

CertainSync: Rateless Set Reconciliation with Certainty

TOMER KENIAGIN, Technion - Israel Institute of Technology, Israel

EITAN YAAKOBİ, Technion - Israel Institute of Technology, Israel

ORI ROTTENSTREICH, Technion - Israel Institute of Technology, Israel

Set reconciliation is a fundamental task in distributed systems, particularly in blockchain networks, where it enables the synchronization of transaction pools among peers and facilitates block dissemination. Existing traditional set reconciliation schemes are either statistical, providing success probability as a function of the communication overhead and the size of the symmetric difference, or require parametrization and estimation of the size of the symmetric difference, which can be prone to error. In this paper, we present *CertainSync*, a novel reconciliation framework that, to the best of our knowledge, is the first to guarantee successful set reconciliation without any parametrization or estimators in use. The framework is rateless and adapts to the unknown symmetric difference size. The set reconciliation is guaranteed to be completed successfully whenever the communication overhead reaches a lower bound derived from the symmetric difference size and the universe size. Our framework is based on recent constructions of Invertible Bloom Lookup Tables (IBLTs) ensuring successful element listing as long as the number of elements is bounded. We provide a theoretical analysis to prove the certainty in the set reconciliation for multiple constructions. The approach is also validated by simulations, showing the ability to synchronize sets with efficient communication costs while maintaining reconciliation guarantees compared to other baseline schemes for set reconciliation. To further improve communication overhead for large universes as blockchain networks, *CertainSync* is extended with a universe reduction technique to minimize communication overhead. We compare and validate the extended framework *UniverseReduceSync* against the basic *CertainSync* framework through simulations using real blockchain transaction hash data from the Ethereum blockchain network. The results illustrate a trade-off between improved communication costs and maintaining reconciliation guarantees without relying on parametrization or estimators, offering a comprehensive solution for set reconciliation in diverse scenarios.

CCS Concepts: • **Networks** → **Network algorithms; Network protocol design.**

Additional Key Words and Phrases: Set Reconciliation; Coding Theory; Blockchain Applications

1 INTRODUCTION

Set reconciliation is essential for synchronizing data across systems like cloud storage services [22], Peer-to-Peer (P2P) networks [5], distributed computing [27], and blockchain networks [30]. Instead of directly exchanging set elements, which incurs high communication overhead $O(|A| + |B|)$ for sets A and B , reconciliation protocols use compact representations based on coding theory and probabilistic data structures (sketches). These representations enable efficient identification of symmetric differences with lower communication costs. For instance, in blockchain networks, efficient reconciliation enables light clients to synchronize their transaction pools and blockchain states effectively, as seen in protocols like MempoolSync [16] and SREP [3]. Additional reconciliation applications in fields like collaborative editing and distributed databases are detailed in Appendix A.

Moreover, data consistency (via guaranteed identification of symmetric differences), as in blockchain systems, is crucial. Reconciliation schemes fall into two main categories: probabilistic and exact. Probabilistic approaches, such as Rateless IBLT [39] or with Cuckoo filter [23, 24], offer low overhead but risk occasional failures in identifying symmetric differences. Exact schemes, like CPISync [37] and PinSketch [9], guarantee the success of identifying symmetric differences by relying on symmetric difference size estimation or an upper bound, which can be computationally expensive and prone to errors. Additionally, existing reconciliation schemes often involve complex

Authors' addresses: Tomer Keniagin, keniagin@campus.technion.ac.il, Technion - Israel Institute of Technology, Haifa, Israel; Eitan Yaakobi, yaakobi@cs.technion.ac.il, Technion - Israel Institute of Technology, Haifa, Israel; Ori Rottenstreich, or@technion.ac.il, Technion - Israel Institute of Technology, Haifa, Israel.

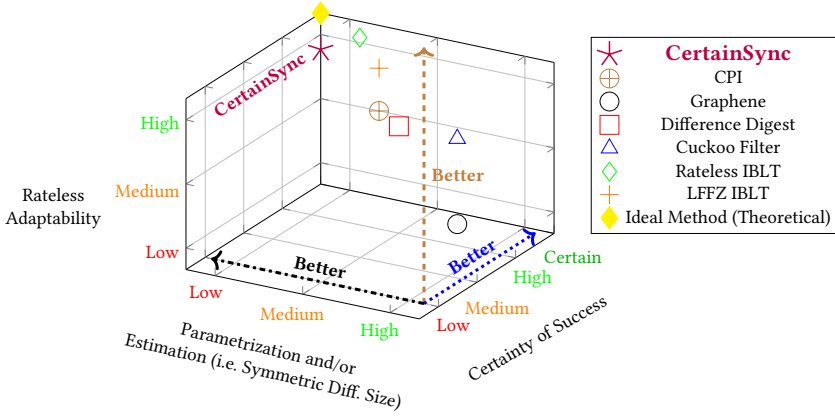


Fig. 1. Comparison of set reconciliation schemes based on three metrics: (i) parametrization tuning and/or symmetric difference size estimation overhead (lower is better), (ii) certainty of success (higher is better), and (iii) rateless adaptability (higher is better) indicates efficiency by dynamically adapting to varying set differences while minimizing communication overhead and retransmissions. A detailed explanation of the parameters and estimators used for each scheme is presented in Table 4 in Appendix B.

parameter tuning as in Graphene [30] for set reconciliation among peers in blockchains and related distributed systems, where there is a parameter search algorithm based on input parameters given, or estimations such as Strata Estimator in Difference Digest [11] for set difference size estimation. In large-scale or real-time set reconciliation applications, where simplicity, scalability, and low latency are prioritized, parametrization and estimators, which incur significant overhead and complexity, should be avoided. This work introduces a novel reconciliation framework called CertainSync that combines two desirable properties: (i) simplicity, with no parameter tuning or difference size estimation, and (ii) guaranteed success in computing symmetric differences through rateless transmission. Fig. 1 illustrates how our approach achieves these goals, ensuring both robustness and efficiency. Our focus is on guaranteeing successful set reconciliation in a parameterless manner, making this the first work to introduce this concept.

CertainSync can be tuned with one of three proposed constructions based on which elements are mapped to the IBLTs. Table 1 summarizes the basic properties of the three constructions as a function of the following parameters: (i) universe size from which elements can appear (n), and (ii) size of the symmetric difference ($|\Delta|$). Each construction is characterized by (i) the maximum supported symmetric difference size ($|\Delta|_{\max}$), (ii) the incremental communication overhead per single element increase in the symmetric difference size (Δm_d), and (iii) the total communication overhead (m_d). In Table 1, the total communication overhead refers to the number of IBLT cells required for synchronization. Throughout the paper, the comparison of set reconciliation schemes is measured in bits, derived from the respective bit size of an individual IBLT cell, with additional overhead bits as necessitated by the distinct requirements of each scheme. To evaluate the characteristics of our proposed framework, we conduct experiments using datasets that include sets of positive numbers to mimic the transaction pools in blockchain systems. Our experimental results demonstrate that our framework successfully computes the symmetric difference with certainty, and communication overhead is determined by a bound derived from the symmetric difference size and the universe size. Since this bound depends on the universe size, which can be large, it may result in inefficiencies in large-scale settings like blockchain networks. We mitigate this issue by extending our framework to include an optional universe size reduction. We provide a comprehensive comparative analysis

Table 1. Proposed constructions for CertainSync and their inherent properties. d represents the symmetric difference size ($|\Delta|$) and n indicates the universe size.

CertainSync Construction	Max Symmetric Difference Size $ \Delta _{max}$	Incremental Communication Overhead Δm_d (Eq. (1))	Total Communication Overhead m_d
Construction I EGH [10] (Subsection 4.1)	n	$O\left(\frac{d \log n}{\log d + \log \log n}\right)$	$O\left(\frac{d^2 \log^2 n}{\log d + \log \log n}\right)$
Construction II OLS [33] (Subsection 4.2)	$\lceil \sqrt{n} \rceil$	$\lceil \sqrt{n} \rceil$	$d \lceil \sqrt{n} \rceil$
Construction III Extended Hamming [12] (Subsection 4.3)	3	$\lceil \log_2 n \rceil$	$(d - 1) \lceil \log_2 n \rceil + 1$

of existing set reconciliation schemes compared to our proposed framework. Also, we apply our framework in the Ethereum blockchain with real blockchain data to synchronize transaction pools. The notations used throughout the paper are presented in Table 5 in Appendix C. We make available the implementation of CertainSync developed in Python and Go as an open source on GitHub¹. It includes algorithms implementations, datasets and test scripts for reproducibility.

2 BACKGROUND AND RELATED WORK

2.1 Set Reconciliation

In set reconciliation, two or more parties hold finite sets and aim to learn the elements missing from each other’s sets. Set reconciliation often utilizes the symmetric difference operation to efficiently identify and reconcile discrepancies between sets. Key performance metrics for evaluating set reconciliation schemes include: (i) decoding accuracy (probability of correctly identifying the symmetric difference), (ii) communication efficiency (minimizing total communication overhead), (iii) scalability (minimizing additional communication overhead as symmetric difference size increases), (iv) computation overhead (processing requirements at each party), and (v) total reconciliation time (the time required until the sets are fully reconciled) are of great interest. However, the primary focus is on decoding accuracy and communication overhead, as less system-independent factors (e.g. network bandwidth, compute capabilities), offering a more general basis for evaluation of different set reconciliation schemes. It is noteworthy that computing the union of sets is a fundamental operation that can be executed by any participant involved in the set reconciliation process. We denote by Δ the symmetric difference of two considered sets.

Definition 2.1 (Symmetric Difference). The symmetric difference for two sets A and B refers to elements that appear in one set and not in the other set, namely $\Delta = (A \setminus B) \cup (B \setminus A)$.

For example, given two sets $A = \{1, 2, 3, 4\}$ and $B = \{3, 4, 6, 7\}$, the symmetric difference consists of elements in A but not in B ($A \setminus B = \{1, 2\}$) and elements in B but not in A ($B \setminus A = \{6, 7\}$). Thus, the symmetric difference is $\Delta = (A \setminus B) \cup (B \setminus A) = \{1, 2, 6, 7\}$.

¹<https://github.com/toto9820/Rateless-Set-Reconciliation-with-Listing-Guarantees>

2.2 Invertible Bloom Lookup Table (IBLT) and its Applications

Sketches, such as Bloom Filter (BF) [2] and Count-Min [7], provide space-efficient, probabilistic representations of sets. These methods trade off some accuracy for substantial gains in efficiency and scalability, making them suitable for applications where approximate results are acceptable. An IBLT [14] is another sketch type that combines efficiency with the ability to identify the elements inserted into it probabilistically. An IBLT consists of a fixed-size array of cells, where each cell contains the following fields: (i) count, an integer representing the number of elements mapped to this cell, (ii) xorSum, the XOR of all elements mapped to this cell, and (iii) checksum, the XOR of the hash values of all elements mapped to this cell. The operations supported by an IBLT are insertion for adding an element to an IBLT, deletion for removing an element from an IBLT, and listing for retrieving elements from an IBLT. An important concept in IBLT listing is a *pure cell*. This is a cell that contains a single inserted element. A pure cell is identifiable by containing only one element mapped to it. The listing procedure repeatedly searches for a pure cell and identifies the element based on the xorSum field when found. The identified element is then removed from other cells it is mapped to, reducing the remaining elements in those cells. The process fails if no pure cells are found before listing all elements.

In this paper, we mainly focus on set reconciliation approaches based on IBLTs. IBLTs offer several advantages for set reconciliation: they provide a compact representation of sets, allowing for minimal communication of set differences [11], they support the insertion and deletion of elements, and there is support for listing elements with success probability. By exchanging IBLT sketches, parties can reconcile their sets efficiently. IBLTs could represent elements that are present in one set but absent in the other, as demonstrated in Difference Digest [11] by the subtraction operation. However, they have a major drawback - the listing is not guaranteed to succeed as there is a success probability derived from the number of elements inserted into the IBLT and the IBLT memory size represented as its number of cells. The listing success probability of an IBLT, as shown in [14], is almost $1 - o(1)$ if the ratio between the number of available cells and the number of inserted elements exceeds a certain threshold c_h , where h is the number of hash values. In contrast, if the ratio falls below c_h , as described in [19], partial listing (namely partial extraction representing listing success probability less than 1) becomes more likely. However, a small number of iterative executions of partial listing can eventually achieve a full listing of all elements inserted in the IBLT.

2.3 Concepts in Coding Theory and Bloom Filters

For any positive integer k , we define $[k] := \{1, \dots, k\}$ and $[k]_0 := \{0, \dots, k-1\}$, and these notations are used consistently throughout the paper to represent sets.

