An Algorithm for Persistent Homology Computation Using Homomorphic Encryption

Dominic Gold[1], Koray Karabina[2,3], and Francis C. Motta[1]

[1] Florida Atlantic University, Boca Raton, FL, USA
dgold2012@fau.edu, fmotta@fau.edu
[2] National Research Council Canada, Ottawa, Ontario, CA
[3] University of Waterloo, Waterloo, Ontario, CA
koray.karabina@nrc-cnrc.gc.ca

Abstract. Topological Data Analysis (TDA) offers a suite of computational tools that provide quantified shape features in high dimensional data that can be used by modern statistical and predictive machine learning (ML) models. In particular, persistent homology (PH) takes in data (e.g., point clouds, images, time series) and derives compact representations of latent topological structures, known as persistence diagrams (PDs). Because PDs enjoy inherent noise tolerance, are interpretable and provide a solid basis for data analysis, and can be made compatible with the expansive set of well-established ML model architectures, PH has been widely adopted for model development including on sensitive data, such as genomic, cancer, sensor network, and financial data. Thus, TDA should be incorporated into secure end-to-end data analysis pipelines. In this paper, we take the first step to address this challenge and develop a version of the fundamental algorithm to compute PH on encrypted data using homomorphic encryption (HE).

Keywords: homomorphic encryption, topological data analysis, secure computing, persistent homology, applied cryptography, privacy enhancing technology

1 Introduction

Topological Data Analysis (TDA) has blossomed into a suite of computational tools, built on firm mathematical theory, that generate quantified, discriminating, shape-based features of data, which can provide interpretable representations of high dimensional data and be taken in by modern statistical and predictive ML models. To apply the flagship approach, known as persistent homology (PH), data—usually in the form of point clouds or scalar-functions defined on a mesh (e.g., images, time series)—are transformed into a binary matrix that encodes the evolution of a family of simplicial complexes. From this matrix a collection of persistence diagrams (PDs) can be derived through a simple reduction algorithm. PDs provide compact representations of the number and size of geometric/topological structures in the data as multisets of planar points, and can be equipped with natural metrics. Models can then be developed either directly

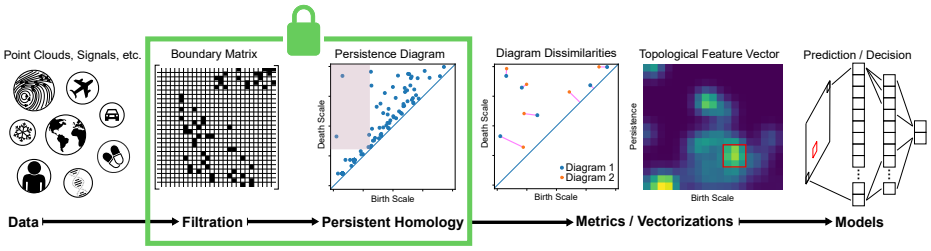

Fig. 1. The TDA-ML Pipeline. Data is first transformed into a family of topological spaces encoded in a binary matrix called a boundary matrix, and then the boundary matrix is transformed into compact representations of the topological structures in the data called persistence diagrams. The distances between diagrams are computed or further transformations produce topological feature vectors. Finally, topological feature vectors are input into downstream models for a desired task. The green box indicates the contribution of this paper, securing the boundary matrix to persistence diagram step in the pipeline.

on PDs using, for example, hierarchical clustering or k-medoids in the metric space of diagrams for classification tasks, or subsequent transformations can be applied to produce topological feature vectors [2, 7, 9, 18, 47, 49] to be used with ML model architectures such as random forests, support vector machines, and neural networks. We refer to these steps as the TDA-ML pipeline, as illustrated in Figure 1.

Crucial to the use of PH in applications are the numerous stability results that establish—under a variety of assumptions about the data and the metrics placed on the data and the PDs—the (Lipschitz) continuity of the map sending data to PDs [11,19,53] and to feature vectors [2]. Due its inherent noise tolerance and suitability across domain and data types, PH has been widely adopted for model development including on sensitive data, such as genomic [52], cancer [8], sensor network [62], and financial data [26]. The reader may refer to recent review articles for references to a variety of PH applications [10, 44].

