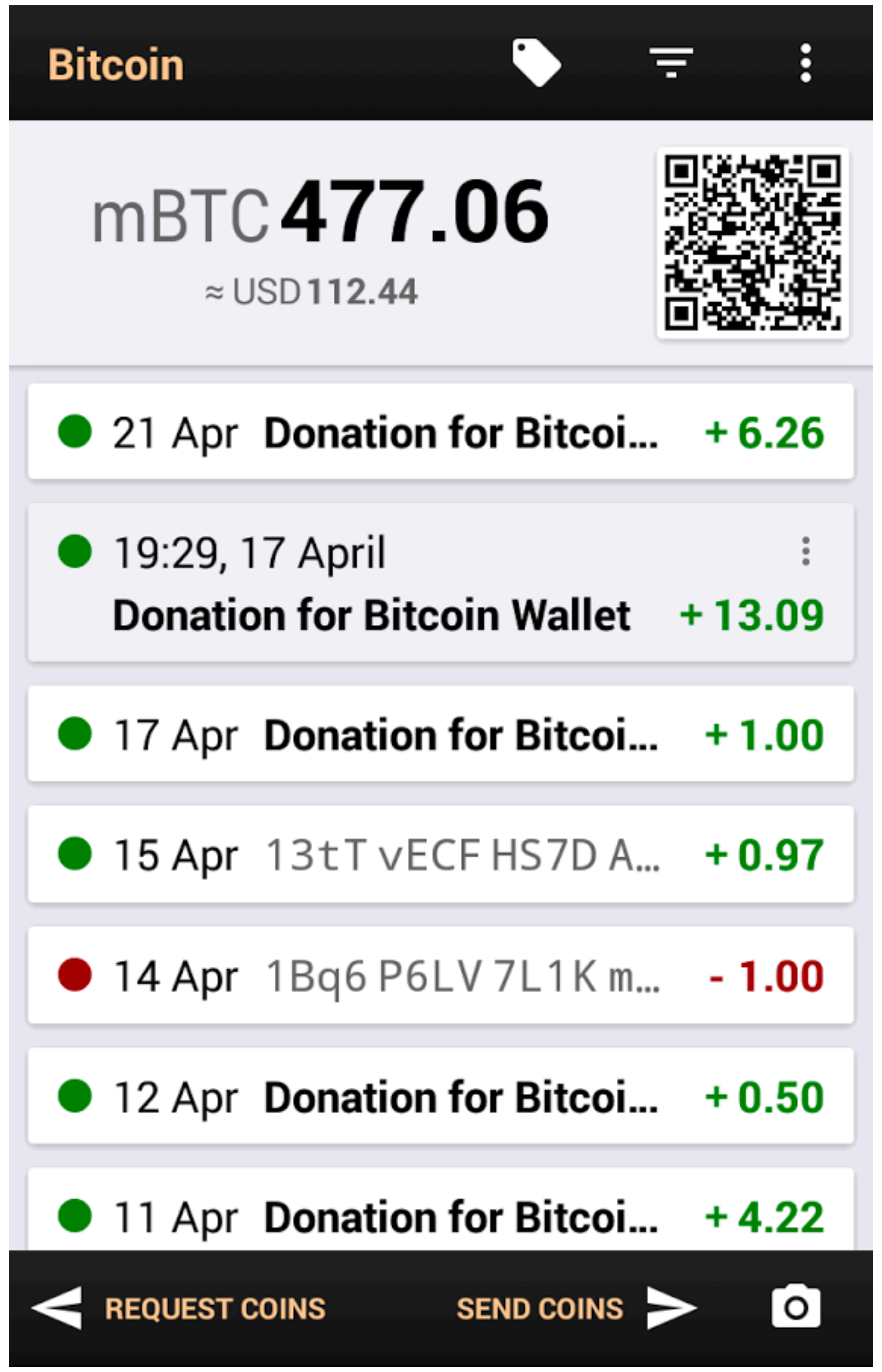
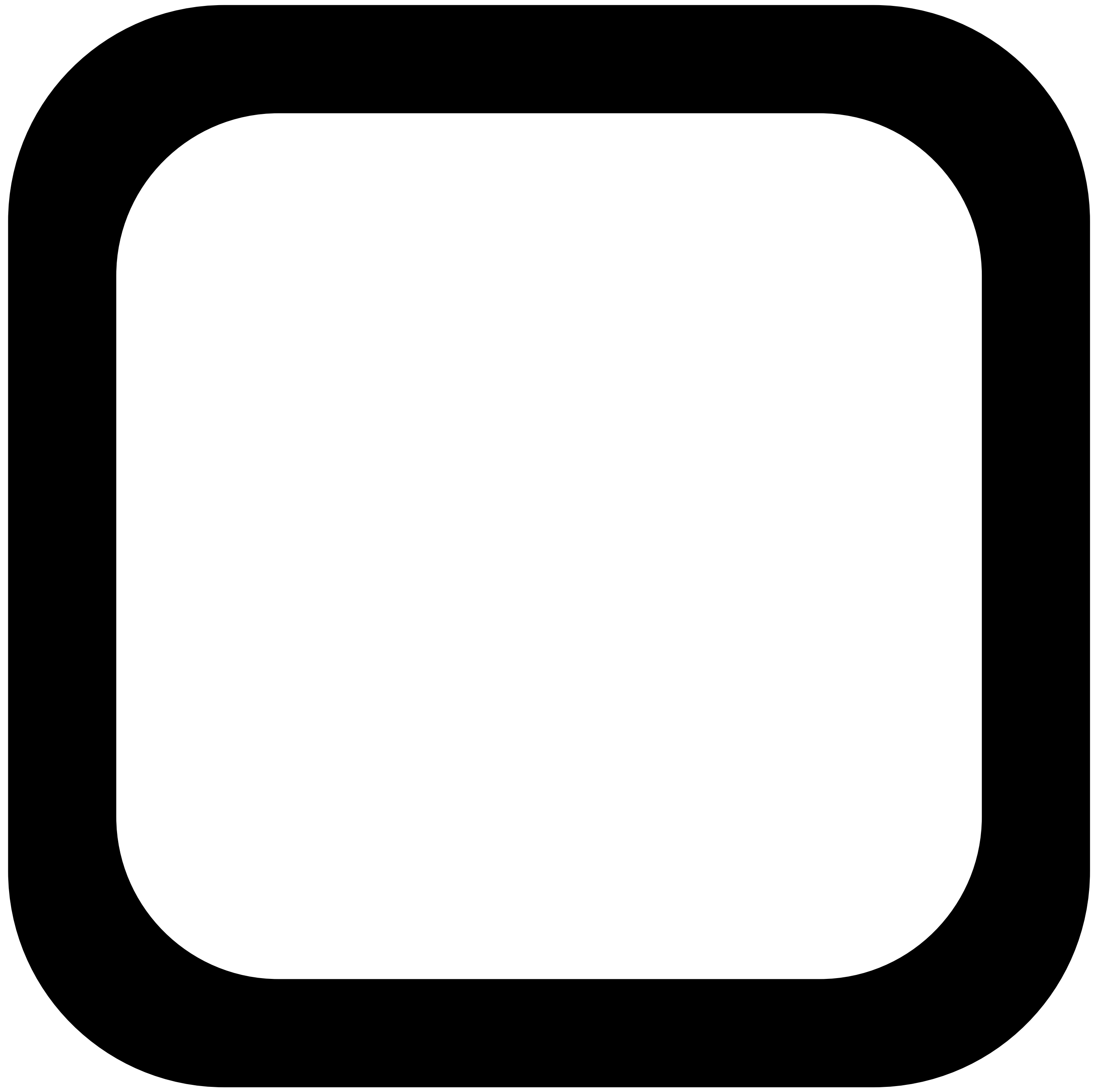
Applying Private Information Retrieval to Lightweight Bitcoin Clients

Kaihua Qin, Henryk Hadass, Arthur Gervais and Joel Reardon

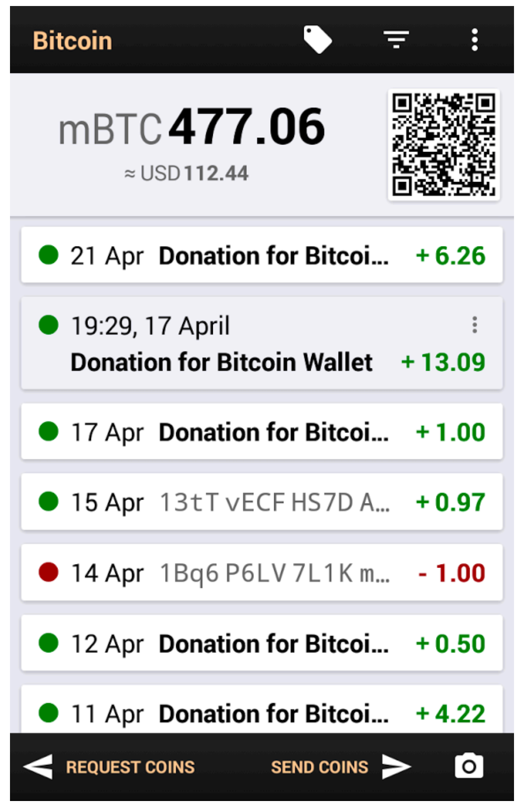
Bitcoin SPV Clients



Simple Payment Verification (SPV) Clients

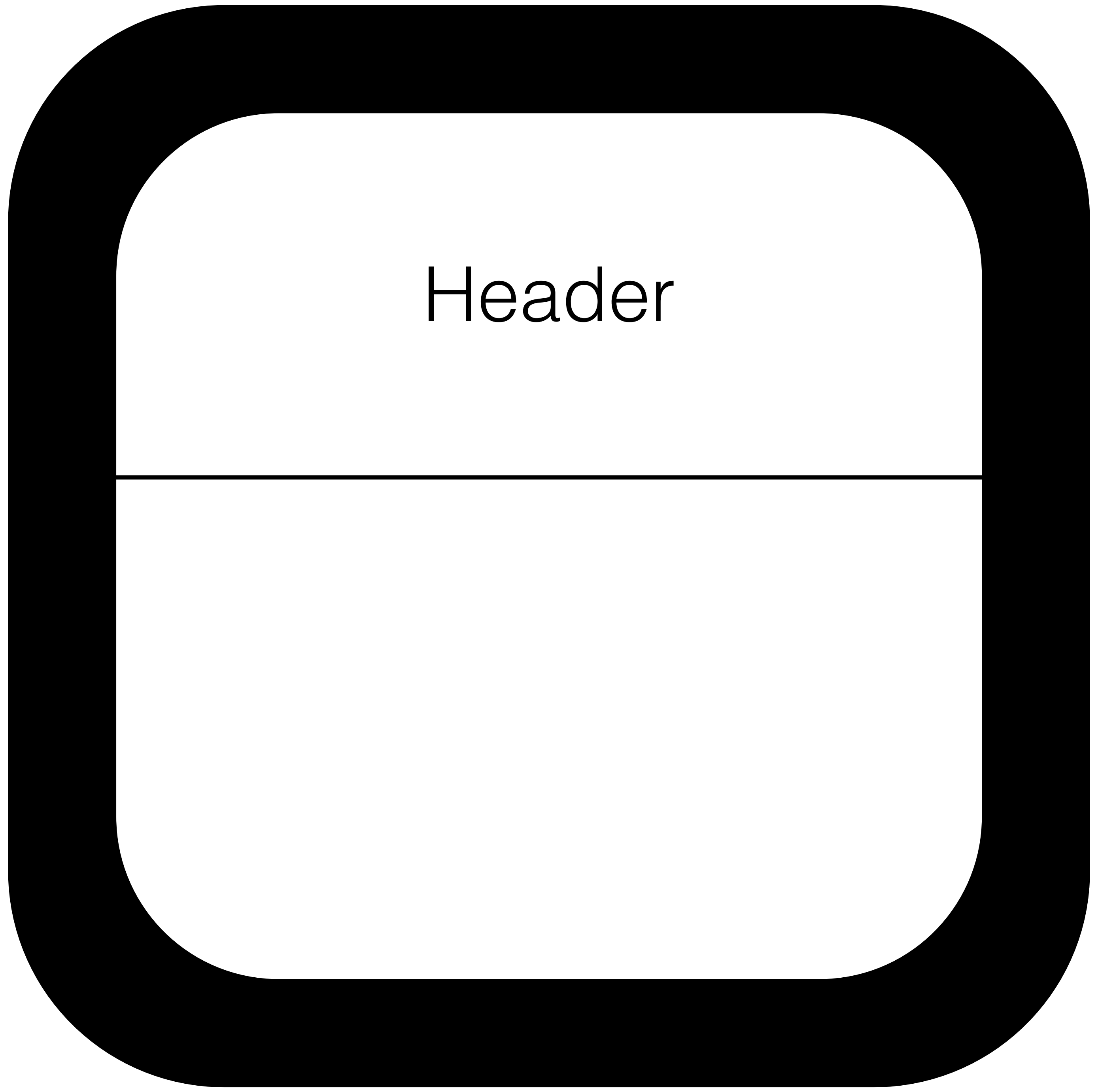


Blockchain Block

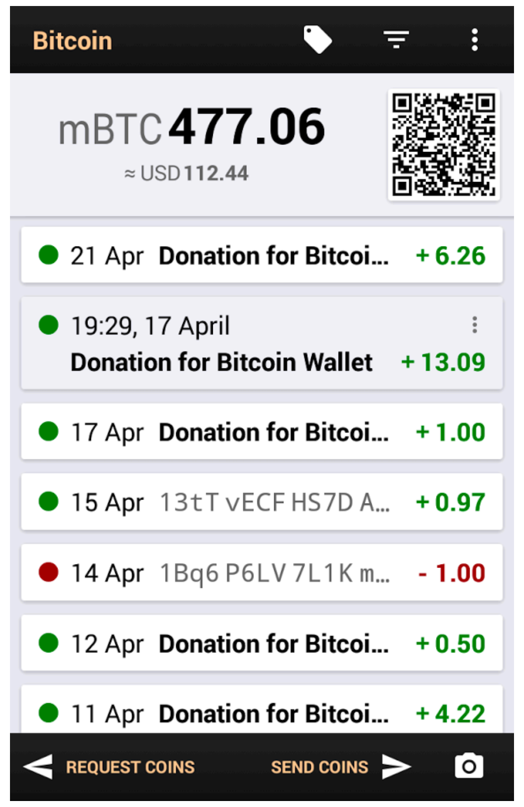


Imperial College London

Simple Payment Verification (SPV) Clients

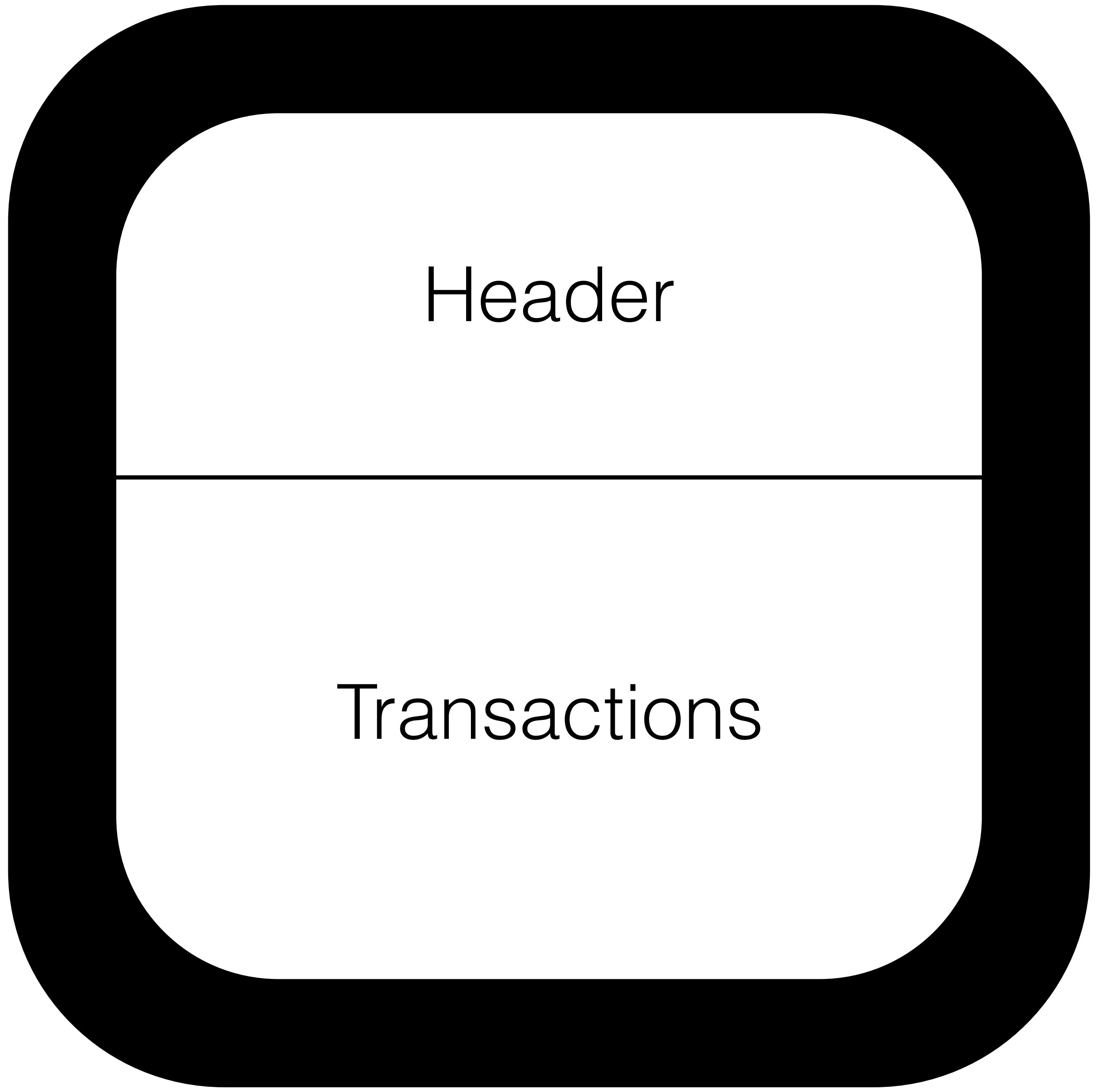


Blockchain Block

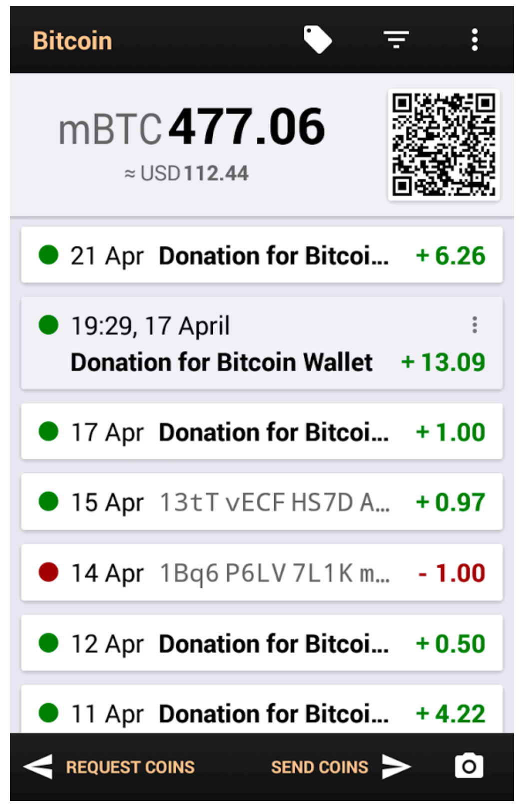


Imperial College London

Simple Payment Verification (SPV) Clients

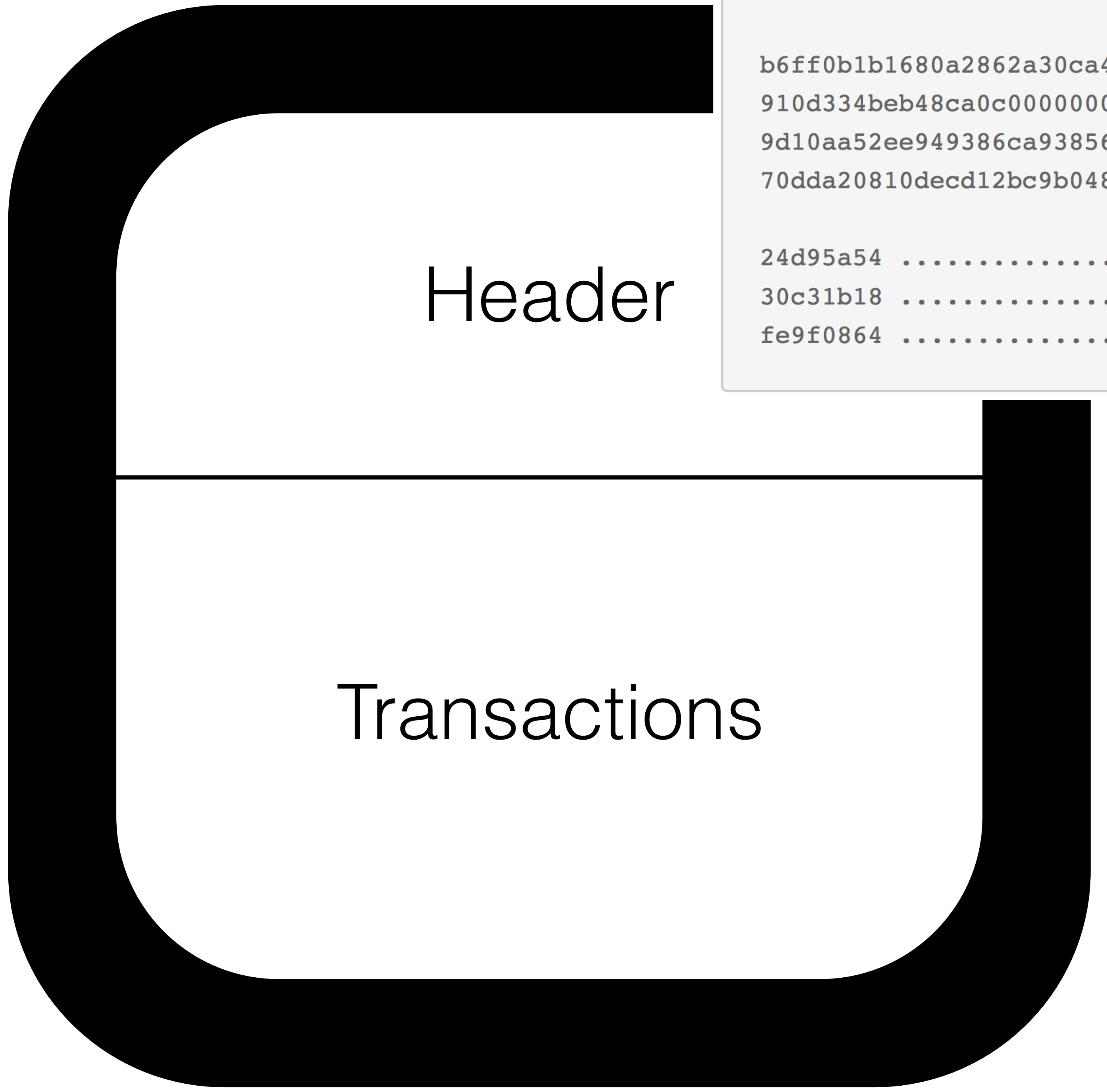


Blockchain Block

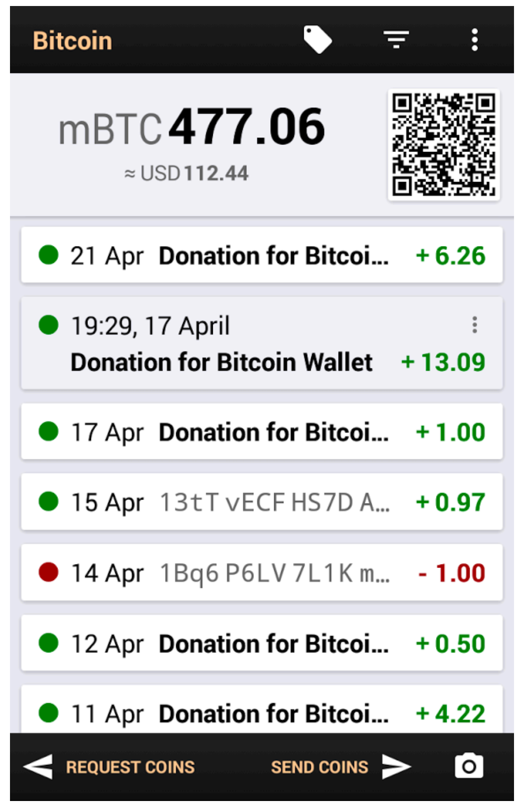


Simple Payment Verification (SPV) Clients

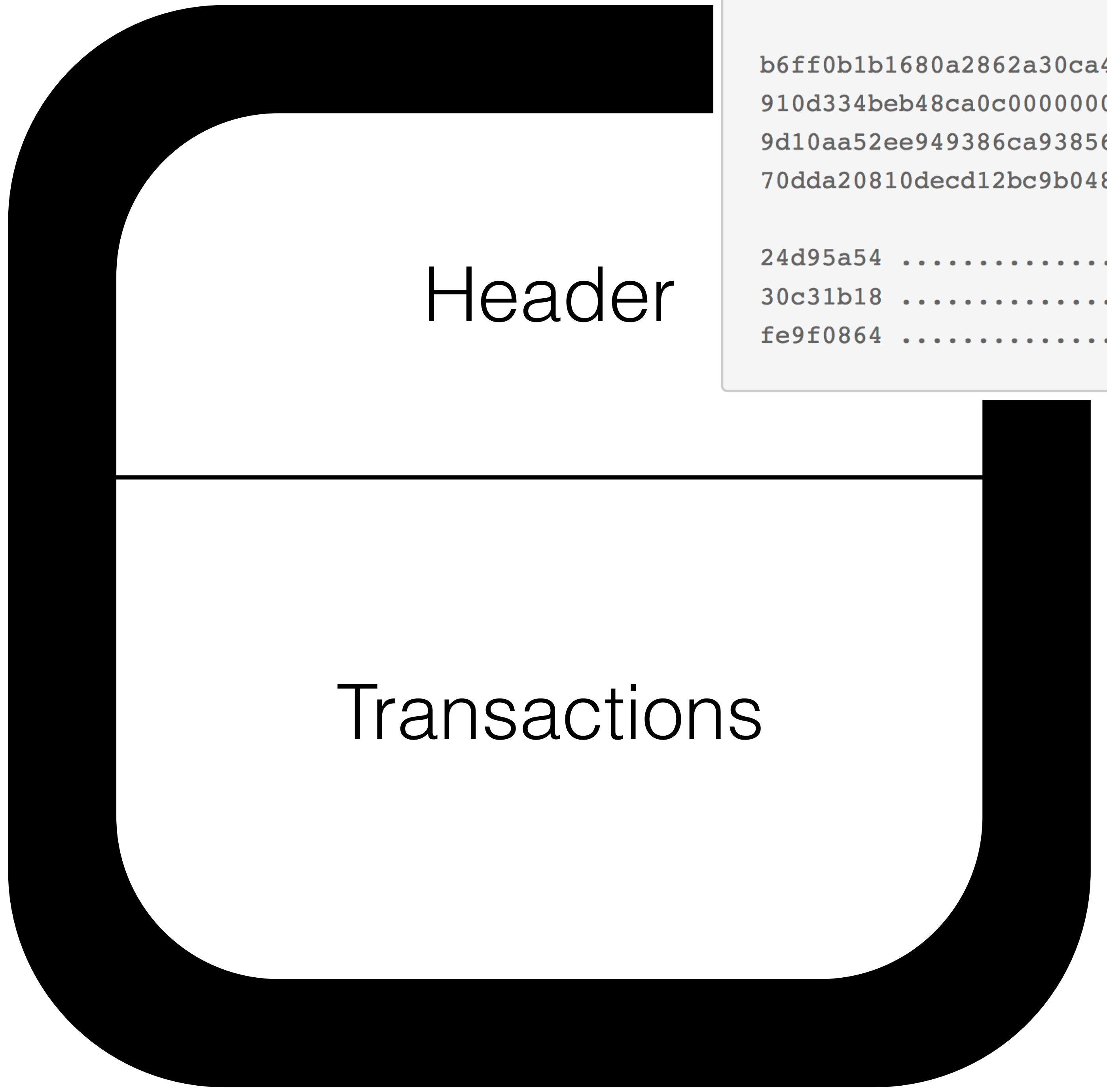
```
02000000 ........................... Block version: 2

b6ff0b1b1680a2862a30ca44d346d9e8
910d334beb48ca0c0000000000000000 ... Hash of previous block's header
9d10aa52ee949386ca9385695f04ede2
70dda20810decd12bc9b048aaab31471 ... Merkle root

24d95a54 ........................... Unix time: 1415239972
30c31b18 ........................... Target: 0x1bc330 * 256**(0x18-3)
fe9f0864 ........................... Nonce
```



