



Privacy Comparison for Bitcoin Light Client Implementations

Arad Kotzer  

The Department of Computer Science, Technion, Haifa, Israel

Ori Rottenstreich  

The Department of Computer Science and the Department of Electrical and Computer Engineering, Technion, Haifa, Israel

Abstract

Light clients implement a simple solution for Bitcoin’s scalability problem, as they do not store the entire blockchain but only the state of particular addresses of interest. To be able to keep track of the updated state of their addresses, light clients rely on full nodes to provide them with the required information. To do so, they must reveal information about the addresses they are interested in. This paper studies the two most common light client implementations, SPV and Neutrino with regards to their privacy. We define privacy metrics for comparing the privacy of the different implementations. We evaluate and compare the privacy of the implementations over time on real Bitcoin data and discuss the inherent privacy-communication tradeoff. In addition, we propose general techniques to enhance light client privacy in the existing implementations. Finally, we propose a new SPV-based light client model, the aggregation model, evaluate it, and show it can achieve enhanced privacy than in the existing light client implementations.

2012 ACM Subject Classification Networks → Network measurement

Keywords and phrases Blockchain, Privacy, Light Clients, Bloom filter

Digital Object Identifier 10.4230/LIPIcs.AFT.2024.15

1 Introduction

Blockchain networks like Bitcoin [41] are composed of a decentralized blockchain, structured as an immutable chain of data blocks. Starting from the first block, each block includes the output of a cryptographic hash function computed over the content of the previous block, making it impossible to alter a block without changing all subsequent blocks. Blockchain blocks are lists of transactions bundled together due to the high cost of the consensus protocol. Blocks are composed of two main components: block header and transactions. The block header stores only metadata, a hash of the previous block and the root of the Merkle tree of the block transactions [36]. The increasing amount of memory required to maintain the full Bitcoin state together with rapid growth in the volume of transactions that must be processed imply a large overhead on full Bitcoin nodes. Hence, not all nodes participating in a blockchain store and process the entire blockchain.

A light client is a node variant that can verify only part of a block, without locally maintaining the complete network state. While *full nodes* process the entire block (both header and transactions), *light clients* process only partial block information. Light clients are connected to full nodes and receive relevant information through them. To do so, light clients reveal in various forms information about the addresses of their interest [23]. This paper focuses on two common light client implementations SPV [41] and Neutrino [44], covering the different approaches used by most existing light client solutions.



© Arad Kotzer and Ori Rottenstreich;
licensed under Creative Commons License CC-BY 4.0
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 15; pp. 15:1–15:23

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

15:2 Privacy Comparison for Bitcoin Light Client Implementations

Our main contributions are as follows:

- We overview the main existing light client implementations: SPV and Neutrino.
- We define privacy metrics for measuring the privacy of the implementations.
- We perform both theoretical and empirical analysis, and compare the privacy of the different light clients over time on real data.
- We discuss how light clients may improve their privacy.
- We present a new model that further improves light client privacy.

We structure the paper as follows. Section 2 overviews methods for set representation, in addition to previous work on light client privacy. Section 3 presents the paper’s threat model and Section 4 describes metrics used to measure the privacy of light clients. Next, we provide a theoretical analysis of the privacy of SPV and Neutrino in Section 5. In subsections 6.1-6.3 we conduct an empirical analysis of the privacy of SPV and Neutrino, based on real data. Subsection 6.4 presents the privacy-communication tradeoff and compare the different implementations. These results are further discussed in Subsection 6.5. Then, we propose a new light client model that improves privacy in Section 7. Section 8 concludes the paper.

2 Background

2.1 Memory-efficient Methods for Set Representation

Bloom Filters. The Bloom filter [10, 13] is a popular data structure widely used in many networking device algorithms [12, 34], in fields as diverse as packet classification, routing, filtering, caching, and accounting, as well as beyond networking in areas like verification and spell checking. The Bloom filter is used for set representation, supporting element insertion and answering membership queries. There are two kinds of errors in membership queries: a false positive (when an element $x \notin U$ is reported as a member of a represented set U) and a false negative (when an element $x \in U$ is reported as a nonmember of U). The Bloom filter encounters false positives and has no false negatives. It is built as an array of bits, where hash functions map elements to bits in the array. With initial values of zero bits, the elements of U are first inserted into the filter, setting the bits pointed by the hash functions. Upon a query, bits mapped by the queried element are examined and a positive answer is returned only when the bits are all set.

Golomb-Coded Set (GCS). *GCS* is a data structure similar to Bloom filters, though it has a more compact in-memory representation that comes at the expense of having a slower query time (compared to Bloom filters) [24]. Given a hash function, N the number of items to be inserted into the set and an expansion parameter M , GCS works as follows:

- (i) Hash all items using hashing function H to integers in the range $[0, N \cdot M)$.
- (ii) Sort hashed values (in ascending order).
- (iii) Calculate the differences between each value and the previous one.
- (iv) Write the differences sequentially, compressed with Golomb coding [24].

Similar to Bloom filters, GCS is a probabilistic data structure that may contain false positives as a tradeoff with the memory size. Assuming the hashing function H has a uniform distribution, elements that were not inserted into the set have a probability of $\frac{N \cdot 1}{N \cdot M} = \frac{1}{M}$ to have the same hash value of an element in the set (after step (i)), and thus to appear in the set. Small M values reduce the GCS size but increase the false positive rate. Table 1 summarizes the main notations of the paper.

■ **Table 1** Summary of main notations.

| Symbol | Meaning | relevant to |
|-----------------|---|--------------------|
| A | address space | SPV, Neutrino |
| $ c $ | number of addresses associated with client c | SPV, Neutrino |
| p_i | probability that an address i belongs to c | SPV, Neutrino |
| H | client entropy | SPV, Neutrino |
| R | detection ratio | SPV, Neutrino |
| X | number of addresses guessed correctly among $ c $ guessed addresses | SPV, Neutrino |
| E_c | expected number of correctly guessed addresses | SPV, Neutrino |
| x | a target probability sum | SPV, Neutrino |
| N | number of inserted elements in a set | Bloom filter, GCS |
| F | number of false positives in a set | Bloom filter (SPV) |
| M | expansion parameter | GCS (Neutrino) |
| p_{FP} | probability of a downloaded block being a false positive | Neutrino |
| K | number of addresses in a block | Neutrino |
| k | number of non-eliminated addresses in a block | Neutrino |
| A' | address space excluding all eliminated addresses | Neutrino |
| z | number of blocks an address appears in | Neutrino |
| p_i^z | probability that non-eliminated address i appearing in z blocks downloaded by client c is associated with c | Neutrino |
| G | number of groups client c participates in | Aggregation Model |
| S | number of clients in each group | Aggregation Model |
| l | group leader | Aggregation Model |
| $p_{colluding}$ | probability light clients might collude with a full node | Aggregation Model |

2.2 Light client implementations

Bitcoin Simplified Payment Verification (SPV). *SPV* clients were first suggested in Bitcoin’s original white paper [41]. Since light clients do not keep track of the entire network state but only of several addresses in their interest, to be familiar with the balance of these addresses, SPV clients request all relevant transactions from a full node. To preserve privacy regarding the addresses associated with each client, light clients do not send an explicit list of relevant addresses but send a filter containing these addresses implicitly. Among all transactions that appear in a block, a full node only forwards those transactions that match the SPV filter, potentially with some false positives (namely transactions beyond the interest of the SPV client). Together with the particular transactions of interest, the full node also provides Merkle-tree-based proofs, demonstrating their inclusion in the block. Once the SPV client receives and validates the transactions (using the Merkle proof), it updates its state according to the transactions.

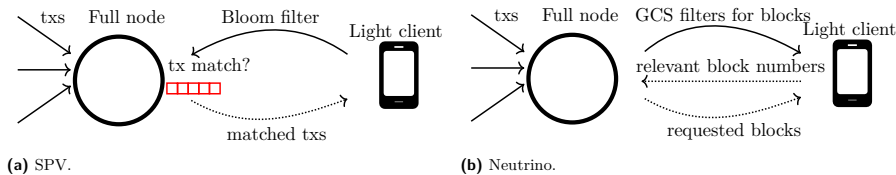
A *Bitcoin Improvement Proposal (BIP)* is a formal proposal to change Bitcoin suggested by the Bitcoin community (recall Bitcoin does not have one centralized leader). In BIP-37 [37], Bloom Filters were suggested as light client address filters. Using Bloom filters allows SPV clients to in-explicitly express the set of addresses they are interested in. The filter length can be selected based on a required false positive probability. Fig. 1a illustrates the process of an SPV client requesting transactions from a full node.

Though multiple Bitcoin light client implementations exist, most implementations, such as Electrum [51], Bither [47] and Mycelium [1], are all SPV-based. Moreover, [15] covers the main blockchain systems supporting light clients, and though these systems differ from each other, and accordingly the light client implementation they support, almost all of the light client implementations, including Binance light client [5], Cosmos - InterBlockchain

15:4 Privacy Comparison for Bitcoin Light Client Implementations

■ **Table 2** A high-level comparison between SPV and Neutrino.

| | SPV [37] | Neutrino [3] |
|---------------------------------|-----------------|--------------------|
| Set representation method | Bloom filter | Golomb-coded set |
| Data downloaded by light client | Transactions | Blocks |
| Who performs most computation? | Full node | Light client |
| Privacy achieved by | False positives | Downloading blocks |



■ **Figure 1** (a) Bitcoin SPV Clients: The light client reports its set of relevant addresses through a Bloom filter. (b) Neutrino Clients: A full node sends GCS (Golomb-Coded Sets) filters of the addresses in newly generated blocks, the light client reports relevant blocks it would like to download and the full node sends these blocks.