As the scale of predictive models and their data demands grow, there is pressure to move to collaborative and cloud-based systems in which analysis is performed remotely and in a distributed fashion (e.g., federated learning [40, 42]). This is especially true for propriety and sensitive models that require large training data. On the other hand, a user—be it an independent data producer or agent lacking the capabilities demanded by the models—may need to keep private both their data and the decisions informed by that data. Thus, there is a growing need in industry and government for efficient, secure end-to-end data analysis pipelines that protect vital information on which sensitive decisions are made; to protect privacy, ensure compliance with personal data management regulations, and prevent hostile interference or misuse.

Example application domains, where bridging topological data analysis and secure end-to-end algorithms will yield more efficient, privacy-preserving, and

robust applications where data analysis, data mining, statistical inference and pattern recognition tasks are performed on private data collected from a large number of, and potentially competing, parties include video surveillance for law enforcement, location and energy use tracking for smart cities and autonomous vehicles [6, 56], financial data [26], and biomedical data such as genomics [52] and cancer [8], to name a few.

In order to address challenges with outsourcing sensitive data analysis, cryptographic researchers have been developing secure multiparty computing tools since the 1980s [61]. A good portion of the theoretical foundations of these primitives have been successfully adapted for practical applications in industry [63]. For example, recent innovations in homomorphic encryption (HE) have expanded the variety and complexity of the operations and algorithms that can compute on encrypted data (e.g., comparisons and conditionals [16, 17, 31, 55]. Secure multiparty computing tools are nowadays interacting with privacy-preserving machine learning (ML) applications [12, 20, 36]. Indeed, there has been a recent surge in the development of secure ML algorithms using HE [4, 24, 34, 41]. Thus, HE promises to expand to support complex algorithms and models that protect the privacy of both input data and model outputs. Similarly, sensitive data may be outsourced to a third party database management system (DBMS), where data owner may not fully trust DBMS but still request DBMS to perform some relational operations on the data such as sort, join, union, intersect, and difference. Specialized (symmetric key) encryption schemes allow data owners to encrypt their data, while preserving the ability of DBMS to perform such operations over the encrypted data [28, 29, 48, 50].

In practice, a hybrid use of public key and symmetric encryption schemes are complementary in creating secure and trustworthy data analytical services and applications, which take encrypted data and perform both training and inference on it. Many such models have been performed this way, like logistic or ridge regression [12, 20, 25, 39, 43], support vector machines [3, 46], random forests [30, 37, 58, 64], and even neural networks [27, 35, 59, 60]. The dual benefits of an HE framework for ML model training and inference are that while the client protects their data, the server protects their models that take in this encrypted data. In the TDA-ML pipeline (Fig. 1), both feature generation and model training/evaluation on those features represent critical components of the model development and deployment. Thus, each step back in the pipeline that can be realized in an HE framework relaxes the preprocessing demands on the client and strengthens the protection of the server's model. Thus securing the boundary matrix to persistence diagram step (green box in Fig. 1) is a critical step to allow a Server to fully protect any model that uses topological data features.

Our contributions: We develop `HE-Reduce` (Algorithm 6) as a first-of-its-kind version of the boundary matrix reduction algorithm (`Reduce`, Algorithm 1), which is at the heart of PH and TDA, and which is suitable for secure computation using HE. We achieve this by modifying the logical structure of Algorithm 1 and by developing new arithmetic circuits to replace its computational

and conditional statements. As a result, `HE-Reduce` traces essentially the same steps as in `Reduce` but in a HE-friendly manner so that computations can be performed securely in the ciphertext space. We prove the correctness of our proposed algorithm and provide a complexity analysis. Our analysis is constructive and provides lower bounds on the implementation parameters that guarantee correctness. We implement our algorithms using the CKKS scheme from the OpenFHE library [5] but our techniques can be adapted for other HE schemes by implementing a compatible comparison function using BGV/BFV or TFHE schemes at a comparable cost; see [32]. Finally, we highlight some limitations of our proposed algorithm and suggest some improvements together with some empirical evidence.