2.3.1 Bloom Filter with FPFZ. Bloom filter with a FPFZ (False Positive Free Zone) [18] is a specific Bloom filter construction where for a set of up to d elements from a finite universe U , membership queries return true if and only if an element is actually in the set, thus guaranteeing no false positives and no false negatives for any membership queries.

2.3.2 EGH & EGH Bloom Filter. The Eppstein, Goodrich, and Hirschberg (EGH) method [10] was originally developed for group testing by leveraging properties of modular arithmetic and the Chinese Remainder Theorem. However, this method has been adapted for various applications, including Bloom filters [18] with EGH filter. In the context of Bloom filters, the EGH method establishes an upper bound d on the maximum number of inserted elements for which a FPFZ is guaranteed. This upper bound is related to the size of the universe n and a product of prime numbers, expressed as $d \leq \log_n \Pi_k = \sum_{j=1}^k \log_n p_j$, where $\Pi_k \triangleq \prod_{j=1}^k p_j$ is the product of the first k prime numbers, and p_j represents the j -th prime number ($p_1 = 2, p_2 = 3, p_3 = 5, \dots$).

2.3.3 OLS & OLS Bloom Filter. Orthogonal Latin Square (OLS) codes [15] have been widely studied to protect memories from errors, as they have a modular construction and can be decoded in parallel with simple circuitry [32]. A Latin square of order n is an $n \times n$ array filled with n different symbols where each symbol belongs to the set $[n]_0$, and each symbol occurs exactly once in each row and exactly once in each column [17] as shown in Fig. 2(a). Two Latin squares are orthogonal if, when superimposed, each ordered pair of symbols appears exactly once as shown in Fig. 2(b). A set of Latin squares of the same order such that every pair of squares is orthogonal is called Mutually Orthogonal Latin Squares (MOLS); it is denoted by $\text{MOLS}(n)$ where n is the order of the Latin squares. The maximum size of $\text{MOLS}(n)$, as shown in [17] (Theorem 5.1.2), is at most $n - 1$. A set of $n - 1$ $\text{MOLS}(n)$ is called a complete set of MOLS. When n is a prime power, we are guaranteed to have at least one complete set of $\text{MOLS}(n)$, as stated in [17] (Theorem 5.2.3). OLS filter is a Bloom filter constructed using OLS code [33]. Besides using MOLS, two additional special matrices (which are not Latin squares) are used for the construction of the filter. One of the two additional special matrices used later we notate as R_n , where each of its rows consists of a single repeated value, ranging from 0 to $n - 1$. Specifically, the i -th row of R_n matrix is filled with the value $i - 1$ for all columns, as illustrated in Fig. 2(c). The second special matrix is its transpose R_n^T .

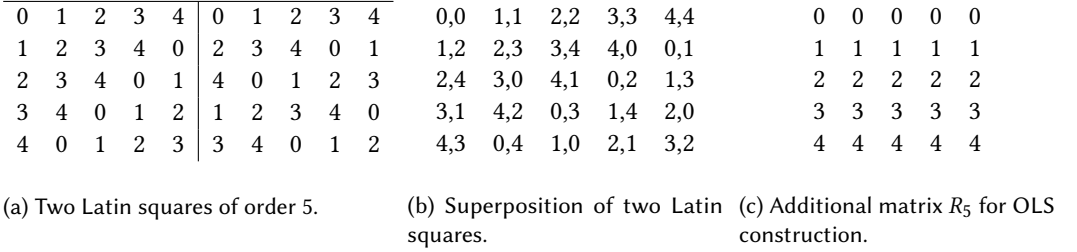


Fig. 2. Latin squares, superposition, and additional matrix for constructing OLS code.

2.3.4 Extended Hamming Code. An Extended Hamming code is an Error Correcting Code (ECC) that extends the standard Hamming code by adding an extra parity bit to have a 1-bit error-correcting or 3-bit error-detecting code.

Definition 2.2 (Extended Hamming Code). An Extended Hamming code is a linear code denoted by $(n = 2^m, k = 2^m - m - 1, d_H = 4)$, where n is the codeword length; k is the information length (the dimension); d_H is the minimum Hamming distance between any two codewords, and $m \geq 2$ is a positive integer. The parity check matrix H_n of the Extended Hamming code is the following:

$$H_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ & & H'_n & & & \end{bmatrix}, \quad H'_n = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 1 & \cdots & 1 & 1 \\ 0 & 1 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

where the matrix H'_n consists of all the 2^m binary column vectors of length $m = \log_2(n)$.

2.3.5 Stopping Set. A stopping set [8] is a combinatorial structure, originally defined in the context of matrices for the decoding procedure of error-correcting codes [35].

Definition 2.3 (Stopping Set). Let M be a matrix and S a non-empty set of its columns. The row's weight in the sub-matrix implied by S is defined as the number of non-zero coordinates in it. S is

called a stopping set if it has no row of weight one. The stopping distance of M , denoted by $s(M)$, is the size of the smallest stopping set in M .

2.4 Blockchain Application of Set Reconciliation

Set reconciliation plays a crucial role in various applications, and one prominent application is in blockchain networks. Blockchain networks rely on efficient and reliable synchronization of transactions from transaction pools (TxPools) and newly mined blocks (block propagation process) between peers, as illustrated in Fig. 3 to maintain consensus and ensure the integrity for proper functioning. IBLTs play an important role in blockchain networks by enabling efficient represen-

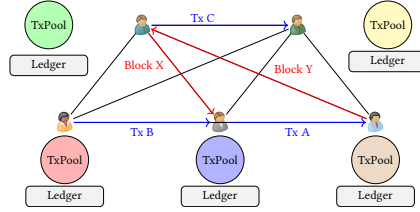


Fig. 3. Blockchain network modeled as a peer-to-peer (P2P) system where each peer maintains a local ledger (immutable chain of blocks) and a transaction pool (TxPool). Peers synchronize transactions and mined blocks to achieve consensus across the network.

tation and reconciliation of transaction pools and blocks between peers, significantly reducing communication overhead as demonstrated in Graphene [30] for interactive set reconciliation in blockchain networks. Various protocols have been proposed for the synchronization of transaction pools outside (and independently) of the block propagation channel [3, 16, 21] due to long block transmission time and block validation time like SREP [3], which leverages out-of-band synchronization of transaction pools, ensuring that only differences between transaction pools are exchanged to minimize communication overhead.

2.5 Rateless Coding & IBLT

Rateless coding, also known as fountain coding, enables efficient and reliable data transmission over unreliable or lossy channels. Unlike traditional fixed-rate coding schemes, rateless codes do not have a predetermined rate or block length. Instead, an unlimited number of encoded symbols are generated from the original data, allowing the recipient to recover the original data by collecting a sufficient number of these encoded symbols, regardless of which specific symbols are received.

Definition 2.4 (Rateless Coding). Let \mathcal{S} be the original data of size k symbols, and $C = \{c_1, c_2, \dots, c_n\}$ be the set of encoded symbols generated by the rateless code. Recipient can reconstruct \mathcal{S} from any subset $C' \subseteq C$ such that $|C'| \geq t(k)$, where $t(k)$ is a threshold function dependent on the size k of the original data and determined by the specific coding scheme.

Rateless coding has been recently applied to the problem of set reconciliation [39]. In this work, the authors propose a rateless set reconciliation protocol based on the original IBLT approach with hash functions. Instead of encoding a set into a fixed-size IBLT, the rateless IBLT approach generates an unbounded stream of IBLT cells as coded symbols. Participant 1 can continuously transmit these coded symbols until Participant 2 has collected enough IBLT cells to decode the set difference successfully. In Definition 2.4, k represents the symmetric difference size $|\Delta|$. Also, Lázaro et al. [20] proposed a rateless solution based on a variant of IBLT named MET IBLT.

2.6 Listing Failure Free Zone (LFFZ) IBLT

One significant challenge in set reconciliation is ensuring that all the set elements are correctly identified, known as the listing guarantee. Traditional methods often fail to provide this guarantee due to their inherent false positive rate. Schemes to provide this listing guarantee have been proposed in the past: Bloom filters with a trie-based mechanism for eliminating false positives [34], changing the traditional decoding method with pure cells by asserting an extra pure cell condition [6], an additional stash data structure based on error correcting codes, which serves as a backup when IBLT fails to decode correctly [1], and replacing the random hash functions for mapping elements of the set to IBLT cells, which contribute to the probabilistic nature of the IBLT, with multiple constructions that are based on various coding techniques [28]. The concept of IBLTs with a Listing Failure Free Zone (LFFZ) [28], provides a guarantee of successful listing for all sets up to a certain size parameter d , thereby enhancing the reliability and robustness of IBLTs. An (n, d) -LFFZ IBLT is defined as follows:

Definition 2.5 ((n, d)-LFFZ). Let $U = [n]$ be a finite universe of size n , and let $S \subseteq U$ be a set of size at most d . If an (n, d) -LFFZ IBLT is constructed to encode the set S , then the decoding process is guaranteed to successfully list all elements in S , regardless of the specific elements or their distribution within the universe U .

LFFZ IBLTs rely on several combinatorial and recursive methods for their construction. They use a binary mapping matrix M of size $m \times n$ to map elements of a set to cells in an IBLT, instead of using hash functions as in the original IBLT [14].

Definition 2.6 (Binary Mapping Matrix M). A binary mapping matrix M of size $m \times n$ is used to map elements of a set to cells in an IBLT. Each row of the matrix corresponds to a cell of an IBLT, and each column corresponds to an element in the universe. The entry $M[i][j]$ indicates whether the j -th element is mapped to the i -th cell, with $M[i][j] = 1$ if the element is mapped to the cell, and $M[i][j] = 0$ otherwise.

An important concept from [28] assures that for any set S of at most d elements from U , the mapping matrix M does not contain any stopping set of size at most d . This leads to the definition of a d -decodable matrix.

Definition 2.7 (d -decodable matrix). An $m \times n$ binary matrix M is called d -decodable if its stopping distance is at least $d + 1$, that is, $s(M) \geq d + 1$. Given $n, d \in \mathbb{N}$, with $d \leq n$, the minimal number of rows of a d -decodable matrix is denoted by $m^*(n, d)$, that is,

$$m^*(n, d) = \min \{m : \exists M \in \{0, 1\}^{m \times n}, M \text{ is } d\text{-decodable}\}.$$

From the definition of a d -decodable matrix, it follows that any d -decodable matrix is also a $(d - 1)$ -decodable matrix.

3 SET RECONCILIATION USING THE CertainSync FRAMEWORK

In the CertainSync framework, we utilize the constructions of d -decodable rateless matrices (where $d = |\Delta|$) from Section 4 as mapping matrices. These matrices map elements to IBLT cells, enabling set reconciliation with certainty. The certainty guarantees the listing success in retrieving the symmetric difference Δ under certain conditions without knowing the size of the symmetric difference $|\Delta|$ in advance.

3.1 Two-Party Problem for the CertainSync Framework

Let $U = [n]$ be a finite universe of size n , and let $S_1, S_2 \subseteq U$ be the sets held by Participant 1 (P_1) and Participant 2 (P_2), respectively. Given a d -decodable rateless matrix $M_{n,d}$ of size $m \times n$, where m ,

which depends on the unknown symmetric difference size, is the finite number of cells in an IBLT. P_1 constructs $IBLT_1$ from set S_1 according to the mapping defined by $M_{n,d}$, and then sends $IBLT_1$ cells to P_2 in a rateless manner. The objective is to construct an IBLT of the symmetric difference ($IBLT\{\Delta\}$) at P_2 by performing the subtraction of the IBLTs of P_2 and P_1 , with the guarantee that the listing operation at P_2 successfully recovers all elements in Δ . Upon successful recovery of Δ , P_2 could transmit the required subset $S_2 \setminus S_1$ of Δ to P_1 to achieve complete set reconciliation.

3.2 Set Reconciliation with Certainty

A set reconciliation with certainty is a more stringent variant of exact set reconciliation, where the goal is not only to fully recover the symmetric difference between two sets, but also to do so without prior knowledge of the size of the symmetric difference, defined formally as follows:

Definition 3.1 (Set Reconciliation with Certainty). Given two participants P_1 and P_2 holding sets $S_1, S_2 \subseteq U$ respectively, where $U = [n]$ is a finite universe of size n . Set reconciliation with certainty is achieved if it correctly recovers the symmetric difference $\Delta = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$ with probability 1, succeeds in recovery without requiring prior knowledge of the symmetric difference size $|\Delta|$, and terminates after a finite number of steps.