Communication [11] and ZCash’s Flyclient [14] light client, are all SPV-based or have a similar implementation to SPV. Ethereum light clients such as Helios [4], Kevlar [50] and Lodestar [6] use Bloom filters too for gathering information transactions from a full node. Cuckoo filter [22] based light clients [48] are similar too, having a client using a filter to request transactions when updating its state. The advantages of SPV clients are that they store locally only a small amount of data, in addition to very little data that is sent as part of the communication with the full node. Additionally, SPV clients perform minimal computations as the full node is the one to process newly generated blocks for finding relevant transactions. However, this light client implementation suffers from several drawbacks. First, SPV clients may observe low privacy as full nodes can infer what addresses are related to the SPV client. Additionally, since the heavy computation is performed on the full node, it is not very rewarding for the full node. This also makes full nodes vulnerable to DOS (Denial of service) attacks. To improve privacy, SPV clients use Bloom filters with a high rate of false positives to make it harder to infer what addresses are associated with the client.

Neutrino. To overcome the privacy drawbacks of SPV light clients, Neutrino clients suggest a different approach: Rather than requesting specific transactions from the full node, light clients download specific blocks, containing the relevant transactions. BIP-157 [44] implements this approach, using Client-Side Block Filtering. Whenever a new block is generated, full nodes create and broadcast a filter of all addresses that appear in the block. When a light client receives a filter, it checks if any of the addresses in the block are relevant. If so, the light client downloads the entire block from the full node and updates its state. BIP-158 [43] suggests using *Golomb-Coded Sets* (GCS) filters since a GCS filter is typically smaller than a Bloom filter with the same amount of elements inserted and the same false positive rate. *Neutrino* [3] is a light client implementation with Client-Side Block Filtering using GCS. Fig. 1b illustrates the process of updating the state of Neutrino clients. Since Neutrino light clients download the entire block and not specific transactions, full nodes have much less information regarding the light client’s addresses. Additionally, as full nodes with Neutrino implementations perform fewer computations than full nodes with SPV implementations, Neutrino full nodes are less vulnerable to DOS attacks. On the other hand, Neutrino requires

more computations on the light client's side when a client checks the relevancy of a received filter. Additionally, the network communication of Neutrino clients is higher than the SPV clients as Neutrino clients download the entire block and not only specific transactions. Moreover, as full nodes know what blocks were downloaded by clients, as we show in this paper Neutrino clients also have privacy concerns. Table 2 summarizes high-level differences between SPV and Neutrino clients.

2.3 Related Work

Security and privacy in Bitcoin have been widely discussed in the literature. Conti et al. [18] perform a comprehensive study on the security and privacy of Bitcoin, reviewing de-anonymization methods by analyzing blockchain data. [39] use a forensics perspective, analyzing Bitcoin wallets for iOS and Android, recovering information such as metadata, installation data, timestamps, and usage traces. Multiple papers [21, 9, 58, 40] try to cluster Bitcoin addresses. [32] link between fungibility and anonymity of cryptocurrencies and put forward a framework to measure the fungibility and anonymity of cryptocurrencies, using Shannon entropy. Additionally, [53] uses a Transaction Directed Acyclic Graph (TDAG) to capture blockchain privacy notions (PDAG) and compare Monero and Zcash, the two most prominent privacy-preserving blockchains. All of these papers though, do not focus on light clients. [31] provides a taxonomy of cryptocurrency wallets, including light clients such as SPV and Neutrino, evaluating their performance and security. The privacy evaluation is given at a high level without formalized privacy or empirical evaluation.

The privacy and anonymity of Bitcoin light clients have been discussed widely [23, 29, 57, 8]. The first to analyze and formalize the privacy issues of light clients using Bloom filters was a study by Geravis [23]. Later, in [26], the BIP-37 proposers who implemented Bloom filters in SPV clients expanded on these privacy issues and discussed the difficulties of solving them. [29] further continue the analysis of [23]. Let N be the total number of addresses inserted in a Bloom filter and F the number of false positives. [23] show the probability of an address with a Bloom filter positive indication actually belonging to c is $p_i = \frac{N}{N+F}$. Accordingly, the probability of guessing j addresses with a positive Bloom filter indication and having them all associated with c is $\prod_{i=0}^{j-1} \frac{N-i}{N+F-i}$. We note that in our paper, we base our initial analysis on these probabilities. Similarly, [29] presents a metric called γ -deniability. They refer to a Bloom filter member $x \in S$ as deniable if for $i \in \{1, \dots, n\}$ there is a nonmember y_i such that $Hash(x) = Hash(y_i)$. Then, a Bloom filter is γ -deniable if an address is deniable with probability γ . That work indicates that privacy is affected not only by the false positive rate of the Bloom filter but also by the number of real addresses. They describe a method for estimating the number of active addresses through a linear regression model. A similar observation about the efficiency of Bloom filters was described in a more general context [46]. Later, [27] provides an evaluation of the privacy of SPV clients using multiple bloom filters with the γ -deniability metric, in addition to an entropy measurement of the Bloom filters. Unlike SPV, the privacy of Neutrino clients is not discussed much as it is generally considered much higher than SPV [45].

To provide better privacy than SPV, several approaches were suggested, such as Neutrino light clients [3], PIR-based light clients and [16, 45, 56] and using trusted execution environment-based light clients [35, 42, 49, 54], and In this paper, we do not focus on the latter approach since it requires suitable special hardware which makes it unusable for most current Bitcoin users, hence it is rarely implemented in practice.

3 Threat Model

In blockchain systems, user privacy is a major concern. However, the relation between Bitcoin transactions and addresses can be used to analyze Bitcoin’s privacy information, seriously jeopardizing Bitcoin anonymity [59]. An adversary may find an association between Bitcoin transactions and addresses using address clustering, further associating groups of addresses with the same entity [25, 28, 59, 52, 33]. Moreover, although the recommendation in Bitcoin is to generate a new address for each transaction, as this results in a large overhead of generating and managing addresses, most Bitcoin users do not generate a new address each time. Similarly to previous work [8, 23, 29, 57], in this paper, the goal of an attacker is to *reveal what addresses typically belong to the wallet of light client c* . This allows the attacker to track every transaction performed by c , and keep track of c ’s financial status. Likewise, we assume the light clients are connected to the Bitcoin P2P network, and receive information regarding the network transaction through full Bitcoin nodes. That attacker gains information from the filters used by c (the Bloom filters for SPV clients and the blocks downloaded for Neutrino clients). Moreover, we assume all relevant filters can be tracked to the same IP address (thus the adversary knows what filters are related).

We argue this model currently represents a real-life scenario. First, for an SPV client c all the attacker needs is a filter of the addresses of c , which is sent to any full node c communicates with. As for Neutrino clients, often a Neutrino client continues to communicate with the same full nodes over and over, allowing them to gain information regarding all of the blocks c is interested in. c communicates with the same nodes for several reasons: First, as the process of full node seeking is DNS-similar and might be time-costly, light clients keep a cache of full nodes they discovered, and communicate with cached full nodes rather than search for new full nodes every time. Additionally, as this threat model is passive, c is not aware of the attacker’s gain of information. An attacker might act as a fast and reliable node, encouraging c to continue using it rather than search for other full nodes. Moreover, if c disconnects from the network and later rejoins, to reconstruct the previous state c might request all relevant blocks at once from the same full node. In addition, Neutrino clients are encouraged to communicate with multiple full nodes and validate the consistency of the data received to lower the risk of data leakage attacks [35]. On top of that, c might communicate with colluding full nodes, sharing information. Hence, Neutrino clients are at risk of having a full node with information regarding all of the blocks c was interested in.

4 Privacy Metrics

In this section, we present light client privacy evaluation metrics. Previous papers [23, 29] have already analyzed the privacy of SPV clients, though they did not use privacy evaluation methods that can be compared to other light client implementations but SPV. We expand the privacy analysis of SPV clients and present different privacy metrics that can be used to analyze the privacy of Neutrino clients as well. These metrics also allow us to compare the privacy of SPV and Neutrino clients. Although Neutrino clients are thought to have much higher privacy since they download full blocks rather than specific transactions [3, 57, 31], as we show in this paper over time Neutrino clients may suffer from severe privacy issues too. To the best of our knowledge, we are the first to measure the privacy of Neutrino clients. Throughout the paper, the *privacy* of light client c refers to the situation where the specific details of the account addresses of c should be hidden from external parties. p_i denotes the probability that an address i is among the addresses of interest of client c . Denote by A the blockchain address space such that $c \subseteq A$. In Section 5 we show how to evaluate p_i for each address in both SPV and Neutrino clients. We present the two following privacy metrics:

(i) Light Client Entropy. Entropy is a metric used to measure the uncertainty or disorder of a dataset or a random variable. The higher the uncertainty regarding each element in the dataset, the higher the entropy gets. When all addresses have the same probability, $p_i = \frac{|c|}{|A|}$, it is the most difficult to distinguish between addresses associated with c and other addresses. Additionally, when some addresses have a higher probability compared to others, there is less uncertainty regarding which addresses belong to c . Assuming the sum of probabilities p_i of all addresses in A is $|c|$ (as c has $|c|$ addresses), the number of addresses needed to cover some probability sum of $x < |c|$ can indicate the uncertainty of the network. Hence, we define $T(x)$ as the minimal number of addresses needed to cover some probability sum x . This minimal number of addresses can be derived based on the addresses with the highest probabilities. On the one hand, a wide range of x values gives a better indication. On the other hand, lower x values distinguish addresses better (for instance $x = |c|$ will always return the number of addresses with $p_i \neq 0$, which is less informative). Hence Definition 1, presenting the *light client entropy* of some light client c , sums $T(x \cdot |c|)$ for x values in the range $[0, 1]$, and gives higher weights for lower x values. Finally, a \ln operation is applied for convenience to scale the entropy (and is not mandatory). We note that addresses that are necessarily not associated with c (satisfying $p_i = 0$) decrease the value of entropy H as they often lead to higher probabilities of other addresses being associated with c , thus decreasing $T(x \cdot |c|)$. Moreover, it is easy to see that the minimal entropy value is achieved when there are $|c|$ addresses with probability $p_i = 1$ (namely, when all addresses associated with c are known), as $T(x \cdot |c|)$ returns the minimal value possible for all x values.