Blockchain Block

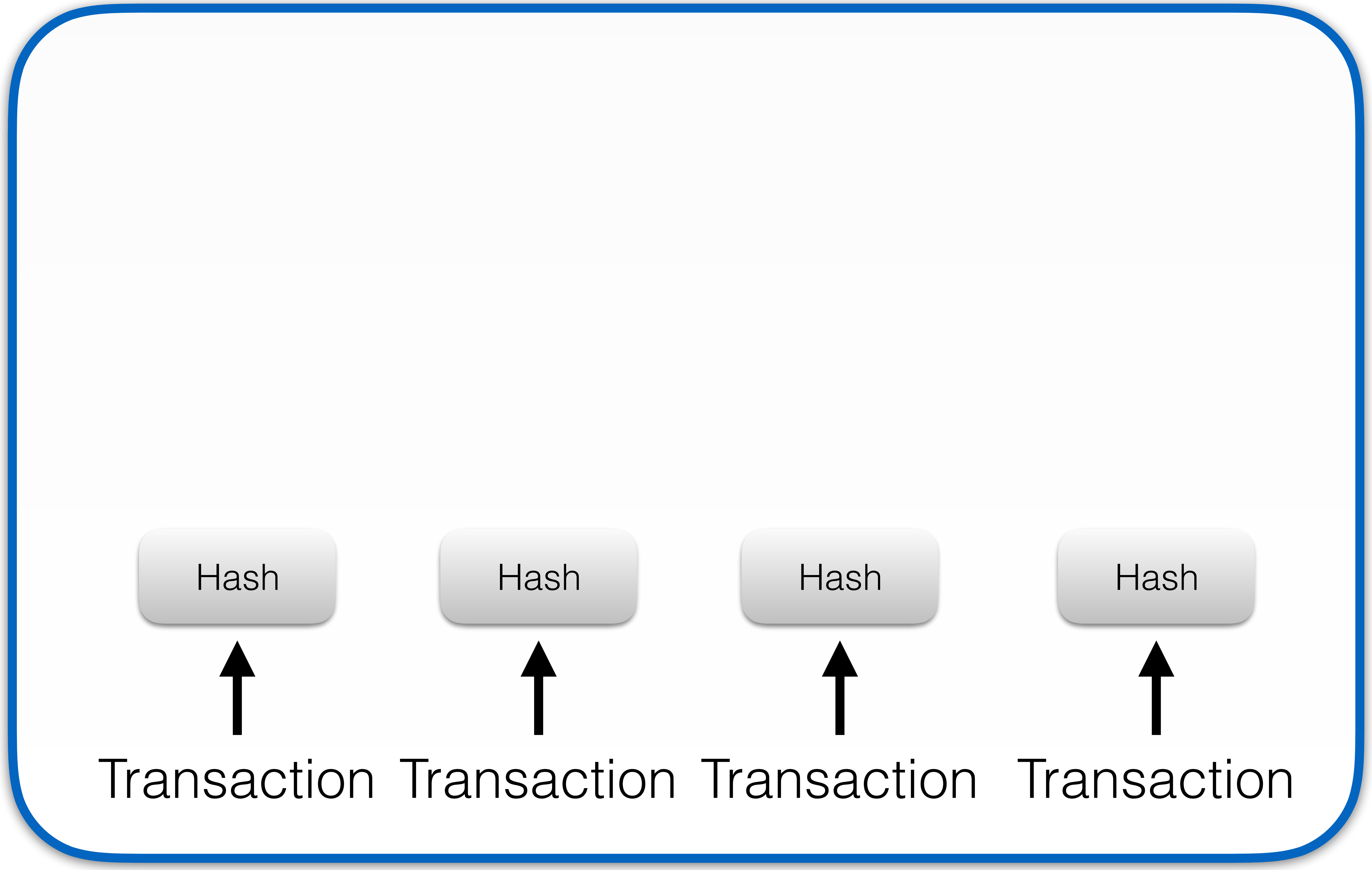


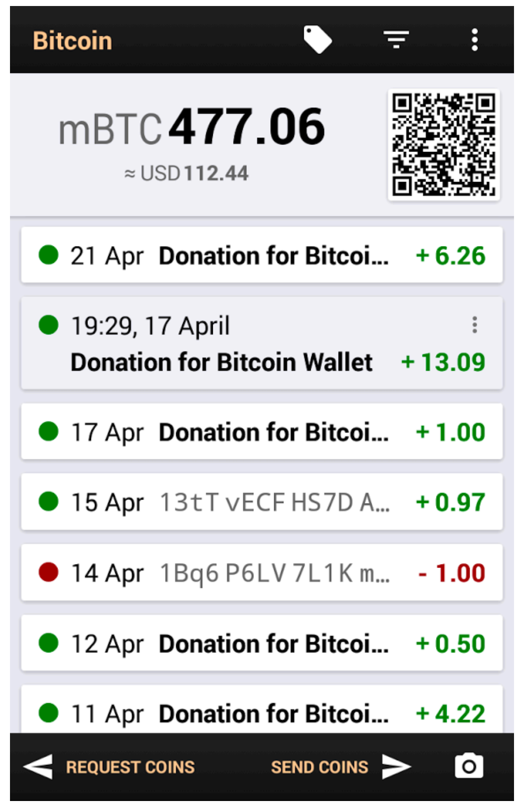
Simple Payment Verification (SPV) Clients



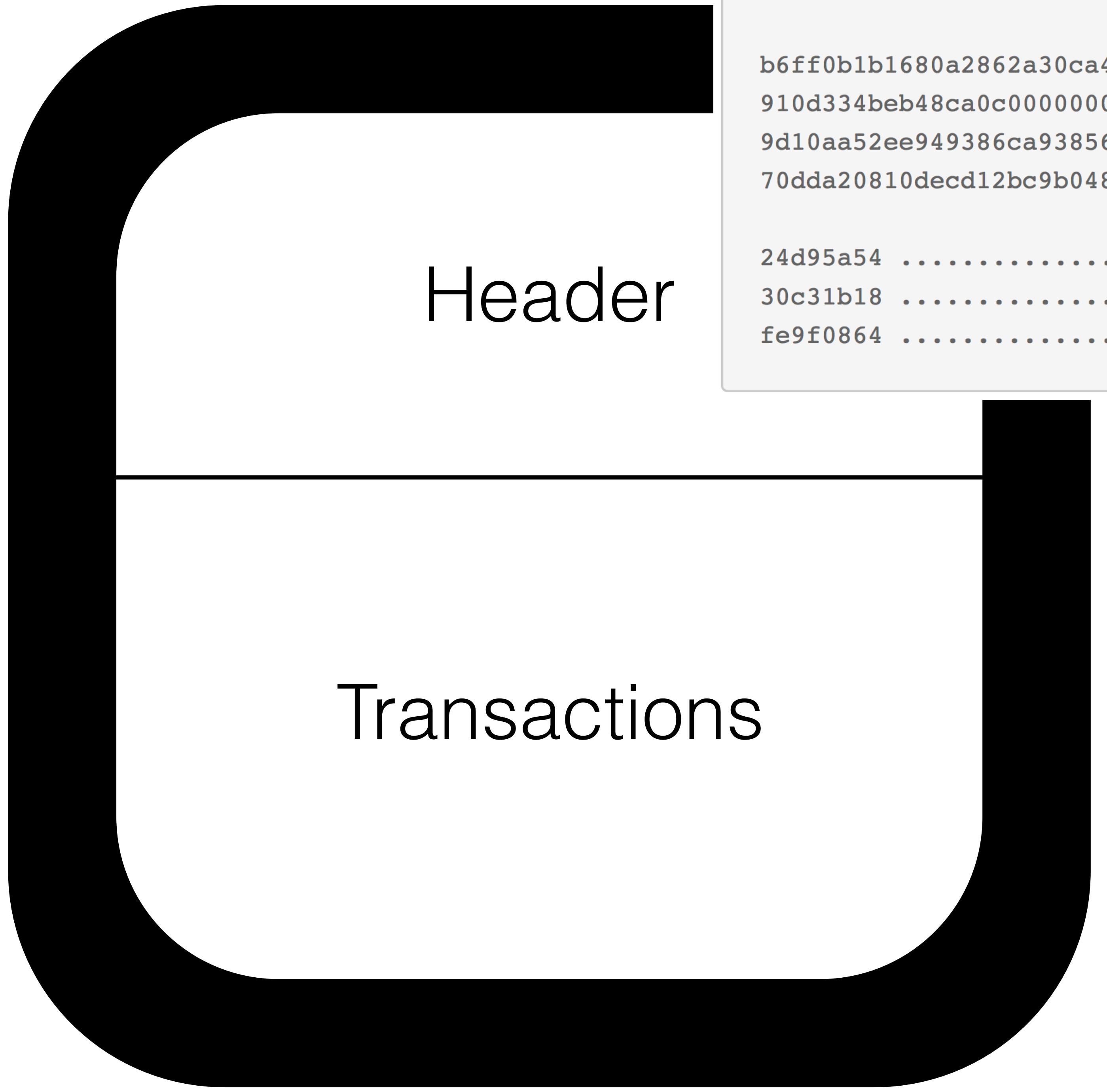
Blockchain Block

```
02000000 ........................... Block version: 2

b6ff0b1b1680a2862a30ca44d346d9e8
910d334beb48ca0c0000000000000000 ... Hash of previous block's header
9d10aa52ee949386ca9385695f04ede2
70dda20810decd12bc9b048aaab31471 ... Merkle root

24d95a54 ........................... Unix time: 1415239972
30c31b18 ........................... Target: 0x1bc330 * 256**(0x18-3)
fe9f0864 ........................... Nonce
```



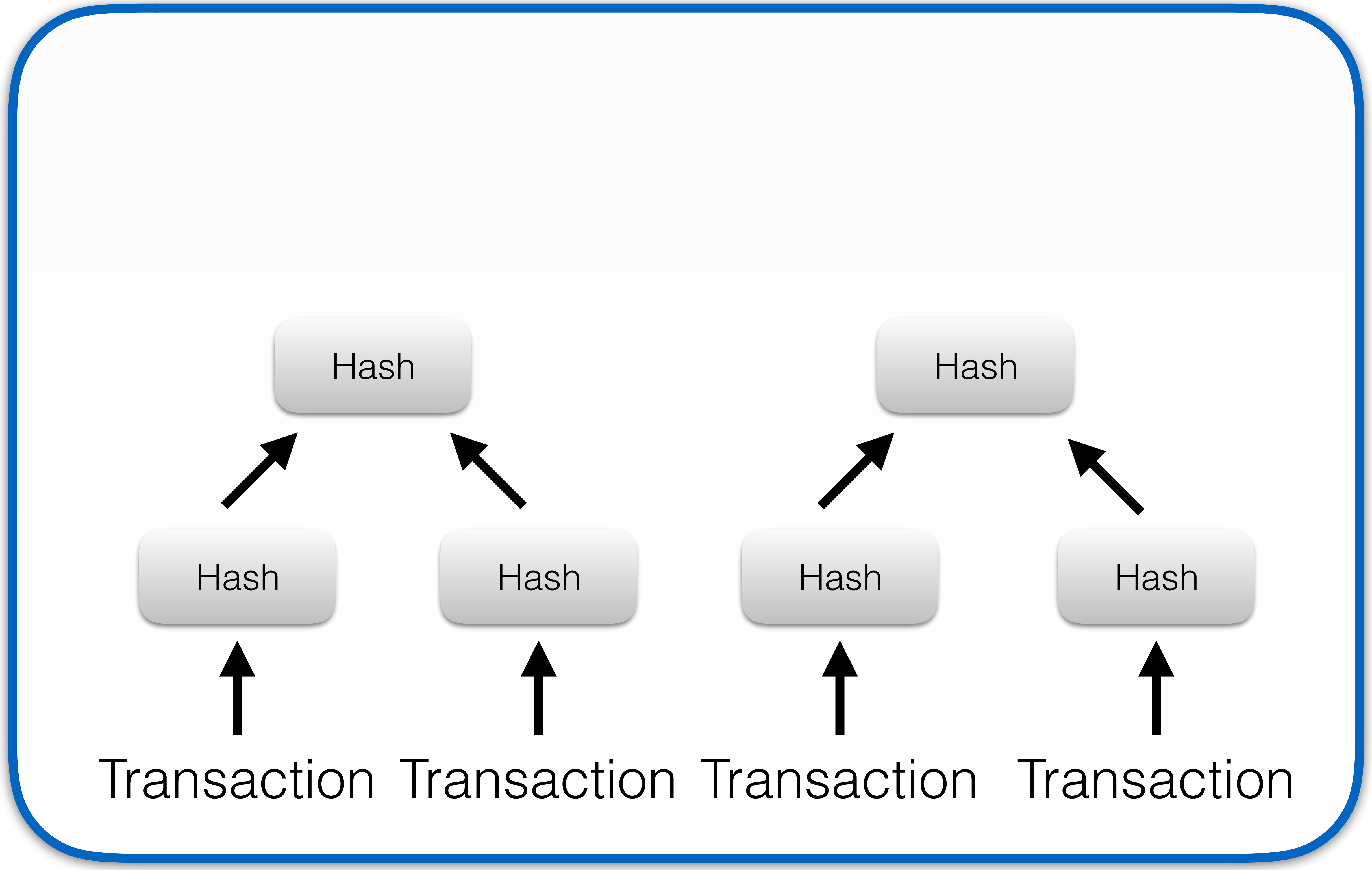


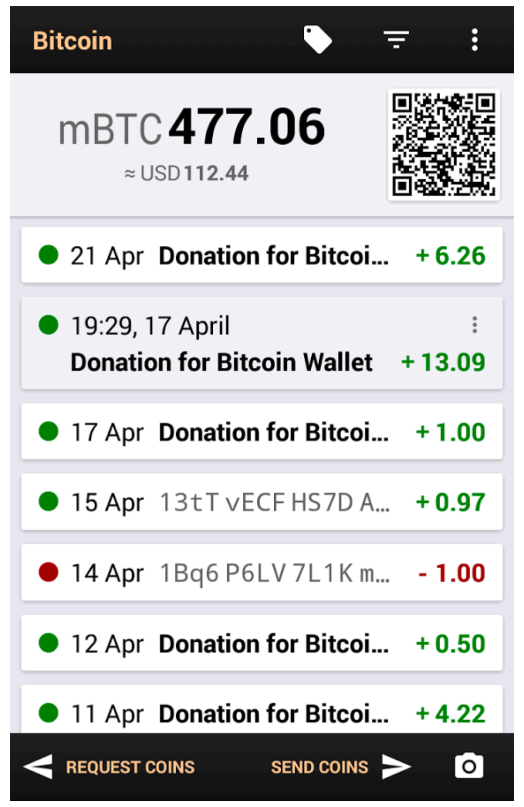
Simple Payment Verification (SPV) Clients



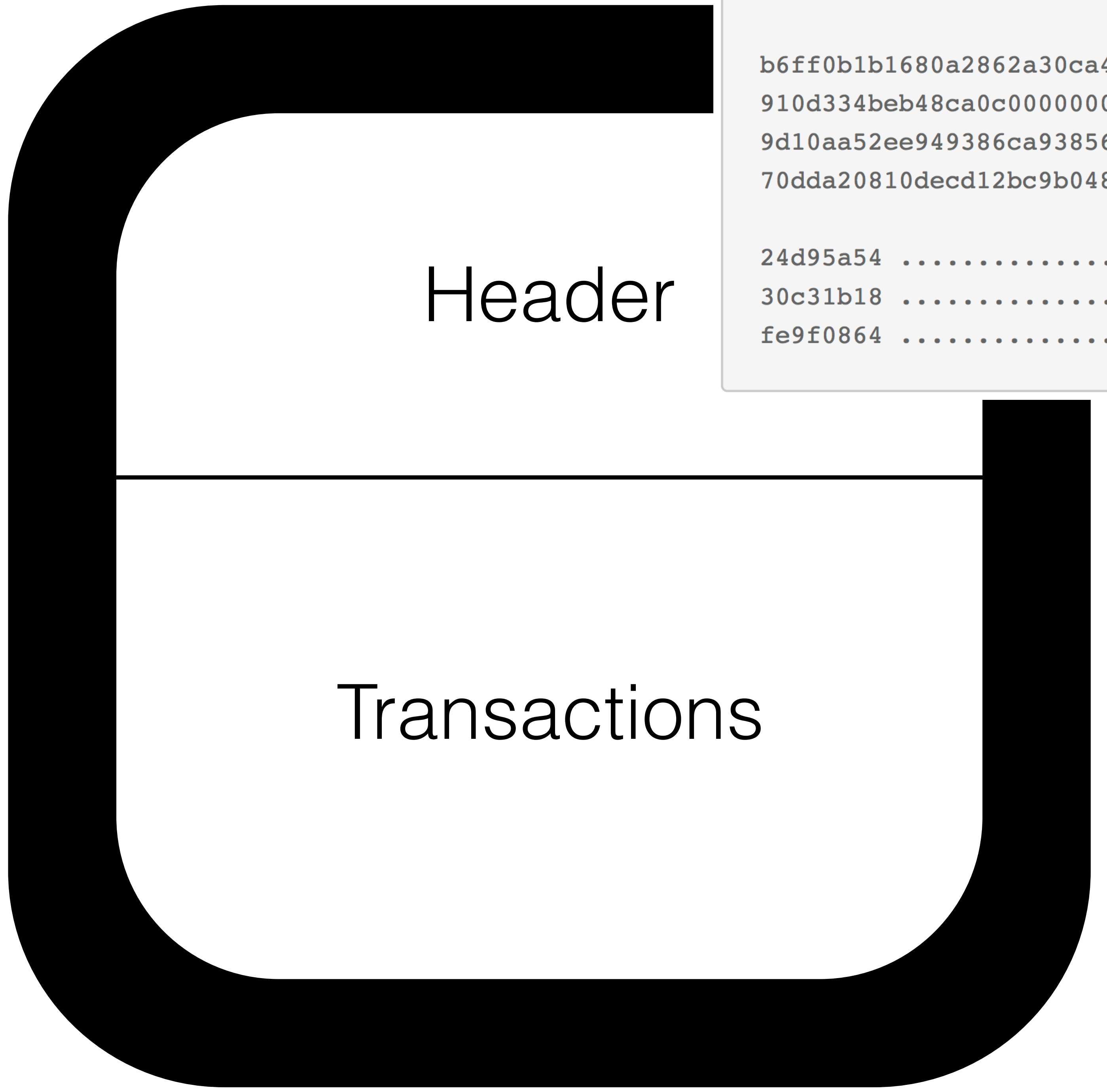
Blockchain Block

```
02000000 ........................... Block version: 2

b6ff0b1b1680a2862a30ca44d346d9e8
910d334beb48ca0c0000000000000000 ... Hash of previous block's header
9d10aa52ee949386ca9385695f04ede2
70dda20810decd12bc9b048aaab31471 ... Merkle root

24d95a54 ........................... Unix time: 1415239972
30c31b18 ........................... Target: 0x1bc330 * 256**(0x18-3)
fe9f0864 ........................... Nonce
```



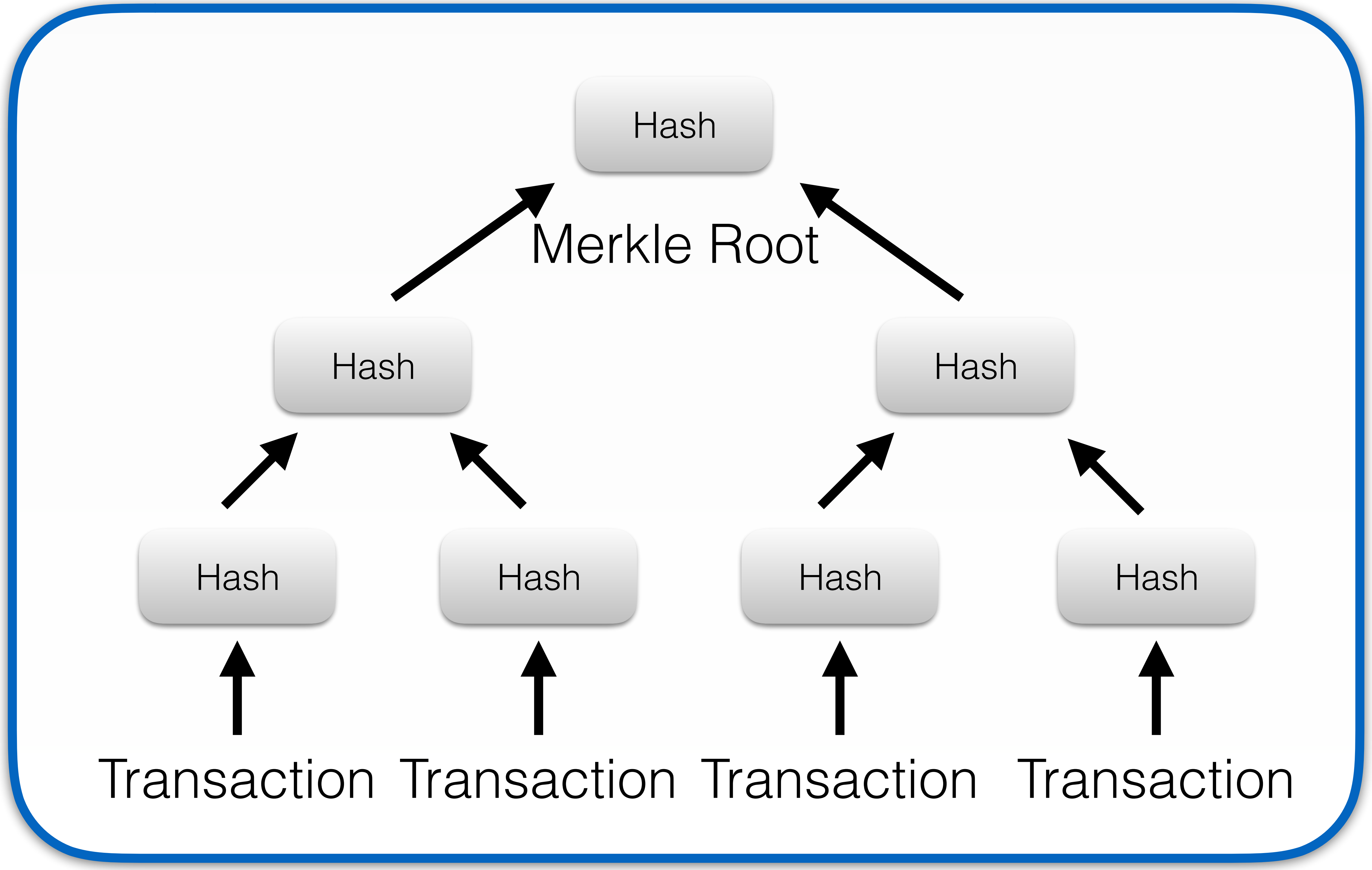


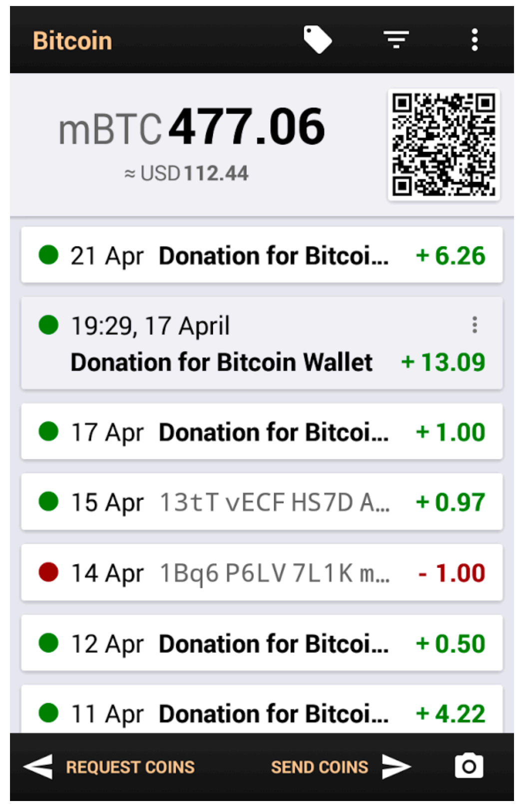
Simple Payment Verification (SPV) Clients



Blockchain Block

```
02000000 ........................... Block version: 2

b6ff0b1b1680a2862a30ca44d346d9e8
910d334beb48ca0c0000000000000000 ... Hash of previous block's header
9d10aa52ee949386ca9385695f04ede2
70dda20810decd12bc9b048aaab31471 ... Merkle root

24d95a54 ........................... Unix time: 1415239972
30c31b18 ........................... Target: 0x1bc330 * 256**(0x18-3)
fe9f0864 ........................... Nonce
```



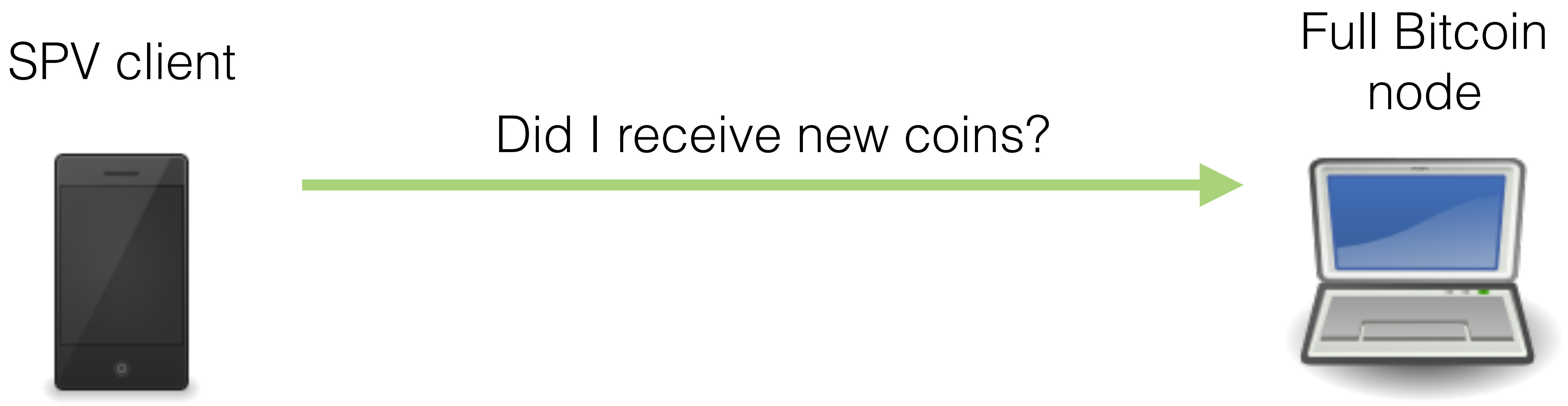


Simple Payment Verification (SPV) Clients

SPV client

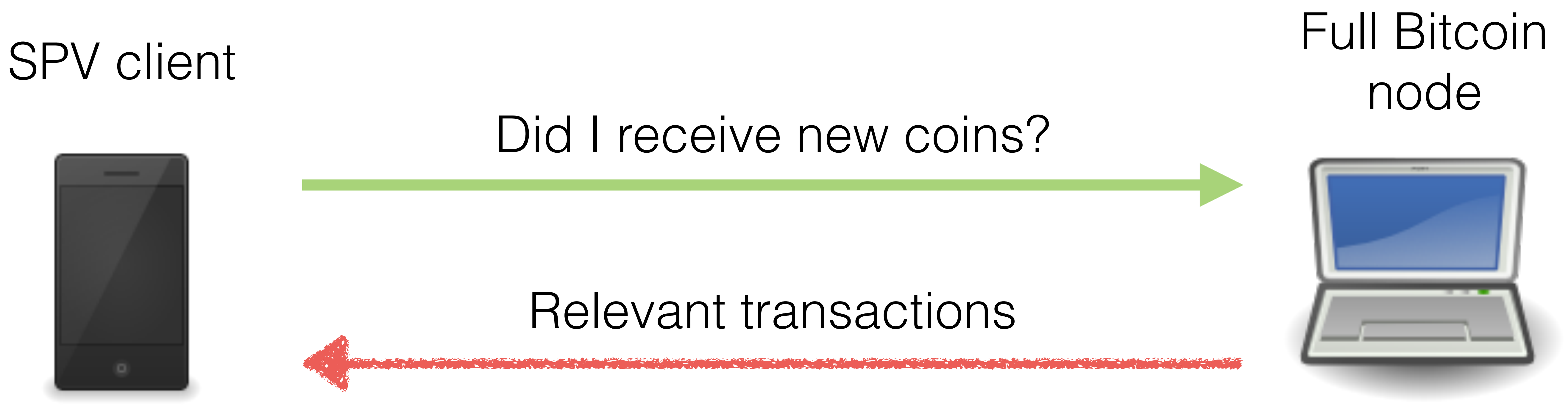
Full Bitcoin node

Simple Payment Verification (SPV) Clients



Simple Payment Verification (SPV) Clients

Simple Payment Verification (SPV) Clients



Privacy!