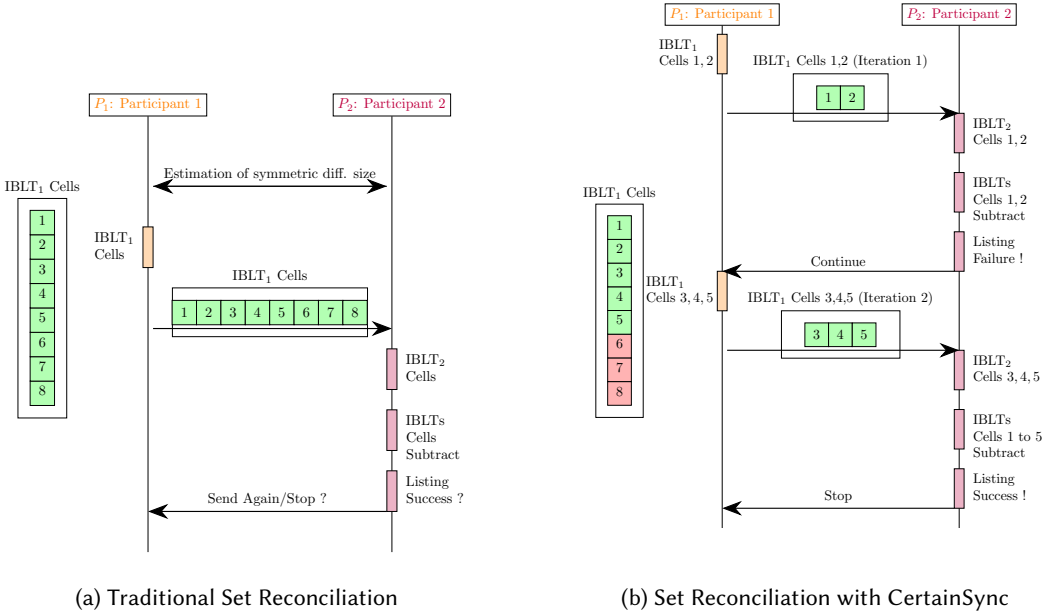


Fig. 4. Illustration of traditional set reconciliation vs. CertainSync.

In the CertainSync framework, set reconciliation with certainty is achieved by leveraging d -decodable rateless matrices, which enable set reconciliation without prior knowledge of the symmetric difference size and termination in a finite number of steps. Fig. 4(a) illustrates the traditional set reconciliation as described in Algorithm 2 in Appendix D, showing how the IBLTs of the participants are constructed with a fixed number of cells based on estimation and/or parametrization, and the probabilistic listing of their subtraction to determine the symmetric difference between them with success probability. In our set reconciliation framework, as described in Fig. 4(b), using CertainSync, Participant 1 constructs at each iteration an amount (according to chosen construction) of $IBLT_1$

cells representing its set and transmits it to Participant 2. There is no estimation of the symmetric difference size at any point like in the traditional set reconciliation. At each iteration, Participant 2 constructs the same amount of cells received from Participant 1, and checks if they have enough total cells to ensure the success of the listing of $\text{IBLT}\{\Delta\}$ to find the symmetric difference between the two sets with certainty (success probability of 1). If so, Participant 2 tells Participant 1 to stop sending cells, and lists successfully all the elements in $\text{IBLT}\{\Delta\}$ to find the symmetric difference between the two sets with certainty.

3.3 Applications for CertainSync Constructions

Constructions of CertainSync and their extended variants with universe reduction (UniverseReduceSync, presented in Section 6) are tailored to address various scenarios detailed in Table 2, therefore offering a comprehensive solution for set reconciliation with certainty. Intuitively, Cer-

Table 2. Comparison of proposed CertainSync constructions and their practical applications.

Property \ CertainSync Construction	Construction I EGH [10] (Subsection 4.1)	Construction II OLS [33] (Subsection 4.2)	Construction III Extended Hamming [12] (Subsection 4.3)
Purpose	Unbounded Symmetric Difference Size	Medium Symmetric Difference Size & Large Bounded Universe	Small Symmetric Difference Size
Key Advantage	Flexible Symmetric Difference	Improved Communication Overhead Scalability	Low Communication Overhead
Small Universe Compatible	✓ CertainSync	✓ CertainSync	✓ CertainSync
Medium Universe Compatible	✓ CertainSync	✓ CertainSync	
Large Universe Compatible	✓ CertainSync ✓ UniverseReduceSync	✓ UniverseReduceSync	
Potential Applications	- Blockchain synchronization - Highly dynamic data systems	- Collaborative document editing - Distributed database sync - Version control systems	- Error correction

tainSync EGH is well-suited for blockchain synchronization due to its ability to handle varying symmetric differences over time in a full universe-size range. CertainSync OLS is useful for the synchronization of distributed databases or files with an overall fixed size (fixed universe size). Meanwhile, CertainSync Extended Hamming is relevant for error correction, offering a low communication overhead solution for managing small symmetric differences effectively.

3.4 Algorithms for CertainSync Framework

We present an overview of the key algorithms for implementing and utilizing CertainSync. These algorithms are designed to ensure guaranteed listing success and utilize rateless adaptability. The detailed algorithms are provided in Appendix D with an example in Appendix E.

ConstructIBLT. This algorithm constructs at each iteration i an amount of IBLT cells (according to chosen construction) from a given set S using submatrix of mapping matrix $M_{n,d}$, denoted as $M_{n,i}$, which is an i -decodable rateless matrix. The IBLT is initialized with cells containing count, xorSum, and checksum fields, which are updated as elements from the set are processed.

IBLTDiff. This algorithm constructs the IBLT of the symmetric difference ($\text{IBLT}\{\Delta\}$) by subtracting the IBLT of P_1 (IBLT_1) from the IBLT of P_2 (IBLT_2).

DecodeDiff. This algorithm lists the symmetric difference Δ by decoding the IBLT of the symmetric difference ($\text{IBLT}\{\Delta\}$). It retrieves elements from pure cells and removes them from other IBLT cells until all elements are decoded or a failure occurs if the IBLT is not empty at the end.

4 CONSTRUCTIONS FOR CertainSync FRAMEWORK

For our constructions, we use a subfamily of d -decodable matrices that inherently possess a rateless structure, defined as follows:

Definition 4.1 (d -decodable Rateless Matrix). An $m \times n$ matrix $M_{n,d}$ is called a d -decodable rateless matrix if there exist positive integers $m_1 \leq m_2 \leq \dots \leq m_d = m$, such that for every $i \in [d]$, the $m_i \times n$ submatrix formed by the first m_i rows of $M_{n,d}$ is i -decodable. We refer to the vector (m_1, m_2, \dots, m_d) as the *decodability profile* of the matrix $M_{n,d}$.

The decodability profile of a d -decodable rateless matrix indicates the additional number of rows required to transition from $(i-1)$ -decodability to i -decodability, where $i \in [d]$, capturing the incremental growth in rows needed for progressively increasing decodability. Specifically, let Δm_i denote the additional number of rows added when transitioning from $(i-1)$ -decodable to i -decodable. With $\Delta m_1 = m_1$ i.e. $m_0 = 0$ the additional number of rows added Δm_i is the following:

$$\Delta m_i = m_i - m_{i-1}, \quad (1)$$

Example (d -Decodable Rateless Matrix). For $n = 8$ columns, we construct a 2-decodable rateless matrix $M_{n=8,d=2}$, given as follows:

$$M_{8,2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

To observe that $M_{8,2}$ is indeed a 2-decodable rateless matrix, note that the matrix $M_{8,2}$ is 2-decodable since any set of two columns has a row of weight one. Hence, $m_2 = 4$. Furthermore, $m_1 = 1$ since the submatrix M_1 of $M_{8,2}$ which consists of its first row, i.e. $M_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$ is 1-decodable because any set containing one column has a row of weight one. In summary, the matrix $M_{8,2}$ is a 2-decodable rateless matrix and its decodability profile is $(m_1, m_2) = (1, 4)$.

Next, we present several CertainSync constructions, where for each construction, we present a d -decodable rateless matrix that is used to map elements to IBLT cells differently. For each construction, an example of a d -decodable rateless matrix is provided in Appendix E.

4.1 Construction I: EGH Rateless Matrix

In this section, we show a construction of a d -decodable rateless matrix using the EGH method [10].

Definition 4.2. [The EGH Matrix] Let $n \in \mathbb{N}$ and $i \in [d]$ be given, where n is the number of columns and i is the decodability parameter. The i -decodable EGH matrix, denoted by $M_{n,i}^I \in \{0, 1\}^{m_i \times n}$, is defined as follows:

- (1) Let k_i be the smallest integer s.t. $\Pi_{k_i} \geq n^i$, where Π_{k_i} is the product of the first k_i prime numbers.
- (2) Let $m_i = \sum_{j=1}^{k_i} p_j$.
- (3) For each $j \in [k_i]$, define the submatrix $M_j \in \{0, 1\}^{p_j \times n}$ as:

$$M_j[x, y] = \begin{cases} 1 & \text{if } y + 1 \equiv x \pmod{p_j} \\ 0 & \text{otherwise} \end{cases}$$

for $x \in [p_j]_0$ and $y \in [n]_0$.

- (4) The EGH matrix $M_{n,i}^I$ is the vertical concatenation of the k_i submatrices M_1, M_2, \dots, M_{k_i} .

The resulting matrix $M_{n,i}^I$ is of size $m_i \times n$, where $m_i = \sum_{j=1}^{k_i} p_j$. The value m_i of the number of rows is also denoted by $m(n, i)$.

In order to show that the EGH matrix $M_{n,d}^I$ is a $(d+1)$ -decodable rateless matrix, we use the following theorem whose proof is deferred to Appendix F.

THEOREM 4.3. *For n and d , the EGH matrix $M_{n,d}^I$ is a $(d+1)$ -decodable rateless matrix. For $2 \leq i \leq d+1$, the decodability profile is $m_i = \sum_{j=1}^{k_i} p_j$, where k_i is the smallest integer such that $\Pi_{k_i} \geq n^i$. Furthermore, $m_1 = m_2$.*

4.2 Construction II: OLS Rateless Matrix

In this section, we present a construction of a $\lceil \sqrt{n} \rceil$ -decodable rateless matrix using the OLS method based on [33] using a combination of mutually orthogonal Latin squares and the aforementioned special matrix R_n , which we interchangeably refer to as L_0 . We present the formal definition.

Definition 4.4. [The OLS Matrix] Let $n \in \mathbb{N}$ and $i \in [\lceil \sqrt{n} \rceil]$ be given, where n is the number of columns and i is the decodability parameter. The i -decodable OLS matrix, denoted by $M_{n,i}^{II} \in \{0, 1\}^{m_i \times n}$, is defined as follows:

- (1) Let $s = \lceil \sqrt{n} \rceil$ be a prime power and $m_i = i \cdot s$.
- (2) For each $j \in [i]_0$, define the submatrix $M_j \in \{0, 1\}^{s \times n}$ as:
 - (a) For $j > 0$, use Latin square j of order s , or the special matrix R_s for $j = 0$, such that $R_s \cup \{L_j \mid 1 \leq j < i\}$ is a mutually orthogonal set.
 - (b) For each $k \in [n]_0$:
 - (i) $x = \lfloor \frac{k}{s} \rfloor$ and $y = k \bmod s$.
 - (ii)

$$M_j[x, k] = \begin{cases} 1 & \text{if } x = L_j[x, y] \\ 0 & \text{otherwise} \end{cases}$$

- (3) The OLS matrix $M_{n,i}^{II}$ is the vertical concatenation of the i submatrices M_0, M_1, \dots, M_{i-1} .

The resulting matrix $M_{n,i}^{II}$ is of size $m_i \times n$, where $m_i = i \cdot s$.

In order to show that the OLS matrix $M_{n,d}^{II}$ is a $\lceil \sqrt{n} \rceil$ -decodable rateless matrix, we use the following theorem whose proof is deferred to Appendix F.

THEOREM 4.5. *For n where $\lceil \sqrt{n} \rceil$ is a prime power, and $d = \lceil \sqrt{n} \rceil$, the OLS matrix $M_{n,d}^{II}$ is a $\lceil \sqrt{n} \rceil$ -decodable rateless matrix. For $1 \leq i \leq \lceil \sqrt{n} \rceil$, the decodability profile is $m_i = i \cdot \lceil \sqrt{n} \rceil$.*

4.3 Construction III: Extended Hamming Rateless Matrix

In this section, we present a construction of a 3-decodable rateless matrix based on Extended Hamming code construction from [12]. For any positive integer n (not necessarily a power of 2), we let $m = \lceil \log_2(n) \rceil$. The matrix H'_n consists of the first n columns of all the 2^m binary column vectors of the matrix H'_{2^m} . Thus, H'_n has dimensions $m \times n$, and H_n is also similarly extended to have dimensions $(m+1) \times n$ by adding the all ones row to the matrix H'_n . This ensures that for any n , we use the parity check matrix of the extended Hamming code for the smallest power of 2 greater than or equal to n , and then take only the first n columns of that matrix.