Our definition of entropy differs from the classic Shannon entropy equation, $Entropy = -\sum_j p(j) \cdot \log p(j)$. The entropy values are maximized when the p_i values get closer to 0.5. As p_i describes the probability of an address being associated with c , and since there are many more addresses that are not associated with c (namely, $|c| \ll |A| - |c|$), lower p_i values, rather values closer to 0.5, are values indicating higher uncertainty. Hence, the classic Shannon entropy equation (while considering the probabilities for addresses in the address space) is less suitable for measuring uncertainty regarding c 's addresses.

► **Definition 1.** The *light client entropy* of light client c is defined as

$$H(c) = \ln \int_0^1 (1-x) \cdot T(x \cdot |c|) dx$$

where $T(x \cdot |c|)$ is the minimal number of addresses needed to cover a probability sum of $x \cdot |c|$.

(ii) Detection Ratio. As the number of addresses of c an adversary can guess correctly is a privacy concern, the detection ratio metric measures the ratio between the number of correctly guessed addresses and the total number of guesses. Assuming an adversary guesses $|c|$ addresses, E_c denotes the expected number of correctly guessed addresses.

► **Definition 2.** The *detection ratio* $R(c)$ of client c is defined as $R(c) = \frac{1}{|c|} \cdot E_c$, assuming out of $|c|$ addresses guessed, the expected number of correctly guessed addresses is E_c .

Intuitively, for lower values of the detection ratio $R(c)$ there is less certainty about which addresses are associated with c , and the privacy of c is higher. We present a simple lemma related to the detection ratio.

► **Lemma 3.** The expected number of correctly guessed addresses E_c equals the sum of probabilities of guessed addresses: $E_c = \sum_{i \in \Omega} p_i$, where Ω is the set of guessed addresses.

Proof. Assume an adversary guesses a set Ω of $|\Omega| = |c|$ addresses. Now, let X be the number of addresses in Ω that were guessed correctly. Let X_i be an indicator for a guessed address $i \in \Omega$ to be indeed in c . In that case, $E_c = E[X] = E[\sum_{i=1}^n X_i] = \sum_{i=1}^n E[X_i]$. Since the expectation on an indicator i is equal to the probability that $X_i = 1$, i.e. $E[X_i] = P(X_i = 1)$, then $E_c = \sum_{i \in \Omega} p_i$ equals the sum of the probabilities of the guessed addresses. ◀

By Lemma 3, the value E_c equals the sum of the probabilities of the guessed addresses. We note the minimal detection ratio is achieved when each address $i \in A$ has the probability $p_i = \frac{|c|}{|A|}$ of being associated with c . In such a case, the detection ratio is $R(c) = \frac{|c|}{|A|}$.

We note that while entropy measures the uncertainty on the entire network and is affected by all addresses, the detection ratio measures the more prominent addresses and is affected mainly by the high-probability addresses in the network.

5 Privacy Theoretical Analysis

5.1 SPV privacy

When evaluating the privacy of SPV clients, the information regarding the addresses of client c is inferred from the Bloom filter c creates. As most SPV-based implementations use a constant filter for a long time, the privacy of SPV clients does not change over time since the information regarding the addresses in c is inferred from the filter alone. We start with a simple property regarding addresses that do not appear in the filter:

► **Property 4.** *Addresses with a negative indication in a filter are not associated with c .*

Property 4 states that the probability of every address with a negative indication being related to c is $p_i = 0$ since c inserts all of its addresses to the filter that has no false negatives. The probability of address i with a positive indication belonging to c was analyzed in [23], and depends on the number of addresses of c and the number of false positives F : $p_i = \frac{|c|}{|c|+F}$. We take the analysis of SPV privacy a step forward, trying to numerically evaluate its privacy in a way that could be compared to other light client implementations. Lemma 5 evaluates the entropy of light client c , showing $H(c) = \ln \frac{|c|+F}{6}$. Lemma 6 shows the number of addresses guessed correctly out of guessing $|c|$ addresses of an SPV client is hypergeometric distributed with parameters $(N, |c|, |c|)$, allowing Lemma 7 to evaluate the detection ratio of light client c as $R(c) = \frac{|c|}{|c|+F}$.

► **Lemma 5.** *For an SPV client with $|c|$ addresses and a Bloom filter containing F false positives, the light client entropy value is $H(c) = \ln \frac{|c|+F}{6}$.*

Proof. We first note that there are $|c| + F$ addresses with a positive filter indication, all with probability $p_i = \frac{|c|}{|c|+F}$ of being associated with c . By property 4, all other addresses in address space A have a probability of $p_i = 0$. Next, we note that there are $x \cdot (|c| + F)$ addresses needed to achieve a probability sum of $x \cdot |c|$, hence for $0 \leq x \leq 1$, $T(x \cdot |c|) = x \cdot (|c| + F)$. Thus, $H(c) = \ln \int_0^1 (1-x) \cdot T(x \cdot |c|) dx = \ln \int_0^1 (1-x) \cdot x \cdot (|c| + F) dx = \ln \frac{|c|+F}{6}$. ◀

► **Lemma 6.** *Let X be the number of addresses guessed correctly out of guessing $|c|$ addresses of an SPV client (all guesses are of addresses with a positive Bloom filter indication). X has a hypergeometric distribution with parameters $(N, |c|, |c|)$.*

Proof. Recall SPV clients create a Bloom filter with a positive indication for $N = |c| + F$ addresses, out of them $|c|$ indeed belong to c . X is the number of addresses that belong to client c that are guessed correctly. The guessing order does not matter, and each guess reduces the address space to guess from (since $|c|$ different addresses are guessed). We note this case

is identical to the classic hypergeometric distribution problem: Given a bin with N balls, out of them $n = |c|$ are black and the rest are white. Let X count the number of black balls drawn (with no returning the balls and with no meaning to the drawing order). Since there are $|c|$ black balls, the maximal number of black balls drawn is $D = |c|$. Therefore, X has a hypergeometric distribution with parameters (N, D, n) , that is $X \sim HG(|c| + F, |c|, |c|)$. ◀

► **Lemma 7.** *For an SPV client with $|c|$ addresses and a Bloom filter containing F false positives, the light client detection ratio is $R(c) = \frac{|c|}{|c|+F}$.*

Proof. By Lemma 6 $X \sim HG(N, |c|, |c|)$. Hence, the expected number of correctly guessed addresses is $E_c = \frac{|c|^2}{|c|+F}$. Therefore, by the definition of the light client detection ratio $R(c) = E_c \cdot \frac{1}{|c|} = \frac{|c|^2}{|c|+F} \cdot \frac{1}{|c|} = \frac{|c|}{|c|+F}$. ◀

5.2 Neutrino privacy

As Neutrino clients receive the GCS (Golomb-coded set) filter from a full node and do not download specific transactions but full blocks, it seems like there is much less information that can be inferred by a full node hence Neutrino clients' privacy is generally considered much higher than of SPV [45]. Matetic et al. [35] show that a Neutrino client c might be at risk of an attack revealing information on its addresses if it receives GCS block filters from a single entity, though privacy is considered high when c communicates with multiple servers (full nodes) and validates the data is consistent between the servers. *This motivates Neutrino clients to request the same information from multiple servers.* We show there yet exists a major privacy issue if some server knows the exact blocks c downloaded. Additionally, in contrast to SPV light clients, we show the privacy of Neutrino clients decreases over time. To the best of our knowledge, we are the first to address and formalize this problem. We assume there exists a server with information about what exact blocks c downloaded and can thus infer what blocks were not downloaded too.

Recall GCS filters have false positives, possibly making clients download blocks without addresses associated with them. Given the probability for the block containing at least one address the client is L , $|c|$ is the number of addresses associated with client c and an expansion parameter M for the GCS filter, Lemma 9 presents the probability of this block being false positive for c . Upon assuming that in address space A all addresses have the same probability to appear in a block, L can be evaluated as shown in Lemma 8.

► **Lemma 8.** *Assume each block contains $K > |c|$ addresses, selected uniformly at random from the address space A . The probability L for the block to contain at least one address of client c is $L = 1 - \binom{|A|-|c|}{K} / \binom{|A|}{K}$.*

Proof. Given a block, let Y be the number of addresses associated with client c that appear in the block. As all addresses have the same probability to appear in the block, when choosing K addresses (out of them $|c|$ addresses are associated with c) for the block the maximal number of addresses associated with c that may appear in a block is $\min\{|c|, K\} = |c|$. Hence, similarly to X in Lemma 6, Y has a hypergeometric distribution, i.e. $Y \sim HG(|A|, |c|, K)$. Thus $P(Y = 0) = \binom{D}{0} \cdot \binom{N-D}{n-0} / \binom{N}{n} = \binom{|A|-|c|}{K} / \binom{|A|}{K}$ and $L = 1 - P(Y = 0) = 1 - \binom{|A|-|c|}{K} / \binom{|A|}{K}$. ◀

► **Lemma 9.** *Consider a Neutrino client c with $|c|$ addresses and let M be the expansion parameter of the GCS filter representation of a block. Let L be the probability of a block containing at least one address of c . The probability of a block being a false positive block (i.e. being downloaded by c without actually containing any address of c) is*

$$p_{FP} = (1 - L) \cdot \left(1 - \left(1 - \frac{1}{M}\right)^{|c|}\right) = \binom{|A| - |c|}{K} / \binom{|A|}{K} \cdot \left(1 - \left(1 - \frac{1}{M}\right)^{|c|}\right).$$

15:10 Privacy Comparison for Bitcoin Light Client Implementations

Proof. For a block to be a false positive for client c , two conditions must be met:

- (i) The block does not contain any address of c (event B).
- (ii) Client c has at least one address with a positive filter indication (event D).