Outline: The rest of this paper is organized as follows. Section 2 establishes the mathematical and computational preliminaries of PH and HE. Section 2 also outlines the main challenges associated with transforming `Reduce` to `HE-Reduce`. In Section 3, we establish an HE-compatible version of the boundary matrix reduction algorithm, presented in Algorithm 6, and establish conditions guaranteeing correctness. Section 4 provides a complexity analysis for Algorithm 6 and notes on the implementation, including limitations of the proposed algorithm and potential improvements. Our plaintext implementation of Algorithm 6 in Section 4.4 simulates an implementation of `HE-Reduce` using HE, verifies the correctness of our theoretical results, and provides some positive evidence for improvements. Our experiments showcase the propagation of errors due to relaxing algorithm parameters; see Figure 3. We make concluding remarks in Section 5 concerning potential future research thrusts in secure TDA. In some cases, we have deferred technical proofs to the Appendix.

2 Preliminaries

Our approach to adapting the PH boundary matrix reduction algorithm into a secure framework is to encrypt the input to the reduction algorithm and to allow computations to be performed on ciphertexts in such a way that the decrypted output of the algorithm is equivalent to the output of the algorithm running on the plaintext input. In Section 2.1, we provide some necessary background information on PH and present the main PH boundary matrix reduction algorithm in Algorithm 1. In Section 2.2, we present an overview of HE and explain some of the challenges that would occur when developing a cryptographic version of Algorithm 1 based on HE.

We denote vectors and matrices with boldface, as in $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{R} \in \mathbb{R}^{n\times n}$, and denote the i-th components of vectors with brackets, e.g., $\mathbf{v}[i]$, and columns of matrices with subscripts, $\mathbf{R}_i$. We denote the infinity norm of $\mathbf{v}$ by $|\mathbf{v}| = \|\mathbf{v}\|_\infty = \max_i \big|\mathbf{v}[i]\big|$. We then define the following metric between any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ in the usual manner:

$$|\mathbf{x} - \mathbf{y}| = \|\mathbf{x} - \mathbf{y}\|_\infty = \max_i \big|\mathbf{x}[i] - \mathbf{y}[i]\big|,$$