Definition 4.6. [The Extended Hamming Matrix] Let $n \in \mathbb{N}$ be given, where $n \geq 8$ is the number of columns. The 3-decodable Extended Hamming matrix, denoted by $M_{n,3}^{III}$, is defined as follows:

$$M_{n,3}^{III} = \begin{bmatrix} H_n \\ \overline{H'_n} \end{bmatrix},$$

where the matrix $\overline{H'_n}$ is the binary complement of the matrix H'_n . The resulting matrix $M_{n,3}^{III}$ is of size $m_3 \times n$, where $m_3 = 2\lceil \log_2 n \rceil + 1$.

In order to show that the Extended Hamming matrix $M_{n,d}^{III}$ is a 3-decodable rateless matrix, we use the following theorem whose proof is deferred to Appendix F.

THEOREM 4.7. For $n \geq 8$ and $d = 3$, the Extended Hamming matrix $M_{n,d}^{III}$ is a 3-decodable rateless matrix, where the decodability profile is $(m_1, m_2, m_3) = (1, \lceil \log_2 n \rceil + 1, 2\lceil \log_2 n \rceil + 1)$.

5 EXPERIMENTAL EVALUATION

This section examines our proposed CertainSync constructions compared to baseline schemes for set reconciliation and demonstrates the construction's comparable performance to the state-of-the-art scheme, Rateless IBLT [39], under different configurations for each construction (universe size, symmetric difference size, and communication overhead).

Schemes Comparison. We compare our CertainSync constructions (EGH, OLS, and Extended Hamming) which do not require any parametrization or estimators, to other set reconciliation schemes. We evaluate our constructions alongside state-of-the-art Rateless IBLT [39] as a rateless scheme, and Difference Digest [11] and Graphene [30] as non-rateless schemes. In particular, Difference Digest incurs additional communication overhead due to the transmission of a Strata Estimator, which is required to estimate the size of the symmetric difference before reconciliation.

Setup. In the experiments, we refer to the common simplified scenario that one set is a superset of another set as a typical case, as observed in prior work [30], which simplifies the computation of the symmetric difference Δ by merely removing elements of one set from the other. Later, in Section 6, which addresses the blockchain applicability of CertainSync, this assumption is not in use anymore. Fig. 5 illustrates two set scenarios. In our experimental setup, each IBLT cell comprises three 8-byte fields (counter, xorSum and checksum), totaling 24 bytes or 192 bits ($24 \text{ bytes} \times 8$) per cell across all evaluated schemes to ensure a fair comparison. To assess metrics of set reconciliation schemes such as accuracy, communication efficiency, and scalability, we developed an experimental design wherein 10 independent trials were conducted for each scheme. Specifically, for each trial, the set of the receiver of IBLT cells was defined as the complete universe of n elements. In contrast, the set of the sender of IBLT cells was constructed by removing $|\Delta|$ (symmetric difference size) unique, randomly selected elements from the receiver set. The experimental results were analyzed by averaging the metrics across the trials, thereby mitigating potential bias in results due to random



Fig. 5. Set scenarios between two participants.

removal of elements from the sender set. The additional communication overhead of sending a subset of Δ elements from the receiver to the transmitter at the end is neglected, as it does not involve the transmission of IBLT cells and is treated as a constant additive factor across all schemes.

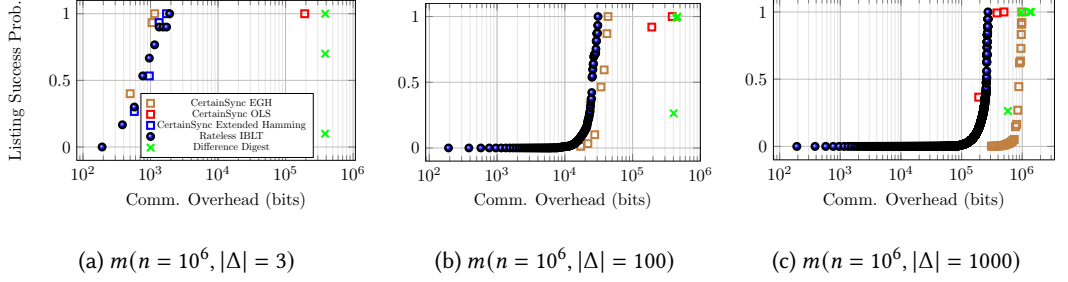


Fig. 6. The trade-off between listing success probability for various schemes as a function of communication overhead (in bits). The universe size is $n = 10^6$ and $|\Delta|$ refers to the symmetric difference size.

Findings. Our experimental results demonstrate that our CertainSync constructions, which are parameterless with a guarantee for successful set reconciliation, achieve comparable performance to state-of-the-art Rateless IBLT scheme across the different configurations, with each construction exhibiting optimal performance for specific symmetric difference size ranges. Specifically, CertainSync EGH and Extended Hamming excel with small to medium symmetric difference size ranges, requiring minimal communication overhead similar to Rateless IBLT, while CertainSync OLS becomes more efficient as the symmetric difference size approaches $\lceil \sqrt{n} \rceil$. The experiments reveal clear trade-offs between decoding accuracy, communication efficiency, and scalability across all schemes, with Rateless IBLT consistently demonstrating superior overall performance but CertainSync constructions offering competitive alternatives for specific use cases as parameterless solutions that eliminate the need for estimators.

5.1 Decoding Accuracy

Fig. 6 presents the listing success probability of various schemes as a function of the communication overhead for different symmetric difference sizes in a linear-log plot. For small symmetric difference sizes, such as $|\Delta| = 3$, CertainSync (EGH and Extended Hamming) and Rateless IBLT demonstrate the lowest communication overhead to reach a success probability of 1. In contrast, CertainSync OLS and Difference Digest exhibit the poorest efficiency with almost double the communication overhead of previous schemes to reach a probability of 1. As the symmetric difference size increases to medium levels ($|\Delta| = 100$), Rateless IBLT maintains superior performance, closely followed by CertainSync EGH. However, CertainSync OLS and Difference Digest continue to underperform, requiring approximately ten times the communication overhead compared to the former schemes to achieve a probability of 1. In the large symmetric difference size regime ($|\Delta| = 1000$), Rateless IBLT remains the most efficient, achieving a success probability converging to 1 as the number of IBLT cells approaches $1.35|\Delta|$. CertainSync OLS shows comparable performance, particularly at lower success probabilities. CertainSync EGH becomes notably inefficient, demanding approximately four times the communication overhead of Rateless IBLT to reach a success probability of 1. Difference Digest maintains the worst efficiency compared to other schemes and reaches a probability of 1 with relatively the same communication overhead as CertainSync EGH. The experimental results reveal a trade-off between symmetric difference size and communication overhead for CertainSync constructions. Each CertainSync construction exhibits an optimal range of symmetric difference

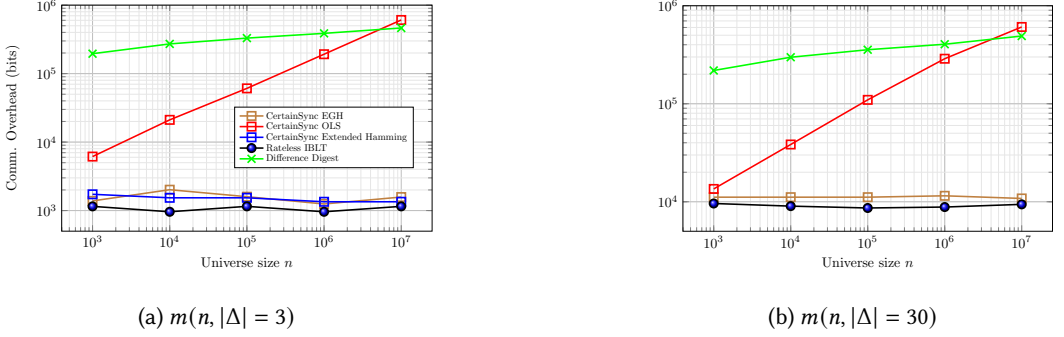
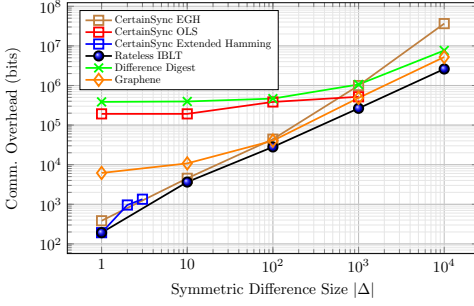


Fig. 7. Trade-off between communication overhead for various schemes as a function of universe size n .

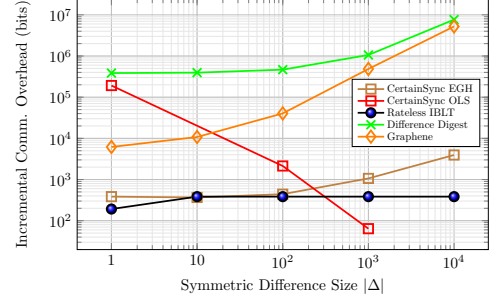
sizes, achieving performance comparable to the Rateless IBLT scheme, which demonstrated the best overall performance. On one hand, CertainSync OLS requires, compared to CertainSync EGH, excessive communication overhead for small to medium symmetric difference sizes ($|\Delta| \ll \lceil \sqrt{n} \rceil = 1000$), due to its dependence on $\lceil \sqrt{n} \rceil$. However, as the symmetric difference size increases, as illustrated for $|\Delta| = 1000$, CertainSync OLS achieves a success probability converging to 1 with significantly lower communication overhead compared to CertainSync EGH, which becomes increasingly inefficient with increasing symmetric difference size.

5.2 Communication Efficiency

Fig. 7 shows the communication overhead of various schemes as a function of the universe size n in a log-log plot. We focus on a small symmetric difference size ($|\Delta| = 3$), and also evaluate larger symmetric difference size ($|\Delta| = 30$) that approaches the theoretical upper bound $|\Delta_{\max}| = \lceil \sqrt{10^3} \rceil = 32$ imposed by CertainSync OLS for minimal universe size $n = 10^3$ experimented with. For $|\Delta| = 3$, Rateless IBLT and CertainSync (EGH and Extended Hamming) exhibit optimal performance with approximately constant growth regardless of the universe size, and require the least communication overhead. For Rateless IBLT, the communication overhead, theoretically based on [39], lies between $1.35|\Delta|$ and $1.72|\Delta|$ on average, confirming that the complexity with respect to the universe size is $O(1)$. In contrast, the theoretical communication overhead complexity for CertainSync Extended Hamming is $O(\log n)$, while for CertainSync EGH it follows $O(\log^2 n)$. Simulation results demonstrate that both constructions achieve practical performance consistent with $O(1)$ complexity, reflecting their dependency on the symmetric difference size ($|\Delta|$) more than the universe size. CertainSync OLS demonstrates a worse performance in comparison to the previous schemes, as it is more susceptible to changes in universe size. Specifically, CertainSync OLS requires communication overhead proportional to $\lceil \sqrt{n} \rceil$, which increases significantly with larger universe sizes. The Difference Digest demonstrates the worst performance in comparison to the other schemes, with the highest communication overhead requirements, even for a small universe size, due to sending an estimator before reconciliation. Fig. 8(a) presents the communication overhead of various schemes as a function of the symmetric difference size $|\Delta|$ in a log-log plot. For low symmetric difference sizes ($|\Delta| < 10$), Rateless IBLT demonstrates the best performance, requiring the least communication overhead, closely followed by CertainSync constructions of EGH and Extended Hamming. Graphene shows moderate efficiency, while CertainSync OLS, with its initial transmission of $\lceil \sqrt{n} \rceil$ cells, and Difference Digest with the transmission of an estimator, perform poorly in this range by having the highest overhead. For medium symmetric



(a) Communication Overhead (in bits)



(b) Incremental Communication Overhead (in bits)

Fig. 8. Communication overhead and incremental communication overhead (in bits) for various schemes as a function of symmetric difference size $|\Delta|$. The universe size is $n = 10^6$.

difference sizes ($10 < |\Delta| < 1000$), Rateless IBLT maintains its superior performance with the lowest communication overhead; Graphene achieves comparable overhead, while CertainSync EGH demonstrates moderate performance. As CertainSync OLS approaches its maximum symmetric difference size $|\Delta|_{max} = \lceil \sqrt{10^6} \rceil = 10^3$, its performance is better than CertainSync EGH. Difference Digest still with the worst performance due to the initial transmission of an estimator. For high symmetric difference sizes ($|\Delta| > 1000$), Rateless IBLT remains the most efficient scheme, requiring the least communication overhead. Graphene performs similarly well but is slightly less efficient than Rateless IBLT. Difference Digest demonstrates moderate performance, and the overhead of transmission of IBLT cells after the estimation phase is comparable to the communication overhead of estimation, thus a rise in overhead as symmetric difference size increases. CertainSync EGH demonstrates the poorest scalability, requiring significantly more communication overhead than all other schemes. Theoretical communication costs almost align with these results. Rateless IBLT achieves $O(|\Delta|)$ complexity, with simulations confirming overhead between $1.35|\Delta|$ and $1.72|\Delta|$ on average, converging to $1.35|\Delta|$ for larger $|\Delta|$. CertainSync EGH, with theoretical complexity $O(|\Delta|^2)$, scales poorly for large $|\Delta|$. CertainSync Extended Hamming exhibits $O(|\Delta|)$ complexity, which makes it effective for small symmetric difference sizes, while CertainSync OLS demonstrates parts of $O(1)$ and $O(|\Delta|)$, which implies superior scaling in parts of $O(1)$ due to a stronger dependence on universe size n . It performs well for medium values of $|\Delta|$. Difference Digest achieves $O(|\Delta|)$ complexity through Strata Estimator based estimation of $|\Delta|$, and Graphene, despite high initial overhead for small $|\Delta|$ as much as 20% higher than its true minimum cost (see Eq. 3 in [30]), converges to $O(|\Delta|)$ for larger symmetric difference sizes.