As L is the probability that at least one address associated with c will appear in the block, the probability for the first condition to be met is $P(B) = 1 - L$. We compute the probability for event D given that event B holds. Recall the probability of a non-associated address to have a positive filter indication is $\frac{1}{M}$, implying that the probability of a non-associated address having a false filter indication is $1 - \frac{1}{M}$. Since client c has $|c|$ addresses, the probability that all of addresses in c have a false filter indication is $(1 - \frac{1}{M})^{|c|}$ and at least one address has a positive filter indication with probability $P(D|B) = (1 - (1 - \frac{1}{M})^{|c|})$. Thus, $p_{FP} = P(B) \cdot P(D|B) = (1 - L) \cdot (1 - (1 - \frac{1}{M})^{|c|})$. ◀

Given that a block was downloaded by a client and is not a false positive, the probability of guessing a single address is $p_i = \frac{1}{K}$ where K is the number of addresses in a block. Since an address appearing on a block may belong to c only if the block is not a false positive, Property 10 is intuitive:

▶ **Property 10.** *Given a block with K addresses that was downloaded by c , with one address that belongs to c and with no other knowledge, the probability of each address being associated with c is $p = (1 - p_{FP}) \cdot \frac{1}{K}$.*

Similarly to Property 4, a full node can infer that addresses in blocks not downloaded by c are not associated with it, as stated in Property 11. For every such address, the probability of being associated with c is $p = 0$. An address in a block downloaded by c that also appears in an earlier block not downloaded by c has a probability of $p = 0$.

▶ **Property 11.** *Addresses appearing in blocks the light client did not download are not associated with the client.*

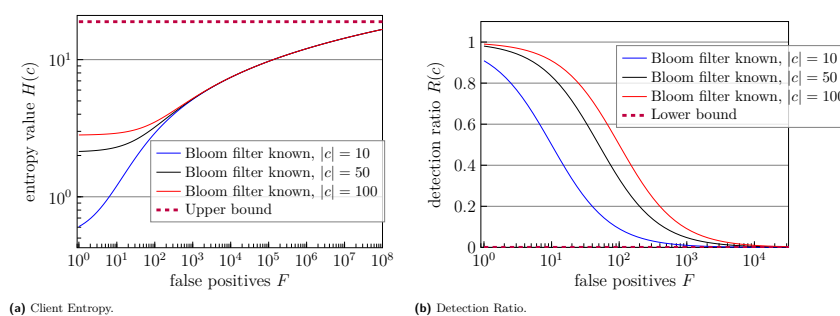
▶ **Definition 12.** *For some light client c and a block B downloaded by c , **non-eliminated addresses** are addresses that appear in B and do not appear in any earlier block that was not downloaded by c .*

Therefore, Property 13 states that only the amount of *non-eliminated addresses* k should be considered when evaluating p_i .

▶ **Property 13.** *Given a block downloaded by c , with one address that belongs to c , knowing the block contains k non-eliminated addresses, the probability of each address being associated with c is $p = (1 - p_{FP}) \cdot \frac{1}{k}$.*

6 Light Client Privacy Measurement

After presenting the privacy analysis of each implementation, we now evaluate and compare the analysis of the different implementations. We examine the main parameters affecting the privacy of each implementation and later empirically evaluate the privacy based on real Bitcoin data. As Bitcoin uses the RIPEMD-160 [20] hash function, the number of possible addresses is 2^{160} . However, in practice, the Bitcoin blockchain consists of about a billion addresses as of April 2023. Hence, the address space is of size $|A| = 10^9$. If only the number of addresses $|c|$ is known, $p_i = \frac{|c|}{|A|}$. As all addresses have the same p_i , this case is equivalent to having a filter using $F = |A| - |c|$ addresses, and by Lemmas 5 and 7 the entropy is



■ **Figure 2** (a) The entropy of an SPV client c compared to the number of positives. The graph also contains the entropy values of C when the Bloom filter is unknown. (b) The number of addresses and the detection ratio compared to the number of positives. For convenience, the detection ratio is presented as a percentage (scaled by 100). Both graphs are calculated for light clients with having $|c| = 10, 50$ and 100 addresses.

$H(c) = \ln \frac{|A|-|c|+|c|}{6} = \ln \frac{10^9}{6} = 18.93$, and the detection ratio is $R(c) = \frac{1}{\lfloor 10^9 \rfloor}$. We now show that by receiving additional information from the client, like a Bloom filter (SPV) client and what blocks c downloaded (Neutrino) the privacy decreases.

6.1 Data

To evaluate the privacy of the different implementations on real-life data, we downloaded all mined Bitcoin blocks during April 2023. A total of 4161 blocks were mined, averaging 137 mined blocks per day. Though the maximal size of a Bitcoin block is 4 MB, the average block size was around 3.2 MB. Since achieving information regarding real-life wallets and the addresses associated with them is not an easy task (as Bitcoin wallets are private), to simulate a Neutrino client wallet, we sampled random addresses appearing in the blocks mined on April 1st for wallets of size 10, 50 and 100. In each analysis, the address sampling and privacy measuring were performed 100 times to neutralize noises. As there is no simple formula for calculating the entropy of a Neutrino client, to evaluate the entropy of Neutrino clients in Section 6.3, we sorted all of the probabilities in the network, implemented $T(x)$ and approximated the entropy value using calculating $H(c) \approx \ln \frac{1}{1000} \cdot \sum_{x \in [0.001, 0.002, \dots, 1]} (1-x) \cdot T(x \cdot |c|)$. To assure uniformity between all entropy measurements, SPV entropy was calculated using both the formula presented in Lemma 5 (derived from Definition 1) and by implementing $T(x)$ and calculating the value of $\ln \frac{1}{1000} \cdot \sum_{x \in [0.001, 0.002, \dots, 1]} (1-x) \cdot T(x \cdot |c|)$. For all SPV entropy experiments, the entropy values of both calculations were at least 99.999% similar, showing the sampling of the entropy evaluation of $\ln \frac{1}{1000} \cdot \sum_{x \in [0.001, 0.002, \dots, 1]} (1-x) \cdot T(x \cdot |c|)$ provides a close enough approximation.

6.2 SPV Measurement

For SPV light clients the main parameter affecting privacy is the number of positives F in the Bloom filter. Fig. 2a presents the entropy of client c compared to the number of positives, for different values of $|c|$. The figure additionally contains an upper bound of the privacy metrics which is achieved when the adversary does not have the client Bloom filter. The larger F is, the higher the entropy is. For instance, for a client with $|c| = 50$ addresses, $F = 104$ false positives imply an entropy of $H(c) = 3.25$. The entropy increases to $H(c) = 7.51$ for a larger amount of false positives, $F = 10^5$. Up to some point (in our case around $F = 40 \cdot 10^3$), an entropy increase can be achieved not only by increasing F , but also by increasing the

number of addresses used by c . For instance, for $|c| = 10$ addresses and $F = 500$ the entropy is equal to $H(c) = 4.52$, while for the same amount of false positives, the entropy increases to $H(c) = 4.69$ if c uses $|c| = 100$ addresses instead. This is because as entropy measures the uncertainty in the network, it is affected by all addresses, and the more addresses with $p_i \neq 0$ there are, the higher entropy gets. More specifically, for SPV clients it is very easy to see from Lemma 5 how the number of addresses in the filter affects entropy $H(c)$, hence for larger $|c|$ values the filter contains more addresses and $H(c)$ increases accordingly. However, from around $F = 40 \cdot 10^3$ all three sizes of $|c|$ achieve similar entropy, as the size of c becomes quite negligible compared to F .

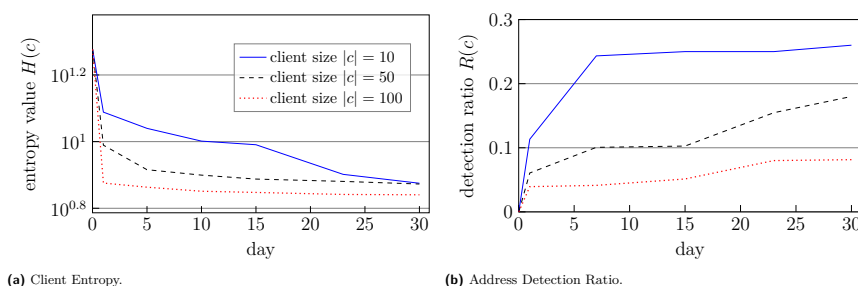
Fig. 2b presents the address detection ratio of client c compared to the number of positives. As expected, the detection ratio decreases as F increases. Though, unlike the entropy, more addresses associated with c result in a higher detection ratio: Having $F = 10$ false positives, the detection ratio is $R(c) = 0.5$ if c uses $|c| = 10$ addresses, namely 50% of the guessed addresses are associated with c . For $|c| = 100$ the detection ratio is $R(c) = 0.91$. To achieve a detection ratio of $R(c) = 0.5$ an amount of $F = 100$ false positives is needed. This is because the detection ratio is affected mainly by the probability p_i of each address, and for the same amount of false positives used in a filter, higher c values have a higher percentage of addresses that belong to c in the filter, hence probability p_i for address increases and as a result $R(c)$ increases too. Similar to the entropy, when using around $F = 40 \cdot 10^3$ false positives, $|c|$ becomes relatively negligible compared to F , in addition to a very low detection ratio, hence the differences between the different $|c|$ values become very small.

6.3 Neutrino Measurement

For Neutrino light clients, privacy is affected mainly by the block size, the amount of non-eliminated addresses k in each block, the number of blocks downloaded by c and the number of blocks each address appears in. Since the privacy of Neutrino clients depends on various parameters and the probability of an address being associated with c varies between the different addresses, Neutrino privacy evaluation is more difficult than SPV clients. We now show that although Neutrino clients are considered to have high privacy, they may suffer privacy issues over time.