Bloom Filter

Bloom filter	0	0	0	0	0	0	0

Bloom Filter

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

Bloom filter

0	0	0	0	0	0	0

Bloom Filter

Insertion

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

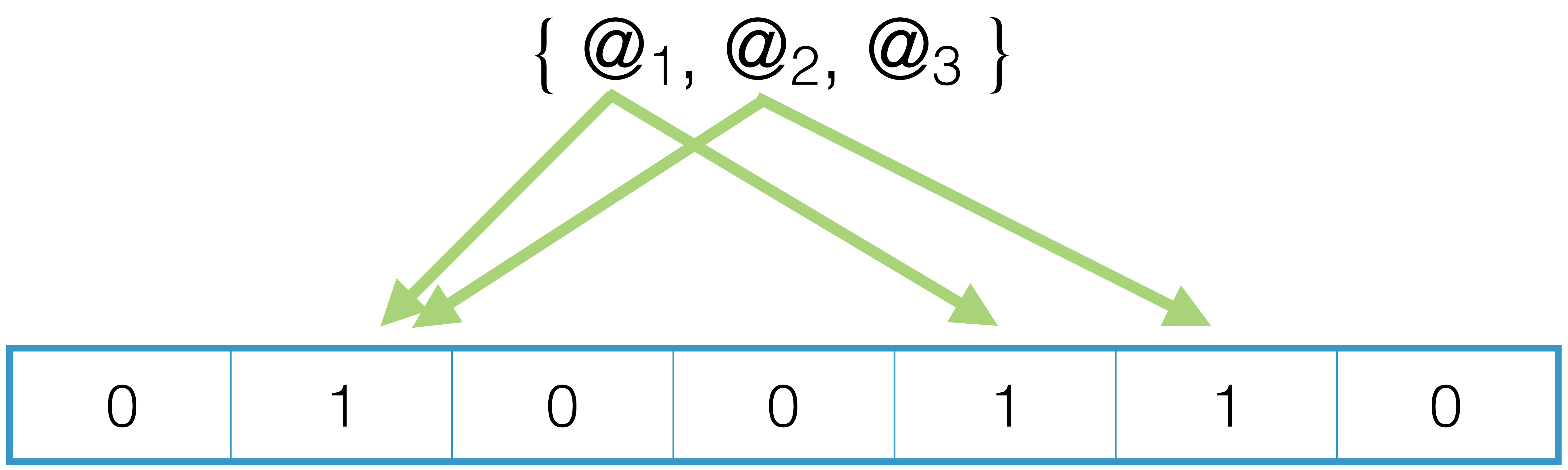
Bloom filter

0	1	0	0	1	0	0

Bloom Filter

Insertion

Bloom filter



Bloom Filter

Insertion

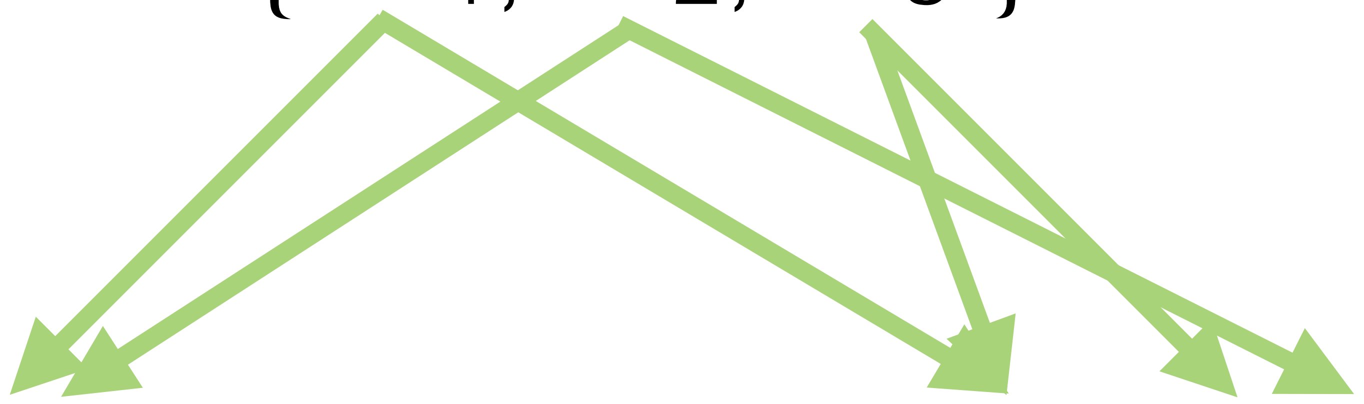
$\{ @_1, @_2, @_3 \}$

Bloom filter

0	1	0	0	1	1	0

Bloom Filter

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

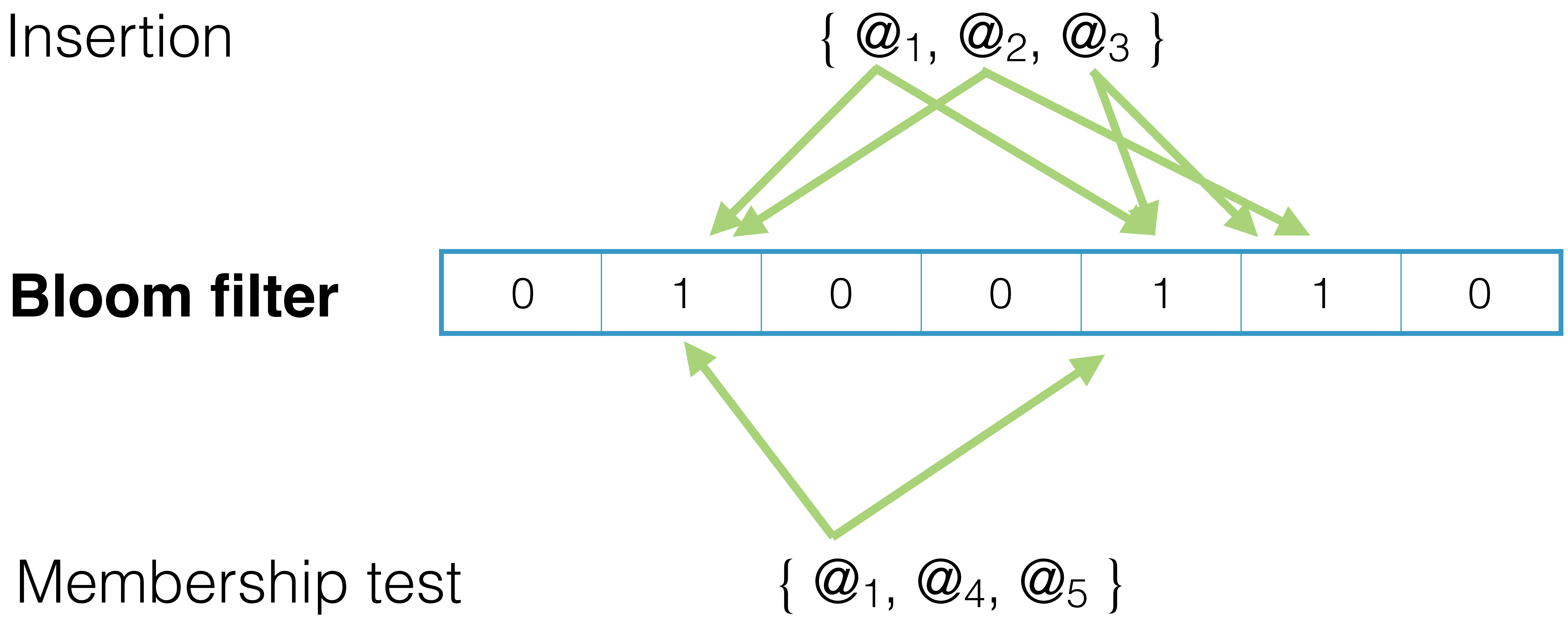


Bloom filter

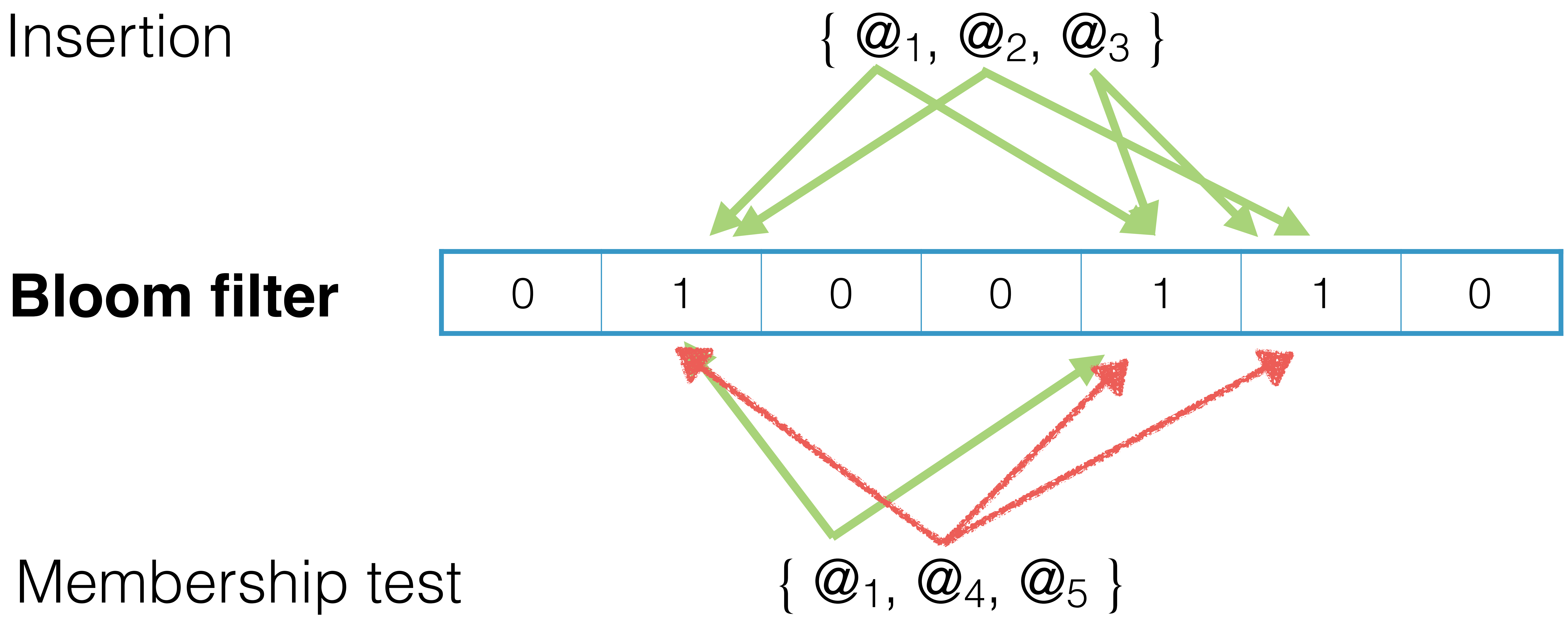
0	1	0	0	1	1	0

Membership test $\{ @_1, @_4, @_5 \}$

Bloom Filter



Bloom Filter

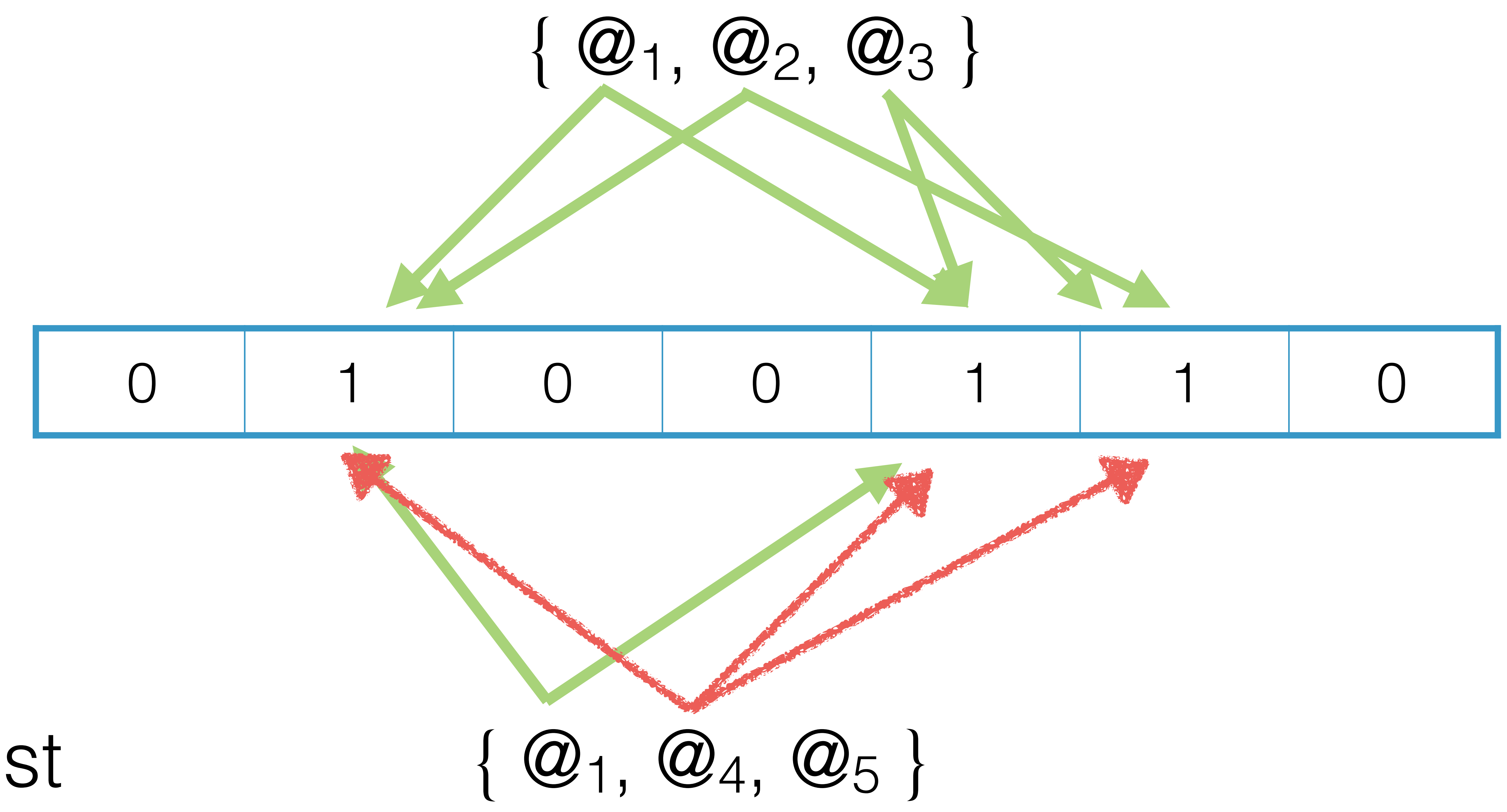


Bloom Filter

Insertion

Bloom filter

Membership test



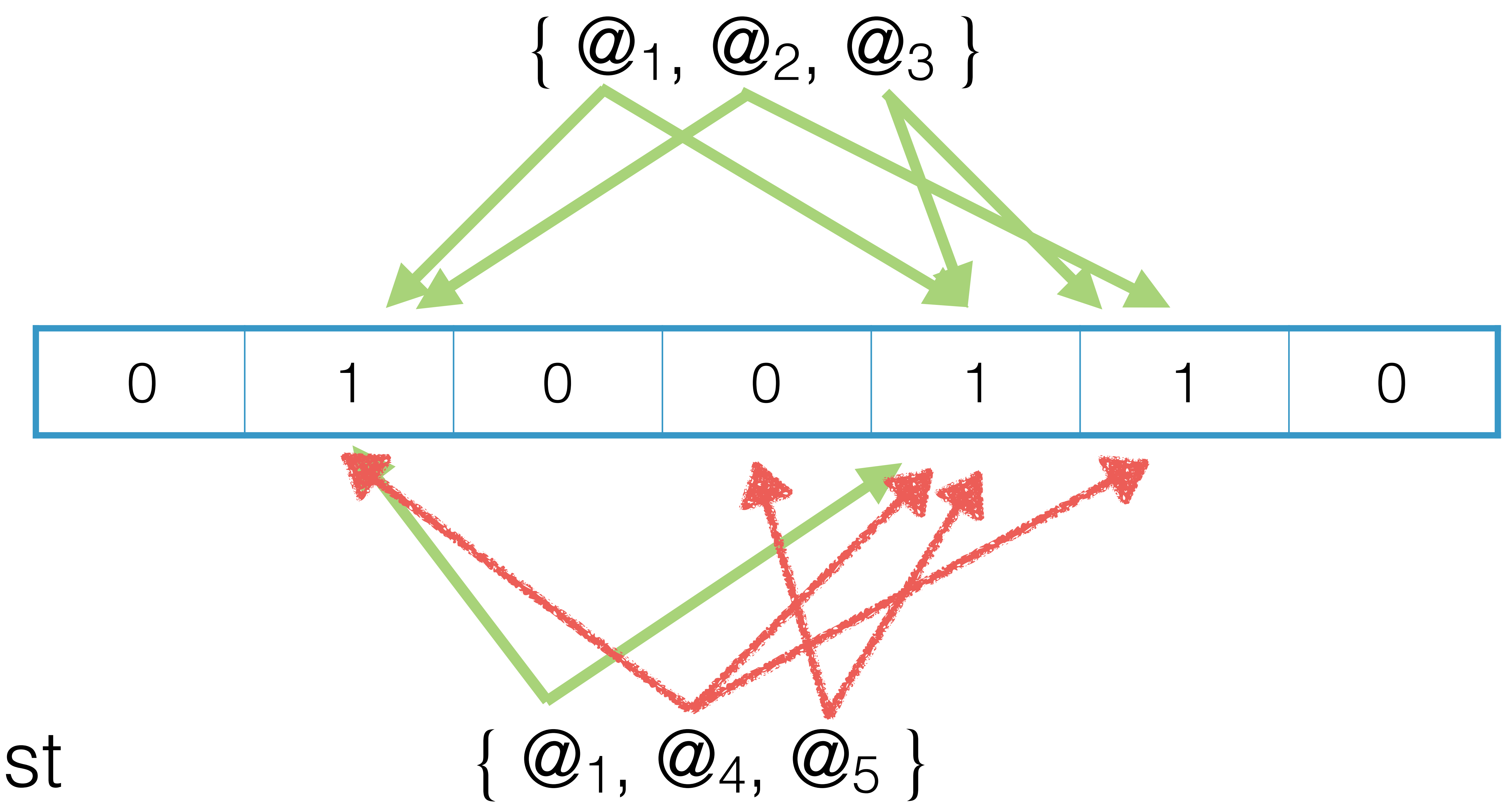
@4 False positive → **target False Positive Rate** (FPR)

Bloom Filter

Insertion

Bloom filter

Membership test



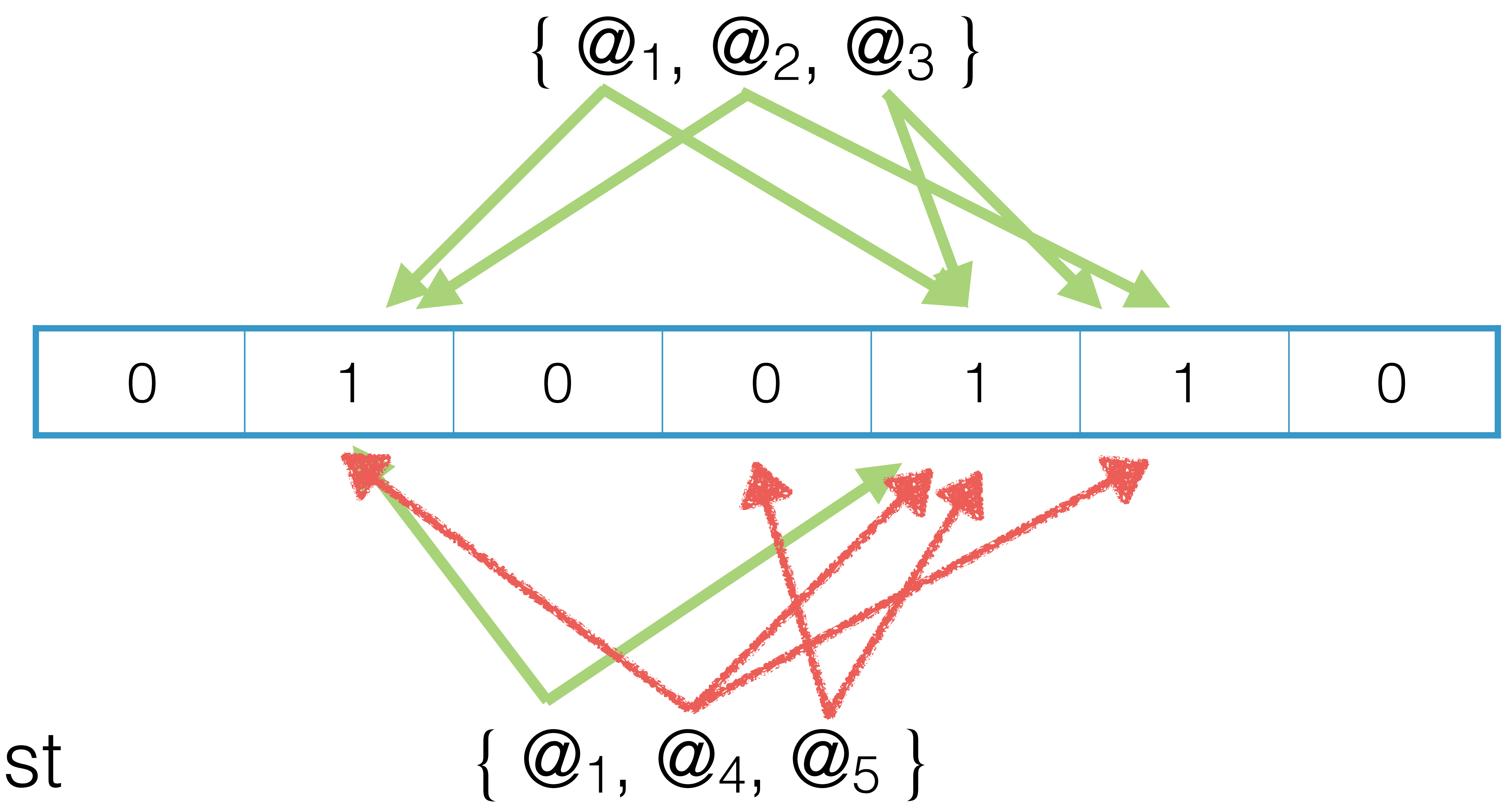
! $@_4$ False positive → **target False Positive Rate** (FPR)

Bloom Filter

Insertion

Bloom filter

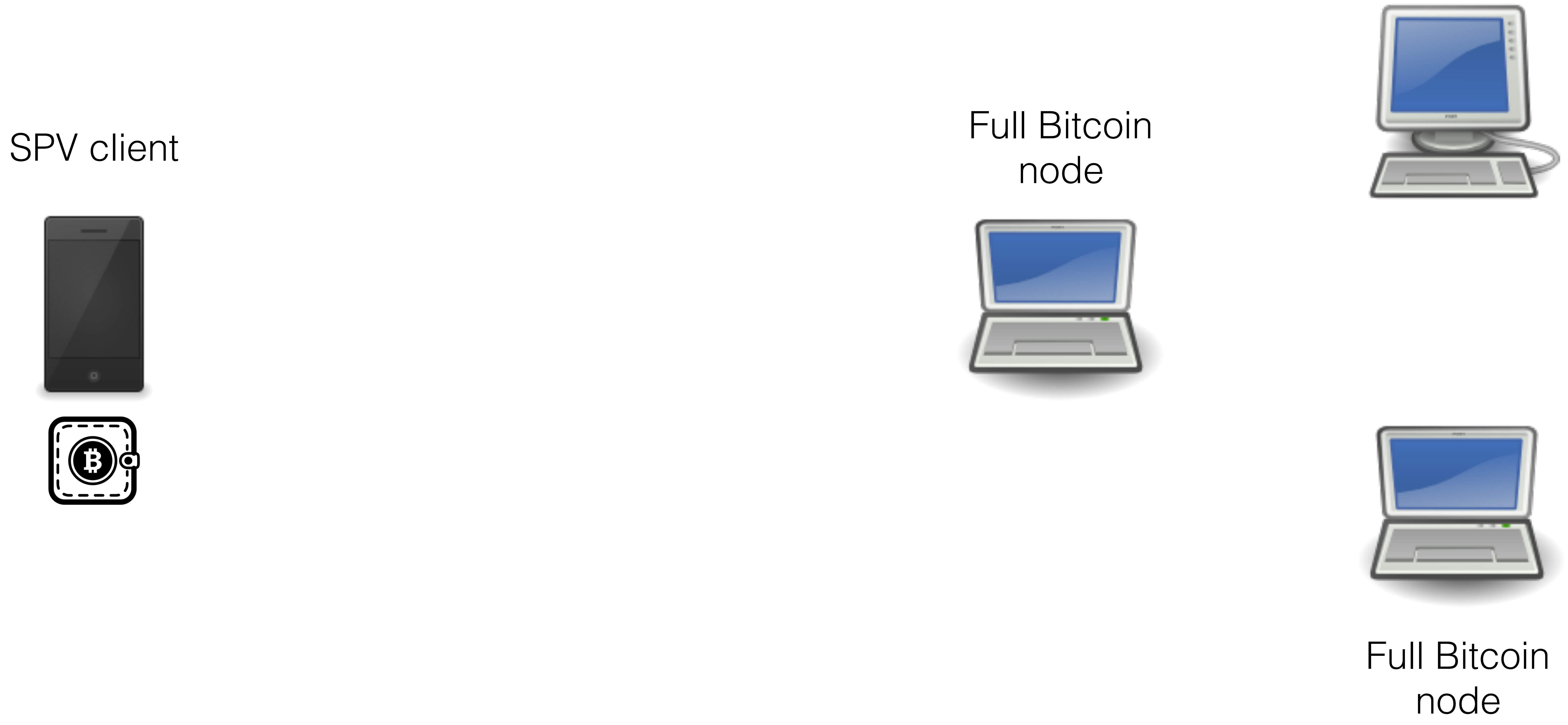
Membership test



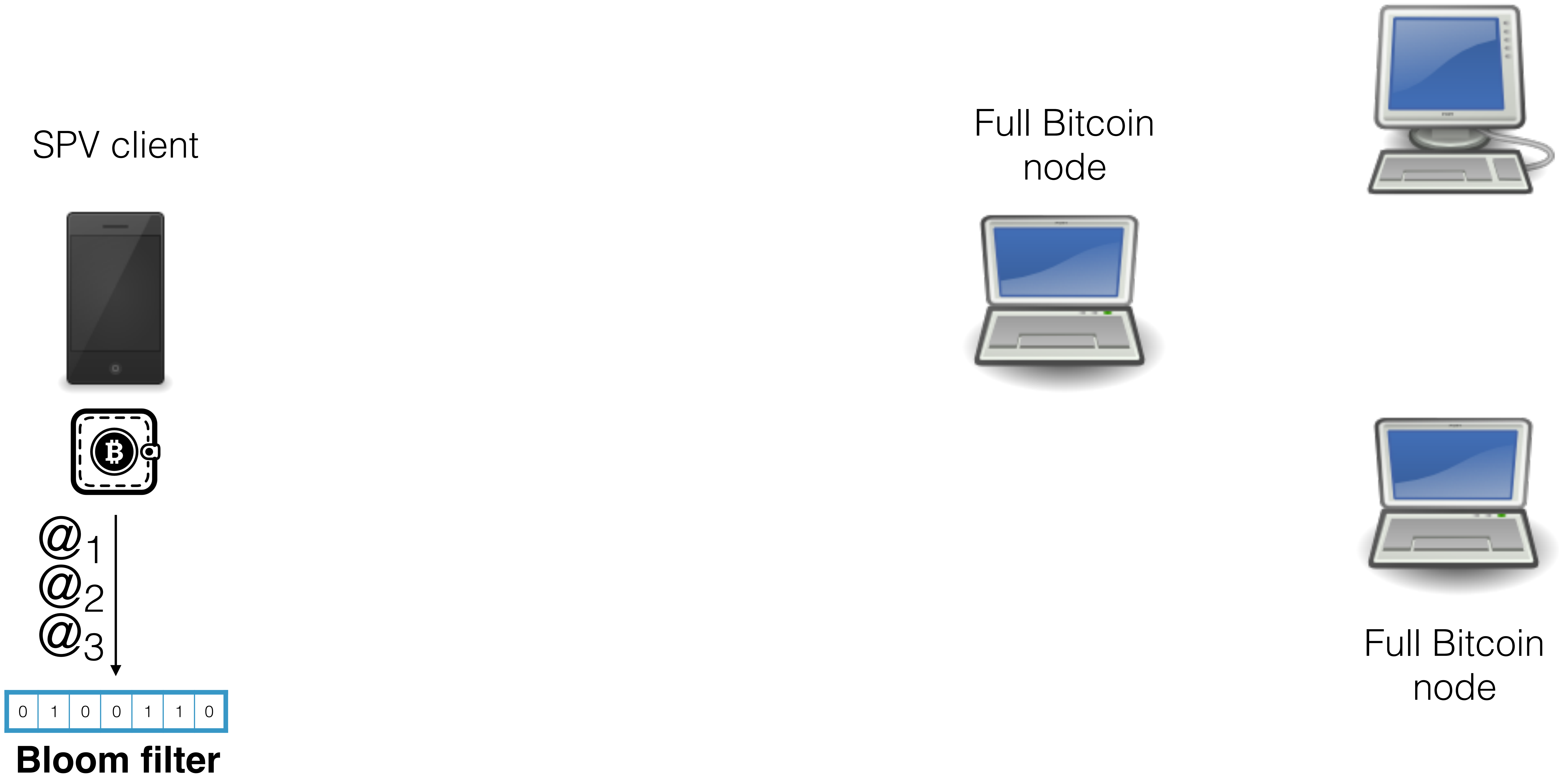
! @4 False positive → **target False Positive Rate** (FPR)