5.3 Scalability

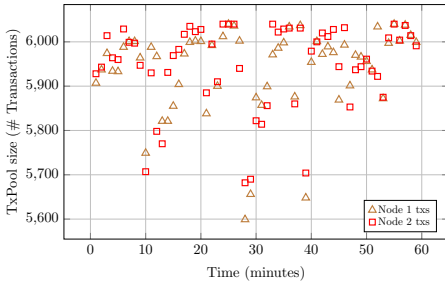
In this context, scalability is measured by the rate at which incremental communication overhead (additional bits) increases concerning the size of the symmetric difference while maintaining a constant universe size. Schemes requiring fewer additional bits as the symmetric difference size grows are considered more scalable. A steeper slope indicates poorer scalability, as the communication overhead increases rapidly. Conversely, a flatter slope indicates better scalability, as the scheme requires fewer additional bits for larger symmetric difference sizes. Fig. 8(b) shows the incremental communication overhead of various schemes as a function of symmetric difference size $|\Delta|$ in a log-log plot. In this simulation, the distinction between rateless and non-rateless schemes becomes evident. The incremental overhead of non-rateless schemes, such as Difference Digest and Graphene,

is the same as the total communication overhead (see Fig. 8(a)) due to their inability to dynamically expand, constrained by either fixed IBLT size based on parametrization (Graphene), or estimation-based allocation (Difference Digest). Consequently, their overall scalability is significantly inferior compared to rateless schemes. Rateless IBLT demonstrates superior overall scalability, exhibiting constant growth in incremental communication overhead as the symmetric difference size increases above $|\Delta| = 10$. CertainSync EGH achieves comparable overhead to Rateless IBLT up to medium symmetric difference size of $|\Delta| = 100$, and beyond it exhibits linear growth, becoming less scalable with complexity $O(|\Delta|)$. CertainSync OLS presents relatively high incremental communication overhead for small symmetric difference sizes due to its dependence on universe size n , performing worse than even the non-rateless Graphene scheme. However, it demonstrates improved scalability compared to non-rateless schemes at medium symmetric difference sizes, and notably, as it approaches its maximum symmetric difference size $|\Delta|_{\max} = \lceil \sqrt{10^6} \rceil = 10^3$, its incremental overhead surpasses that of Rateless IBLT.

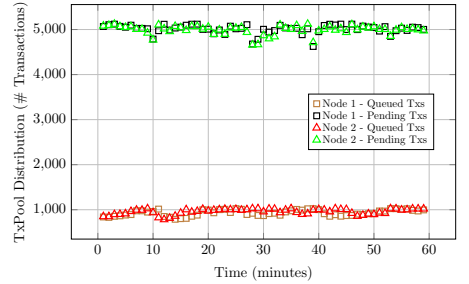
6 CertainSync FOR BLOCKCHAIN SYNCHRONIZATION

6.1 Setup

We utilize an architecture comprising two Ethereum blockchain nodes, each consisting of an execution client and a consensus client. The execution client used is Geth, which facilitates the processing of transactions and the execution of smart contracts on the Ethereum blockchain. The consensus client employed is Prysm, which implements Ethereum’s Proof of Stake (PoS) consensus mechanism. Each node operates independently but participates in the same network. By utilizing the Geth *admin.peers* API, we can confirm that these nodes are not peers of one another. This separation is advantageous, as it might lead to more distinct transaction pools for each node as we mainly focus on synchronization of transactions rather than blocks. For this experiment, we utilize the Sepolia testnet, which is one of the Ethereum test networks. Sepolia provides a sandbox environment that allows testing applications and smart contracts without incurring real costs or risks associated with the main Ethereum network (Mainnet). In blockchain, the transaction pool (TxPool), analogous to



(a) The size of the Ethereum transaction pools (TxPools), combining both queued and pending transactions over time (in minutes).



(b) The count of queued and pending transactions in transaction pools (TxPools) over time (in minutes).

Fig. 9. Characteristics of Ethereum transaction pools (TxPools).

the Mempool in Bitcoin, is a critical component that holds all transactions that have been submitted but not yet included in a block. We collected and analyzed the real content of the TxPools at two Ethereum nodes on November 27, 2024, for an hour (which ended around noon EST). The TxPool is divided into two types: queued transactions, which wait for processing and inclusion in a block,

prioritized by gas price for miners' selection, and pending transactions, which have been selected by a miner and are in the process of being included in a block but are not yet confirmed or added to the blockchain. For the purpose of synchronizing transaction pools, we place less emphasis on transaction types, but rather on the total number of them at any given time as shown in Fig. 9(a). We stick to the default maximum values as specified in the Geth client version 1.14.11-stable-f3c696fa, where the maximum pending transactions (controlled by `txpool.globalslots`) is limited to 1024 transactions, the maximum queued transactions (controlled by `txpool.globalqueue`) is limited to 5120 transactions, and queued transactions are removed after 3 hours (controlled by `txpool.lifetime`). After giving each node enough time to be fully synchronized to the chain, from Fig. 9(b), we can observe that the number of queued transactions is indeed close to 1024 with declines due to the discarding of queued transactions, or due to becoming a pending transaction, while pending transaction count is around 5000 with declines due to appending transactions to a block, and increases due to queued transactions becoming pending. By utilizing this setup, we intend to explore transaction pool synchronization between the two nodes.

6.2 The UniverseReduceSync Framework

6.2.1 Motivation. Each transaction in a blockchain network includes a hash field, which is generated using a cryptographic hash function like SHA256. This function processes the transaction details, such as the transaction value and the sender's and receiver's addresses, to produce a unique identifier. In the Ethereum network, a transaction includes a hash field of 256 bits. At first glance, this implies a universe size of $n = 2^{256}$ and substantial inefficiency in the basic CertainSync framework, with communication overhead affected by the universe size (Table 1). For example, in CertainSync OLS construction, communication overhead is proportional to $\lceil \sqrt{n} = 2^{128} \rceil = 2^{128}$. Therefore, we extend our CertainSync framework to incorporate universe size reduction, resulting in an extended framework for large-scale universe size scenarios such as blockchain networks, which we refer to as *UniverseReduceSync*. By reducing the universe size to a scale proportional to the actual size of the sets involved in synchronization, such as the size of the transaction pools in blockchain networks as demonstrated by using the Ethereum blockchain as a case study, we significantly reduce the communication overhead compared to the basic CertainSync framework.

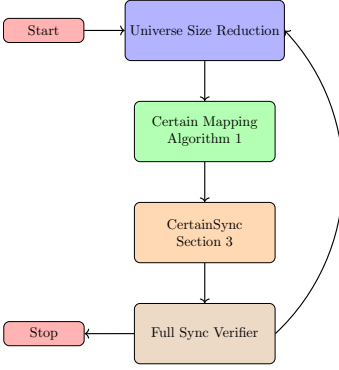
6.2.2 Architecture. In *UniverseReduceSync*, as described in Fig. 10, there is a cyclic process designed to facilitate synchronization in multiple rounds, where in each round CertainSync is used. The key components of the framework are the following:

UniverseSizeReduction. This component estimates the new reduced universe size (n_r) based on the total elements from the original universe size n of sets S_1 and S_2 . Its estimation aims to minimize the number of collisions in the reduced universe below a threshold denoted as δ .

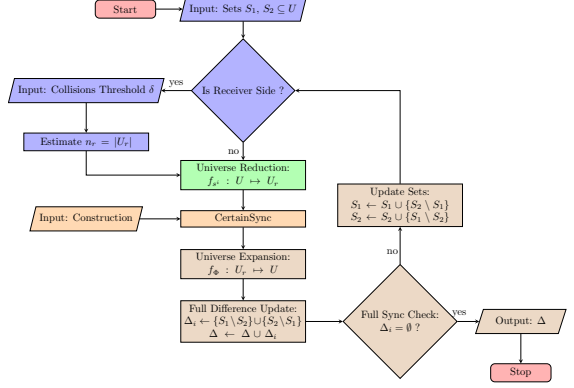
Certain Mapping. In this component, each element e in a set undergoes a mapping using a predefined hash function \mathcal{H} parameterized by a hash salt s^i , where i denotes the round number. The hash salt s^i is generated using an agreed pseudorandom number generator (PRNG), ensuring consistent and reproducible values for s^i among all participants. Specifically, for each element e , we compute a reduced element value $e_r = \mathcal{H}(e, s^i)$. Additionally, it constructs a reverse mapping Φ from the reduced universe back to the original universe, ensuring that each element in the reduced universe can be mapped back to its corresponding original element or elements in case of duplicates.

CertainSync. The CertainSync component performs the synchronization process within the reduced universe, leveraging the reduced elements S_r from the previous step on each side.

Fully Sync Verification. The Fully Sync verification component determines whether the synchronization is fully done. If so, the process yields the symmetric difference Δ . However, if the



(a) Block diagram illustrating the architecture and workflow of the UniverseReduceSync.



(b) The detailed flowchart for UniverseReduceSync. The colors match the components in the block diagram.

Fig. 10. UniverseReduceSync illustration via block diagram and flowchart.

synchronization is not fully done, the process returns to the Universe Size Reduction stage with less data to synchronize than the previous round.

6.2.3 Trade-offs. Table 3 illustrates the key trade-offs between CertainSync and UniverseReduceSync frameworks. For CertainSync, EGH is the sole applicable construction, as it avoids the limitations of Extended Hamming with a small maximum symmetric difference size ($|\Delta_{max}| = 3$), and is significantly less dependent on the universe size n with polylogarithmic communication overhead complexity $O(\log^2 n)$, compared to OLS proportional to $\lceil \sqrt{n} \rceil$. In UniverseReduceSync, while Extended Hamming remains irrelevant due to its small maximum symmetric difference size, both EGH and OLS constructions prove applicable. Moreover, UniverseReduceSync achieves lower communication costs through reduced elements but introduces potential collisions and requires parameter tuning and estimation of the reduced universe size. In contrast, CertainSync offers simpler, collision-free operation with guaranteed single-round synchronization at the expense of higher communication overhead due to the use of elements in the original large universe.

Table 3. Framework Characteristics: CertainSync vs. UniverseReduceSync. **Green** indicates a preferable characteristic, while **red** highlights a less favorable one.

Property \ Framework	CertainSync (Section 3)	UniverseReduceSync (Section 6)
Constructions	EGH	EGH, OLS
Universe Size	Original	Reduced
Communication Complexity	Higher	Lower
Parametrization & Estimation	None	Includes
Collisions	No collisions	Possible within and between participants
Synchronization Rounds	Exactly 1 round	≥ 1 rounds (collision-dependent)
Memory Overhead	Original elements only	Original and reduced elements

6.3 Design of UniverseReduceSync

UniverseSizeReduction. The reduced universe size n_r must be at least $2^{\lceil \log_2(m) \rceil}$ where $m = |S_1| + |S_2|$ serves as an upper bound on $|S_1 \cup S_2|$ due to potential overlaps of same elements. This bound ensures: (i) avoiding guaranteed collisions by the pigeonhole principle when $n_r < m$, (ii) optimal bit representation since any $n_r < 2^{\lceil \log_2(m) \rceil}$ would not fully utilize the minimum bits needed for m distinct values. While $2^{\lceil \log_2(m) \rceil}$ provides sufficient capacity for unique representations, the actual number of collisions depends on the statistical properties of the chosen hash used to map to the reduced universe, the reduced universe size n_r and the hash salt per round s_i .

