom Filters)

SPV client

Connection

Bloom filter

Relevant transactions

Is transaction relevant for Bloom filter?

transactions

Full Bitcoin node

Full Bitcoin node
BIP-37 (Bloom Filters)

**Promise:** 33 mio addresses in the Blockchain

```
"User addresses hidden amongst 33 000" false positives
```
Model and Privacy measure

SPV client

Blockchain
Model and Privacy measure

Blockchain

SPV client

Adversary
Model and Privacy measure

SPV client → Bloom filter + parameters (seed, FPR) → Adversary

Blockchain
Model and Privacy measure

Blockchain

+ parameters (seed, FPR)

0 1 0 0 1 1 0

SPV client

Bloom filter

Adversary

All addresses of the Blockchain
Model and Privacy measure

SPV client → Bloom filter + parameters (seed, FPR) → Adversary → Blockchain → All addresses of the Blockchain → Total positives
Model and Privacy measure

SPV client

Blockchain

+ parameters (seed, FPR)

Bloom filter

All addresses of the Blockchain

Total positives

Adversary

True positives
Model and Privacy measure

SPV client

Blockchain

All addresses of the Blockchain

Bloom filter

+ parameters (seed, FPR)

Adversary

Total positives

Measure privacy

True positives

0 1 0 0 1 1 0

0 0 1 0 0 0
Model and Privacy measure

Blockchain

All addresses of the Blockchain

Total positives

Measure privacy

True positives

SPV client

Bloom filter

+ parameters (seed, FPR)

0 1 0 0 1 1 0

Adversary

+ @

Imperial College London
Model and Privacy measure

SPV client

Blockchain

All addresses of the Blockchain

Total positives

Measure privacy

True positives

Bloom filter

Bloom filter 2

Adversary

+ parameters (seed, FPR)

0 1 0 0 1 1 0

0 1 0 0 1 1 0

0 1 0 0 1 1 0

0 1 0 0 1 1 0
1. Given one Bloom filter, Bitcoin addresses partially linkable
   - Addresses linkable if < 20 addresses in wallet

2. Given multiple Bloom filter, addresses nearly always linkable

Gervais et al., On the privacy provisions of bloom filters in lightweight bitcoin clients, ACSAC'14
Proposed solution

- Pre-generate Bitcoin addresses and insert into filter
- Keep state about outsourced Bloom filter
- Overhead: For 100 addresses, < 1 kb
Private Information Retrieval

Query the Database
without learning anything about the query

Blockchain
Private Information Retrieval

Query the Database without learning anything about the query

Solution: Download the entire DB!
PIR Versions

PIR
PIR Versions
PIR Versions

- IT-PIR
- C-PIR
- PIR
PIR Versions

- IT-PIR
- C-PIR

Q1 → Q2 → Q3
Information Theoretic security, assuming no collision up to a certain degree
PIR Versions

Information Theoretic security, assuming no collision up to a certain degree

Computational PIR, query privacy via cryptographic means
## IT-PIR vs. C-PIR

<table>
<thead>
<tr>
<th></th>
<th>IT-PIR</th>
<th>C-PIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication costs</td>
<td>Smaller</td>
<td>Bigger</td>
</tr>
<tr>
<td>Computation costs</td>
<td>Smaller</td>
<td>Bigger</td>
</tr>
<tr>
<td>Risk of Collusion</td>
<td>Server can mount sybil attack</td>
<td>Non existent</td>
</tr>
<tr>
<td>Robust to missing or incorrect server response</td>
<td>Robust</td>
<td>Non-robust</td>
</tr>
</tbody>
</table>
The best of both worlds: Hybrid PIR

- combines the IT-PIR protocols with the C-PIR protocols
- queries are operated in a recursive way
- maintains partial privacy of client query information if the assumptions made by one of the inner protocols is broken
- open-source Percy++ library

Devet, C. and Goldberg, I., The best of both worlds: Combining information-theoretic and computational PIR for communication efficiency. PETS’14
(a) PIR servers download the blockchain, constructs PIR databases. For each database, the PIR server creates a description file called manifest file.

(b) The user collects all available block headers from e.g., full node peers.

(c) The user fetches the manifest files from the PIR servers to later efficiently query the PIR database.

(d) The user executes the PIR-SPV protocol, decodes the PIR responses for servers and then performs SPV validation.