@5 True negative

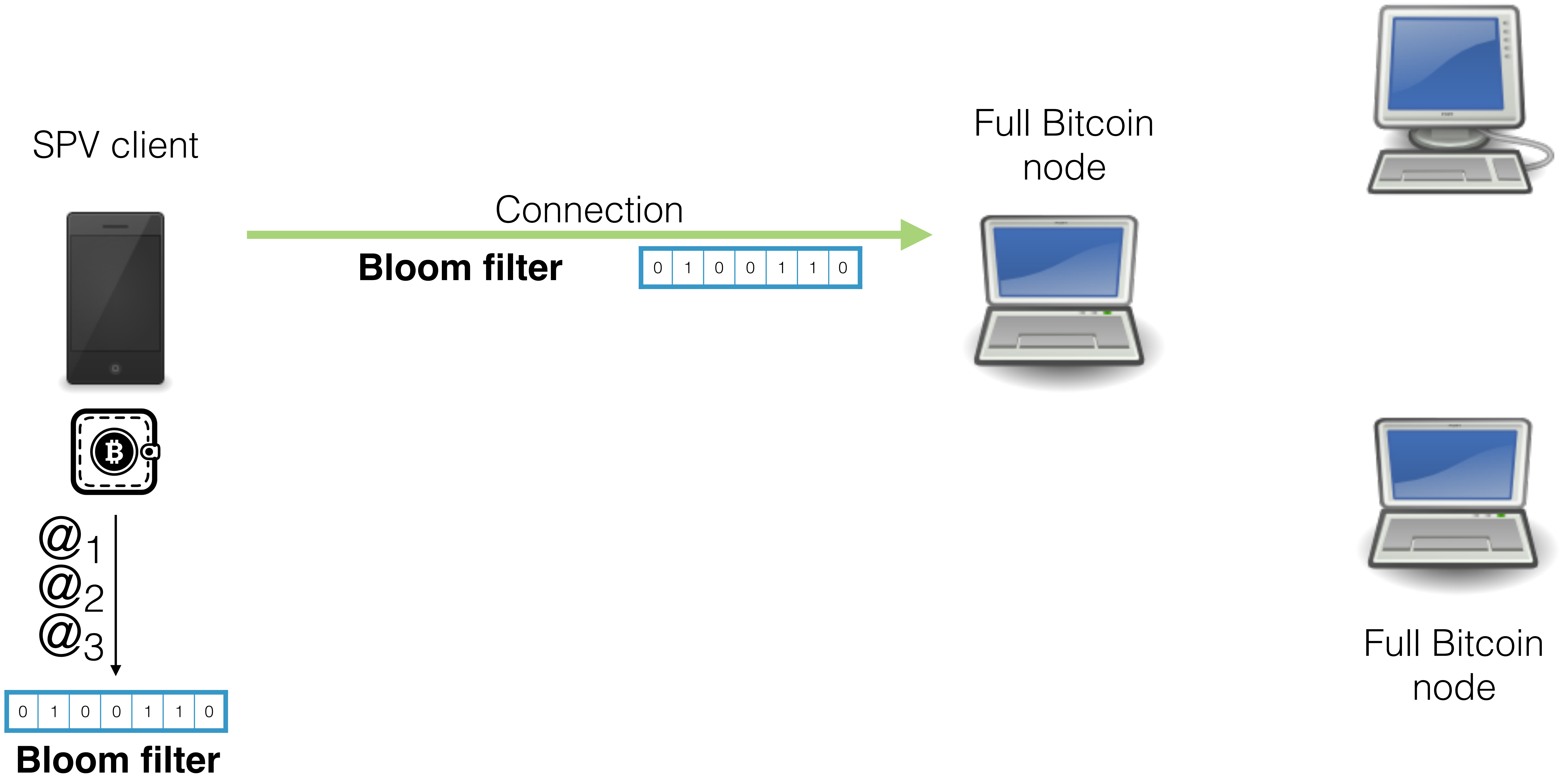
BIP-37 (Bloom Filters)



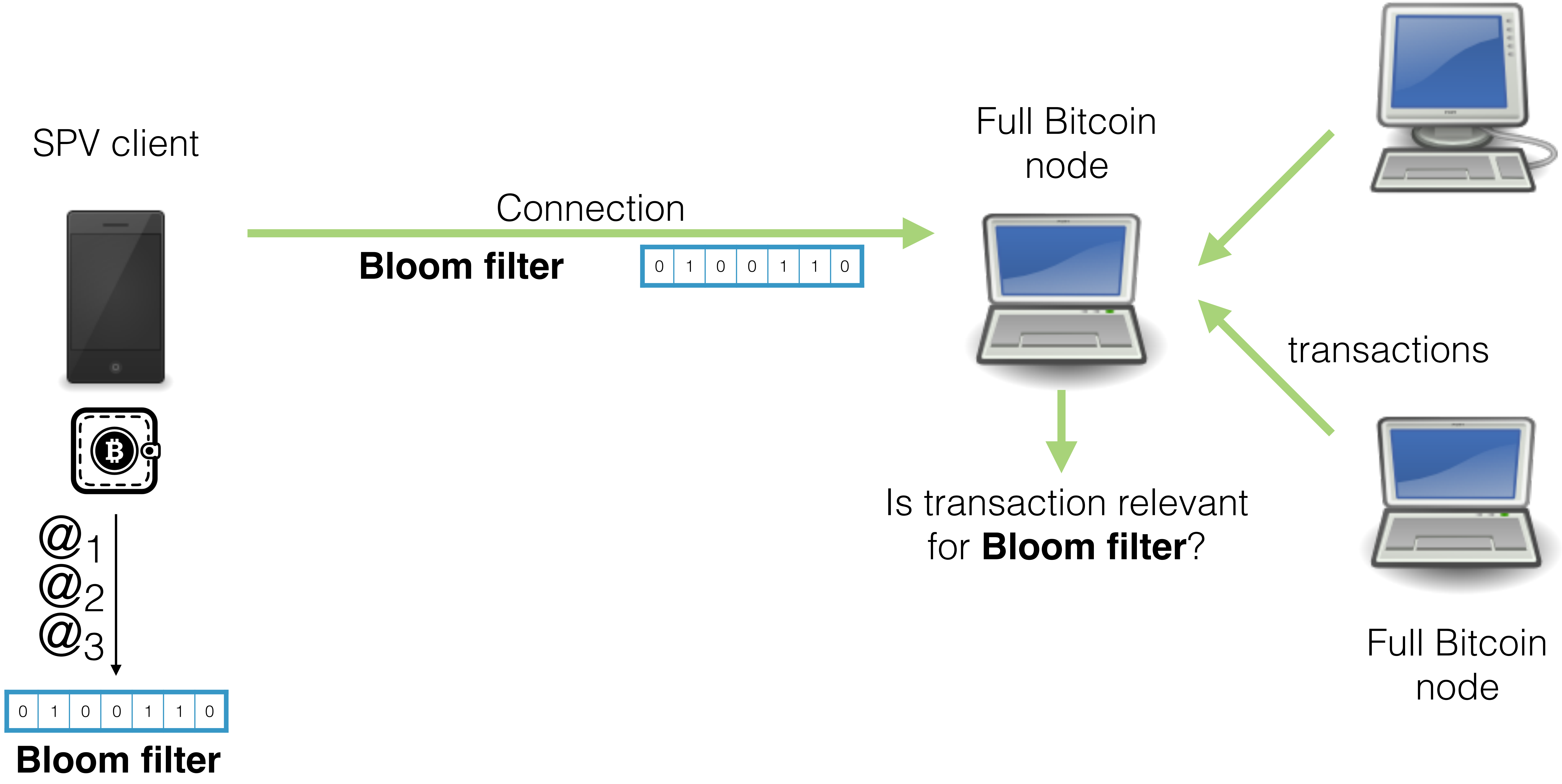
BIP-37 (Bloom Filters)



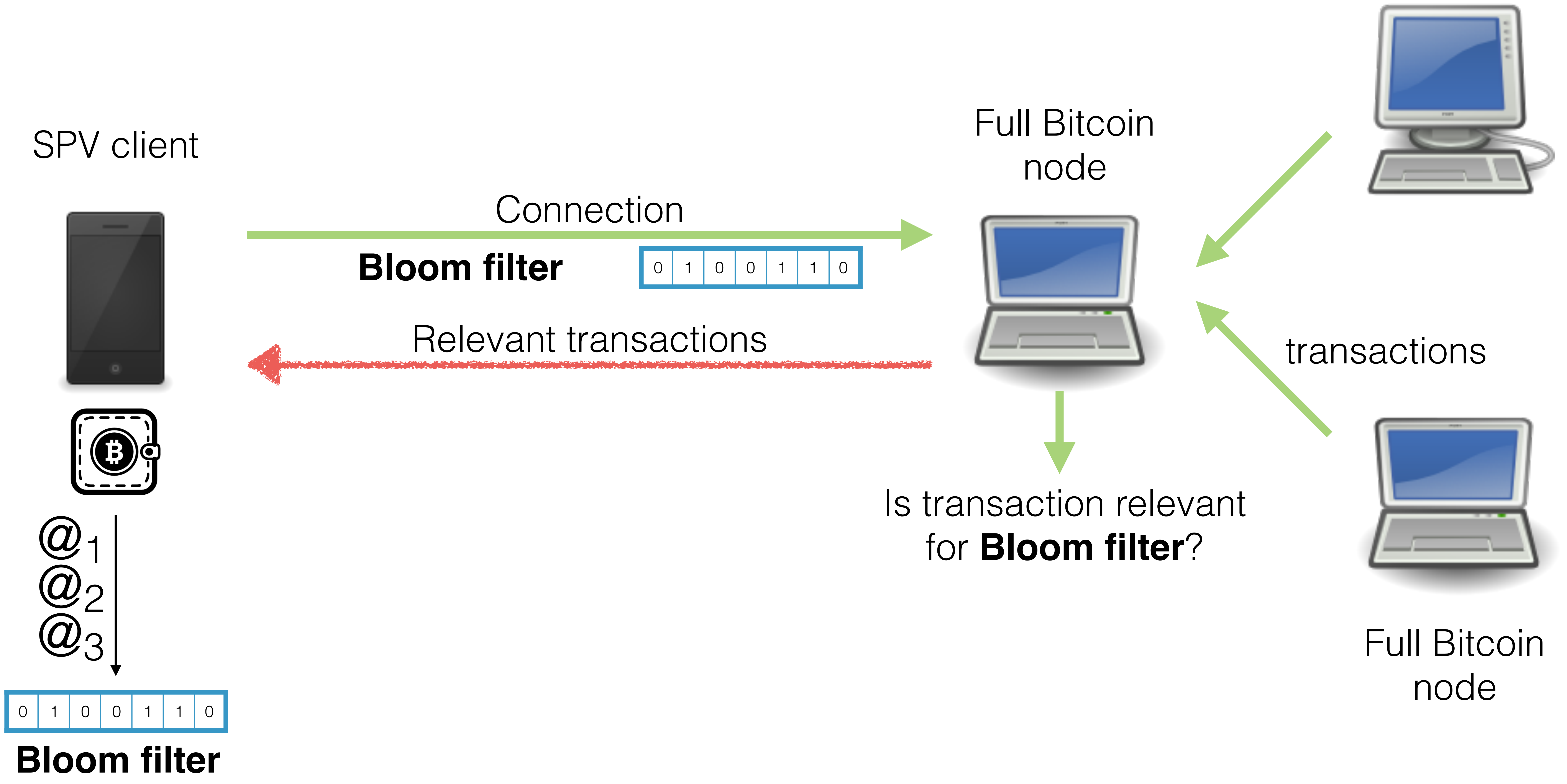
BIP-37 (Bloom Filters)



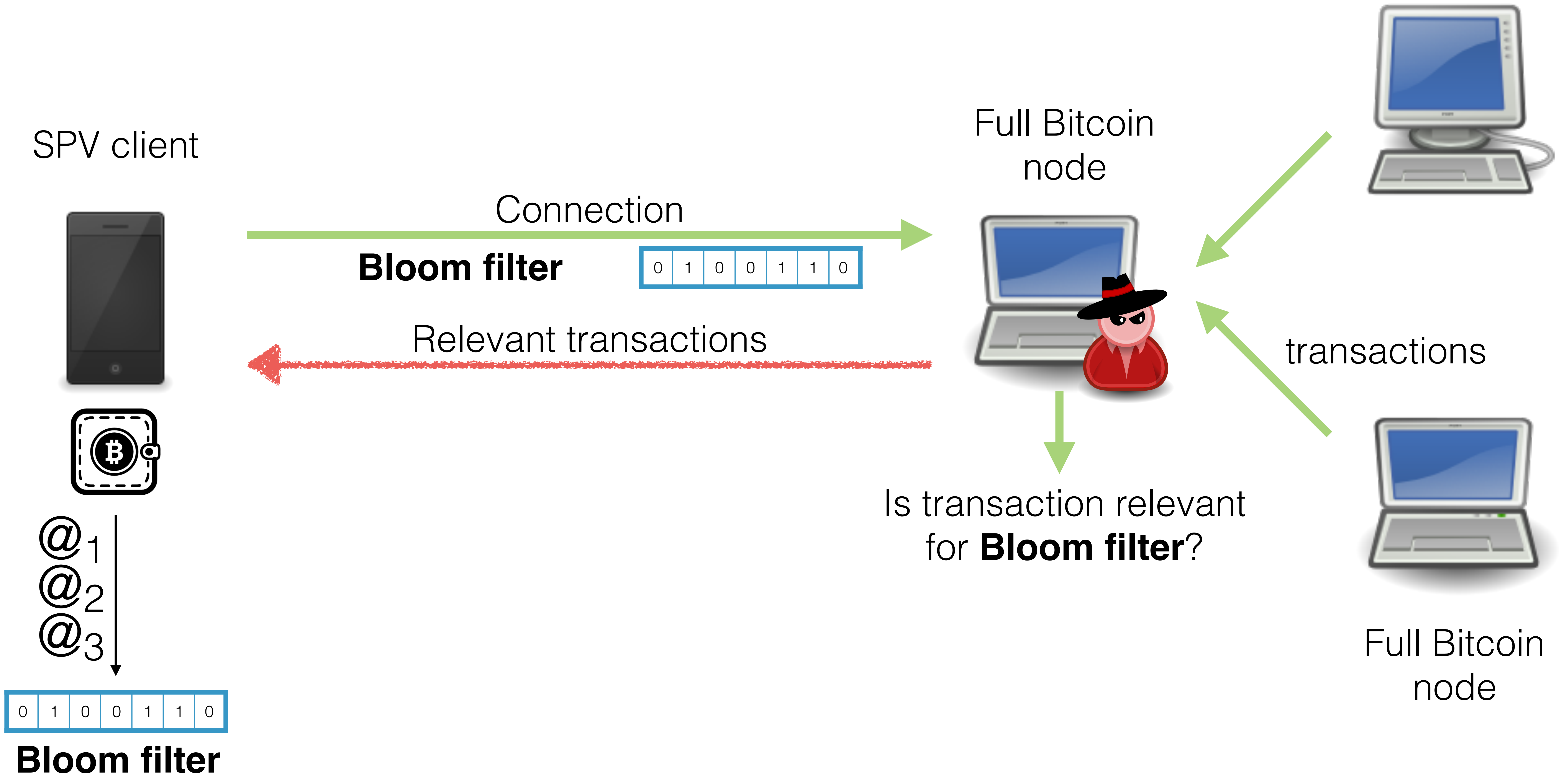
BIP-37 (Bloom Filters)



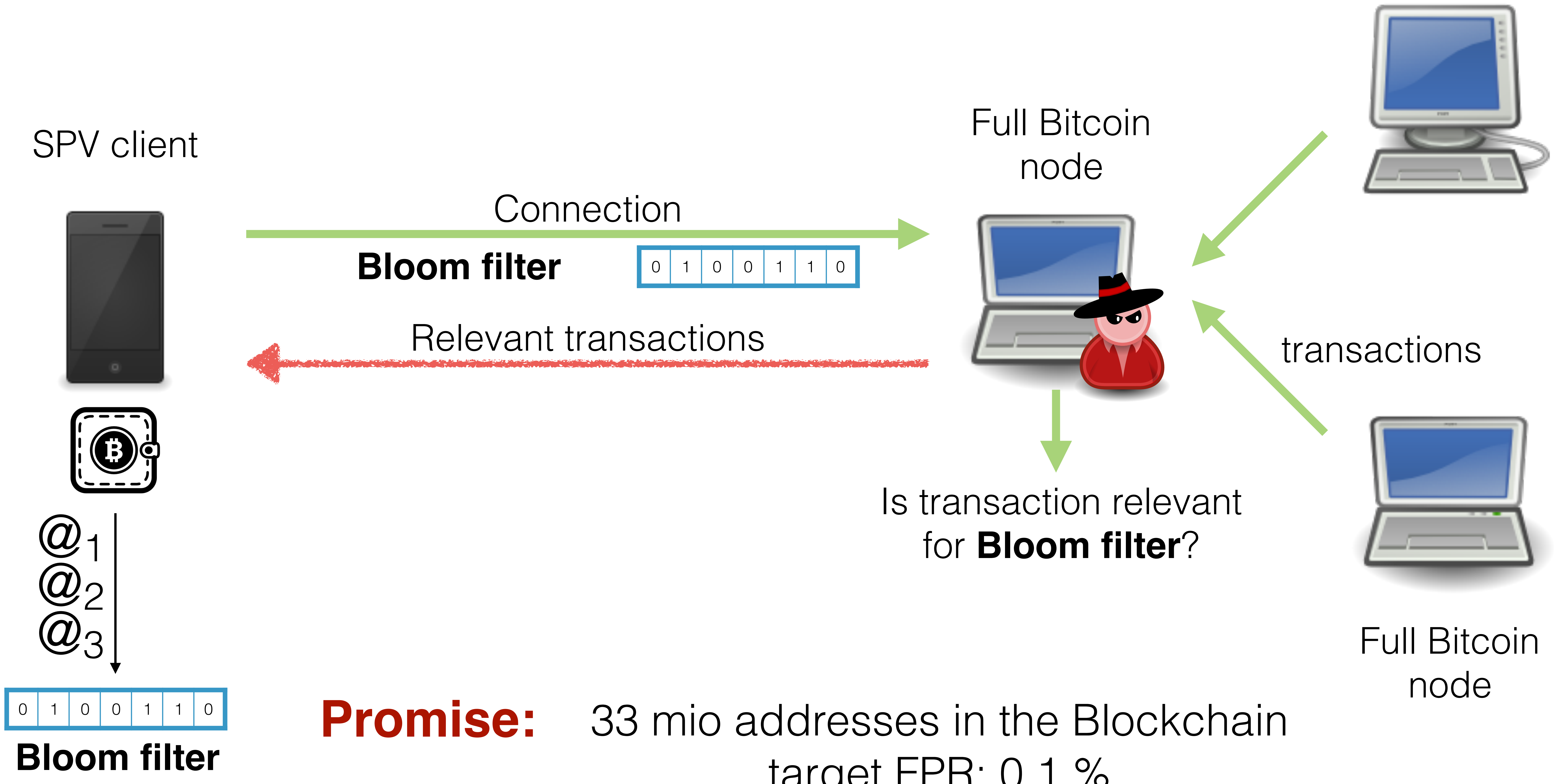
BIP-37 (Bloom Filters)



BIP-37 (Bloom Filters)



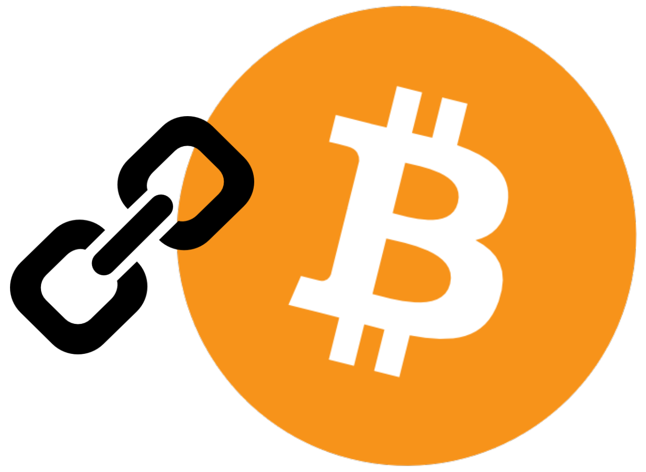
BIP-37 (Bloom Filters)