THEOREM 6.1 (HASH COLLISIONS EXPECTATION). *Let \mathcal{H} be a hash function that maps elements to a universe of size n_r , and let m be the number of elements. Assuming a uniform hash distribution, the expected number of element collisions is $E[\text{Collisions}] = \frac{m(m-1)}{2n_r}$.*

PROOF. The probability of a specific pair's collision is $\frac{1}{n_r}$, with $\binom{m}{2} = \frac{m(m-1)}{2}$ total possible element pairs. Consequently, the expected number of element collisions is $\binom{m}{2} \cdot \frac{1}{n_r} = \frac{m(m-1)}{2n_r}$. \square

To determine the minimal value of n_r , we start with the constraint that the expected number of collisions, given by $E[\text{collisions}] = \frac{m(m-1)}{2 \cdot n_r}$, must not exceed δ . Through algebraic manipulation of this inequality, we find that $n_r \geq \frac{m(m-1)}{2\delta}$. Since n_r must be a positive integer, we take the ceiling function of this expression, which yields $n_r = \left\lceil \frac{m(m-1)}{2\delta} \right\rceil$ as the minimal value.

CertainMapping. Given two sets S_1 and S_2 , if elements $e_1 \in S_1$ and $e_2 \in S_2$ map to the same reduced element e_r under CertainMapping, then either $e_1 = e_2$ or a collision occurred.

Algorithm 1: CertainMapping

Input: Elements $S \subseteq U$, Hash Salt s^i , Reduced Universe size n_r

Output: Reduced elements S_r , Mapping Φ

Initialize $S_r \leftarrow \emptyset$, $\Phi \leftarrow \text{dictionary}\{\}$

for $e \in S$ **do**

$e_r \leftarrow (\mathcal{H}(e, s^i) \bmod n_r) + 1$
 $S_r \leftarrow S_r \cup \{e_r\}$
 $\Phi(e_r) \leftarrow \Phi(e_r) \cup \{e\}$

return (S_r, Φ)

Fully Sync Verifier. Algorithm 5 in Appendix D is extended by leveraging the counter's sign in pure cells to determine the side-association of each reduced element $e_r \in \text{IBLT}\{\Delta\}$. For the first round with an assumption for discrete uniform distribution for hash \mathcal{H} , the symmetric difference size in this round $|\Delta|_1$ is bounded by $\max(0, |\Delta| - 2\delta) \leq |\Delta|_1 \leq |\Delta|$, where the lower bound represents collisions of elements in Δ , and the upper bound represents element collisions in the intersection. Consequently, the probability of a full sync check succeeding is $\mathcal{P}(\text{Success}) \geq \frac{\max(0, |\Delta| - 2\delta)}{|\Delta|}$.

6.4 Results

To enable validation of the correctness of our results, the symmetric difference size $|\Delta|$ is required in advance, but in real time, it is unknown. We collected, as mentioned earlier, the content of TxPools at two Ethereum nodes over time at one-minute intervals, and calculated in advance their symmetric difference size as shown in Fig. 11(a). The transaction unique identifier, which is the hash field, is used as the new element, in contrast to previous experiments with a positive integer. In our

experimental setup, UniverseReduceSync constructions use an IBLT cell structure with an 8-byte counter, 32-byte xorSum, and 32-byte checkSum, totaling 72 bytes or 576 bits per cell. In contrast, CertainSync constructions utilize an IBLT cell with three 8-byte fields (counter, xorSum, checkSum), resulting in 24 bytes or 192 bits per cell. The difference in sizes is due to UniverseReduceSync utilizing original transaction hashes of 256 bits (32 bytes), whereas CertainSync utilizes reduced transaction hashes of 64 bits (8 bytes) for its xorSum and checkSum fields.

In Fig. 11(b) and (c), there is a comparison of the communication overhead for three synchronization schemes: CertainSync EGH (original universe) and UniverseReduceSync EGH & OLS (reduced universe) under varying number of collision constraints $\delta \in \{1, 100\}$. The subplots show the communication overhead as a function of time, with CertainSync EGH consistently requiring the most overhead with the same overhead costs across δ values, UniverseReduceSync EGH consistently requiring the least communication overhead with the same overhead costs across δ values, and UniverseReduceSync OLS demonstrating decreased communication overhead as δ increases, which implies a lower reduced universe size n_r .

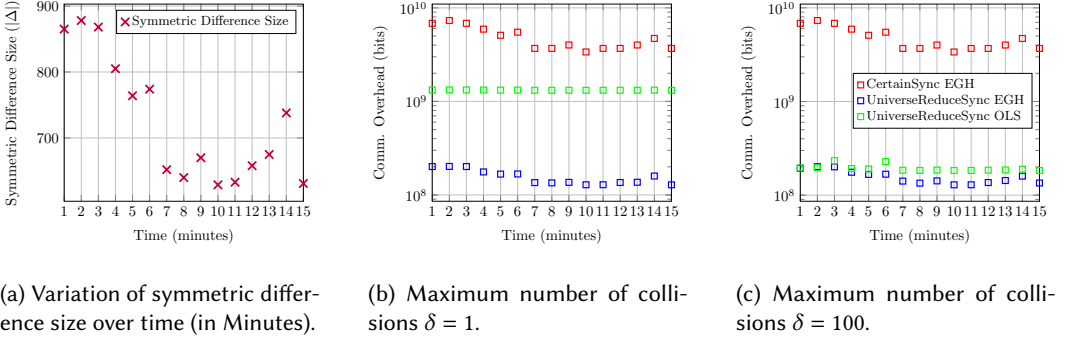


Fig. 11. Symmetric Difference Size $|\Delta|$ and communication overhead (in bits) for various synchronization schemes as a function of maximum number of collisions (δ) and time in minutes.

7 CONCLUSIONS AND FUTURE WORK

This paper introduced CertainSync, a novel framework for set reconciliation that guarantees success when communication overhead reaches a bound derived by the symmetric difference size and the universe size, unlike traditional schemes that offer only statistical guarantees. We proposed three rateless constructions without requiring parametrization or symmetric difference size estimation based on group testing, Latin squares, and error correction codes. We analyzed their performance and validated their effectiveness through experiments compared to other baseline schemes for set reconciliation. Additionally, we presented UniverseReduceSync, an extended framework of CertainSync for large-scale universe size reconciliation to minimize communication overhead. We evaluated it alongside the basic CertainSync framework on the Sepolia Ethereum network to compare the trade-offs between the two frameworks. A natural open question for future work refers to the development of other families of constructions for CertainSync, like combinatorial or recursive, that can be applicable to set reconciliation with certainty. We would also like to study the reconciliation of more than two sets, often known as multi-party set reconciliation, with certainty due to its practical relevance in blockchain networks. Moreover, evaluating the total reconciliation time of CertainSync compared to other baseline set reconciliation schemes and in real-world scenarios, particularly under high network churn, remains an important direction that will further validate the practical applicability.

REFERENCES

- [1] Djamal Belazzougui, Gregory Kucherov, and Stefan Walzer. 2024. Better Space-Time-Robustness Trade-Offs for Set Reconciliation. In *International Colloquium on Automata, Languages, and Programming (ICALP)*.
- [2] Burton H. Bloom. 1970. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM* 13, 7 (1970), 422–426.
- [3] Novak Boskov, Sevval Simsek, Ari Trachtenberg, and David Starobinski. 2023. SREP: Out-Of-Band Sync of Transaction Pools for Large-Scale Blockchains. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*.
- [4] Novak Boskov, Ari Trachtenberg, and David Starobinski. 2022. GenSync: A New Framework for Benchmarking and Optimizing Reconciliation of Data. *IEEE Trans. Netw. Serv. Manag.* 19, 4 (2022), 4408–4423.
- [5] John W. Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost. 2002. Informed content delivery across adaptive overlay networks. In *ACM SIGCOMM*.
- [6] Eunji Choi, Jungwon Lee, Changhoon Yim, and Hyesook Lim. 2024. Decoding Errors in Difference-Invertible Bloom Filters: Analysis and Resolution. *IEEE Access* 12 (2024), 40622–40633.
- [7] Graham Cormode and S. Muthukrishnan. 2004. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. In *Latin American Theoretical Informatics (LATIN)*.
- [8] Changyan Di, David Proietti, I. Emre Telatar, Thomas J. Richardson, and Rüdiger L. Urbanke. 2002. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Inf. Theory* 48, 6 (2002), 1570–1579.
- [9] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. 2006. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *CoRR abs/cs/0602007* (2006).
- [10] David Eppstein, Michael T. Goodrich, and Daniel S. Hirschberg. 2007. Improved Combinatorial Group Testing Algorithms for Real-World Problem Sizes. *SIAM J. Comput.* 36, 5 (2007), 1360–1375.
- [11] David Eppstein, Michael T. Goodrich, Frank C. Uyeda, and George Varghese. 2011. What’s the Difference? Efficient Set Reconciliation without Prior Context. In *ACM SIGCOMM*.
- [12] Tuvi Etzion. 2006. On the Stopping Redundancy of Reed-Muller Codes. *IEEE Trans. Inf. Theory* 52, 11 (2006), 4867–4879.
- [13] Long Gong, Ziheng Liu, Liang Liu, Jun Xu, Mitsunori Ogihara, and Tong Yang. 2020. Space- and Computationally-Efficient Set Reconciliation via Parity Bitmap Sketch (PBS). *Proc. VLDB Endow.* 14, 4 (2020), 458–470.
- [14] Michael T. Goodrich and Michael Mitzenmacher. 2011. Invertible Bloom lookup tables. In *Allerton Conference on Communication, Control, and Computing*.
- [15] M. Y. Hsiao, D. C. Bossen, and R. T. Chien. 1970. Orthogonal latin square codes. *IBM J. Res. Dev.* 14, 4 (1970), 390–394.
- [16] Muhammad Anas Imtiaz, David Starobinski, Ari Trachtenberg, and Nabeel Younis. 2021. Churn in the Bitcoin Network. *IEEE Trans. Netw. Serv. Manag.* 18, 2 (2021), 1598–1615.
- [17] A. Donald Keedwell and József Dénes. 2015. Chapter 5 - The concept of orthogonality. In *Latin Squares and their Applications (Second Edition)*.
- [18] Sándor Z. Kiss, Éva Hosszu, János Tapolcai, Lajos Rónyai, and Ori Rottenstreich. 2021. Bloom Filter With a False Positive Free Zone. *IEEE Trans. Netw. Serv. Manag.* 18, 2 (2021), 2334–2349.
- [19] Ivo Kubjas and Vitaly Skachek. 2020. Failure Probability Analysis for Partial Extraction from Invertible Bloom Filters. *CoRR abs/2008.00879* (2020).
- [20] Francisco Lázaro and Balázs Matuz. 2023. A Rate-Compatible Solution to the Set Reconciliation Problem. *IEEE Trans. Commun.* 71, 10 (2023), 5769–5782.
- [21] Yixin Li, Liang Liang, Yunjian Jia, and Wanli Wen. 2024. Presync: An Efficient Transaction Synchronization Protocol to Accelerate Block Propagation. *IEEE Trans. Netw. Serv. Manag.* 21, 5 (2024), 5582–5596.
- [22] Zhenhua Li, Cheng Jin, Tianyin Xu, Christo Wilson, Yao Liu, Linsong Cheng, Yunhao Liu, Yafei Dai, and Zhi-Li Zhang. 2014. Towards Network-level Efficiency for Cloud Storage Services. In *Internet Measurement Conference (IMC)*.
- [23] Lailong Luo, Deke Guo, Ori Rottenstreich, Richard T.B. Ma, and Xueshan Luo. 2019. Set Reconciliation with Cuckoo Filters. In *ACM International Conference on Information and Knowledge Management (CIKM)*.
- [24] Lailong Luo, Deke Guo, Yawei Zhao, Ori Rottenstreich, Richard T. B. Ma, and Xueshan Luo. 2021. MCFsyn: A Multi-Party Set Reconciliation Protocol With the Marked Cuckoo Filter. *IEEE Trans. Parallel Distributed Syst.* 32, 11 (2021), 2705–2718.
- [25] Aljoscha Meyer. 2023. Range-Based Set Reconciliation. In *International Symposium on Reliable Distributed Systems (SRDS)*.
- [26] Y. Minsky, A. Trachtenberg, and R. Zippel. 2003. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory* 49, 9 (2003), 2213–2218.
- [27] Michael Mitzenmacher and Rasmus Pagh. 2018. Simple multi-party set reconciliation. *Distributed Comput.* 31, 6 (2018), 441–453.
- [28] Avi Mizrahi, Daniella Bar-Lev, Eitan Yaakobi, and Ori Rottenstreich. 2024. Invertible Bloom Lookup Tables with Listing Guarantees. In *ACM Sigmetrics*.