Devet, C. and Goldberg, I., The best of both worlds: Combining information-theoretic and computational PIR for communication efficiency. PETS’14
Database structure

Manifest files

- Address PIR Manifest
- Merkle Tree PIR Manifest
- Transaction PIR Manifest

PIR databases

- Address PIR DB row entry - 62 bytes
  - address - 25 bytes
  - TXID - 32 bytes
  - blockheight - 3 bytes
  - vout index - 2 bytes

- Merkle Tree PIR DB row
  - TXID - 32 bytes
  - TXID - 32 bytes
  - TXID - 32 bytes
  - ..."
Temporal Division
PIR Protocol - Pre-requisites

1) Download Manifest files
   - Address PIR Manifest
   - Merkle Tree PIR Manifest
   - Transaction PIR Manifest

2) Download all Block Header
PIR Protocol

Address PIR DB row entry - 62 bytes
- address - 25 bytes
- TXID - 32 bytes
- blockheight - 3 bytes
- vout index - 2 bytes

Merkle Tree PIR DB row
- TXID - 32 bytes
- TXID - 32 bytes
- TXID - 32 bytes
- TXID - 32 bytes

Transaction PIR DB row
- transaction bytes
PIR Protocol

Choose @
PIR Protocol

Address PIR DB row entry - 62 bytes
- address - 25 bytes
- TXID - 32 bytes
- blockheight - 3 bytes
- vout index - 2 bytes

Merkle Tree PIR DB row
- TXID - 32 bytes
- TXID - 32 bytes
- TXID - 32 bytes
- ... TXID - 32 bytes

Transaction PIR DB row
- transaction bytes

Choose @

Address PIR Manifest
Merkle Tree PIR Manifest
Transaction PIR Manifest
PIR Protocol

Address PIR DB row entry - 62 bytes
address - 25 bytes  TXID - 32 bytes  blockheight - 3 bytes  vout index - 2 bytes

Merkle Tree PIR DB row
TXID - 32 bytes  TXID - 32 bytes  TXID - 32 bytes  ⋯  TXID - 32 bytes

Transaction PIR DB row
transaction bytes

Choose @
Row indices

Choose @
Address PIR Manifest

Choose @
Merkle Tree PIR Manifest

Choose @
Transaction PIR Manifest
PIR Protocol

- Address PIR DB row entry: 62 bytes
  - address: 25 bytes
  - TXID: 32 bytes
  - blockheight: 3 bytes
  - vout index: 2 bytes

- Merkle Tree PIR DB row:
  - TXID: 32 bytes
  - TXID: 32 bytes
  - TXID: 32 bytes

- Transaction PIR DB row:
  - transaction bytes

Choose @

Row indices

Block height

Choose @

Address PIR Manifest

Merkle Tree PIR Manifest

Transaction PIR Manifest
PIR Protocol

Row indices
Block height

Choose @

Address PIR Manifest
Merkle Tree PIR Manifest
Transaction PIR Manifest

Block height
Row indices
Row indices
**PIR Protocol**

<table>
<thead>
<tr>
<th>Address PIR DB row entry - 62 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>address - 25 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Merkle Tree PIR DB row</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXID - 32 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transaction PIR DB row</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction bytes</td>
</tr>
</tbody>
</table>

Choose @

- Row indices
- TXIDs

Choose @

- Row indices
- Block Height
- TXIDs

Choose @

- Address PIR Manifest
- Merkle Tree PIR Manifest
- Transaction PIR Manifest
PIR Protocol

Choose @

Address PIR Manifest
Merkle Tree PIR Manifest
Transaction PIR Manifest

Row indices
TXIDs

Row indices
TXIDs

Row indices
Transactions

Row indices
Block height

Choose @

Address PIR Manifest
Merkle Tree PIR Manifest
Transaction PIR Manifest
PIR Protocol
PIR Protocol

We have:

- Transaction IDs
- Block Headers
- Our raw transactions
PIR Protocol

We have:
- Transaction IDs
- Block Headers
- Our raw transactions
PIR Protocol

We have:

- Transaction IDs
- Block Headers
- Our raw transactions
PIR Protocol

We have:

- Transaction IDs
- Block Headers
- Our raw transactions
Costs - PIR Database sizes

Bitcoin Data — Block 1~ 513502 (March 2018)

<table>
<thead>
<tr>
<th></th>
<th>All-Time blocks 1 to 508462</th>
<th>Monthly blocks 508463 to 512494</th>
<th>Weekly blocks 512495 to 513502</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Address PIR DBs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total size of DB</td>
<td>3.16 GB</td>
<td>176.04 MB</td>
<td>59.11 MB</td>
</tr>
<tr>
<td>Size of manifest file</td>
<td>65.99 MB</td>
<td>175.33 MB</td>
<td>66.97 MB</td>
</tr>
<tr>
<td><strong>Merkle Tree PIR DBs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total size of DB</td>
<td>10.18 GB</td>
<td>1.46 GB</td>
<td>1.44 GB</td>
</tr>
<tr>
<td>Size of manifest file</td>
<td>40.90 MB</td>
<td>0.32 MB</td>
<td>78.62 KB</td>
</tr>
<tr>
<td><strong>Transaction PIR DBs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total size of DB</td>
<td>15.87 GB</td>
<td>1.30 GB</td>
<td>448.91 MB</td>
</tr>
<tr>
<td>Size of manifest file</td>
<td>3.03 GB</td>
<td>218.68 MB</td>
<td>72.45 MB</td>
</tr>
</tbody>
</table>