Promise: 33 mio addresses in the Blockchain
target FPR: 0.1 %

"User addresses hidden amongst 33 000" false positives

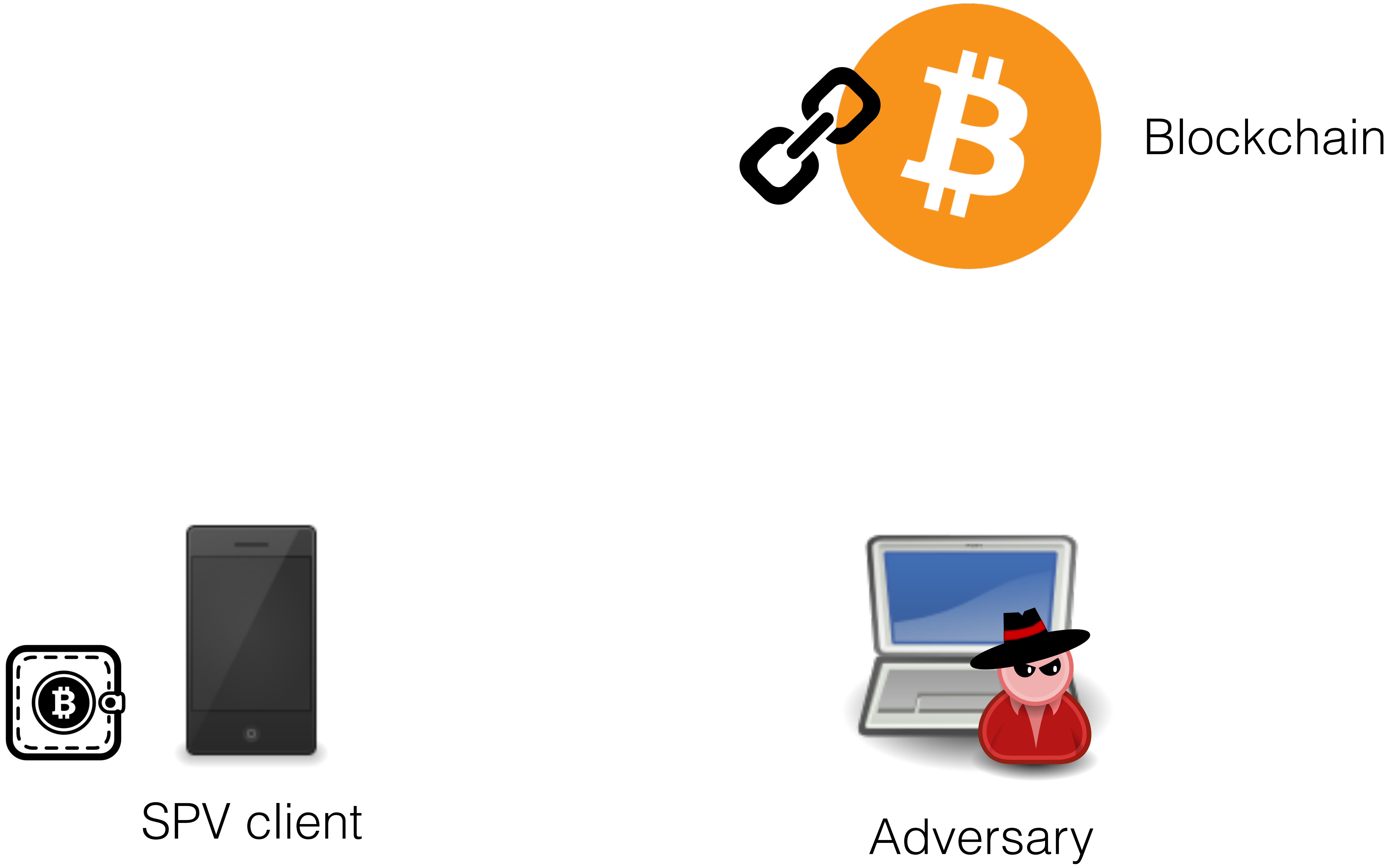
Model and Privacy measure



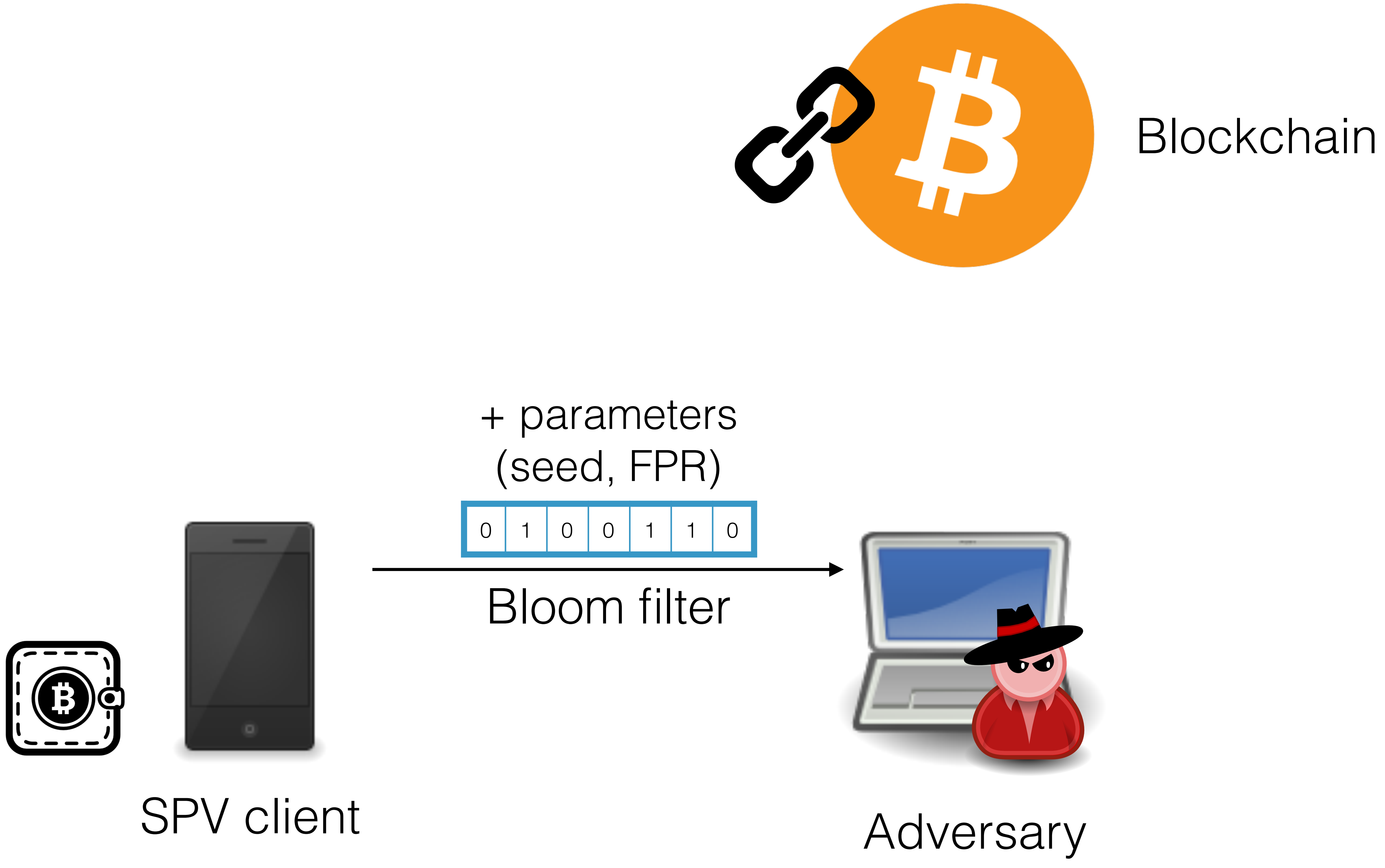
Blockchain

SPV client

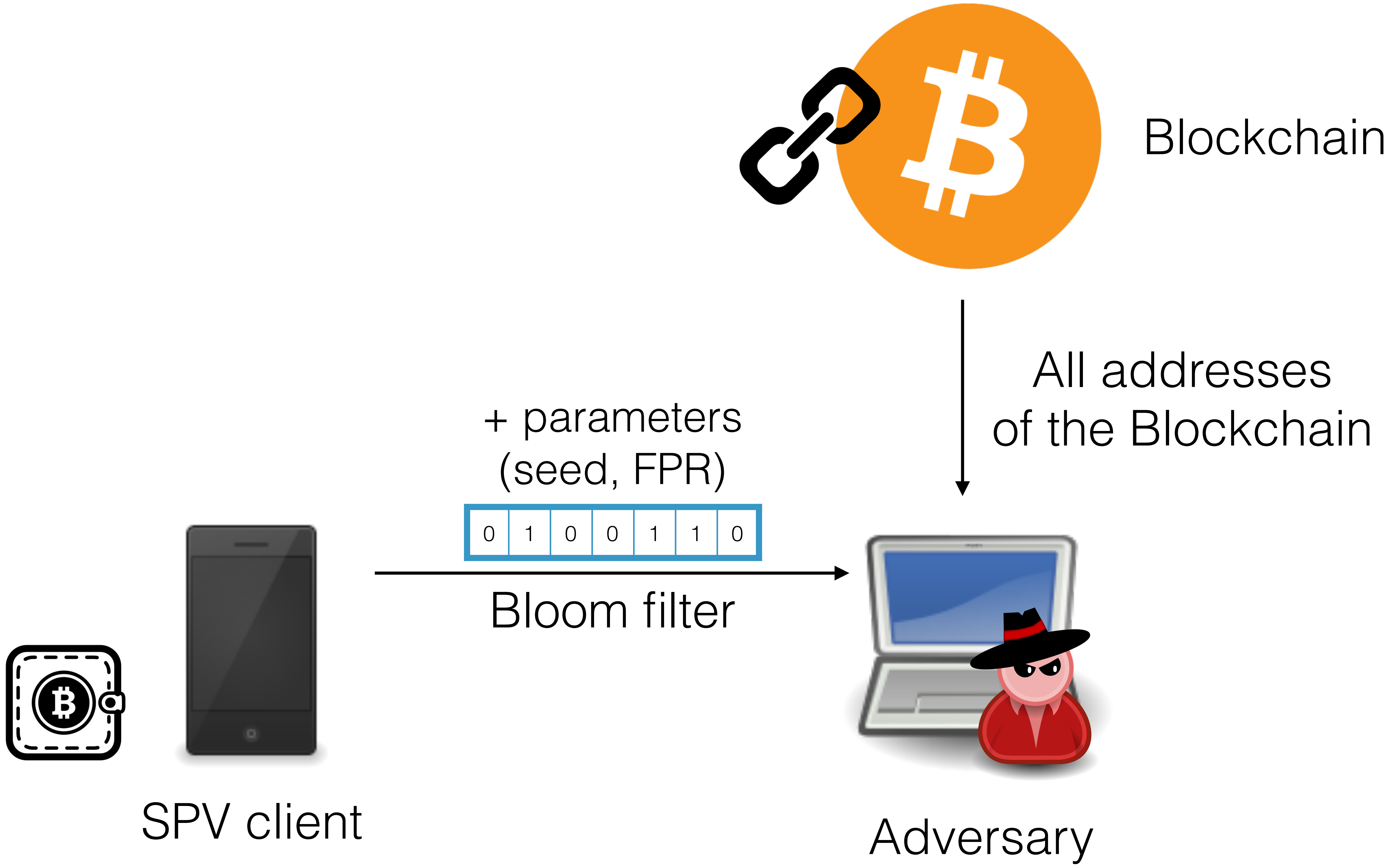
Model and Privacy measure



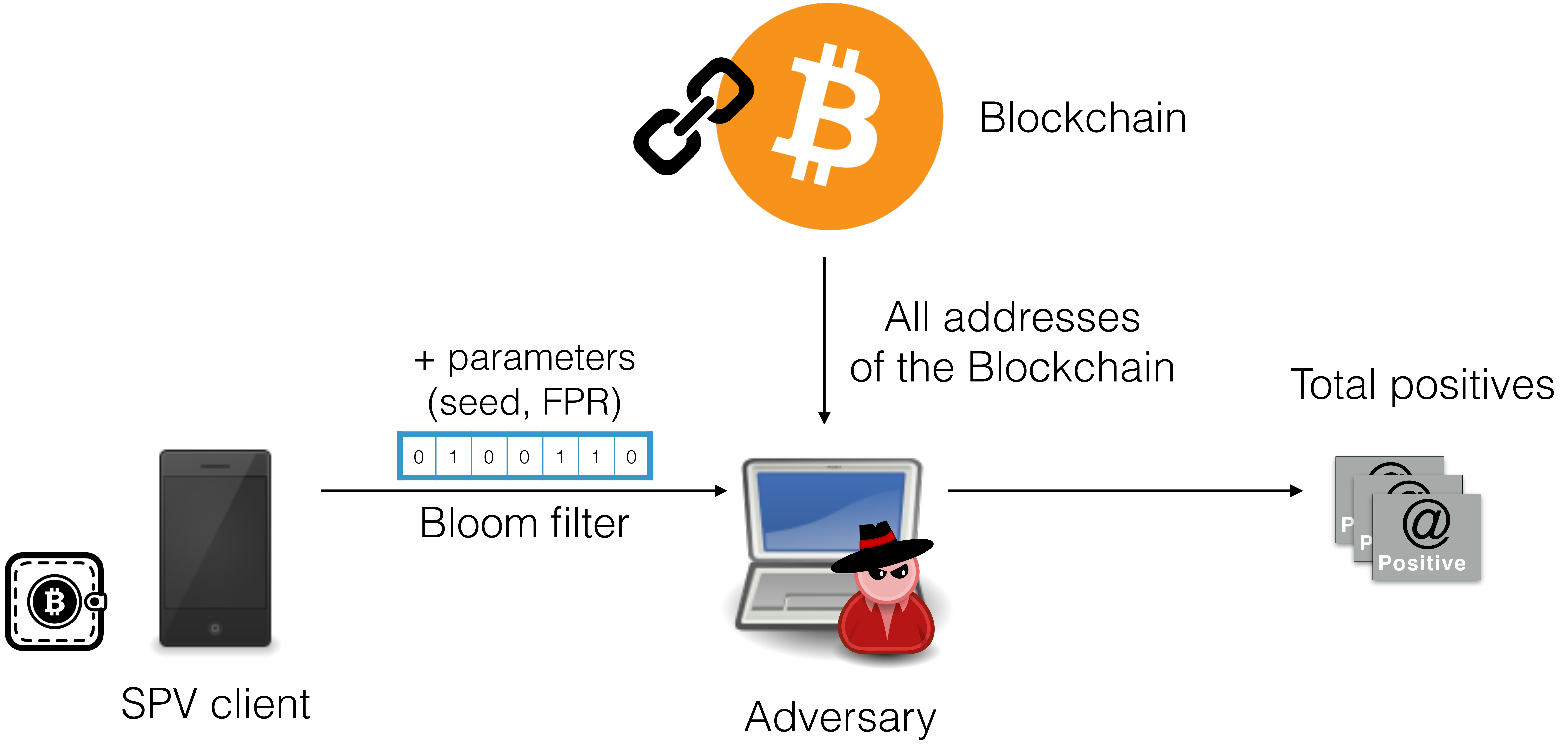
Model and Privacy measure



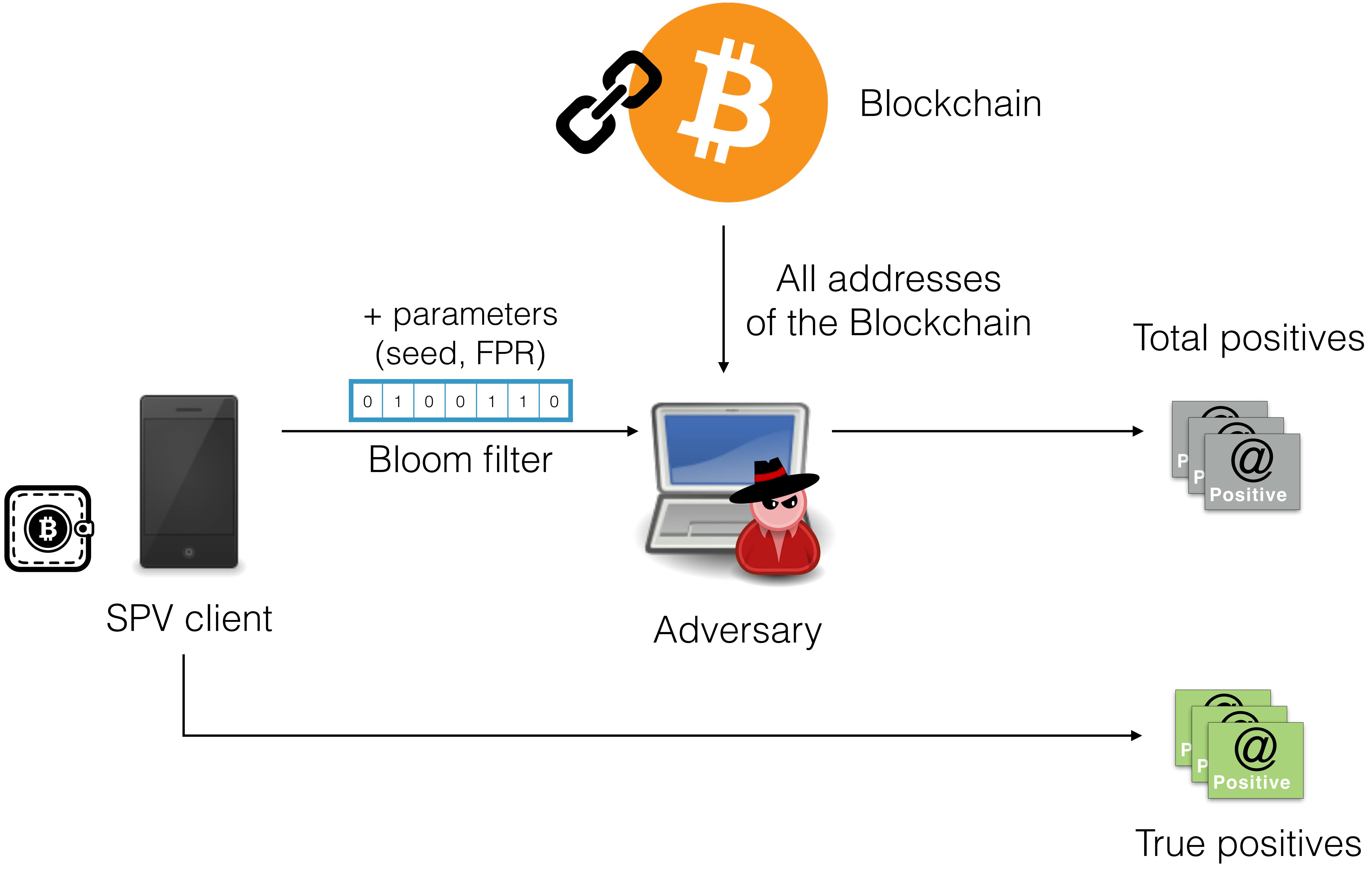
Model and Privacy measure



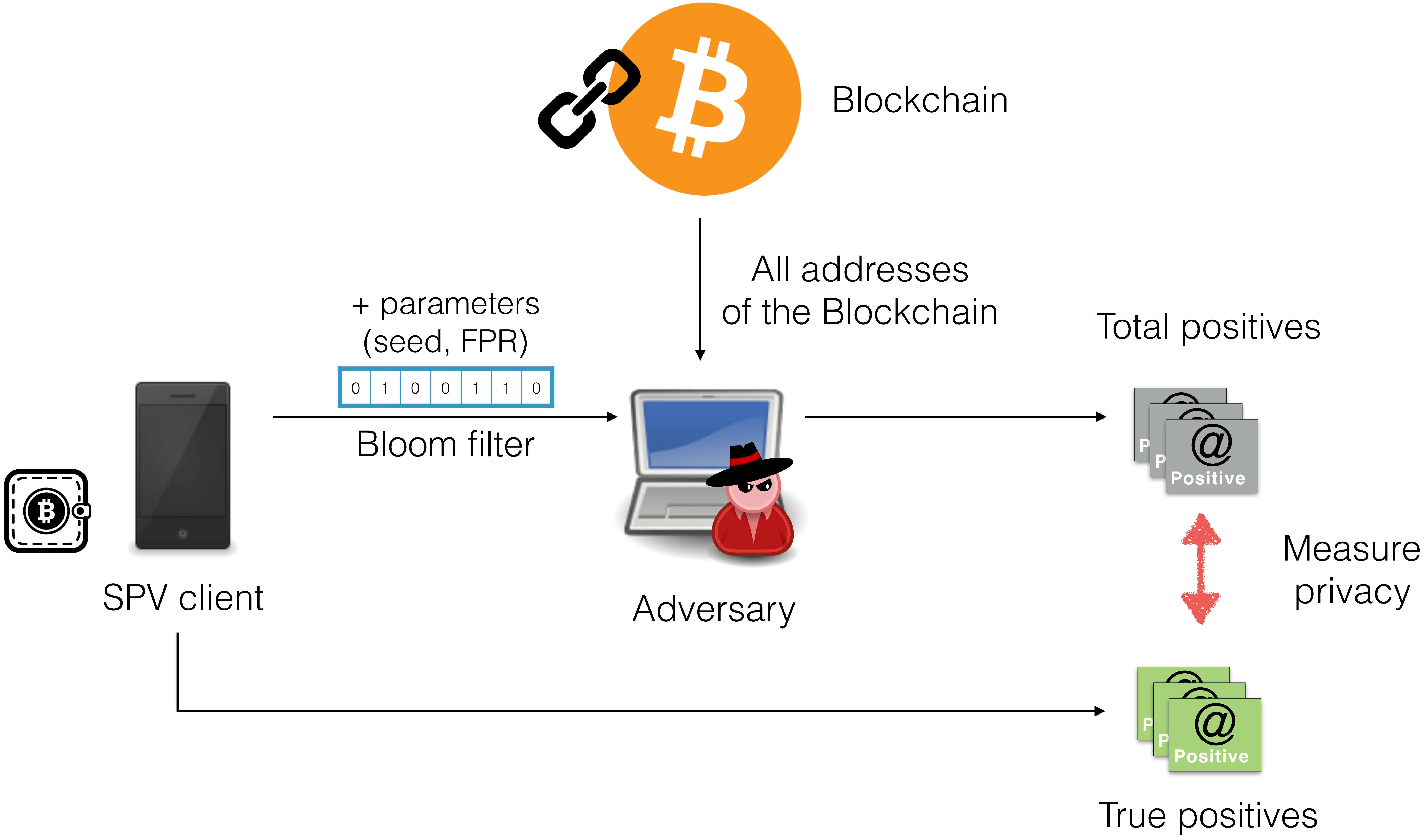
Model and Privacy measure



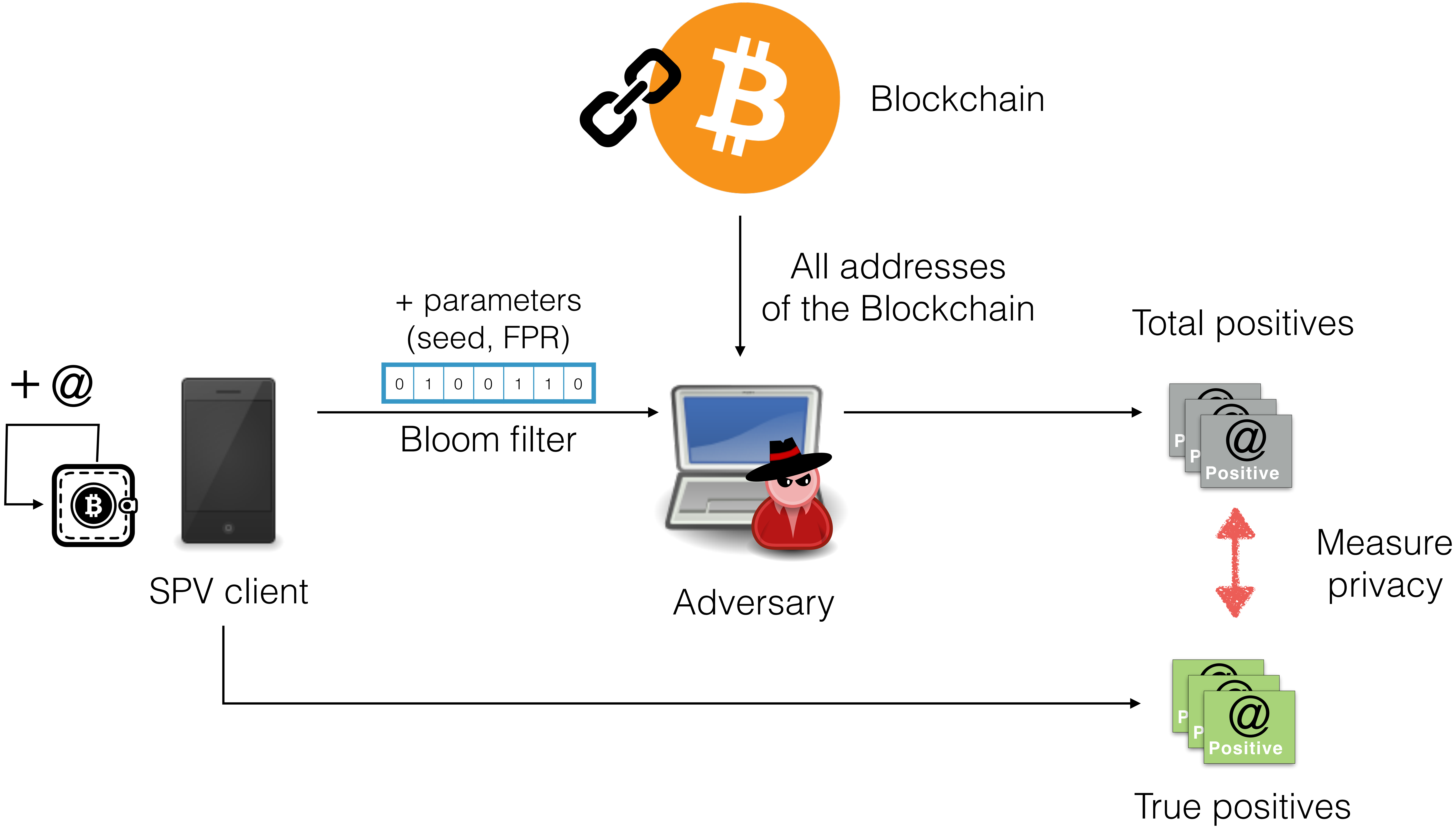
Model and Privacy measure



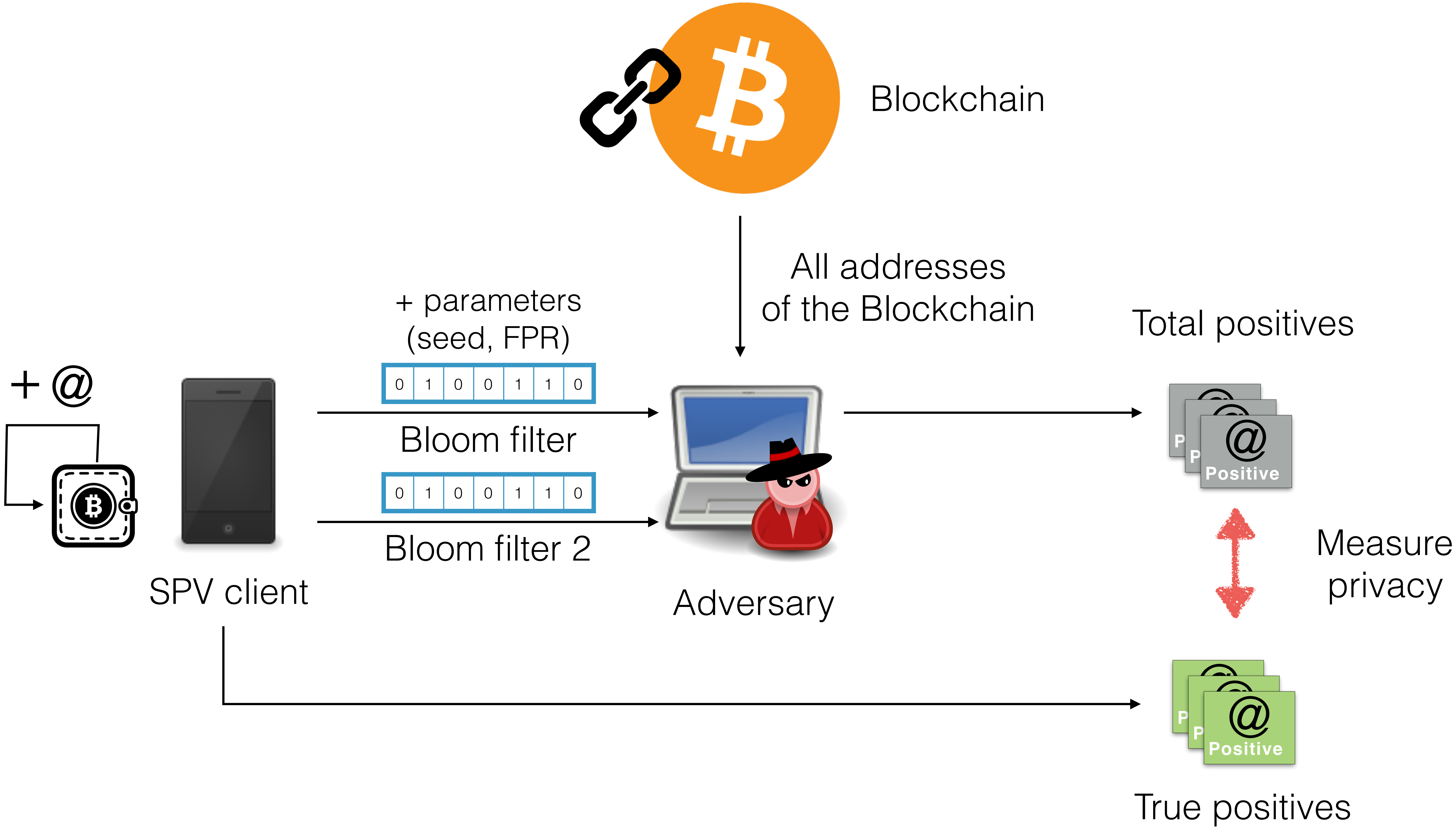
Model and Privacy measure



Model and Privacy measure



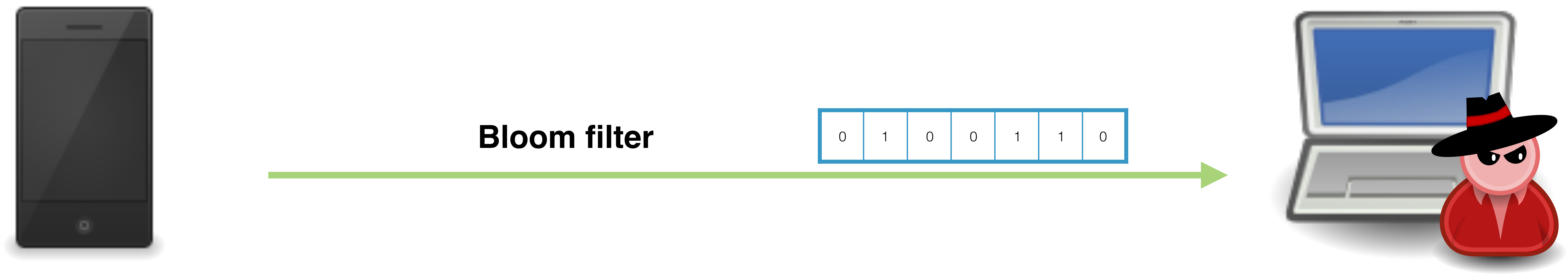
Model and Privacy measure



SPV + Bloom Filter

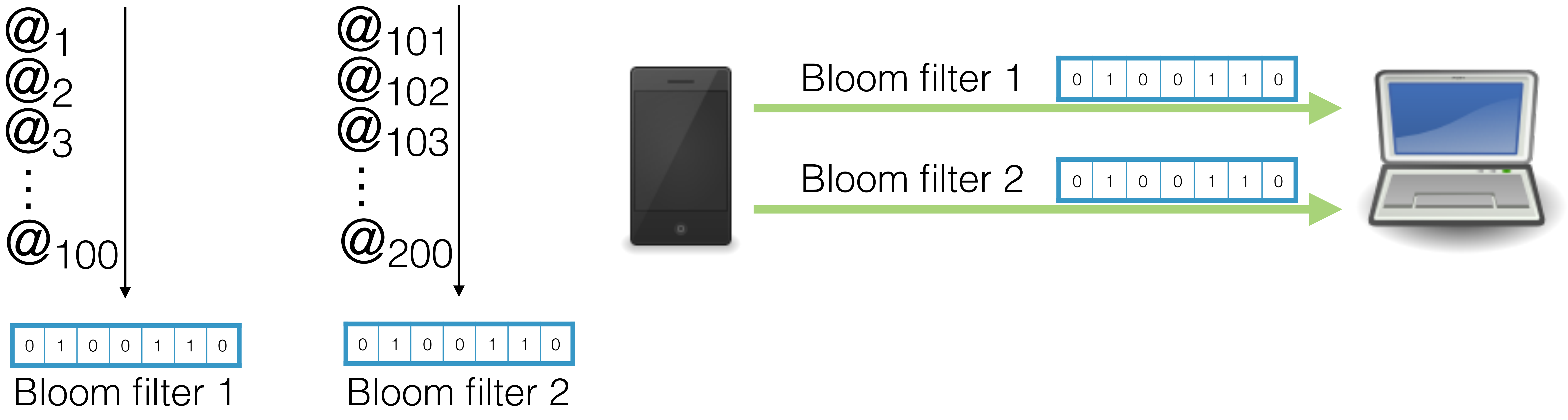
1. Given **one** Bloom filter, Bitcoin addresses partially linkable
   - Addresses **linkable** if < 20 addresses in wallet

2. Given **multiple** Bloom filter, addresses nearly always linkable



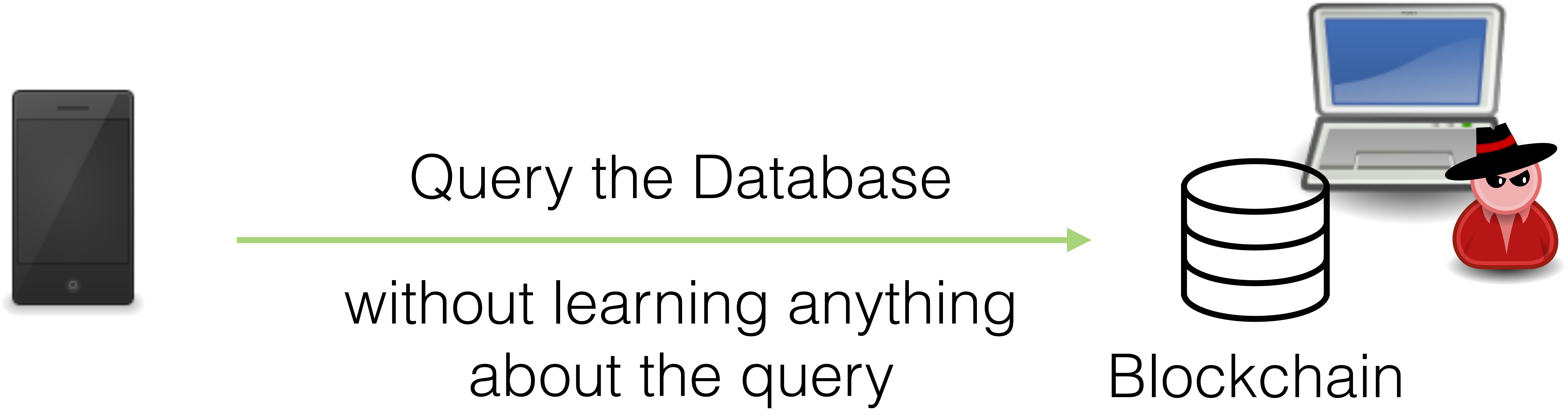
Gervais et al., On the privacy provisions of bloom filters in lightweight bitcoin clients, ACSAC'14

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



Private Information Retrieval

Query the Database

without learning anything
about the query

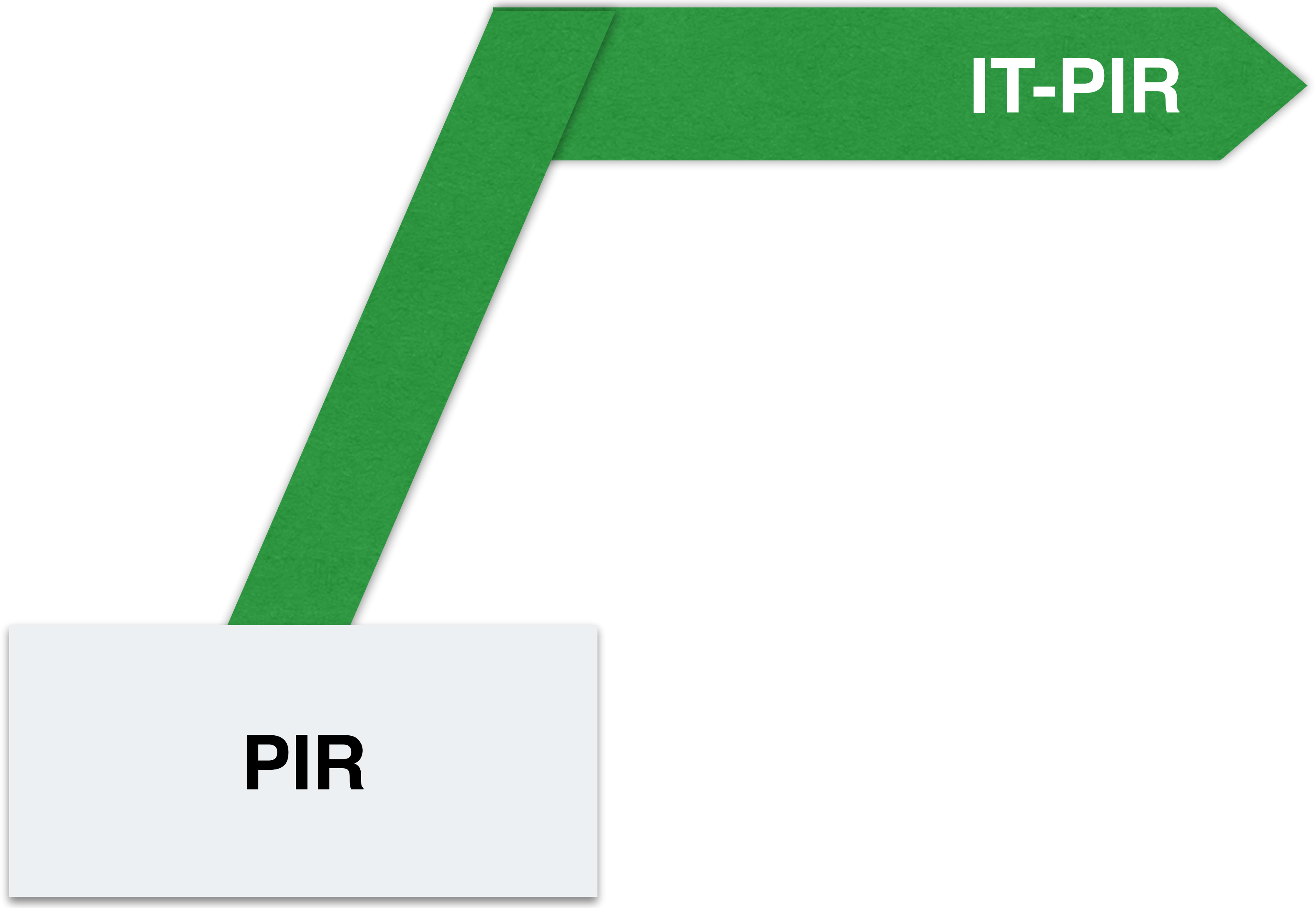
Blockchain

Solution: Download the entire DB!