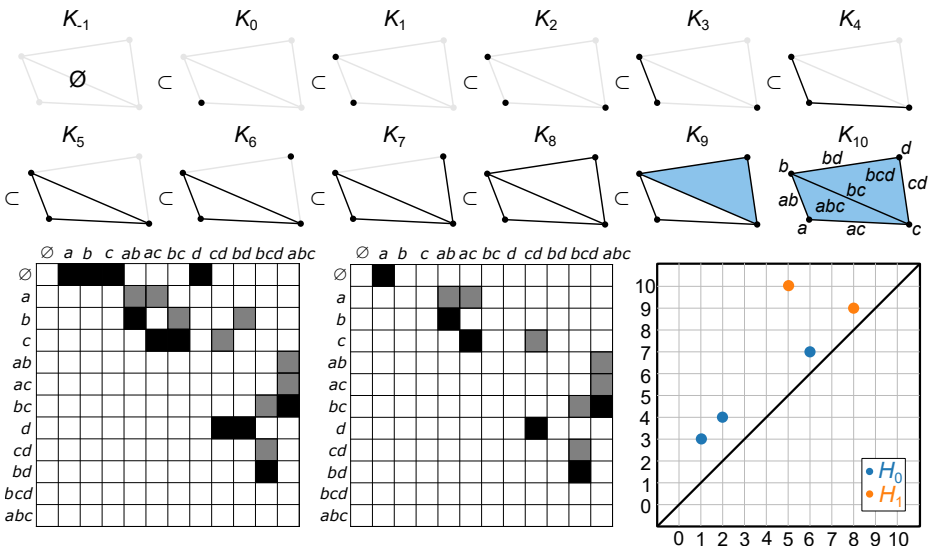

Fig. 2. (Rows 1,2) Example filtration given by an ordering of the simplices in a simplicial complex that consists of 4 points, 5 edges, and 2 triangles. (Row 3) From left to right, the exact binary boundary matrix, the binary reduced boundary matrix, and the H_0 and H_1 persistence diagrams corresponding to the given filtration. White boxes in the matrices indicate 0s and shaded boxes represent 1s, with the lowest 1 in each column shaded with black.

where $|\cdot|$ in the final expression is the usual absolute value of a real number. Furthermore, for $\mathbf{v} \in [0,1]^n$, we denote $l_{\mathbf{v}} = low(\mathbf{v})$ as the integer-valued maximum index containing a 1 to help ease notation when appropriate.

2.1 Persistent Homology

PH, a mathematical device from algebraic topology, provides a means of comparing data through its latent shape-based structures. This is achieved by first associating to a dataset an ordered, nested family of combinatorial objects that are equipped with well-defined notions of shape. In particular, these shape features will be representations of k-dimensional holes in the data. Intuitively, a k-dimensional hole is a vacancy left by a $(k+1)$-dimensional object whose k-dimensional boundary remains. In this way, PH can be regarded as a feature extraction tool which pulls from data topological/geometric features which may provide scientific insights and can be used to train discriminating or predictive models. Although there are different forms of (persistent) homology theory, we restrict our attention to simplicial homology because of its intuitive appeal and practical computability.

Definition 1. *An abstract simplicial complex, K, is a finite collection of finite subsets (called simplices) such that if $\sigma \in K$, then $\tau \in K$ for all $\tau \subset \sigma$. A k-*

simplex, or a simplex of dimension k, is a set of size $k+1$, and the dimension of a complex, $dim(K)$, is the maximum dimension of any of its simplices. A proper subset, $\tau \subsetneq \sigma \in K$, is called a face of σ. If τ is a codimension-1 face of σ, i.e., $\tau \subset \sigma \in K$ and $|\tau| = |\sigma| - 1$, we call τ a boundary face of σ. For simplicity, we will denote the k-simplex $\{x_0, x_1, \ldots, x_k\}$ by $x_0 x_1 \ldots x_k$.

One may regard 0-simplices (singleton sets) as points in some Euclidean space, 1-simplices (pairs) as edge segments between points, 2-simplices (sets of size 3) as filled-triangles, 3-simplices (sets of size 4) as filled tetraheda and so on, with the requirement that simplices in the geometric realization intersect only along common faces. Figure 2 illustrates such geometric realizations of abstract simplicial complexes. For example, K_5 is the geometric realization of the abstract simplicial complex $\{\emptyset, a, b, c, ab, ac, bc\}$. The empty triangle formed by the edges ab, bc, and ac at index 5 in Figure 2 provides an example of a 1-dimensional hole formed by the vacancy of the missing 2-simplex, abc, enclosed by its three boundary edges, ab, ac, and bc. The holes in a simplicial complex, K are collected into a group, denoted $H_1(K)$, composed of equivalence classes of collections of 1-simplices that form cycles (e.g., ab, ac, and bc in K_5) that could be the boundary faces of some collection of 2-simplices, but aren't. Similarly, a collection of triangles in K that enclose a void become represents of elements in $H_2(K)$. More generally, for each dimension k, the k-dimensional homology group $H_k(K)$ comprises equivalence classes of k-dimensional cycles that are not boundaries of a collection of $(k+1)$-dimensional simplices. $H_0(K)$ encodes the connected components of K.