Platform — Ubuntu 16.04 (64-bit) OS, Intel Core i7 3.4 GHz CPU, 16 GB of RAM and a 512 GB hard drive
Manifest File Trie Scheme

- In some cases, manifest files are very big
- Transform the simple manifest files into a format suitable for PIR queries.
- Clients perform interpolation search on these manifest files, under PIR.
**Bandwidth comparison to verify a single transaction**

<table>
<thead>
<tr>
<th></th>
<th>BIP-37</th>
<th>PIR: 1 server</th>
<th>PIR: 3 servers</th>
<th>Naive SPV</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All-Time</strong></td>
<td>102.80 KB</td>
<td>21.54 MB</td>
<td>64.61 MB</td>
<td>80.27 GB</td>
</tr>
<tr>
<td><strong>Monthly</strong></td>
<td>132.43 KB</td>
<td>1.70 MB</td>
<td>5.11 MB</td>
<td>159.19 GB</td>
</tr>
<tr>
<td><strong>Weekly</strong></td>
<td>128.52 KB</td>
<td>666.07 KB</td>
<td>2.00 MB</td>
<td>161.47 GB</td>
</tr>
</tbody>
</table>

Platform — Ubuntu 16.04 (64-bit) OS, Intel Core i7 3.4 GHz CPU, 16 GB of RAM and a 512 GB hard drive
Bandwidth costs to verify a single transaction
Cumulative distribution of bandwidth costs to verify a single transaction

(a) All-Time

(b) Monthly

(b) Weekly
Further Improvements

- Interpolative Search to skip manifest files
- More fine granular separation of databases
Thank you!
Manifest Files

• Format

```json
{ ...
    "address" / "block-height" / "txid" : [
        "row index start",
        "row index end",
        "column index start",
        "column index end",
    ]
} ...
```

• Manifest File Trie Scheme

- In some cases, manifest files have a nearly equivalent or even larger size compared with the database. This may incur a poor performance in bandwidth.
- Transform the simple manifest files into a format suitable for PIR queries.
- Clients perform interpolation search on these manifest files, under PIR
Protocol

- **Data:** SPV client C, one or multiple PIR servers S
- **Result:** C obtains the necessary data for a SPV privately
- **Initialization:** S constructs the PIR databases and associated manifest files; C downloads the manifest files from S;

### a. Query the Address PIR DB
1. C selects an address to fetch a record from the Address PIR DB manifest file and generates the PIR queries based on row indices of the selected record;
2. S computes the result using the PIR queries on the Address PIR DB;
3. C parses and decodes the result to obtain one or more Address PIR DB entries;

### b. Query the Merkle Tree PIR DB
1. C uses the value of the block height field of an entry to fetch the corresponding record from the Merkle Tree PIR DB manifest file and generates the PIR queries based on the row indices;
2. S computes the result using the PIR queries on the Merkle Tree PIR DB;
3. C parses and decodes the result to obtain the requested list of TXIDs;

### c. Query Transaction PIR DB
1. C uses the value of the TXID field from the same entry that was selected in step b.1, to fetch the corresponding record from the Transaction PIR DB manifest file and generates the PIR queries based on the row indices;
2. S computes the result using the PIR queries on the Transaction PIR DB;
3. C parses and decodes the result to obtain the requested transaction.
Evaluation

- Cumulative bandwidth cost required to verify 1~100 transactions
- Latency of PIR-SPV (1-server and 3-servers)

- Weekly
  - Single server PIR-SPV has a similar bandwidth cost to BIP-37 SPV when verifying between 20 and 100 transactions.
  - In particular, if a client is interested in performing single-sever PIR SPV on 100 transactions which occurred in the past week, it will take approximately 2 minutes for this query to be executed, with a bandwidth cost of 11.18 MB. In contrast, BIP-37 SPV will require 12.85 MB in the same scenario.
  - Clients who are interested in a smaller number of transactions, such as 20, would incur a bandwidth cost of 8.31 MB in the multi-server setting, with a latency of approximately 62 seconds.

- Monthly
  - In the multi-server setting, a 20-transaction query would incur a bandwidth cost of 12.20 MB with a latency of approximately 2.3 minutes.

- All-time
  - Cost is relatively high, but all-time queries are infrequent.