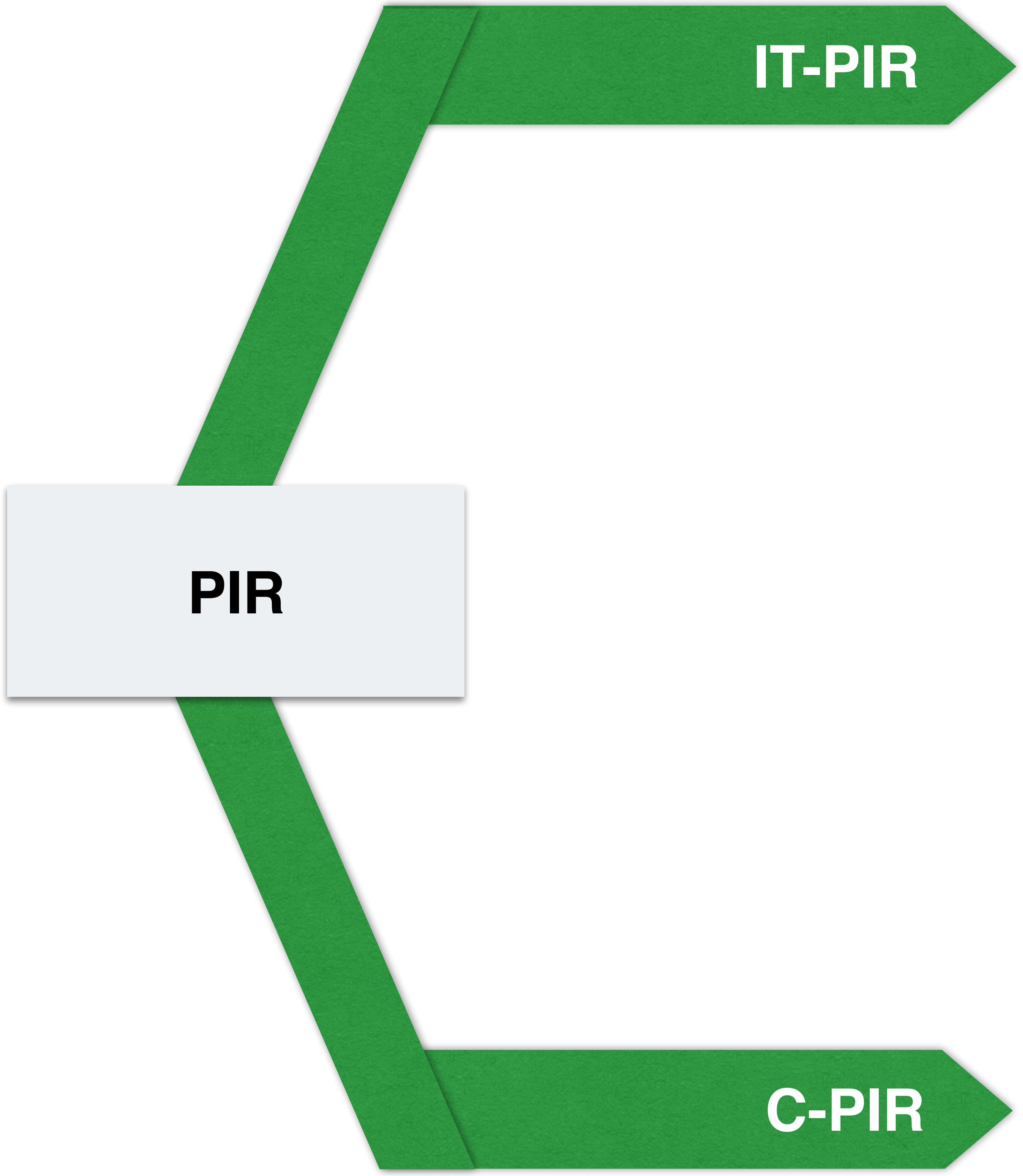
PIR Versions

PIR

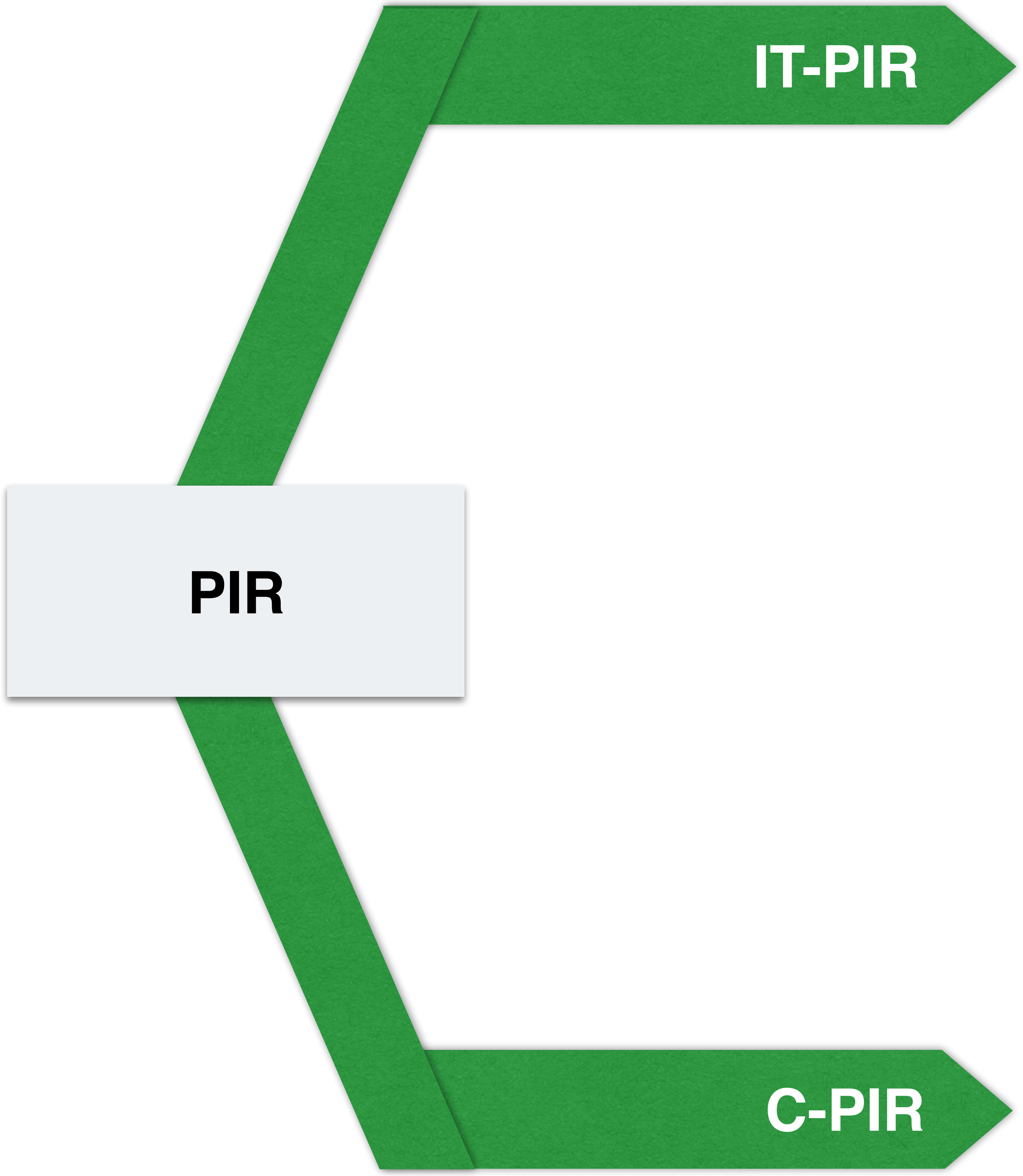
PIR Versions



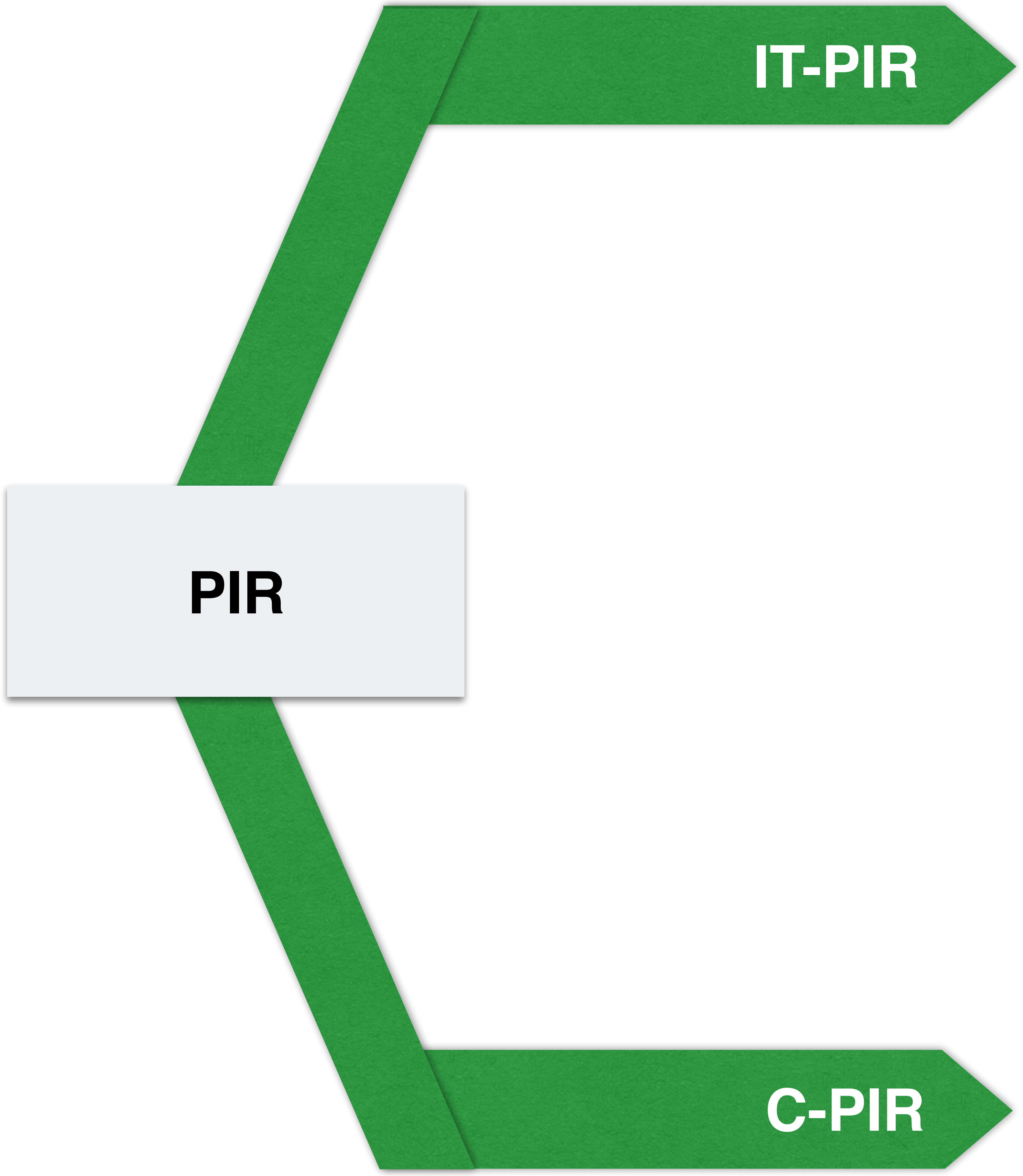
PIR Versions

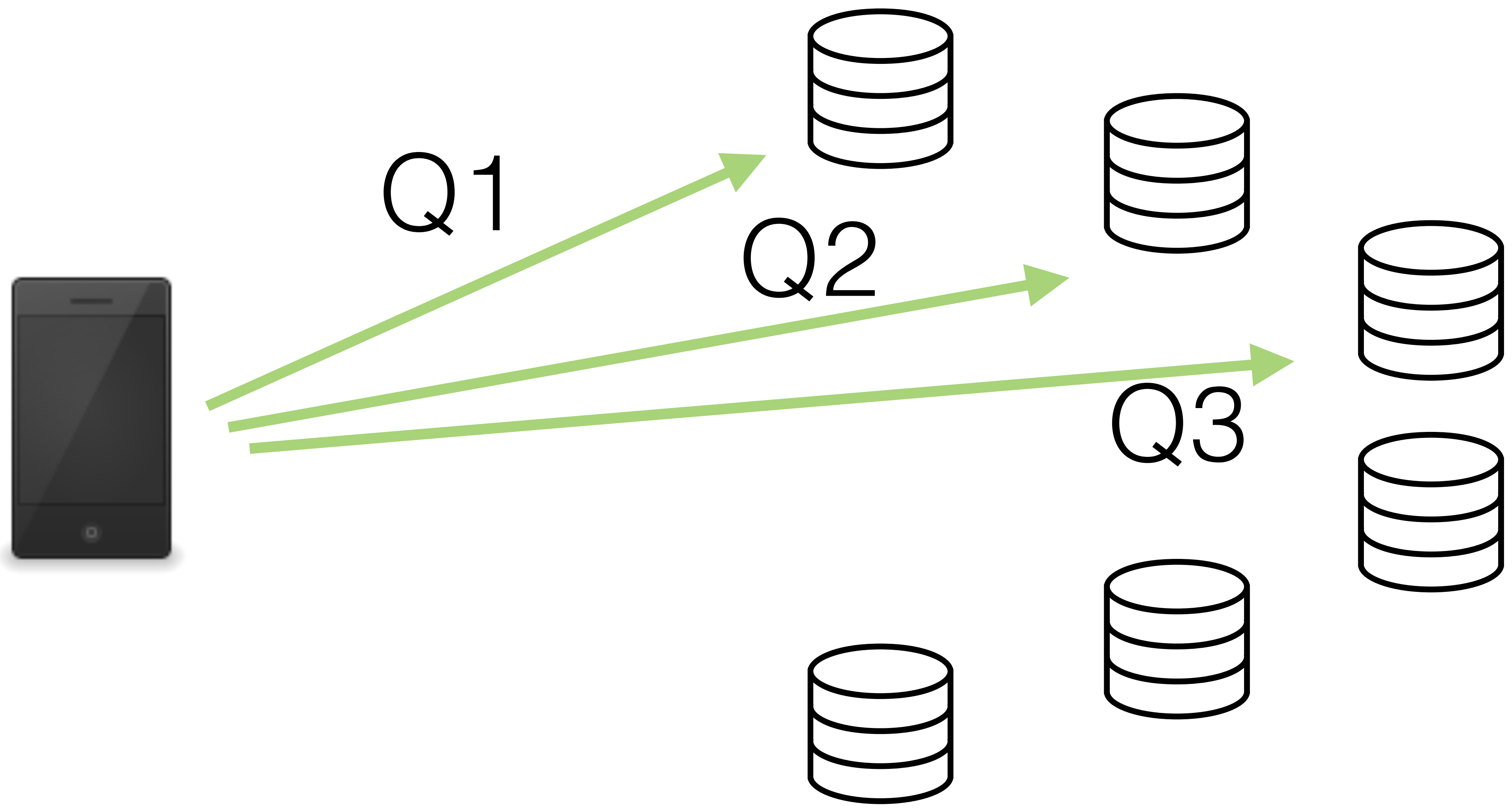


PIR Versions

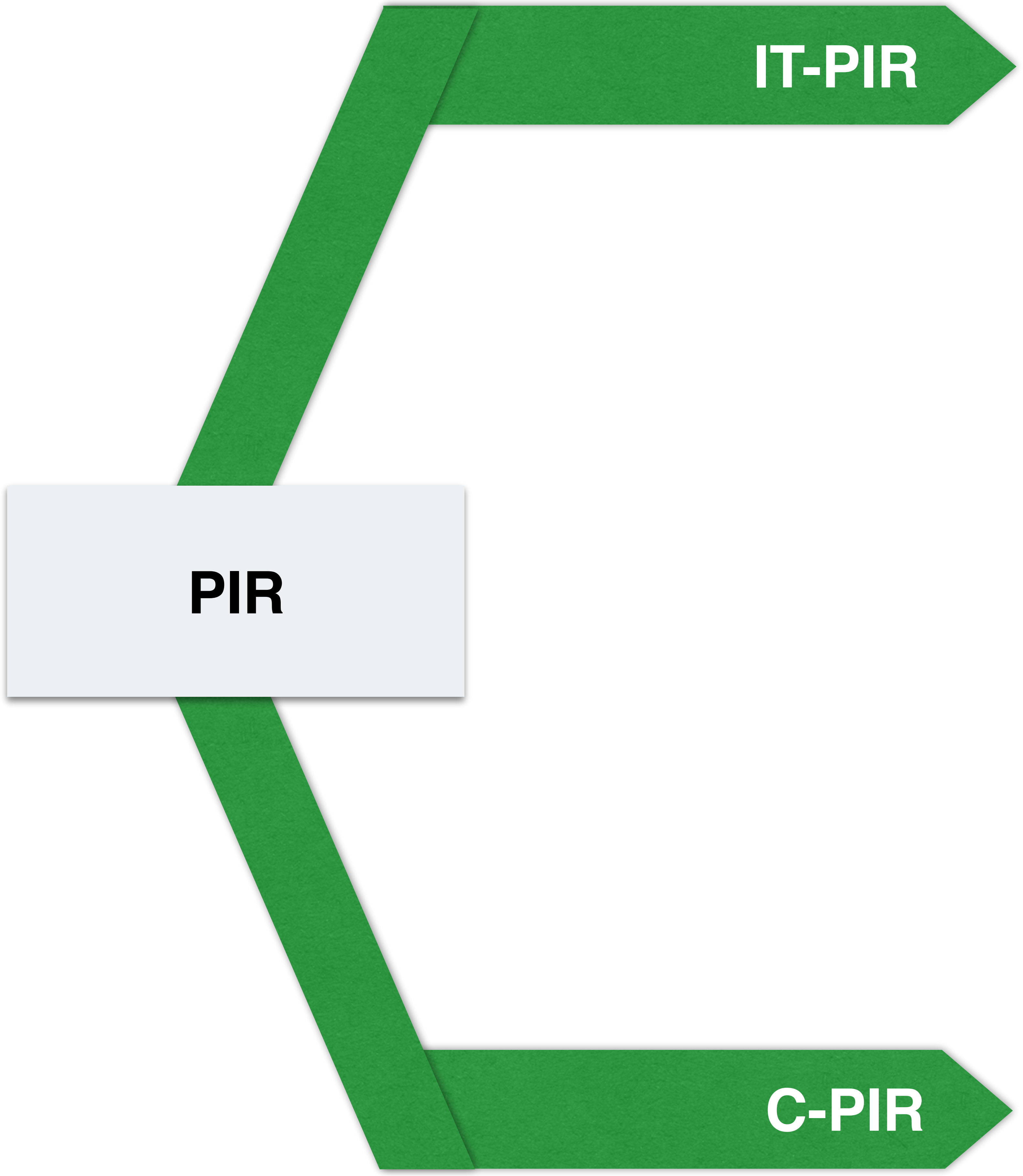


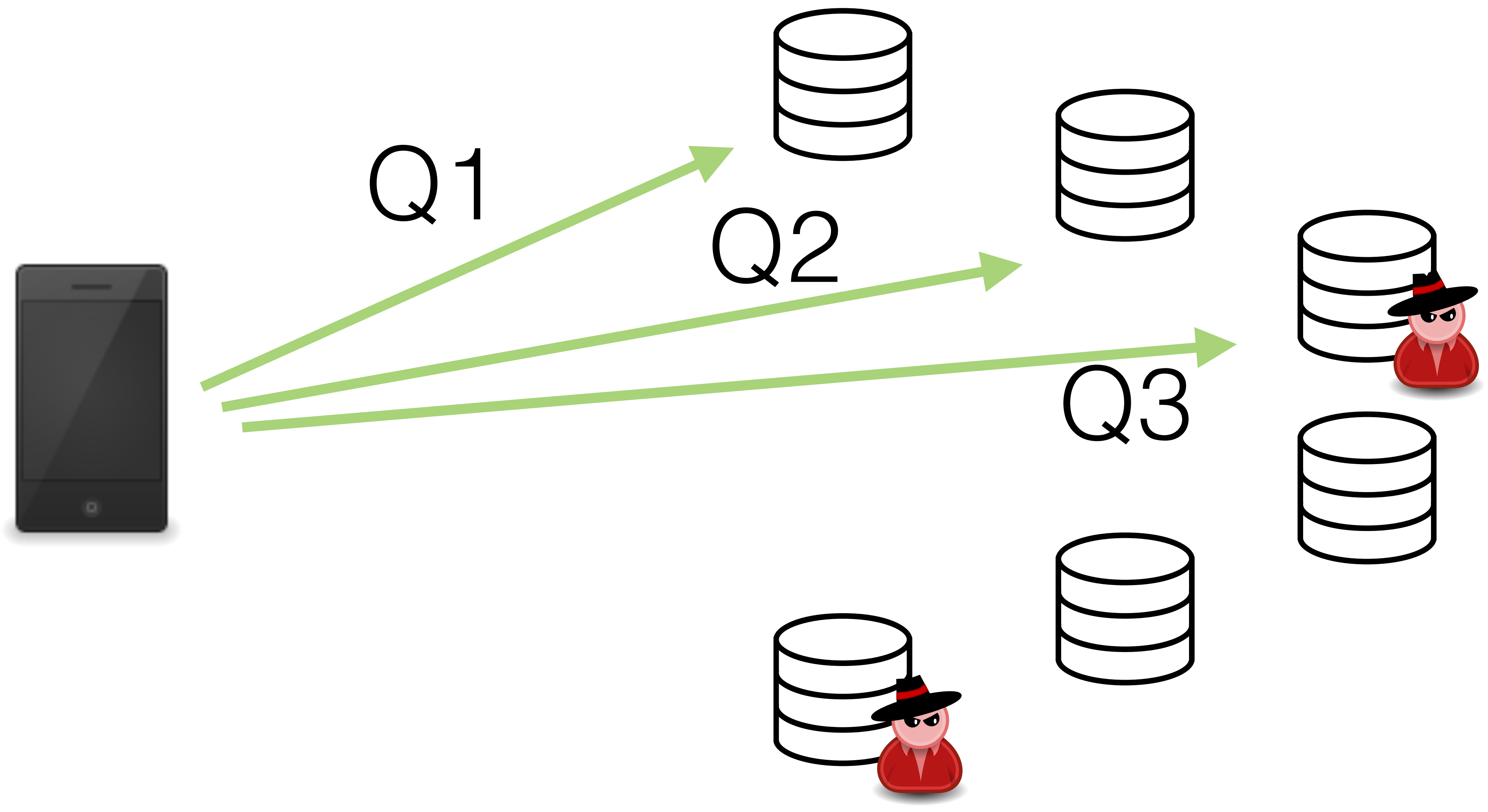
PIR Versions





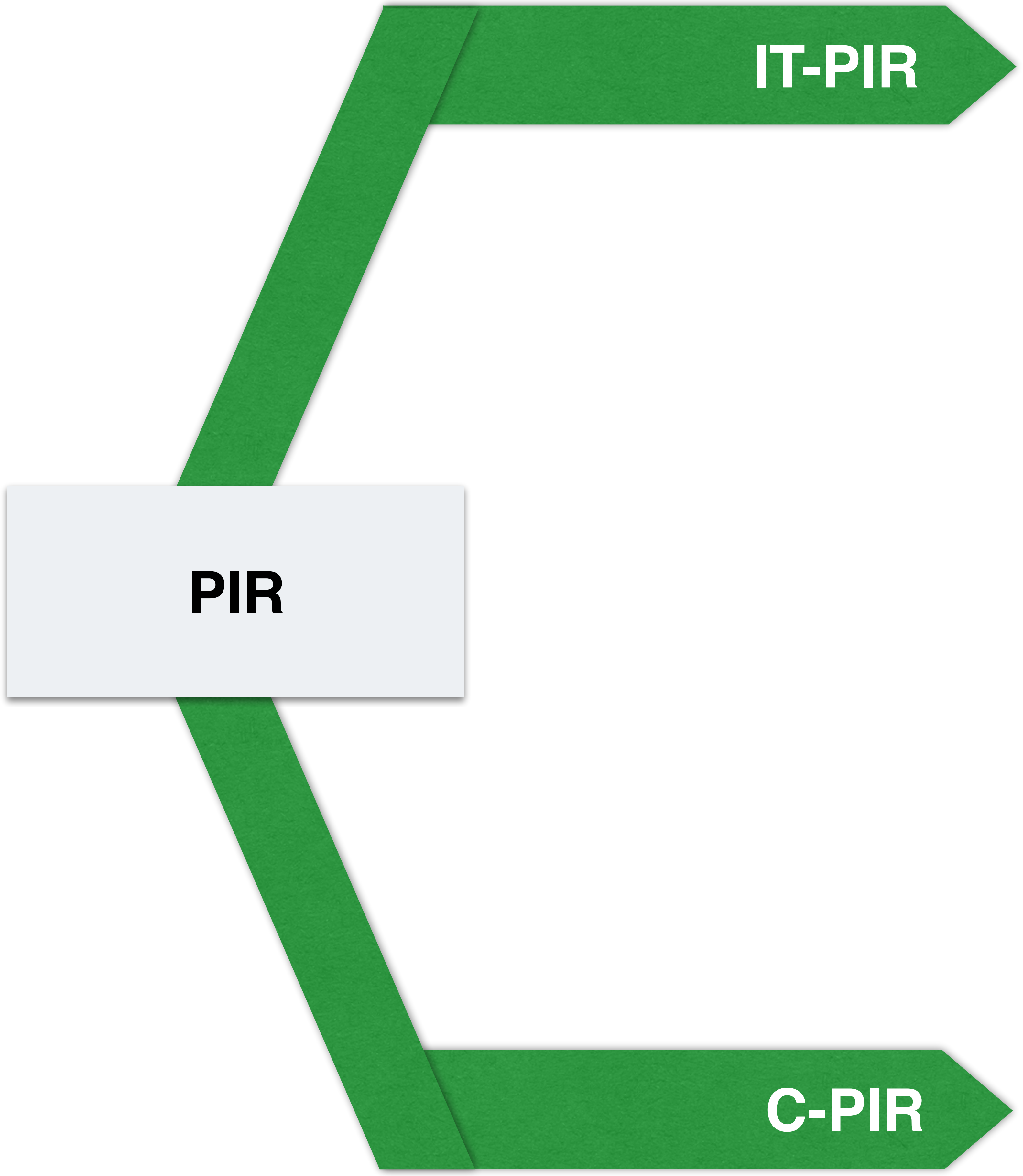
PIR Versions

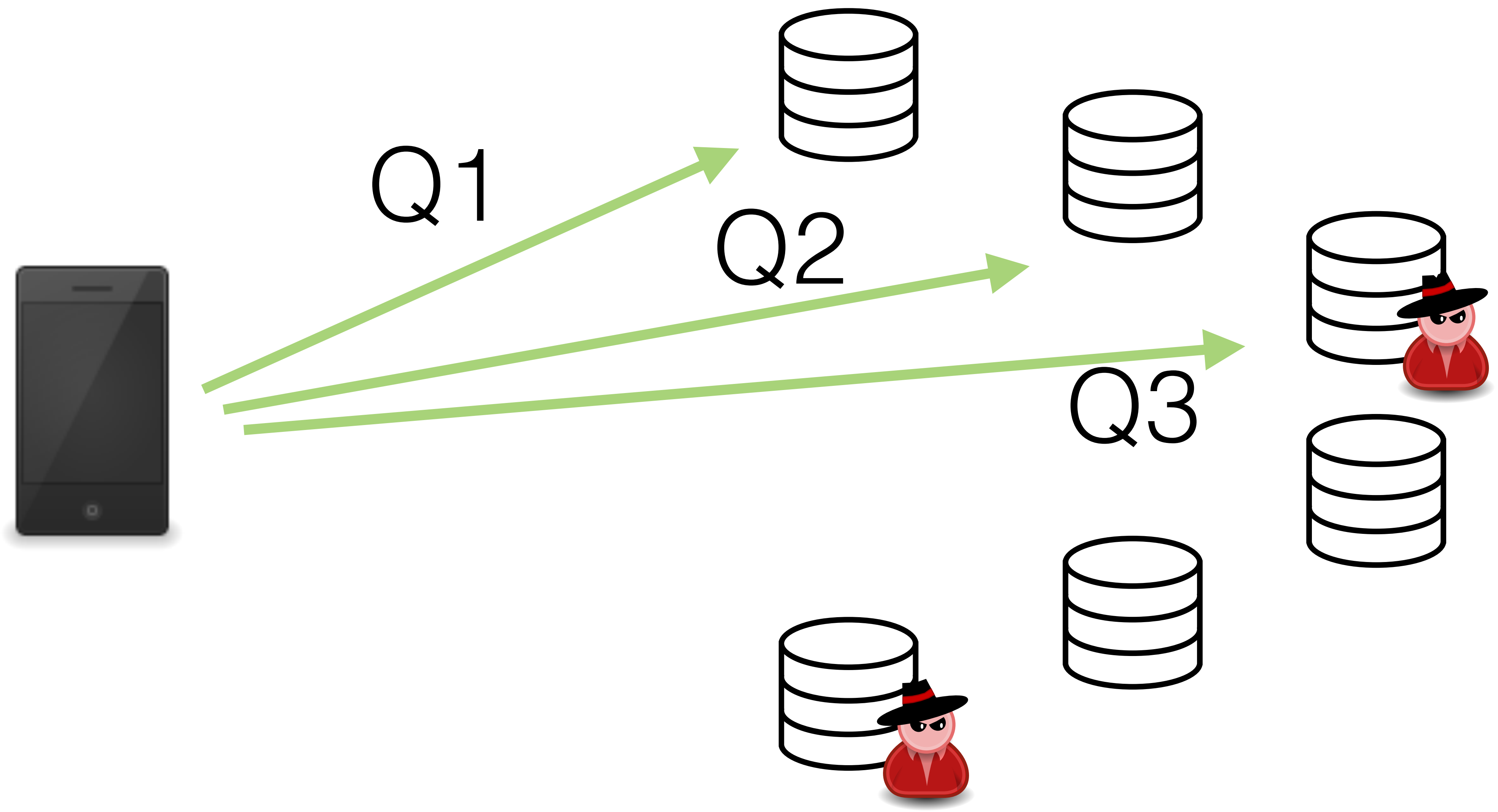




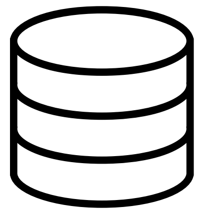
Information Theoretic security, assuming no collision up to a certain degree