By ordering the simplices of a simplicial complex so that no simplex appears before any of its faces, one forms a nested sequence of simplicial complexes, which we'll call a *filtration*. Across this filtration one can track which simplices gave birth to homological features and which simplices kill off those homological features to determine (birth, death) pairs that track the persistence of each homological feature. For example, in Figure 2, $H_1(K_4)$ is trivial since K_4 contains no holes. This is in contrast to the complexes $K_5 - K_9$ that have a non-trivial H_1 element represented by the boundary edges ab, bc, and ac that was born with the introduction of bc at index 5. In K_8 there appears another hole with the introduction of the edge bd, which then disappears in K_9 when the triangle bcd fills the cycle formed by bc, bd, cd.

In practice one usually defines a complex, K, from a dataset and computes a filtration from a real-valued function $f : K \to \mathbb{R}$ that satisfies $f(\tau) \leq f(\sigma)$ if $\tau \subseteq \sigma \in K$. f encodes the 'scales' at which each simplex appears in the filtration gotten by ordering simplices according to their scales and sorting ties arbitrarily while ensuring each simplex never appears before its faces. A multitude of methods have been proposed to derive such filtrations [45], both from point cloud data (e.g., Vietoris-Rips filtration [65], alpha filtration [21]) and related filtrations for functions on a cubical mesh [57]. However determined, the structures in the filtration can be encoded in a square, binary matrix $\mathbf{\Delta}(\mathbf{K})$ called a *boundary matrix*, whose rows and columns are indexed by the simplices in K, ordered $\sigma_1, \ldots, \sigma_n$ so that $i < j$ if $f(\sigma_i) < f(\sigma_j)$ or if $\sigma_i \subset \sigma_j$. The entries

Algorithm 1 `Reduce`($\mathbf{\Delta}$)

Input: A boundary matrix $\mathbf{\Delta} = [\mathbf{\Delta}_0 \mid \mathbf{\Delta}_1 \mid ... \mid \mathbf{\Delta}_{n-1}] \in \mathbb{Z}_2^{n \times n}$
Output: A reduced matrix $\mathbf{R} \in \mathbb{Z}_2^{n \times n}$

```
R ← Δ
for j ← 0 to n − 1 do
    while exists j_0 < j with low(R_{j_0}) = low(R_j) do
        R_j ← (R_j + R_{j_0}) mod 2
    end while
end for
return R
```

of the boundary matrix are

$$\mathbf{\Delta}_{i,j} = \begin{cases} 1, & \text{if } \sigma_i \text{ is a boundary face of } \sigma_j \\ 0, & \text{otherwise} \end{cases}.$$

Thus, $\mathbf{\Delta}$ encodes the order in which simplices appear in the filtration and the relationship between each simplex and its boundary simplices. We let the first row and column correspond to the empty simplex, $\emptyset$, so that the vertices have boundary equal to $\emptyset$. Thus, vertices are encoded by a column [1,0,...,0], while $\mathbf{\Delta}_0$ is then necessarily a zero column, which could be omitted. The scales, $f(\sigma_i)$, at which each simplex is added to the complex may be regarded as a real-valued vector in $\mathbb{R}^n$ and can be held separately from the combinatorial information encoded in the boundary matrix.

It is shown in [22,23] that calculation of the persistence pairs can be achieved through a straightforward algorithm (Algorithm 1) that brings a boundary matrix into a reduced form. The critical operation needed to transform a filtered simplicial complex $\mathbf{K}$—given by the monotonic filtration function $f : K \to \mathbb{R}$ and encoded in a boundary matrix $\mathbf{\Delta}$—into its PDs is the function