Following the analysis in Section 5.2, we analyze the probability of an address appearing in z blocks being associated with c . Intuitively, when the false-positive ratio is low, the more blocks an address appears in the higher the chances it is associated with c . Assuming (for simplicity) c is interested in (at most) one address from each downloaded block, since in a non false-positive block the probability of a non-eliminated address not being associated with client c is $p_i = \frac{k-1}{k}$ (for a false positive block $p_i = 0$). This is assuming independence between blocks, meaning the probability that an address that appears in z blocks is not associated with c is $P = \prod_{i=1}^z \frac{k_i-1}{k_i}$, where k_i is the number of non-eliminated addresses and 1 is the number of addresses associated with c in a non false-positive block. Hence, if all blocks are non false-positive the probability that the address is associated with c is $p = 1 - \prod_{i=1}^z \frac{k_i-1}{k_i}$. Since the probability for z blocks not to be false positive is $P = (1 - p_{FP})^z$, the probability of a non-eliminated address appearing in z blocks downloaded by c being associated with c is $p_i^z(c) = (1 - p_{FP})^z \cdot \left(1 - \prod_{i=1}^z \frac{k_i-1}{k_i}\right)$.

Fig. 3 presents the average entropy and detection ratio over time. We consider probability $p = 0$ for eliminated addresses, and for each non-eliminated address $i \in A'$ that did not appear yet in a block, we consider probability $p = \frac{|c|}{|A'|}$, where A' is the address space excluding all eliminated addresses. When evaluating the entropy, all address probabilities were normalized to ensure the sum of probabilities equals c .



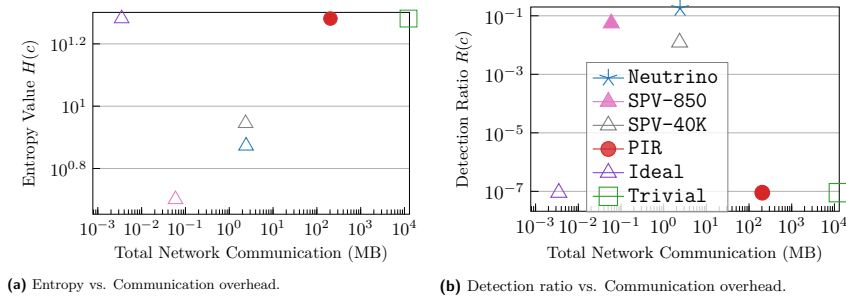
■ **Figure 3** (a) Entropy of a Neutrino client c over a month. (b) Detection ratio of a Neutrino client c over a month. Both graphs were calculated for client size of $|c| = 10, 50$ and 100 addresses.

Fig. 3a shows the more blocks are mined the more information there is regarding the addresses of c and the entropy decreases. For instance, while for $|c| = 50$ the entropy was $H(c) = 9.75$ at the end of the first day (after 139 mined blocks), by the 30th day the entropy decreased to $H(c) = 7.46$. This indicates that the more blocks are mined the more certainty there is regarding what addresses belong to c . We additionally notice there is a large drop in entropy on the first day. This is because before a full node has any information on c , all addresses are candidates of being associated with c . After a day, many addresses become eliminated addresses hence $|A'|$ decreases by much, and as the entropy is affected by all addresses, eliminating that many addresses decreases the entropy by much. Moreover, we see the slopes of the graphs become milder over time. This is because over time there are fewer and fewer addresses that are eliminated, hence the entropy decreases slower over time.

The detection ratio evaluation shows how the detection ratio increases over time too, indicating a privacy decrease. As shown in Fig. 3b, after one day the detection ratio for $|c| = 10$ was $R(c) = 0.11$, meaning an adversary could identify correctly an average of 11% of the addresses of c . By the 30th day, the detection ratio increased to $R(c) = 0.26$. Unlike the entropy, there is a difference in the detection ratio for different $|c|$ values, as the more addresses c uses the lower the chances of guessing correctly the addresses that belong to c . For instance, on the 30th day with $|c| = 100$ the detection ratio is only $R(c) = 0.08$, more than three times lower than $|c| = 10$. Moreover, the graphs show that for each day, for larger c values the entropy and detection ratio have lower values, whereas for smaller c values they increase. This is because larger c values result in downloading more blocks. The main addresses that contribute to the entropy values are the non-eliminated addresses that did not appear in any block. Hence, as for larger c values more blocks are downloaded, fewer addresses have not appeared yet in a block and thus the entropy is relatively larger. As the detection ratio is mostly affected by unique addresses that appear in blocks c downloaded more than others, when more blocks are downloaded the higher the chances of addresses appearing, thus reducing the chances of unique addresses that significantly appear more than other addresses. To conclude, for all c values the entropy decreases over time while the detection ratio increases, both indicating a privacy decrease over time in Neutrino.

6.4 Privacy and Communication Overhead

We now analyze the privacy and network communication of SPV and Neutrino. Additionally, we compare this analysis to a PIR light client, which provides maximal privacy. *Private Information Retrieval* (PIR) protocols, introduced by [17], allow clients to query a server and retrieve data from the server's database without revealing information regarding the data the client was interested in. There have been several suggestions for light client implementations using PIR [55, 56, 45]. We note that PIR implementations require a change in the current



■ **Figure 4** Privacy compared to communication overhead comparison between different light client implementations, for a client with $|c| = 50$ addresses. (a) Entropy vs. communication overhead. (b) Detection ratio vs. communication overhead.

Bitcoin full node protocol, in addition to much higher increased computational requirements on both the server and clients. As light clients should stay light (usually running on devices with small computation power), they are not yet popular as a light client solution. Hence, we do not focus on PIR client implementations and use them as a solution providing ideal privacy while improving the trivial solution of downloading the entire blockchain. We analyze the PIR-based light client protocol presented in [45], provided in Percy++ [2], an open-source library. Since we assume the privacy of PIR-based clients is ideal, the probability p_i of each address for PIR implementations is $p_i = \frac{|c|}{|A|}$ with $|A|$ addresses that appear in the blockchain.

In our analysis, we generate clients with $|c| = 50$ random addresses similarly to Section 6.3. We note the communication overhead of the filters used by SPV and Neutrino clients is negligible compared to the downloaded transactions' data. Fig. 4 presents the privacy of the different implementations, in addition to the *Trivial* (downloading the entire blockchain) and the *Ideal* implementations (achieving maximal privacy with downloading minimal data), compared to the communication overhead after 30 days. For the *SPV* implementation, we used two filter sizes with $F = 850$ and $F = 40 \cdot 10^3$ false positives, denoted as *SPV-850* and *SPV-40K*, respectively. We observe in Fig. 4a that the *Ideal*, *Trivial* and *PIR* solutions reach an entropy of $H(c) = 18.91$. While *Trivial* requires downloading the entire blockchain of size 12.3 GB, *PIR* reduces this overhead to 204 MB. An ideal solution, though, would require only 0.004 MB. *SPV-40K* achieves an entropy of $H(c) = 8.81$ for only 2.36 MB, and *Neutrino* achieves an entropy of $H(c) = 7.46$, higher only than *SPV-850*, having an overhead of 2.4 MB. The lowest entropy implementation was *SPV-850* with an entropy of $H(c) = 5.02$, though having a small overhead of 0.06 MB. Fig. 4b presents the detection ratio of each solution. As we have already seen in Fig. 3, the detection ratio of *Neutrino* increases over time, making it the highest detection ratio implementation, with $R(c) = 0.18$ after 30 days. *SPV-850* and *SPV-40K* achieve a detection ratio of $R(c) = 0.055$ and 0.012 , respectively, while *PIR* has a detection ratio of approximately zero.

6.5 Discussion: Additional Insights from the Measurement and Analysis

We now compare the privacy of the light client implementations, suggesting insights to improve privacy in the existing light client implementations.

6.5.1 Privacy Comparison

Subsection 6.4 shows that in general, there is a privacy and communication overhead tradeoff for light clients. For example, *PIR* provides better privacy than *SPV* and *Neutrino*, though with a much higher communication overhead. That said, several implementations are

comparable to others and some light clients perform better than others. Our analysis shows that, unlike the common conception that Neutrino clients preserve more privacy compared to SPV, under our threat model, *SPV clients provide better privacy* than Neutrino: While SPV-40K has the same communication overhead as Neutrino, it provides a higher entropy value and a detection ratio almost 15 times lower than Neutrino clients. SPV clients have an additional advantage over other implementations, as there is only one main parameter that determines privacy and communication overhead- the number of false positives F . This allows SPV clients the opportunity to easily tune F based on what is more important for their use- privacy or low network communication. Similarly, PIR light clients achieve similar privacy as Trivial, having a much lower communication overhead, and can thus perform better than SPV in the extreme case c wants maximal privacy.

6.5.2 Improving Light Client Privacy

Both SPV and Neutrino clients can improve their privacy by increasing the false-positive rate of their filters. This results in additional transactions downloaded in SPV, and additional blocks downloaded for Neutrino clients. We now suggest other efficient techniques to improve light client privacy.

Neutrino. Recall by Section 6.3 that the probability p_i of address i being associated with c mainly depends on two parameters: z , the number of blocks an address appears in, and k , the number of non-eliminated addresses. Increasing the number of addresses used by c and thus decreasing z for many addresses will result in higher entropy and lower detection ratio, indicating privacy improvement without any communication increase. Additionally, we suggest methods for decreasing the chances of our threat model occurring. Recall Neutrino clients are motivated to request information from multiple servers to reduce changes of some privacy attacks [35]. We suggest Neutrino light clients should both *limit themselves to a non-large amount of full nodes* and additionally *divide the blocks download between multiple servers* rather than querying the same full nodes every time, especially when rejoining the network. Moreover, we suggest Neutrino clients should keep track of the full nodes they approached for information, and try being as diverse as possible when choosing a full node to communicate with. In addition, occasionally clearing the full node cache will help avoid reaching the same full nodes every time.

SPV. The main parameter determining the privacy of SPV clients is the number of false positives, F . Hence, besides increasing F , section 7 suggests an SPV-based light client model, that improves privacy for a similar communication overhead.