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

	IT-PIR	C-PIR
Communication costs	Smaller	Bigger
Computation costs	Smaller	Bigger
Risk of Collusion	Server can mount sybil attack	Non existent
Robust to missing or incorrect server response	Robust	Non-robust

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

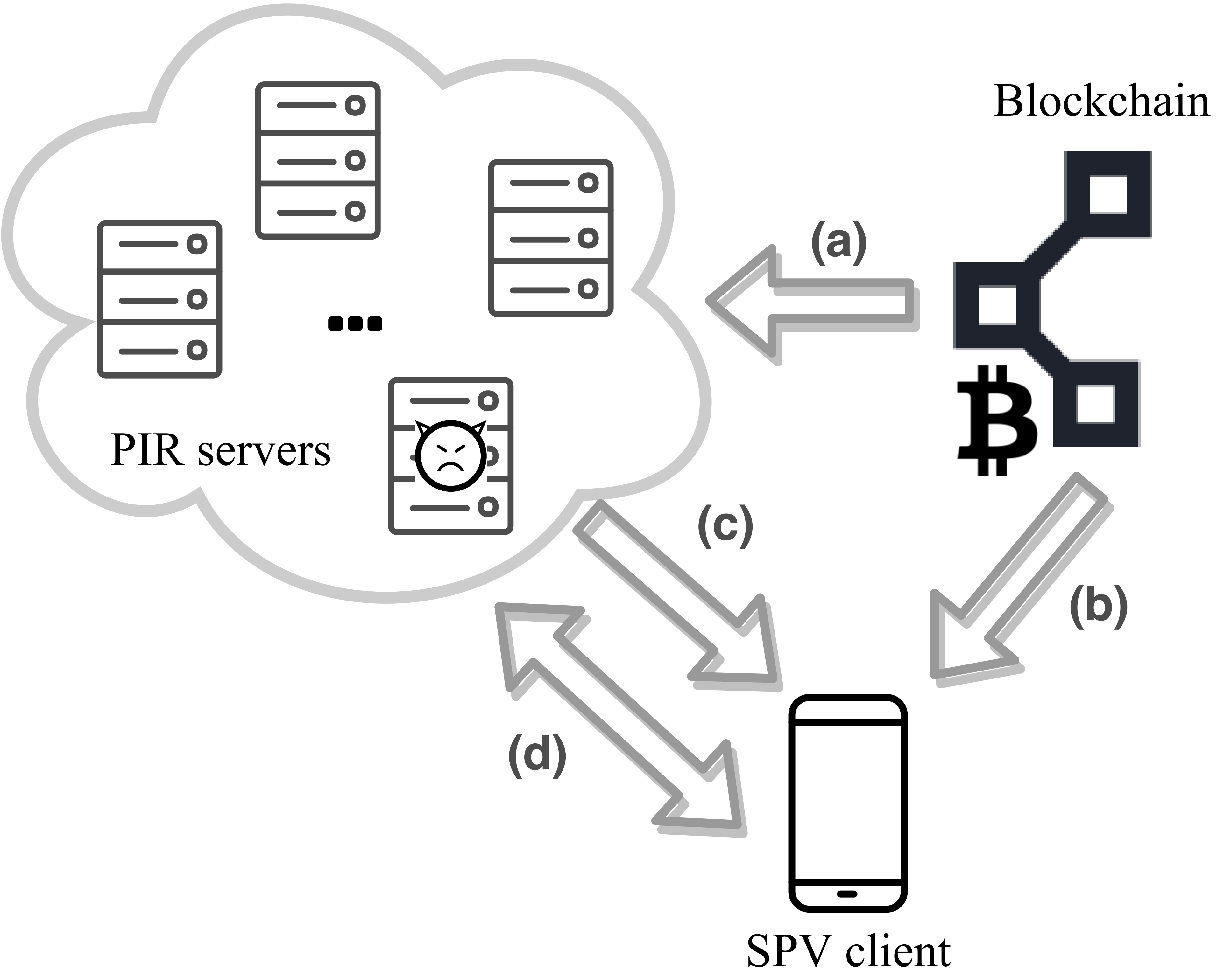
System Overview

(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 collect all available block headers from e.g., full node peers.

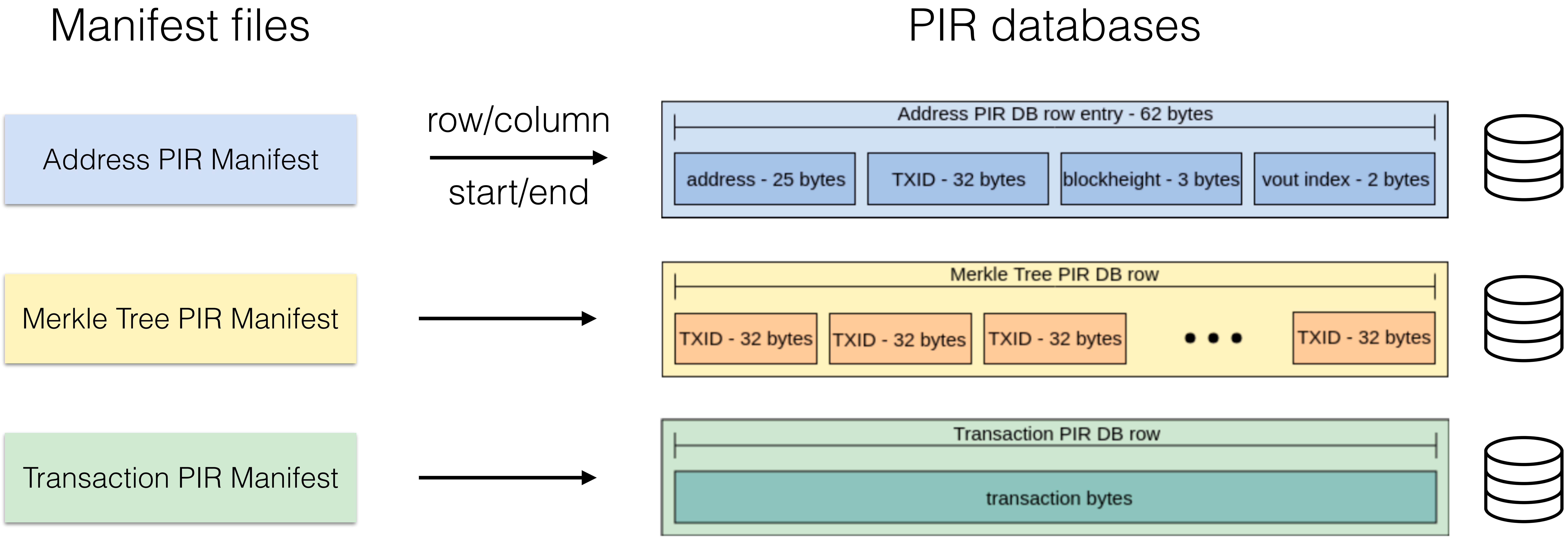
(c) The user fetch the manifest files from the PIR servers to later efficiently query the PIR database.

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

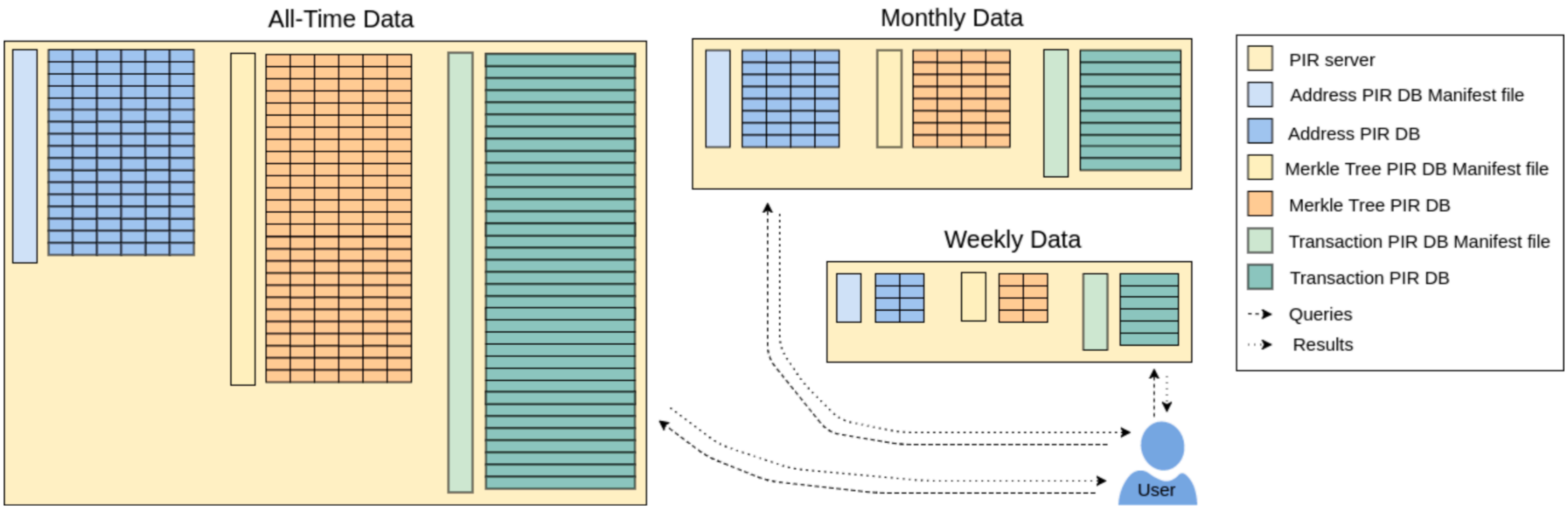


Devet, C. and Goldberg, I., The best of both worlds: Combining information-theoretic and computational PIR for communication efficiency. PETS'14