- [29] Patrick Mukherjee, Christof Leng, Wesley W. Terpstra, and Andy Schürr. 2008. Peer-to-Peer Based Version Control. In *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*.
- [30] A. Pinar Ozisik, Gavin Andresen, Brian N. Levine, Darren Tapp, George Bissias, and Sunny Katkuri. 2019. Graphene: Efficient interactive set reconciliation applied to blockchain propagation. In *ACM SIGCOMM*.
- [31] Nalin Ranjan, Zechao Shang, Sanjay Krishnan, and Aaron J. Elmore. 2021. Version Reconciliation for Collaborative Databases. In *ACM Symposium on Cloud Computing (SoCC)*.
- [32] Pedro Reviriego, Salvatore Pontarelli, Alfonso Sánchez-Macián, and Juan Antonio Maestro. 2014. A Method to Extend Orthogonal Latin Square Codes. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 22, 7 (2014), 1635–1639.
- [33] Ori Rottenstreich, Pedro Reviriego, Ely Porat, and S. Muthukrishnan. 2021. Avoiding Flow Size Overestimation in Count-Min Sketch With Bloom Filter Constructions. *IEEE Trans. Netw. Serv. Manag.* 18, 3 (2021), 3662–3676.
- [34] Sebastian Schildt, Johannes Morgenroth, and Lars C. Wolf. 2013. Efficient false positive free set synchronization using an extended Bloom filter approach. *Comput. Commun.* 36, 10-11 (2013), 1245–1254.
- [35] Moshe Schwartz and Alexander Vardy. 2005. On the Stopping Distance and the Stopping Redundancy of Codes. *CoRR abs/cs/0503058* (2005).
- [36] Mounir Tlili, Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. 2010. Scalable P2P Reconciliation Infrastructure for Collaborative Text Editing. In *International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA)*.
- [37] Ari Trachtenberg, David Starobinski, and Sachin Agarwal. 2002. Fast PDA Synchronization Using Characteristic Polynomial Interpolation. In *IEEE INFOCOM*.
- [38] Xiaobo Xing. 2021. Financial Big Data Reconciliation Method. In *International Symposium on Advances in Informatics, Electronics and Education (ISAIEE)*.
- [39] Lei Yang, Yossi Gilad, and Mohammad Alizadeh. 2024. Practical Rateless Set Reconciliation. In *ACM SIGCOMM*.

A ADDITIONAL APPLICATIONS OF SET RECONCILIATION

Set reconciliation is crucial in domains that require efficient data synchronization, particularly in blockchain networks. Minimizing communication overhead for synchronization between parties is essential to ensure efficiency and scalability. Some notable applications include:

Distributed Databases. Large-scale distributed databases like in finance often employ reconciliation to maintain consistency across replicas [38]. Also, schemes for reconciliation in distributed versioned database systems like MindPalace [31] are used for auto-mergeability, where branches may be reconciled (analogous to a branch merging in Git) without human intervention.

Collaborative Editing. Real-time collaborative editing systems use reconciliation techniques to synchronize user changes efficiently, ensuring consistency with minimal conflicts. While centralized systems like Google Docs handle this via a server, P2P collaborative editing in decentralized systems [29, 36] lacks a central mediator. Peers must directly reconcile local changes, making efficient set reconciliation critical for identifying and resolving differences without full document transfers.

B ALTERNATIVE APPROACHES TO SET RECONCILIATION

Advances in set reconciliation have focused significantly on improving communication efficiency and minimizing overhead. In addition to the IBLT-based techniques, three other modern methods have emerged.

Characteristic Polynomial Interpolation (CPI). This method encodes sets as polynomials to minimize the data exchanged during the reconciliation process [26]. CPI has been explored in several works, including research on out-of-band synchronization of transaction pools for large-scale blockchains [3], studies on benchmarking and optimizing data reconciliation [4], and efforts to achieve fast Personal Digital Assistant (PDA) synchronization using CPI [37].

Parity Bitmap Sketch (PBS). This is an ECC-based set reconciliation scheme [13] which reduces both space and computational overheads by leveraging parity bits for detecting and correcting errors, ensuring accurate reconciliation even in the presence of errors. It uses a parity bitmap sketch, which is a compact data structure that encodes set differences using parity information.

Recursive Partitioning and Fingerprinting. This method optimizes the process of computing set unions over a network [25]. It employs a divide-and-conquer approach, recursively partitioning the sets, computing fingerprints for each partition, and comparing fingerprints of the partitions to determine which partition should be sent to the other side.

B.1 Parameterization Tuning & Estimations

Table 4 compares various set reconciliation schemes based on the required parameters and estimators. Notably, CertainSync constructions stand out due to their lack of parameters or estimators. This simplicity makes CertainSync constructions particularly advantageous for applications where minimizing implementation complexity and tuning is critical.

Table 4. Comparison of parametrization and estimators across different set reconciliation schemes. The hash count parameter refers to the number of cells each element should be mapped to.

Scheme	Parameters Needed	Estimators Used
CPI [26]	(i) Upper bound for symmetric diff. size \bar{m}	None
Graphene [30]	(i) Hedge factor τ (ii) Hash count (iii) Success prob. p	(i) IBLT-Param-Search
Difference Digest [11]	(i) Hash count (ii) α	(i) Strata Estimator for symmetric diff. size
Cuckoo Filter [23]	(i) # bucket m (ii) # fingerprint b (iii) Hash count	None
Rateless IBLT [39]	(i) Hash count	None
LFFZ IBLT [28]	(i) Hash count	None
CertainSync EGH	None	None
CertainSync OLS	None	None
CertainSync Extended Hamming	None	None

C NOTATIONS

Table 5. Summary of main notations

Symbol	Meaning
Δ	Symmetric difference
Δm_d	Incremental communication overhead
H_n	Extended Hamming code parity check matrix
H'_n	Submatrix of H_n containing all binary column vectors of length $\log_2(n)$
\mathcal{H}	Hash function
Φ	Reverse mapping to universe reduction
Π_k	The product of first k primes
P_1, P_2	Participant 1 and 2
R_n	Special matrix where each row consists of a single repeated value (0 to $n - 1$)
R_n^T	Transpose of R_n matrix
S	Set of elements
S_r	Set of reduced elements
U	Universe from which elements are selected
$\text{IBLT}\{\Delta\}$	IBLT containing elements from symmetric difference
$\text{MOLS}(n)$	Mutually Orthogonal Latin Squares of order n
M	Binary mapping matrix
$M_{n,d}$	d -decodable rateless matrix
e	Element value
e_r	Reduced element value
m_d	Total communication overhead
$m^*(n, d)$	Minimal number of rows of a d -decodable matrix
n	Universe size - $ U $
n_r	Reduced universe size
$s(M)$	Stopping distance of matrix M
s^i	Hash salt for round i
p_j	The j -th prime number
$[k]$	For positive integer k , denotes the set $\{1, \dots, k\}$
$[k]_0$	For positive integer k , denotes the set $\{0, \dots, k - 1\}$
(m_1, \dots, m_d)	Decodability profile of matrix $M_{n,d}$

D ADDITIONAL ALGORITHMS

We present algorithms that illustrate key concepts discussed in the paper.

Algorithm 2: Traditional Set Reconciliation using IBLTs subtraction [11]

Input: S_1, S_2 (Sets held by Participant 1 (P_1) and Participant 2 (P_2) respectively)
Output: $\Delta = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$ (Symmetric Difference set)
 P_1 estimates the symmetric difference size $|\Delta|$ locally or by communication with P_2 .
 P_1 constructs $IBLT_1$ representing S_1 with $\alpha|\Delta|$ cells, where $\alpha \geq 1$
 P_1 sends $IBLT_1$ to P_2
 P_2 constructs $IBLT_2$ representing S_2 with $\alpha|\Delta|$ cells, where $\alpha \geq 1$
 P_2 calculates $IBLT\{\Delta\} = IBLT_1 \setminus IBLT_2$
 P_2 lists the elements in $IBLT\{\Delta\}$ to find the symmetric difference between the two participants with some success probability p .
 $\Delta = ListElements(IBLT\{\Delta\}, p)$
return Δ

ConstructIBLT algorithm to construct an IBLT by mapping elements from set S using a d -decodable rateless matrix.

Algorithm 3: ConstructIBLT: Construct an IBLT from a set using a d -decodable rateless matrix construction

Input: S (set of elements), i (iteration number)
Output: $IBLT\{S\}$ Cells
 $n \leftarrow$ Universe size
 Initialize IBLT with Δm_i cells, each having fields: count, xorSum, checksum
 /* Assuming x can be indexed */
foreach $x \in S$ **do**
 $k \leftarrow$ Index of x in S
 for $j \leftarrow m_{i-1}$ **to** $m_i - 1$ **do**
 if $M_{n,i}[j][k] == 1$ **then**
 $IBLT[j].count \leftarrow IBLT[j].count + 1$
 $IBLT[j].xorSum \leftarrow IBLT[j].xorSum \oplus x$
 $IBLT[j].checksum \leftarrow IBLT[j].checksum \oplus Hash(x)$
return $IBLT\{S\}$ Cells

IBLTDiff algorithm to compute the IBLT of the symmetric difference ($IBLT\{\Delta\}$) by performing IBLTs subtraction.

Algorithm 4: IBLTDiff: Construct IBLT of Symmetric Difference ($IBLT\{\Delta\}$)**Input:** $IBLT_2$ (IBLT of P_2), $IBLT_1$ (IBLT of P_1), i (iteration number)**Output:** $IBLT\{\Delta\}$

```

for  $j \leftarrow 0$  to  $m_i - 1$  do
     $IBLT_2[j].count \text{ -= } IBLT_1[j].count$ 
     $IBLT_2[j].xorSum \oplus = IBLT_1[j].xorSum$ 
     $IBLT_2[j].checksum \oplus = IBLT_1[j].checksum$ 
return  $IBLT_2$ 

```

DecodeDiff algorithm to recover the symmetric difference Δ by iteratively identifying and removing elements in pure cells from $IBLT\{\Delta\}$ until successful decoding or failure when nonempty cells remain which are not pure.

Algorithm 5: DecodeDiff: List Symmetric Difference**Input:** $IBLT\{\Delta\}$ (IBLT of symmetric difference), i (iteration number)**Output:** Δ (Symmetric difference) or FAIL $\Delta \leftarrow \emptyset$ **while true do**

```

    /* Peeling Decoder - retrieve symbol from pure cell if found */

```

```

     $symbol \leftarrow \text{peelingDecoder.decode}(IBLT\{\Delta\})$ 

```

```

    if  $symbol$  is None then

```

```

        if  $IBLT\{\Delta\}$  is not empty then

```

```

            return FAIL

```

```

        else

```

```

            break

```

```

    Add  $symbol$  to  $\Delta$ 

```

```

    /* Remove  $symbol$  from IBLT cells */

```

```

    for  $j \leftarrow 0$  to  $m_i - 1$  do

```

```

        if  $M_{n,i}[j][symbol - 1] == 1$  then

```

```

             $IBLT\{\Delta\}[j].remove(symbol)$ 

```

```

return  $\Delta$ 

```

E ADDITIONAL EXAMPLES OF CERTAINSYNC CONSTRUCTIONS

We present a collection of examples that demonstrate various concepts discussed in different sections throughout the paper.

Example (EGH Rateless Matrix).

For $n = 5$ columns, we construct a 2-decodable rateless matrix $M_{5,2}^I$ using the EGH method.

Step 1: We need to find the smallest integer k_2 such that the product of the first k_2 prime numbers Π_{k_2} is greater than or equal to n^2 . For $n = 5$ and $i = 2$, it is possible to show that the value of k_2 is 3, since $\Pi_3 = 30 \geq n^2 = 25$.

Step 2: Let $m_2 = \sum_{j=1}^{k_2} p_j$, where p_j is the j -th prime. The first three primes are 2, 3, 5: $m_2 = 2 + 3 + 5 = 10$.

Step 3: For each $j \in [k_2]$, we define the submatrix $M_j \in \{0, 1\}^{p_j \times n}$ as:

$$M_j[x, y] = \begin{cases} 1 & \text{if } y + 1 \equiv x \pmod{p_j} \\ 0 & \text{otherwise} \end{cases}$$

for $x \in [p_j]_0$ and $y \in [n]_0$.