7 Proposal: The Aggregation Model (AM)

7.1 Overview

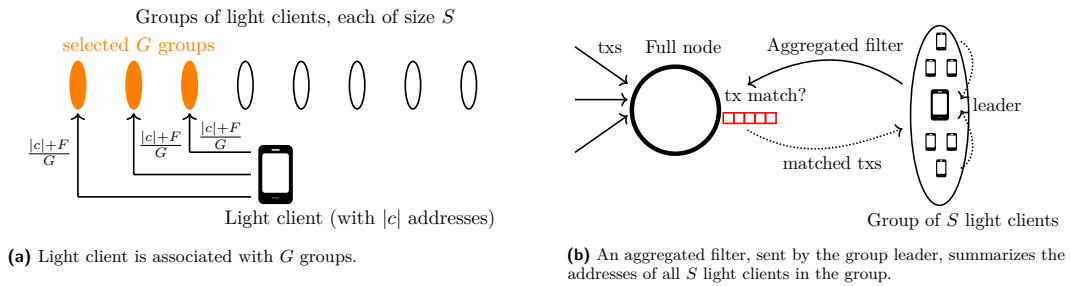
As existing light client implementations suffer from privacy issues, we suggest a new SPV-based light client model, *the aggregation model (AM)*, to potentially improve the privacy of light clients, for a similar communication overhead. In short, we suggest light clients should group up with other light clients, create an aggregated filter for all the clients and have one client representative that communicates with the full node. We suggest light client c should join G groups (each of size s light clients), equally split its addresses between these groups and additionally add some false positives to each group filter.

15:16 Privacy Comparison for Bitcoin Light Client Implementations

We assume each light client c uses F false positives. Additionally, c joins G groups, each group of size S , meaning there are S light clients in each group. The following steps are performed when a light client wants to download transactions from a full node:

1. c creates G non-intersecting filters, composed of $\frac{|c|}{G}$ addresses of c and $\frac{F}{G}$ false positives.
2. c joins G groups, each of size S .
3. For each group, the following is performed:
 - a. A leader l is randomly chosen (this can be done using several distributed algorithms such as [7, 30, 38]).
 - b. All clients send l their (bloom) filters.
 - c. l creates an aggregated Bloom filter and sends it to a full node.
 - d. l receives transactions from the full node and sends each client its transactions (with their Merkle-proofs), as received from the full node.

The implementation is illustrated in Fig. 5.



■ **Figure 5** Illustration of the aggregation model (a) Light client is associated with G groups and sends $\frac{|c|+F}{G}$ addresses to each group. (b) An aggregated filter, sent by the group leader, summarizes the addresses of all S light clients in the group.

When a new Bitcoin node connects to the network, it queries several DNS servers (which are operated by volunteer nodes and provide a random selection of bootstrap nodes that are active in the Bitcoin network). Once connected, the joining node learns about other nodes by asking their neighbors for known addresses and listening for spontaneous advertisements of new addresses [19]. Hence, we suggest light clients advantage of this mechanism to form G groups of S clients each. We note the main advantage of the aggregation model is that only the leader exposes itself to a full node, providing anonymity for other clients. This way light clients receive transactions without any full node gathering information about non-leader clients. Additionally, even at the worst scenario, where all nodes in the groups c participates in collude with full nodes, using the aggregation model *the privacy of light client c is at least as high as the original SPV model*.

7.2 Privacy analysis

In the aggregated model, both the full nodes and the leader gain information regarding the light client addresses. As long as there are no colluding light clients, full nodes cannot infer anything about the group light clients besides information concerning the leader. Hence, assuming there are no colluding clients in the network, when evaluating the privacy of light client c , the addresses in address space A can be separated into two groups: addresses in filters of groups where c is the leader and all other addresses. We assume there are G groups that c participates in, each of size S , and all clients have the same amount of addresses $|c|$ and use the same amount of false positives F , equally split between G groups. Property 14 calculates the expected number of leader groups. For each such group, Property 15 evaluates

the number of addresses in the aggregated filter c creates and the probability p_i of each address in the filter being associated with c . As address space A is much larger than $|c|$, Property 16 evaluated the total number of addresses associated with c in groups where c is the leader, and Property 17 evaluates the probability p_i of each address that does not appear in a group c is the leader of being associated with c .

► **Property 14.** *Given there are G groups c participates in, each of size S . The expected number of groups c will be the leader of is $\frac{G}{S}$.*

► **Property 15.** *Given there are G groups c participates in, each of size S , assuming all clients have the same amount of addresses $|c|$ and use the same amount of false positives F , equally split between G groups. In this case, the group leader creates an aggregated filter containing $\frac{|c| \cdot S}{G} + \frac{F \cdot S}{G} = \frac{(|c|+F) \cdot S}{G}$ addresses. As the group leader l has $\frac{|c|}{G}$ addresses on the aggregated filter associated with him, the probability p_i of any address in the aggregated filter being associated with l is $\frac{|c|}{G} / \frac{(|c|+F) \cdot S}{G} = \frac{|c|}{(|c|+F) \cdot S}$.*

► **Property 16.** *Assume the addresses in c are equally split between G groups, each of size S . As c is the leader of $\frac{G}{S}$ groups (Property 14) and there are $\frac{|c|}{G}$ addresses associated with c in each group, the total number of addresses associated with c in groups c is the leader of is $\frac{G}{S} \cdot \frac{|c|}{G} = \frac{|c|}{S}$.*

► **Property 17.** *Assuming there are $|c'| \leq |c|$ addresses that appear in groups c is the leader of (c' is evaluated in Property 16), and assuming $|A| \gg |c|$, when there are no colluding light clients the probability p_i of an address that did not appear in a group c is the leader of is $\frac{|c'|}{|A|}$.*

We note that the previous analysis relies on the light client being honest. However, some light clients might be adversaries too, and collude with each other or with a full node. We denote by $p_{colluding}$ the percentage of colluding light clients in the network (or similarly the probability of a light client colluding). In such a case, the addresses in address space A can be separated into three groups: addresses in filters of groups where c is the leader, addresses in groups where c is not the leader and leader l is colluding, and all other addresses. Property 18 calculates the expected number of groups with a colluding leader, and Property 19 evaluates the probability p_i of all addresses used by c in this group. Similar to Property 17, assuming there are $|c'| < |c|$ addresses of c that appear in groups c is the leader of (evaluated in Property 16), and $|c''| < |c|$ addresses of c that appear in a group with a colluding leader (evaluated in Property 20), the probability p_i of any other address is $\frac{|c|-|c'|-|c''|}{A}$. We note we assume here the strictest colluding assumption, where some full node has the entire information gained by all colluding nodes.

► **Property 18.** *Given there are G groups c participates in, each of size S , and a colluding probability of $p_{colluding}$ for each node that is not c . The number of groups c participates in with a colluding leader is $p_{colluding} \cdot \frac{G \cdot (S-1)}{S}$.*

► **Property 19.** *Assume client c has $|c|$ addresses and uses a total of F false positives, equally split between G groups. For each group that c is not the leader of, c sends a filter to the leader. The probability p_i of each address in the filter being associated with $|c|$ is $p_i = \frac{|c|}{G} / \frac{|c|+F}{G} = \frac{|c|}{|c|+F}$.*

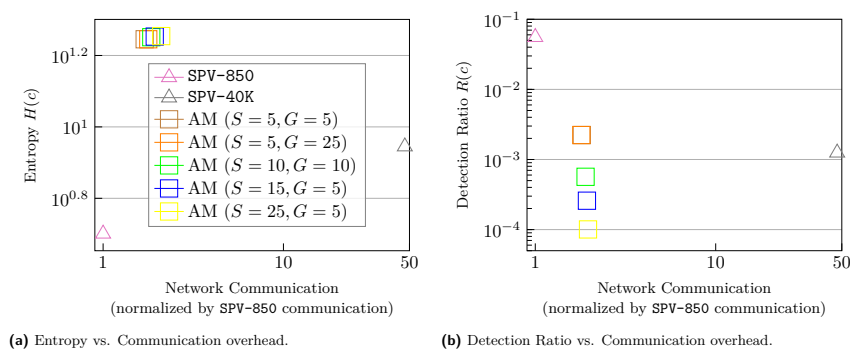
► **Property 20.** *Assume the addresses in c are equally split between G groups, each of size S . As by Property 18 the number of groups c is not the leader of is $G \cdot \frac{S-1}{S}$, each having $\frac{|c|}{G}$ addresses associated with c . With a probability of $p_{colluding}$ of the leader colluding, the number of addresses associated with c in groups where c is not the leader and there is a colluding leader is $p_{colluding} \cdot G \cdot \frac{|c|}{G} \cdot \frac{S-1}{S} = p_{colluding} \cdot |c| \cdot \frac{S-1}{S}$.*

► **Property 21.** *When all nodes communicating with c are colluding, by Property 19 the probability p_i of each address used in a group c is not the leader of is $p_i = \frac{|c|}{|c|+F}$. Additionally, for groups c is the leader of, as all addresses that do not belong to c are known to the leader, the probability of each address being associated with c is $p_i = \frac{|c|}{G} / \frac{|c|+F}{G} = \frac{|c|}{|c|+F}$. Thus, as c uses $|c| + F$ addresses in total each with probability $\frac{|c|}{|c|+F}$ being associated with c , the privacy when all nodes collude equals the privacy of the original SPV model using F false positives.*