Database structure



Temporal Division



PIR Protocol - Pre-requisites

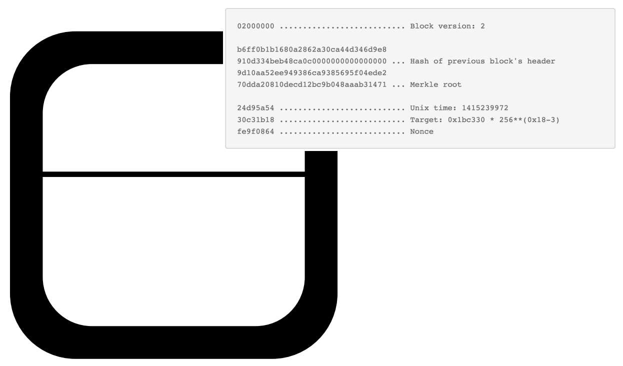
1) Download Manifest files

Address PIR Manifest

Merkle Tree PIR Manifest

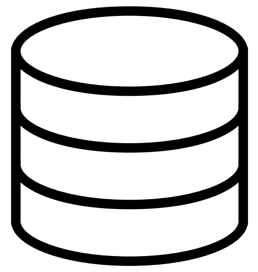
Transaction PIR Manifest

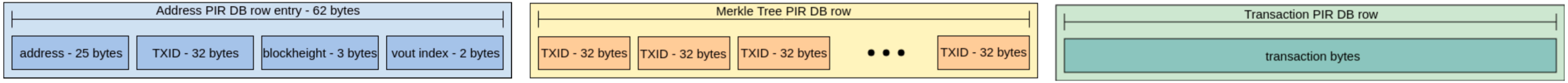
2) Download all Block Header



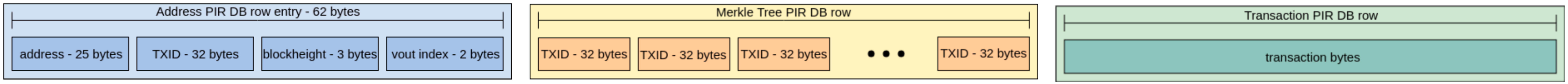
Imperial College London

PIR Protocol



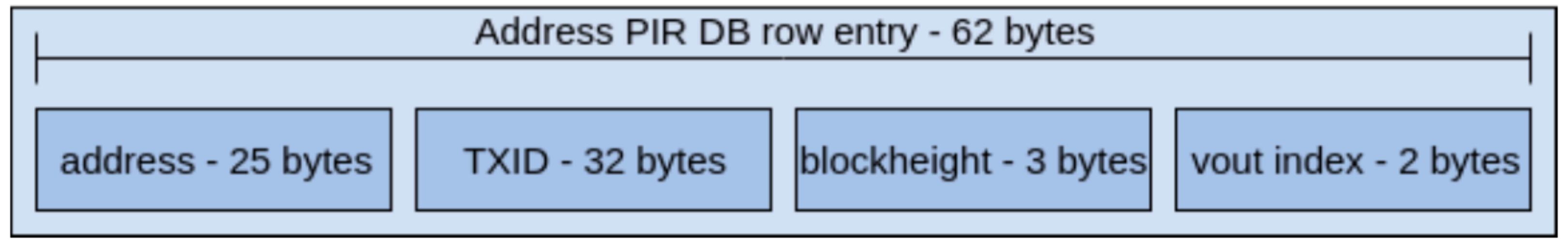


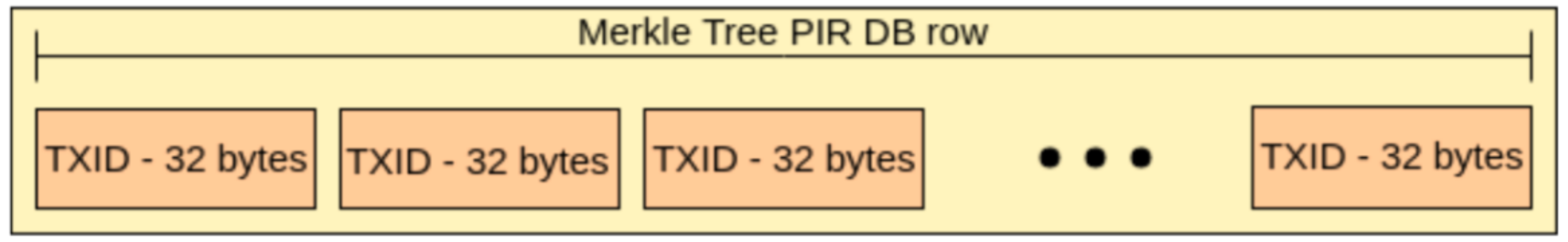
PIR Protocol

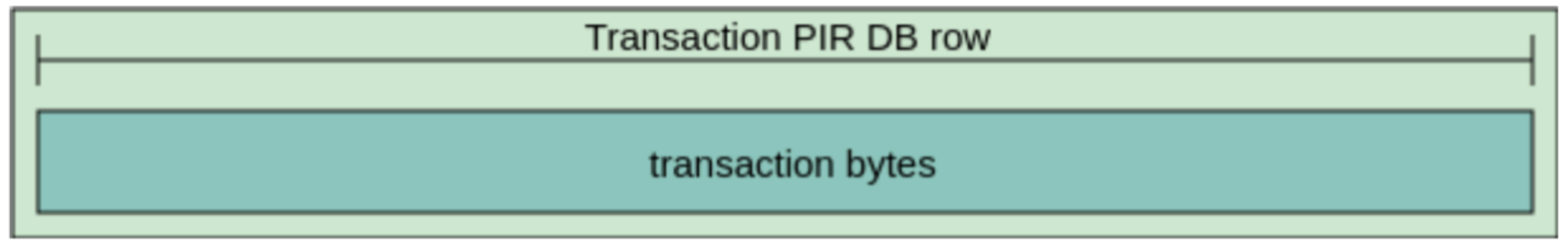


Choose @

PIR Protocol







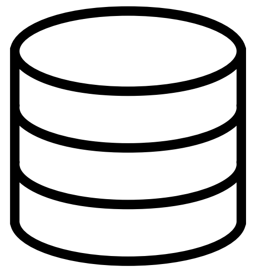
Choose @

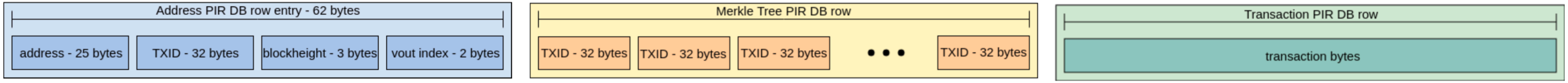
Address PIR Manifest

Merkle Tree PIR Manifest

Transaction PIR Manifest

PIR Protocol





Choose @

Choose @ →

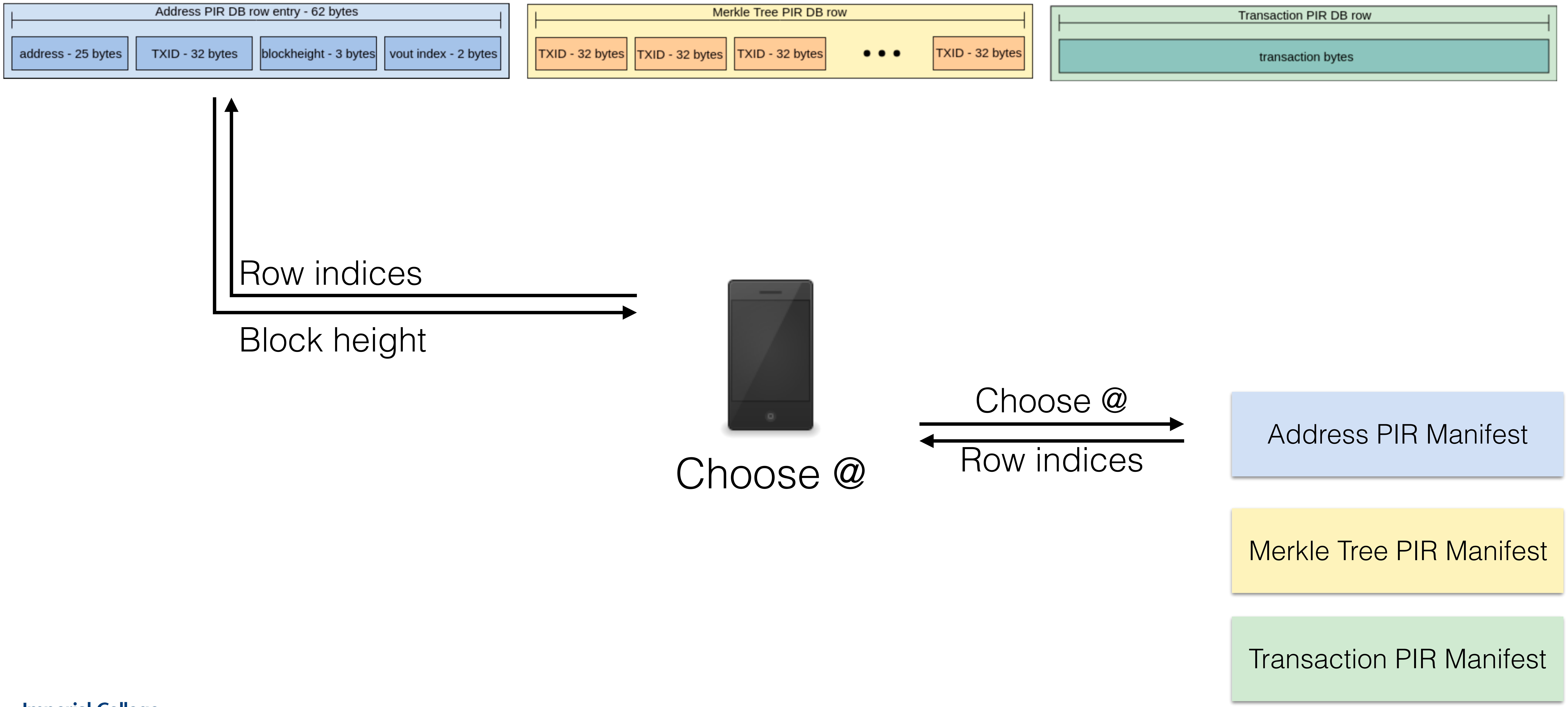
← Row indices

Address PIR Manifest

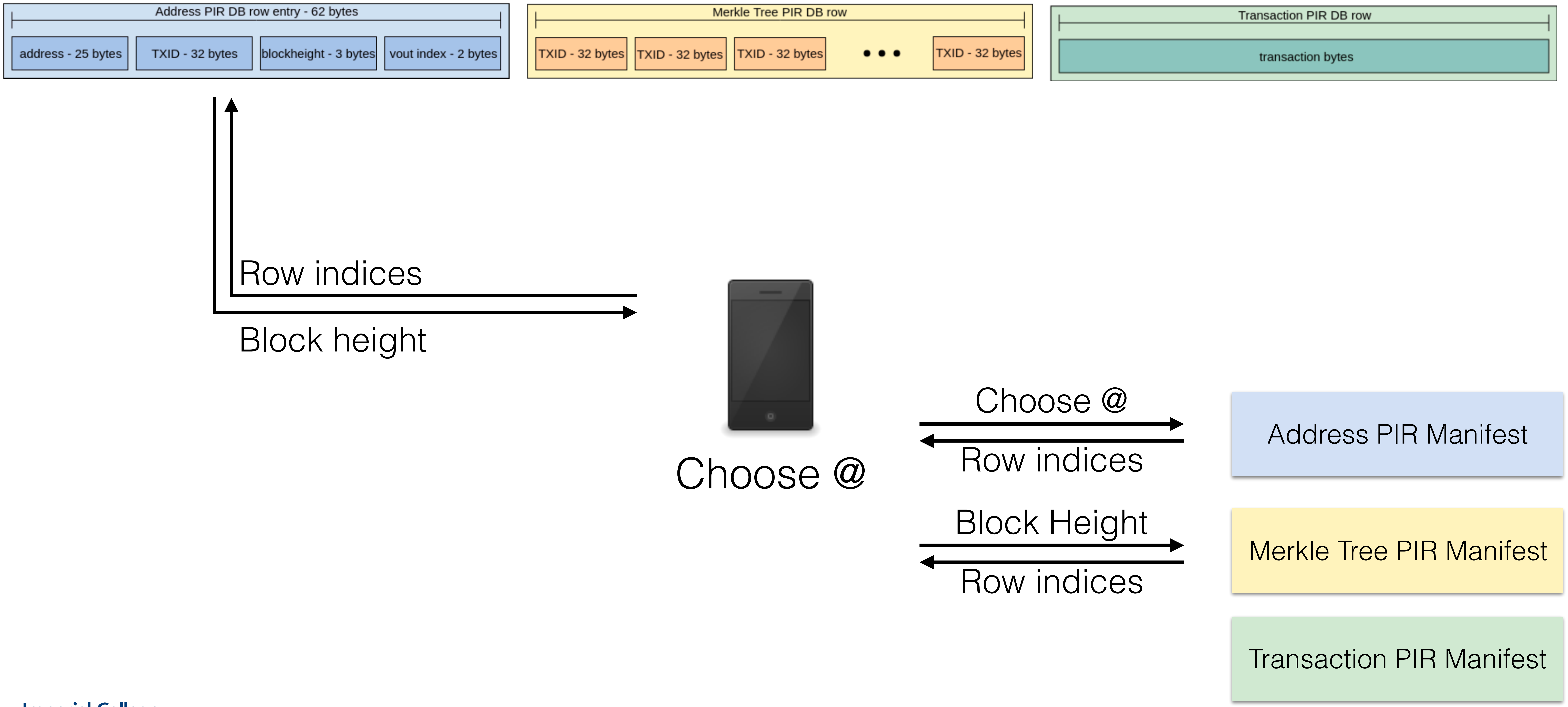
Merkle Tree PIR Manifest

Transaction PIR Manifest

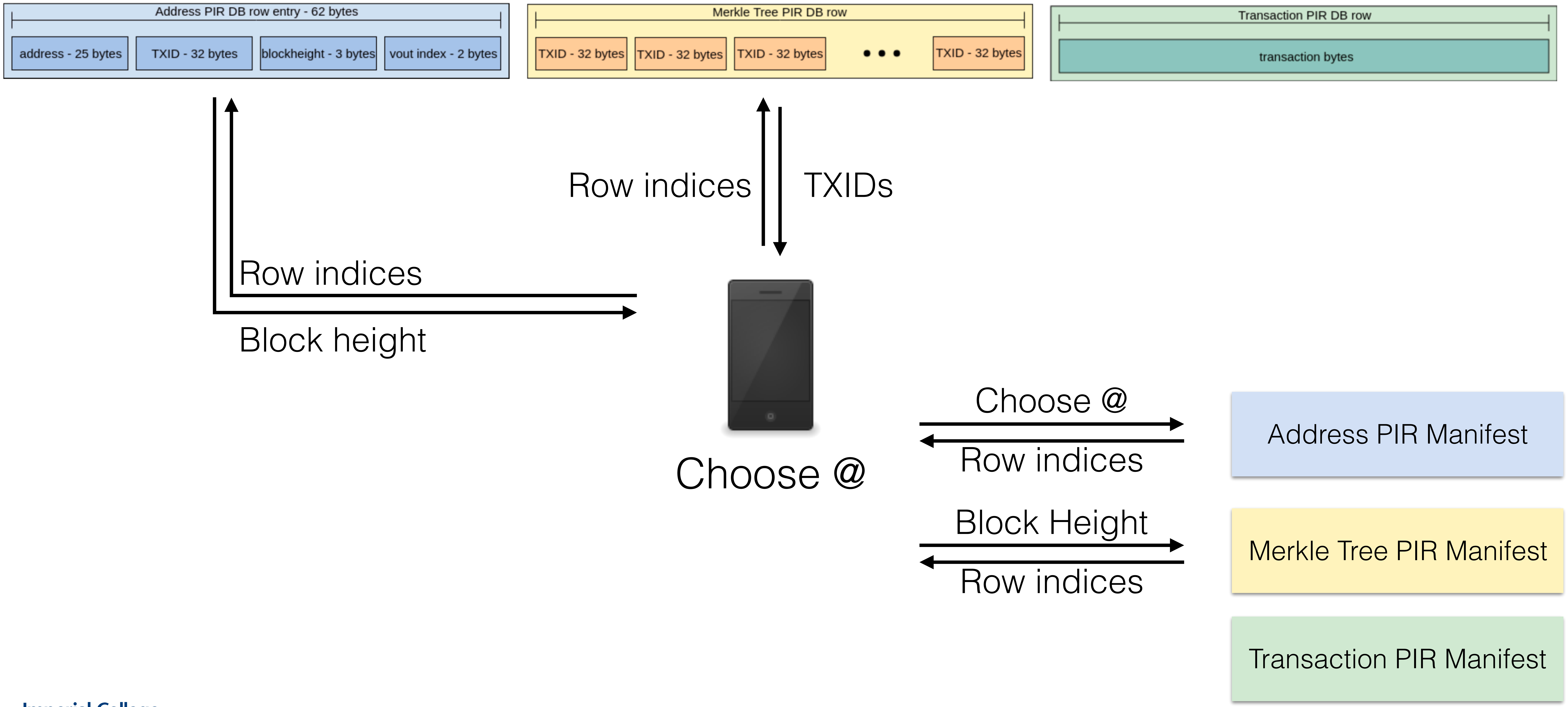
PIR Protocol



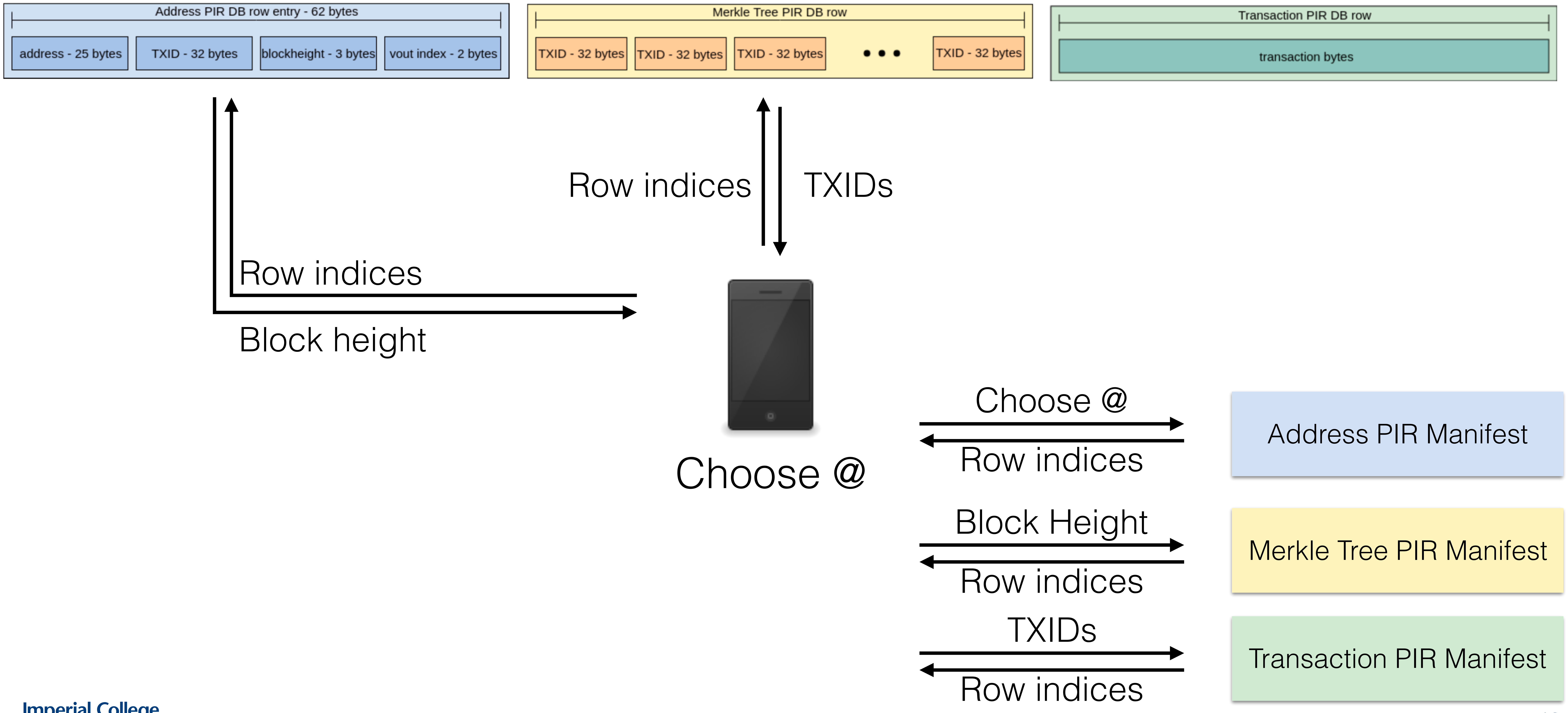
PIR Protocol



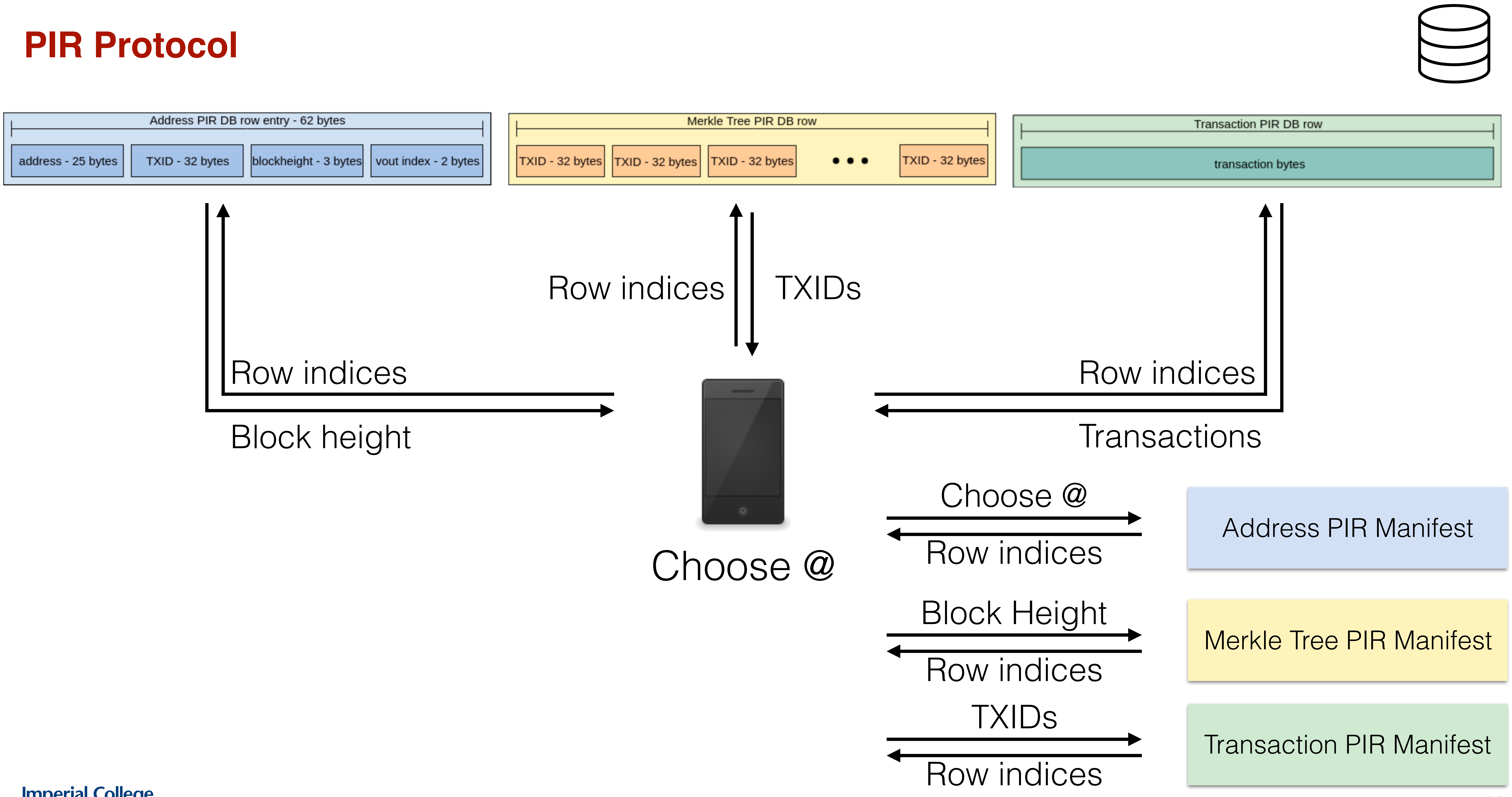
PIR Protocol



PIR Protocol



PIR Protocol



PIR Protocol

PIR Protocol

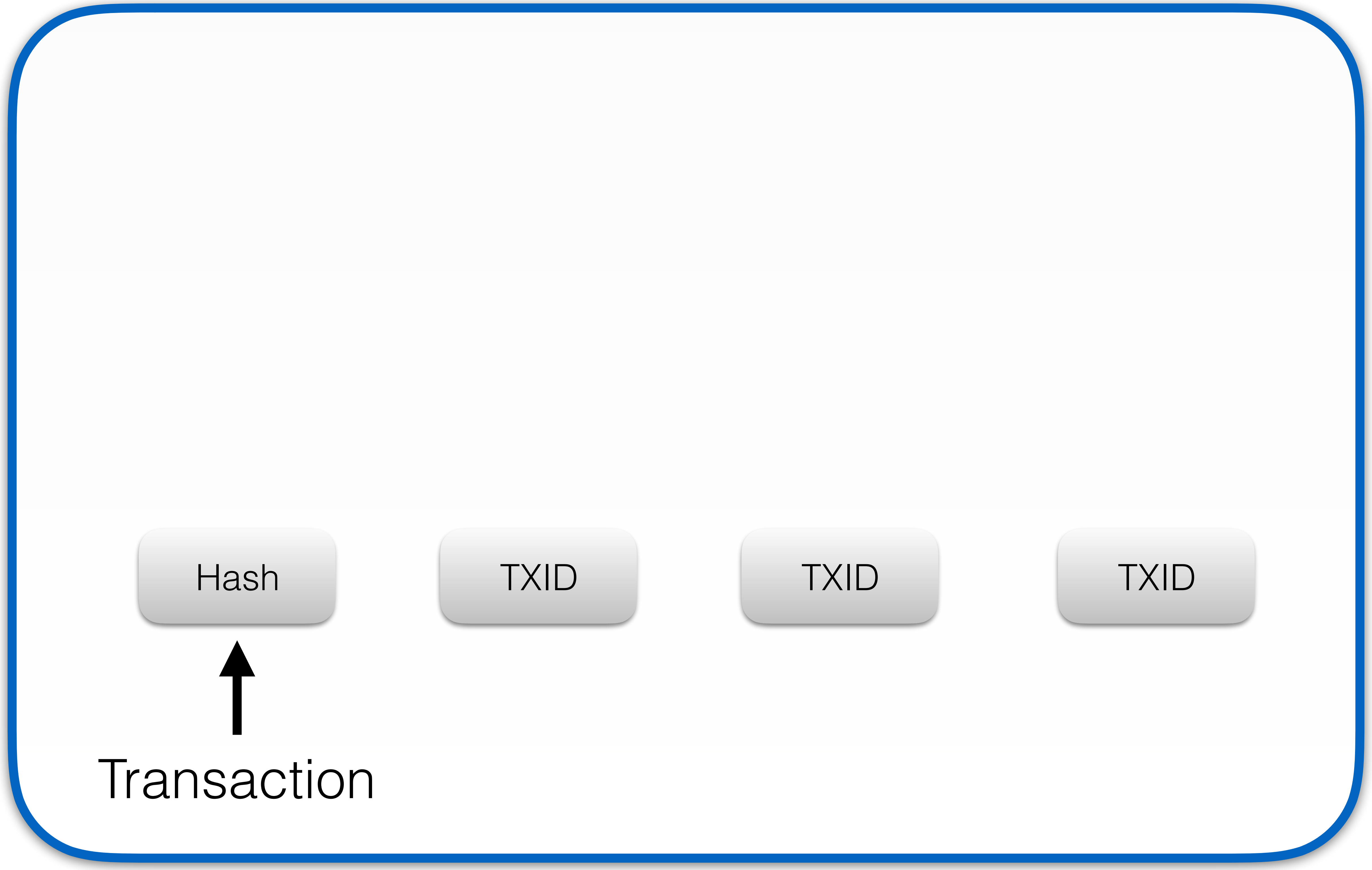
We have:

- Transaction IDs
- Block Headers
- Our raw transactions

PIR Protocol

We have:

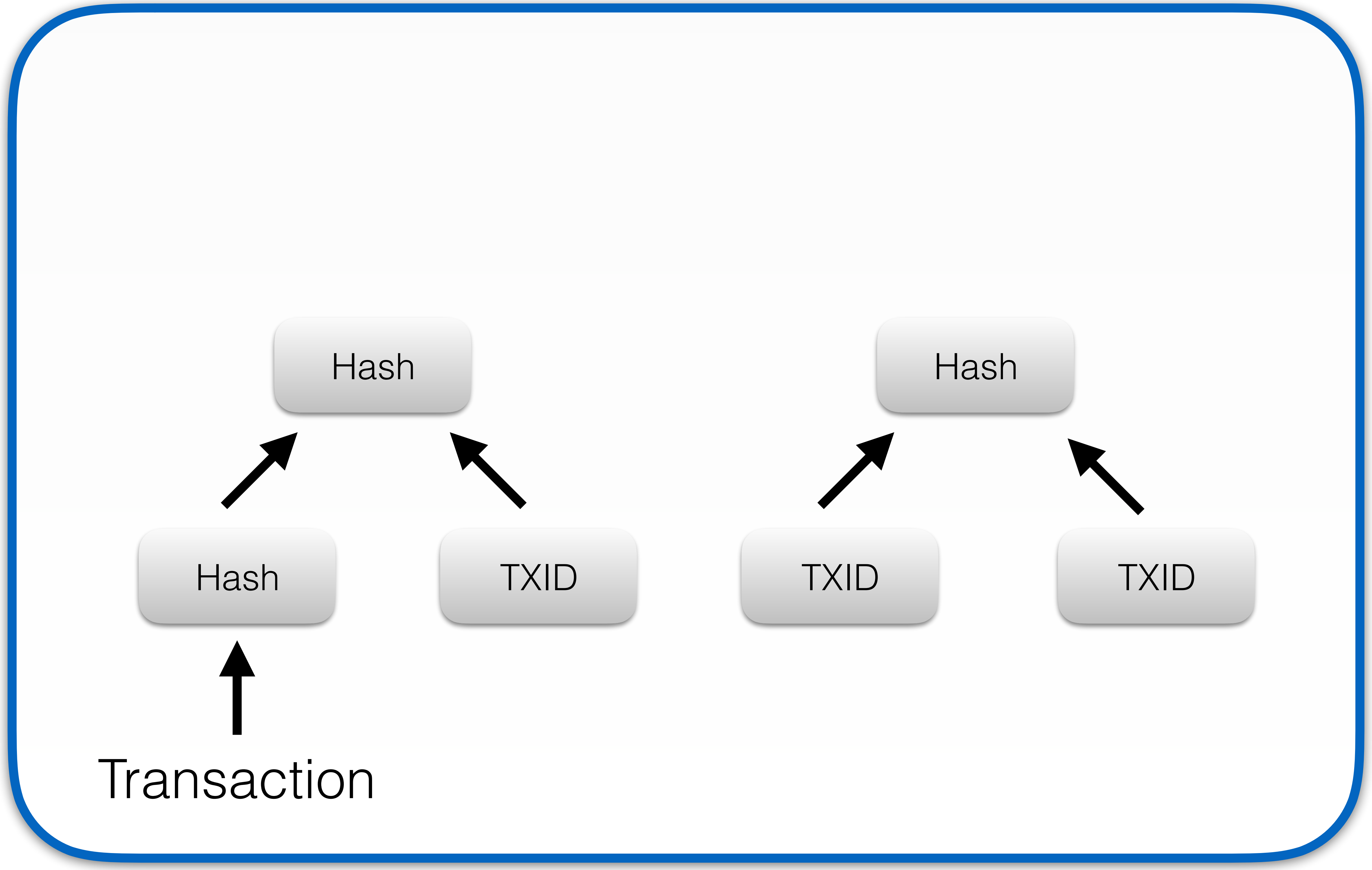
- Transaction IDs
- Block Headers
- Our raw transactions



PIR Protocol

We have:

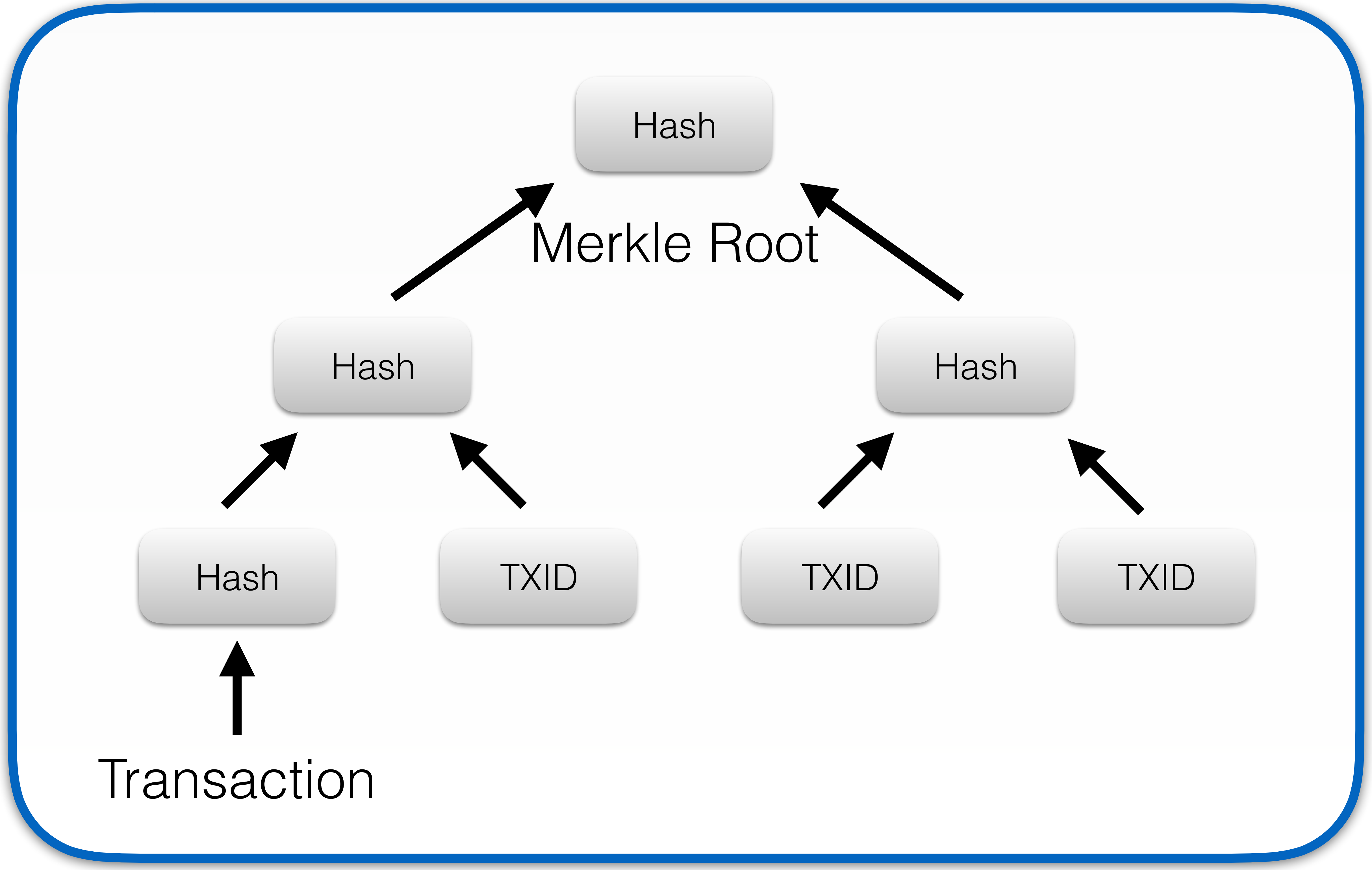
- Transaction IDs
- Block Headers
- Our raw transactions



PIR Protocol

We have:

- Transaction IDs
- Block Headers
- Our raw transactions



Costs - PIR Database sizes

Bitcoin Data — Block 1~ 513502 (March 2018)

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

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

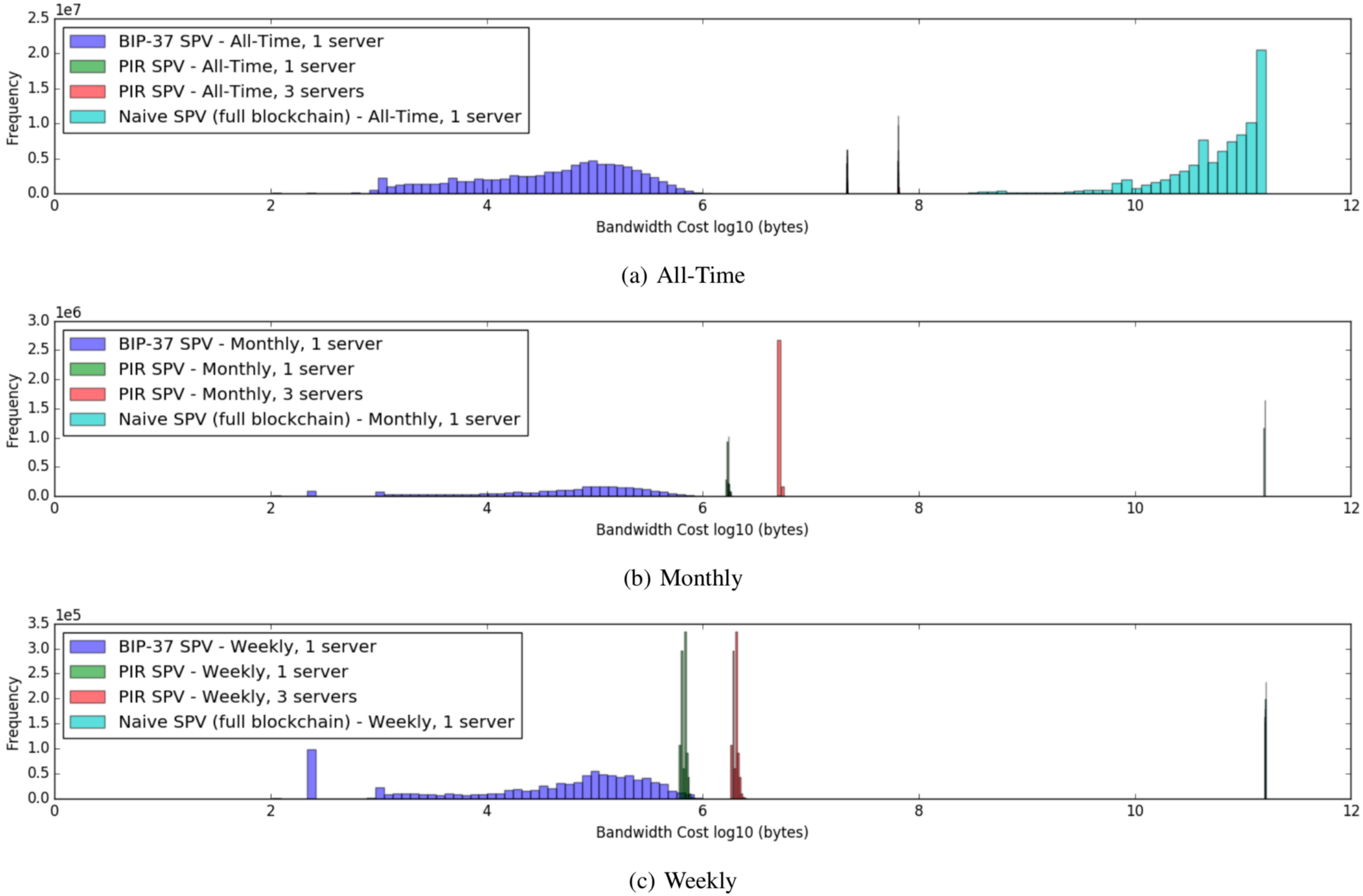
- In some cases, manifest files are very big
- Transform the simple manifest files into a format suitable for PIR queries.
- Clients perform interpolation search on these manifest files, under PIR.

Bandwidth comparison to verify a single transaction

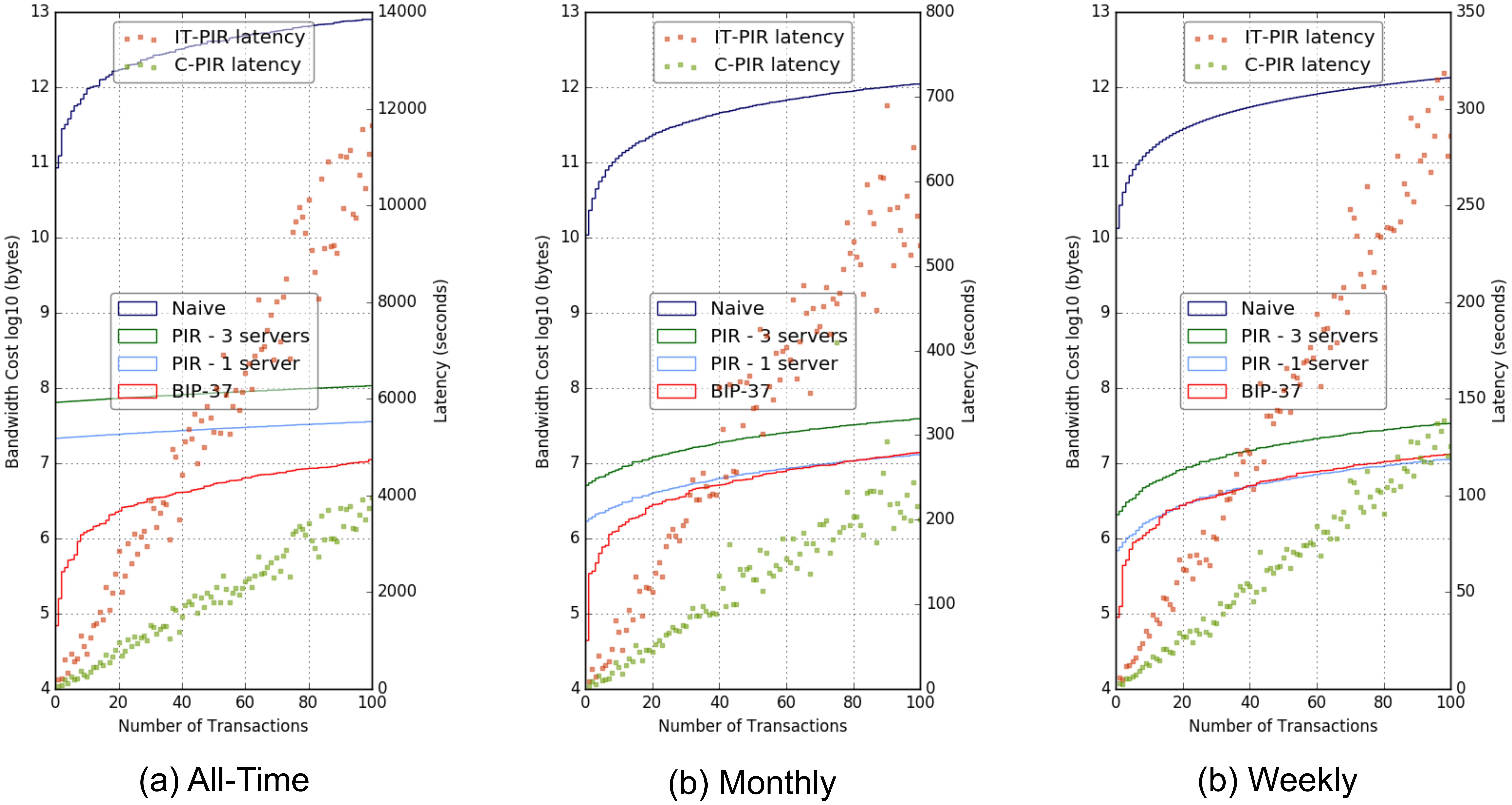
	BIP-37	PIR: 1 server	PIR: 3 servers	Naive SPV
All-Time	102.80 KB	21.54 MB	64.61 MB	80.27 GB
Monthly	132.43 KB	1.70 MB	5.11 MB	159.19 GB
Weekly	128.52 KB	666.07 KB	2.00 MB	161.47 GB

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



Imperial College London

Further Improvements

- Interpolative Search to skip manifest files 👀
- More fine granular separation of databases 🧐

Thank you!

Imperial College London

Manifest Files

- Format

```
{ …
        "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

Imperial College
London

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.