The resulting submatrices are:

$$M_1 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}, M_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, M_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Step 4: Finally, the EGH matrix $M_{5,2}^I$ is formed by the vertical concatenation of these submatrices:

$$M_{5,2}^I = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

The resulting matrix $M_{5,2}^I$ is of size 10×5 , where $m_2 = 10$.

Example (OLS Rateless Matrix). For $n = 6$ columns, we construct a 3-decodable rateless matrix $M_{6,3}^{II}$ using the OLS method.

Step 1: We calculate the value $s = \lceil \sqrt{6} \rceil = 3$, which is a prime (also prime power by definition), and $m_i = i \cdot s = 3 \cdot 3 = 9$.

Step 2: For each $j \in \{0, 1, 2\}$, we define the submatrix $M_j \in \{0, 1\}^{s \times n}$ as:

For $j = 0$

a) Use the special matrix R_3 :

$$R_3 = L_0 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix}$$

b) For each $k \in [6]_0$:

(1) $x = \lfloor k/s \rfloor = \lfloor k/3 \rfloor$

(2) $y = k \bmod s = k \bmod 3$

(3) Set $M_0[x, k] = 1$ if $x = R_3[x, y]$, otherwise 0.

Resulting M_0 :

$$M_0 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For $j = 1$

a) Use the Latin square L_1 :

$$L_1 = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix}$$

b) For each $k \in [6]_0$:

(1) $x = \lfloor k/s \rfloor = \lfloor k/3 \rfloor$

(2) $y = k \bmod s = k \bmod 3$

(3) Set $M_1[x, k] = 1$ if $x = L_1[x, y]$, otherwise 0.

Resulting M_1 :

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

For $j = 2$

a) Use the Latin square L_2 :

$$L_2 = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix}$$

b) Similar to the construction of M_1 , the resulting M_2 :

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

It is easily seen that $R_3 \cup \{L_j \mid 1 \leq j < 3\}$ is a mutually orthogonal set.

Step 3: The OLS matrix $M_{6,3}^{II}$ is formed by the vertical concatenation of these submatrices:

$$M_{6,3}^{II} = \begin{bmatrix} M_0 \\ M_1 \\ M_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

The resulting matrix $M_{6,3}^{II}$ is of size 9×6 , where $m_2 = 9$.

Example (Extended Hamming Rateless Matrix). For $n = 8$ columns, using the Extended Hamming method, we construct the 3-decodable rateless matrix $M_{8,3}^{III}$, given as follows:

$$M_{8,3}^{III} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

The resulting matrix $M_{8,3}^{III}$ is of size 7×8 .

Example. Suppose we want to transmit a set of source symbols. Let the set of Participant 1 (P_1) be $S_1 = \{1\}$.

Encoding. We start by encoding the source symbols into a chunk of IBLT cells that represent the code symbols. The mapping itself is performed with Construction I (EGH). Let us assume that the universe size n is 5. Also, we use the simplifying assumption mentioned above, where the set of Participant 2 (P_2) be S_2 which is a superset of S_1 . Specifically, let $S_2 = \{1, 2, 4\}$ such that the symmetric difference size $|\Delta| = 2$.

The mapping matrix $M_{n,d}^I$ is responsible for mapping elements of S_1 and S_2 to IBLT cells, where at each iteration, some amount of IBLT cells (each cell here denoted with C_i where i is its number) are transmitted between the participants. For each iteration, a specific number of rows from the mapping matrix is used for encoding, as the total number of rows is unknown in advance, as the symmetric difference size is unknown in advance.

$M_{5,2}^I$	1	2	3	4	5
$C1$	0	1	0	1	0
$C2$	1	0	1	0	1
$C3$	0	0	1	0	0
$C4$	1	0	0	1	0
$C5$	0	1	0	0	1
\vdots			...		

In this example with EGH, prime numbers are used such that each amount of IBLT cells transmitted at each iteration is a prime number. Here, P_1 constructs cells according to algorithm 3 in Appendix D, and transmits the first 2 cells, then 3, 5, and so on. Note that the elements row is not a field of an IBLT cell - just for convenience to see which elements each cell represents.

Iteration 1	C1	C2
count	0	1
xorSum	0	1
checkSum	0	$H(1)$
Elements	-	1

Iteration 2	C3	C4	C5
count	0	1	0
xorSum	0	1	0
checkSum	0	$H(1)$	0
Elements	-	1	-

Iteration 3	5 cells
count	...
xorSum	...
checkSum	...
Elements	...

Transmission. P_1 transmits at each iteration a prime amount of cells in ascending order (2, 3, 5, 7...) and waits for acknowledgment from P_2 before sending another chunk of a prime number of cells.

Reception and Decoding. P_2 collects the transmitted cells at each iteration. In our example, after the first two iterations, the receiver has collected 5 cells:

	Iteration 1		Iteration 2		
	C1	C2	C3	C4	C5
count	0	1	0	1	0
xorSum	0	1	0	1	0
checkSum	0	$H(1)$	0	$H(1)$	0
Elements	-	1	-	1	-

P_2 saves P_1 's cells it has received and constructs more cells of its own $IBLT_2$ from its set $\{1, 2, 4\}$ using the same mapping matrix as P_1 .

	Iteration 1		Iteration 2		
	C1	C2	C3	C4	C5
count	2	1	0	2	1
xorSum	$2 \oplus 4$	1	0	$1 \oplus 4$	2
checkSum	$H(2) \oplus H(4)$	$H(1)$	0	$H(1) \oplus H(4)$	$H(2)$
Elements	2, 4	1	-	1, 4	2

P_2 calculates $IBLT\{\Delta\}$ according to algorithm 4 in Appendix D.

	Iteration 1		Iteration 2		
	C1	C2	C3	C4	C5
count	2	0	0	1	1
xorSum	$2 \oplus 4$	0	0	4	2
checkSum	$H(2) \oplus H(4)$	0	0	$H(4)$	$H(2)$
Elements	2, 4	-	-	4	2

P_2 performs listing according to algorithm 5 in Appendix D to the calculated $IBLT\{\Delta\}$, and in case of failure, asks P_1 for more IBLT cells; otherwise, yielding the symmetric difference set $\Delta = \{2, 4\}$. This means that elements $\{2, 4\}$ are present in P_2 's set but not in P_1 's, thus representing the symmetric difference between the two sets. In this example, P_2 asks P_1 for more cells after iteration 1 due to the lack of pure cells in iteration 1 of $IBLT\{\Delta\}$.

F ADDITIONAL PROOFS

We present mathematical properties and proofs of CertainSync constructions, that establish the certainty of those constructions for set reconciliation, including the EGH, OLS, and Extended Hamming. As stated in [28] (Corollary 2) and derived from Lemma 1 in [18], for the EGH matrix from Definition 4.2, the following holds:

COROLLARY F.1. *For n and d , the EGH matrix $M_{n,d}^I$ is a $(d+1)$ -decodable matrix.*

Moreover, as shown in [10] (Theorem 1), there is an upper bound on the number of rows in the EGH matrix $M_{n,i}^I$, which is given by $m(n, i) < \frac{\lceil 2i \ln n \rceil^2}{2 \ln \lceil 2i \ln n \rceil} \cdot \left(1 + \frac{1.2762}{\ln \lceil 2i \ln n \rceil}\right)$. This upper bound results in the following asymptotic complexity $m(n, i) = O\left(\frac{i^2 \log^2 n}{\log i + \log \log n}\right)$. The incremental number of rows, Δm_i , of the EGH matrix, also has an upper bound, as shown in Theorem 1 in [10], and is given by $\Delta m_i < \frac{\lceil 2i \ln n \rceil}{2 \ln \lceil 2i \ln n \rceil} \cdot \left(1 + \frac{1.2762}{\ln \lceil 2i \ln n \rceil}\right)$. Thus, asymptotically we have that $\Delta m_i = O\left(\frac{i \log n}{\log i + \log \log n}\right)$. In order to show that the EGH matrix $M_{n,d}^I$ is also a $(d+1)$ -decodable rateless matrix, we use the observation that for i and n , the EGH matrix $M_{n,i}^I$ is a submatrix of the EGH matrix $M_{n,i+1}^I$.

THEOREM F.2. *For n and d , the EGH matrix $M_{n,d}^I$ is a $(d+1)$ -decodable rateless matrix, where for $2 \leq i \leq d+1$, the decodability profile is $m_i = \sum_{j=1}^{k_i} p_j$, where k_i is the smallest integer such that $\Pi_{k_i} \geq n^i$. Furthermore, $m_1 = m_2$.*

PROOF. To prove that the EGH matrix $M_{n,d}^I$ is a $(d+1)$ -decodable rateless matrix, we need to show that there exist positive integers $m_1 \leq m_2 \leq \dots \leq m_{d+1}$ such that for each $1 \leq i \leq d+1$, the $m_i \times n$ submatrix formed by the first m_i rows of the EGH matrix is i -decodable. For $2 \leq i \leq d+1$, it holds that the submatrix of $M_{n,d}^I$ which is formed by the first $m(n, i)$ rows of the matrix $M_{n,d}^I$ is the matrix $M_{n,i-1}^I$ and hence, by Corollary F.1 it is i -decodable. Lastly, since we don't have a submatrix of $M_{n,1}^I$ which is 1-decodable we simply set $m_1 = m_2$. \square

As stated in [28] (Corollary 2), and based on Theorem 3.1 from [33], for the OLS matrix from Definition 4.4, the following holds:

COROLLARY F.3. *For n and d , the OLS matrix $M_{n,d}^{II}$ is a d -decodable matrix.*

THEOREM F.4. *For n where $\lceil \sqrt{n} \rceil$ is a prime power, and $d = \lceil \sqrt{n} \rceil$, the OLS matrix $M_{n,d}^{II}$ is a $\lceil \sqrt{n} \rceil$ -decodable rateless matrix, where for $1 \leq i \leq \lceil \sqrt{n} \rceil$, the decodability profile is $m_i = i \cdot \lceil \sqrt{n} \rceil$.*

PROOF. In order to prove that the OLS matrix $M_{n,d}^{II}$ is a $\lceil \sqrt{n} \rceil$ -decodable rateless matrix, we need to show that there exist positive integers $m_1 \leq m_2 \leq \dots \leq m_{\lceil \sqrt{n} \rceil}$ such that for each $1 \leq i \leq \lceil \sqrt{n} \rceil$, the $m_i \times n$ submatrix formed by the first m_i rows of the OLS matrix is i -decodable. For $1 \leq i \leq \lceil \sqrt{n} \rceil$, it holds that the submatrix of $M_{n,d}^{II}$ which is formed by the first $i \cdot s$ rows of the matrix $M_{n,d}^{II}$ is the matrix $M_{n,i}^{II}$ and hence, by Corollary F.3 it is i -decodable. \square

Following Theorem F.4, the incremental number of rows is $\Delta m_i = m_i - m_{i-1} = \lceil \sqrt{n} \rceil$.

In order to show that the Extended Hamming matrix $M_{n,3}^{III}$ from Definition 4.6 is a 3-decodable rateless matrix, we use the property that the stopping redundancy of an extended Hamming code of length 2^m is $2m - 1$, as proved in Theorem 1 by [12], and according to Theorem 4 from [28], there exists a 3-decodable matrix with $2m - 1$ rows.

THEOREM F.5. *For $n \geq 8$ and $d = 3$, the Extended Hamming matrix $M_{n,d}^{III}$ is a 3-decodable rateless matrix, where the decodability profile is $(m_1, m_2, m_3) = (1, \lceil \log_2 n \rceil + 1, 2\lceil \log_2 n \rceil + 1)$.*

PROOF. In order to prove that the Extended Hamming matrix $M_{n,d}^{III}$ is a 3-decodable rateless matrix, we need to show that there exist positive integers $m_1 \leq m_2 \leq m_3$ such that for each $1 \leq i \leq 3$, the $m_i \times n$ submatrix formed by the first m_i rows of the Extended Hamming matrix is i -decodable. For $i = 1$, the submatrix $M_{n,1}^{III}$ consists of the first row of $M_{n,3}^{III}$. This row is a vector of all ones, and thus it is 1-decodable because each column has a weight of 1. For $i = 2$, the submatrix $M_{n,2}^{III}$ consists of the first $\lceil \log_2 n \rceil + 1$ rows of $M_{n,3}^{III}$. According to the definition of the Extended Hamming matrix, each pair of distinct columns in this submatrix differs in at least one row, implying that the matrix is 2-decodable; for any pair of columns, there exists at least one row with a weight of 1. For $i = 3$, it holds that the matrix $M_{n,3}^{III}$ is 3-decodable as it includes the rows of the specific parity check matrix presented in [12] (Corollary 1), which is 3-decodable based on Theorem 4 from [28]. \square