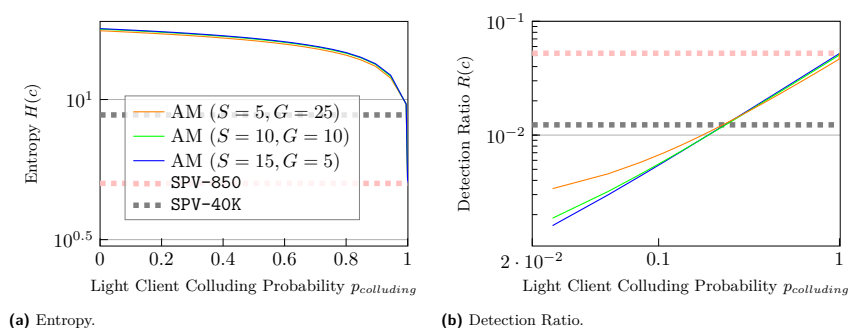
Using these properties, we evaluated the entropy and detection ratio of the aggregated transaction proposed model, similar to the previous privacy evaluation of the other light client implementations. In the entropy calculation evaluation, we evaluated the probability of each address in the network, implemented $T(x)$ and calculated $H(c) \approx \ln \frac{1}{1000} \cdot \sum_{x \in [0.001, \dots, 1]} (1 - x) \cdot T(x \cdot |c|)$. Fig. 6 evaluates the privacy of the aggregation model using several S and G parameters, compared to the communication overhead (normalized by the communication overhead of SPV-850) assuming there are no colluding nodes. Additionally, the entropy and detection ratio of SPV-850 and SPV-40K are added to show the privacy improvement. Fig. 6a presents the entropy compared to the communication overhead. As we see, the aggregation model using all S and G parameters has a similar entropy of around $H(c) = 17.5$, while having a communication overhead 1.8 – 1.96 times higher than SPV-850, and 25 times lower than SPV-40K. Recall, SPV-850 and SPV-40K have an entropy of $H(c) = 8.81$ and $H(c) = 5.02$, respectively. The communication network overhead is derived directly from the ratio $\frac{G}{S}$, as the communication overhead increases when the number of groups c is the leader of increases. The entropy of the aggregation model is very similar for all S, G parameters since the entropy evaluates the knowledge regarding all addresses, and when assuming there are no colluding nodes a full node does not have information about all of the filters used by c , hence all addresses are eligible to be associated with c . The increase in the probabilities of the addresses that appear in groups c is the leader of (and a full node has information about) are relatively negligible to all other addresses. Hence, the entropy values are high and similar for all S, G values.

In the detection ratio, shown in Fig. 6b, there are some differences between parameters. While for using $S, G = (5, 5)$ the detection ratio is $R(c) = 0.0022$, having a communication overhead 1.8 times higher than SPV-850, for $S, G = (25, 5)$ the detection ratio decreases to $R(c) = 0.0001$, though with a communication overhead 1.96 times higher than SPV-850. As the detection ratio is mainly affected by the high-probability addresses a full node knows about (rather than of all addresses), the differences in the detection ratio values are derived from the relation between S and G , which affects the probability of c being the leader of the group. The lower the ratio $\frac{G}{S}$ is, the lower the expected number of times c is a leader of the group, and the detection ratio decreases accordingly.

Fig. 7 presents the entropy and detection ratio of the aggregation model, using $S, G = (5, 25), (10, 10), (15, 5)$, compared to $p_{colluding}$, the probability the leader colluding with full nodes. While all three parameter sets have a similar entropy, using $S, G = (5, 15)$ achieves higher entropy for lower $p_{colluding}$ values: As for $p_{colluding} = 0.05$ $S, G = (5, 15)$ has an entropy of $H(c) = 17.79$, the entropy of $S, G = (10, 10)$ and $S, G = (5, 25)$ is $H(c) = 17.71$ and $H(c) = 17.49$, respectively. This is because for smaller $\frac{G}{S}$ values the expected number of groups c is the leader of is lower, hence c is exposed to the full node less times. Yet, recall by Properties 15 and 19, when c is the leader a full node receives a filter with more addresses than when c is not the leader and the leader colludes. Hence, when $p_{colluding}$ is high, full nodes get more information from groups c is not the leader of, thus the entropy is higher when c is the leader of more groups, i.e. when $\frac{G}{S}$ is larger. For instance, when $p_{colluding} = 1$, $H(c) = 9.65$ for $S, G = (5, 25)$, compared to $H(c) = 9.63$ and $H(c) = 9.62$ for $S, G = (10, 10)$



■ **Figure 6** The entropy and detection ratio of SPV-850, SPV-40K and the aggregation model (AM) using $S, G = (5, 5), (10, 10), (15, 5), (25, 5), (5, 25)$, compared to the network communication overhead normalized by the communication of SPV-850, assuming there are no colluding nodes.



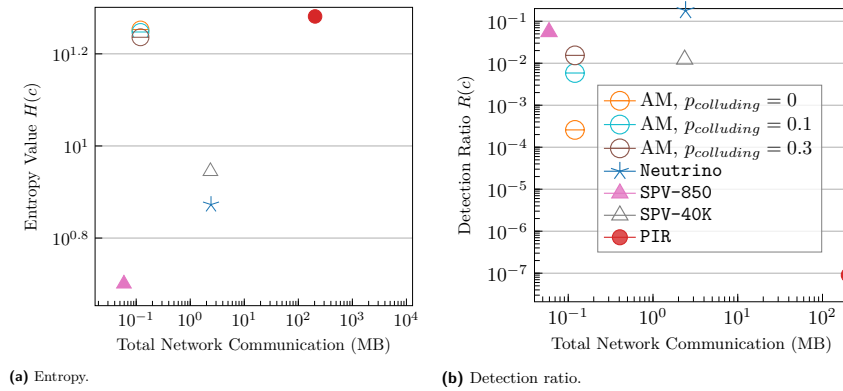
■ **Figure 7** The entropy and detection ratio of the aggregation model (AM) using $S, G = (10, 10), (15, 5), (5, 25)$, compared to the light client colluding probability $p_{colluding}$.

and $(15, 5)$, respectively. We note there is an entropy dropdown when $p_{colluding} = 1$ to $H(c) = 5.02$, as when $p_{colluding} = 1$ all addresses used in filters c sent are known, hence the address space A is decreased to only 900 addresses (the number of addresses used in filters of c).

The detection ratio increases too as $p_{colluding}$ increases. As for $S, G = (10, 10)$ the detection ratio is $R(c) = 0.0006$ when there are no colluding nodes, when 50% of the nodes collude the detection ratio increases to $R(c) = 0.026$, and when all nodes collude the detection ratio increases to $R(c) = 0.05$, similar to the detection ratio of SPV using 950 false positives (as derived by Lemma 7). For the same reasons as the entropy, for small $p_{colluding}$ values, the detection ratio is lower (implying better privacy) when $\frac{G}{S}$ is smaller, and for a high $p_{colluding}$, the detection ratio is lower when $\frac{G}{S}$ is larger. Finally, following Property 21, the figure shows the privacy of the aggregation model is always *at least as good as the privacy of SPV-850*, as the entropy is higher and the detection ratio is lower. We additionally note that compared to SPV-40K, the entropy of AM was higher except for the case where $p_{colluding} = 1$. Additionally, when $p_{colluding} \leq 0.23$, the detection ratio of SPV-40K is higher than the AM.

7.3 Aggregation Model Discussion and Limitations

Section 7.2 shows the great potential of the aggregation model, as a privacy increase is achieved with very low communication overhead. Fig. 8 summarizes the entropy and detection ratio of the different implementations compared to the communication overhead. The figure includes



■ **Figure 8** Privacy compared to communication overhead comparison between the different light client implementations and the *AM* model using ($S = 15, G = 5$) and $F = 850$ false positives.

AM using parameters ($S = 15, G = 5$) and $F = 850$ false positives, for $p_{colluding} = 0, 0.1, 0.3$. Though the *AM* communication overhead is 20 times lower than *SPV-40K*, for all three $p_{colluding}$ values, the entropy is higher. Moreover, for $p_{colluding} = 0, 0.1$, the detection ratio was lower too. Assuming the percentage of colluding nodes is lower than 23%, the *AM* model is only second to the *PIR* model in its privacy (having a communication overhead 1700 times smaller), with a communication overhead of approximately only twice the communication overhead of *SPV-850*.

We acknowledge the drawbacks of this model, as it might suffer from a time delay when creating the client groups, and when waiting for an answer from the leader. Moreover, in the scenario where all clients participating in groups with c are colluding, there is no privacy improvement. That said, the aggregation model, the best advantage of the aggregation model is it does not require any changes in the Bitcoin network protocols, and can work side-by-side with the original *SPV* protocol and full nodes. Clients preferring time over privacy can always use the original *SPV* protocol, as at any point light client c can send a filter to the full node. Property 21 states the *AM* model guarantees privacy at least as good as the original *SPV* model even when all other clients collude with full nodes. We additionally note that the *AM* protocol is designed to be very simple and light to run, as light clients are not meant to run heavy computations. As this paper mainly focuses on the privacy issue of light clients, the aggregation model presented here is not complete, and future work will expand this model. However, it contains the main building blocks of a new model that has great potential in increasing light client privacy.

8 Conclusion

In this work, we analyzed and compared the privacy of the *SPV* and *Neutrino* light client implementations. We defined two metrics to evaluate the privacy of light clients: light client entropy and detection ratio. Based on these metrics we analyzed and evaluated the privacy of the different implementations on real data and evaluated the privacy-communication tradeoff, comparing the different implementations and discussing these results and ways to improve privacy. Finally, we suggested a new *SPV*-based light client model that improves privacy. In future work, we intend to deepen the privacy analysis, including cases when an adversary with a full node has only partial information. We aim to further enhance the technical details and the analysis of the aggregation model, dealing with issues such as disconnecting nodes

and other malicious behaviors.

References

- 1 Mycelium - Bitcoin wallet. <https://mycelium.com/index.html>.
- 2 Percy++ / PIR in C++. <https://percy.sourceforge.net/readme.php>.
- 3 Neutrino: Privacy-preserving Bitcoin light client. <https://github.com/lightninglabs/neutrino>, 2017.
- 4 Helios. <https://github.com/a16z/helios>, 2023.
- 5 Run a light client to join binance chain. <https://docs.binance.org/light-client.html>, 2023.
- 6 Lodestar ethereum consensus implementation. <https://github.com/ChainSafe/lodestar>, 2024.
- 7 Ittai Abraham, Danny Dolev, and Joseph Y Halpern. Distributed protocols for leader election: A game-theoretic perspective. In *International Symposium on Distributed Computing (DISC)*, 2013.
- 8 André Augusto, Rafael Belchior, Miguel Correia, André Vasconcelos, Luyao Zhang, and Thomas Hardjono. Sok: Security and privacy of blockchain interoperability. *Authorea Preprints*, 2023.
- 9 Alex Biryukov and Sergei Tikhomirov. Transaction clustering using network traffic analysis for Bitcoin and derived blockchains. In *IEEE INFOCOM Workshops*, 2019.
- 10 Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.
- 11 Sean Braithwaite, Ethan Buchman, Ismail Khoffi, Igor Konnov, Zarko Milosevic, Romain Ruetschi, and Josef Widder. A tendermint light client. *arXiv preprint arXiv:2010.07031*, 2020.
- 12 Andrei Z. Broder and Michael Mitzenmacher. Network applications of Bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2003.
- 13 Jehoshua Bruck, Jie Gao, and Anxiao Jiang. Weighted Bloom filter. In *IEEE International Symposium on Information Theory (ISIT)*, 2006.
- 14 Benedikt Bünz, Lucianna Kiffer, Loi Luu, and Mahdi Zamani. Flyclient: Super-light clients for cryptocurrencies. In *IEEE Symposium on Security and Privacy (SP)*, 2020.
- 15 Panagiotis Chatzigiannis, Foteini Baldimtsi, and Konstantinos Chalkias. SoK: Blockchain light clients. In *International Conference on Financial Cryptography and Data Security (FC)*, 2022.
- 16 Archana Chhabra, Rahul Saha, Gulshan Kumar, and Tai-Hoon Kim. Navigating the maze: Exploring blockchain privacy and its information retrieval. *IEEE Access*, 2024.
- 17 Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.
- 18 Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of Bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, 2018.
- 19 Christian Decker and Roger Wattenhofer. Information propagation in the Bitcoin network. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2013.
- 20 Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A strengthened version of RIPEMD. In *International Workshop on Fast Software Encryption*, 1996.
- 21 Dmitry Ermilov, Maxim Panov, and Yury Yanovich. Automatic Bitcoin address clustering. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017.
- 22 Bin Fan, Dave G Andersen, Michael Kaminsky, and Michael D Mitzenmacher. Cuckoo filter: Practically better than Bloom. In *ACM International on Conference on emerging Networking Experiments and Technologies (CoNext)*, 2014.
- 23 Arthur Gervais, Srdjan Capkun, Ghassan O. Karame, and Damian Gruber. On the privacy provisions of Bloom filters in lightweight Bitcoin clients. In *ACM Annual Computer Security Applications Conference (ACSAC)*, 2014.
- 24 Solomon Golomb. Run-length encodings (corresp.). *IEEE transactions on information theory*, 12(3):399–401, 1966.
- 25 Xi He, Ketai He, Shenwen Lin, Jinglin Yang, and Hongliang Mao. Bitcoin address clustering method based on multiple heuristic conditions. *IET Blockchain*, 2(2):44–56, 2022.

- 26 Mike Hearn. Bloom filter privacy and thoughts on a newer protocol. <https://groups.google.com/forum/#!msg/bitcoinj/Ys13qkTwcNg/9qxnHwnkeoIJ>, 2015.
- 27 Kilan M Hussein and MF Al-Gailani. Evaluation performance of bloom filter in blockchain network. *Iraqi Journal of Information and Communication Technology*, 6(2):17–30, 2023.
- 28 MJ Jeyasheela Rakkini and K Geetha. Detection of Bitcoin miners by clustering crypto address with google bigquery open dataset. In *Soft Computing: Theories and Applications (SoCTA)*. 2022.
- 29 Kota Kanemura, Kentaroh Toyoda, and Tomoaki Ohtsuki. Design of privacy-preserving mobile Bitcoin client based on γ -deniability enabled Bloom filter. In *Int. Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017.
- 30 Bruce M Kapron, David Kempe, Valerie King, Jared Saia, and Vishal Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. *ACM Transactions on Algorithms (TALG)*, 6(4):1–28, 2010.
- 31 Kostis Karantias. SoK: A taxonomy of cryptocurrency wallets. *Cryptology ePrint Archive*, 2020.
- 32 Domokos Miklós Kelen and István András Seres. Towards measuring the fungibility and anonymity of cryptocurrencies. *arXiv preprint arXiv:2211.04259*, 2022.
- 33 Feng Liu, Zhihan Li, Kun Jia, Panwei Xiang, Aimin Zhou, Jiayin Qi, and Zhibin Li. Bitcoin address clustering based on change address improvement. *IEEE Transactions on Computational Social Systems*, 10:1–13, 2023.
- 34 Lailong Luo, Deke Guo, Richard T. B. Ma, Ori Rottenstreich, and Xueshan Luo. Optimizing Bloom filter: Challenges, solutions, and comparisons. *IEEE Communications Surveys and Tutorials*, 21(2):1912–1949, 2019.
- 35 Sinisa Matetic, Karl Wüst, Moritz Schneider, Kari Kostianen, Ghassan Karame, and Srdjan Capkun. Bite: Bitcoin lightweight client privacy using trusted execution. In *USENIX Security Symposium*, 2019.
- 36 R. C. Merkle. Secrecy, authentication, and public key systems. *PhD thesis, Stanford*, 1979.
- 37 Matt Corallo Mike Hearn. Connection Bloom filtering. <https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki>, 2012.
- 38 Deepanjan Mitra, Agostino Cortesi, and Nabendu Chaki. ALEA: An anonymous leader election algorithm for synchronous distributed systems. In *Progress in Image Processing, Pattern Recognition and Communication Systems (CORES, IP&C, ACS Conference)*, 2021.
- 39 Angelica Montanez. Investigation of cryptocurrency wallets on iOS and Android mobile devices for potential forensic artifacts. *Forensic Science Center, Marshall University*, 2014.
- 40 Malte Möser and Arvind Narayanan. Resurrecting address clustering in Bitcoin. In *International Conference on Financial Cryptography and Data Security (FC)*, 2022.
- 41 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 2008.
- 42 Yukun Niu, Chi Zhang, Lingbo Wei, Yankai Xie, Xia Zhang, and Yuguang Fang. An efficient query scheme for privacy-preserving lightweight Bitcoin client with Intel SGX. In *IEEE GLOBECOM*, 2019.
- 43 Alex Akselrod Olaoluwa Osuntokun. Compact block filters for light clients. <https://github.com/bitcoin/bips/blob/master/bip-0158.mediawiki>, 2017.
- 44 Jim Posen Olaoluwa Osuntokun, Alex Akselrod. Client side block filtering. <https://github.com/bitcoin/bips/blob/master/bip-0157.mediawiki>, 2017.
- 45 Kaihua Qin, Henryk Hadass, Arthur Gervais, and Joel Reardon. Applying private information retrieval to lightweight Bitcoin clients. In *Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2019.
- 46 Ori Rottenstreich and Isaac Keslassy. The Bloom paradox: When not to use a Bloom filter. *IEEE/ACM Trans. Netw.*, 23(3):703–716, 2015.
- 47 Maroufi Saeid and Nemat Moein. Go bither. <https://github.com/bitherhq/go-bither>, 2017.

- 48 Sehrish Shafeeq, Sherali Zeadally, Masoom Alam, and Abid Khan. Curbing address reuse in the iota distributed ledger: A cuckoo-filter-based approach. *IEEE Transactions on Engineering Management*, 67(4):1244–1255, 2019.
- 49 Ke Shao, Wei Lv, and Yu Li. Addressing blockchain privacy and efficiency challenges in mobile environments: an optimization strategy for lightweight clients and full nodes. *Advances in Engineering Technology Research*, 7(1):1–1, 2023.
- 50 Apostolos Tzinas Angel Leon Dimitris Lamprinos Ardis Lu shrestha agrawal, Justin Martin. Kevlar. <https://github.com/lightclients/kevlar>, 2022.
- 51 Thomas Voegtlin. Electrum - lightweight Bitcoin client. <https://github.com/spesmilo/electrum>, 2018.
- 52 Kai Wang, Maike Tong, Changhao Wu, Jun Pang, Chen Chen, Xiapu Luo, and Weili Han. Exploring unconfirmed transactions for effective Bitcoin address clustering. *arXiv preprint arXiv:2303.01012*, 2023.
- 53 François-Xavier Wicht. Blockchain privacy notions using the transaction graph model. *Master Thesis, University of Fribourg*, 2023.
- 54 Karl Wüst, Sinisa Matetic, Moritz Schneider, Ian Miers, Kari Kostiainen, and Srdjan Čapkun. ZLite: Lightweight clients for shielded Zcash transactions using trusted execution. In *International Conference on Financial Cryptography and Data Security (FC)*, 2019.
- 55 Yankai Xie, Chi Zhang, Lingbo Wei, Yukun Niu, and Faxing Wang. Private transaction retrieval for lightweight Bitcoin client. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019.
- 56 Yankai Xie, Chi Zhang, Lingbo Wei, Yukun Niu, Faxing Wang, and Jianqing Liu. A privacy-preserving Ethereum lightweight client using PIR. In *IEEE/CIC International Conference on Communications in China (ICCC)*, 2019.
- 57 Yiyin Zhang. SoK: Anonymity of lightweight clients in cryptocurrency systems. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2023.
- 58 Yuhang Zhang, Jun Wang, and Jie Luo. Heuristic-based address clustering in Bitcoin. *IEEE Access*, 8:210582–210591, 2020.
- 59 Baokun Zheng, Liehuang Zhu, Meng Shen, Xiaojiang Du, and Mohsen Guizani. Identifying the vulnerabilities of Bitcoin anonymous mechanism based on address clustering. *Science China Information Sciences*, 63:1–15, 2020.