$$low(\mathbf{v}) = \max(\{i \mid (\mathbf{v}[i] = 1\}),$$

which returns the largest index among those coordinates of the binary vector $\mathbf{v}$ that are equal to 1. Progressing from $j = 1$ to n (i.e., in the order of the simplices given by the monotonic function f), each column $\mathbf{\Delta}_j$ is replaced with the mod-2 sum $\mathbf{\Delta}_i + \mathbf{\Delta}_j$, whenever $\text{low}(\mathbf{\Delta}_i) = \text{low}(\mathbf{\Delta}_j)$ and $i < j$, until the lowest 1 in column j is distinct from all lowest 1s in the preceding columns. The lowest 1s in the reduced boundary matrix then specify the indices of the pair of simplices at which each PH class of the corresponding dimension is born and dies. More precisely, let $\mathbf{R} = \texttt{Reduce}(\mathbf{\Delta})$ be the reduction of the boundary matrix $\mathbf{\Delta}$ after applying Algorithm 1. Then $(f(\sigma_i), f(\sigma_j))$ is a (finite persistence) point in the k-dimensional PD $\text{dgm}_k(\mathbf{K})$ if and only if σ_i is a simplex of dimension k and $i = \text{low}(\mathbf{R}_j)$. In other words, a k-dimensional homology class was born with the introduction of the simplex $\sigma_i = \sigma_{\text{low}(\mathbf{R}_j)}$ and died when σ_j was added to the filtration.

In Figure 2 we illustrate the original boundary matrix, its reduced form after applying Algorithm 1, and H_0 and H_1 PDs associated to the given filtration. In the reduced matrix, columns b and c consist of all zeros, since their appearance created homology (H_0) classes[4]. The connected components represented by vertices b and c are then killed by the introduction of ab and ac respectively, since these edges merge the connected component into the component represented by a, which was born earlier. This is encoded in the reduced boundary matrix by the low 1s at indices (b, ab) and (c, ac) respectively. The edge bc likewise gives birth to an H_1 class, that is later killed off by the introduction of the triangle abc. This is why, in $\texttt{Reduce}(\mathbf{\Delta})$, column bc consists of all zeros and the low 1 in abc is in row bc.

The low 1s in the reduced matrix encode the birth-death simplex pairs appearing in the PDs of the filtration. Here we take the scale of each simplex to be the index of the complex in which it first appears so that the low 1 at (b,ab) is sent to the point (1,3) in the H_0 diagram. Similarly, (bd, bcd) maps to (8,9) and (bc, abc) maps to (5,10) in the H_1 PD, $\text{dgm}_1(\mathbf{K})$. If the scales of each simplex were determined instead by some geometric information in the data (e.g., using pairwise distances between points as is the case for the Vietoris-Rips filtration), the positions of the points in the PDs would capture these scales, rather than merely the indices.

2.2 Homomorphic Encryption

Let $\mathcal{M}$ be a message (plaintext) space and $\mathcal{C}$ be a ciphertext space. We assume that $\mathcal{M}$ and $\mathcal{C}$ are commutative rings with their respective identity elements, and addition and multiplication operations, denoted $(\mathcal{M}, 1_{\mathcal{M}}, +, \times)$ and $(\mathcal{C}, 1_{\mathcal{C}}, \oplus, \otimes)$. When the underlying ring is clear from the context, we simply denote the identity element by 1 and so by abuse of notation the scalar space of the ring consists of elements $s = \sum_{i=1}^{s} 1 \in \mathbb{Z}$. For a given parameter set $\texttt{params}$, an HE scheme consists of algorithms as described in the following:

- $\texttt{KeyGen}(\texttt{params})$: Takes $\texttt{params}$ as input, and outputs a public key and secret pair ($\texttt{pk}$, $\texttt{sk}$), and an evaluation key $\texttt{evk}$.
- $\texttt{Enc}_{\texttt{pk}}(\mathbf{m})$: Takes a plaintext message $\mathbf{m} \in \mathcal{M}$ and the public key $\texttt{pk}$ as input, and outputs a ciphertext $c \in \mathcal{C}$.
- $\texttt{Dec}_{\texttt{sk}}(c)$: Takes a ciphertext $c \in \mathcal{C}$ and the public key $\texttt{sk}$ as input, and outputs a plaintext message $\mathbf{m} \in \mathcal{M}$.
- $\texttt{Add}_{\texttt{evk}}(c_1, c_2)$: Takes a pair of ciphertexts (c_1, c_2), $c_i \in \mathcal{C}$ and the evaluation key $\texttt{evk}$ as input, an outputs a ciphertext $c_{\texttt{add}} \in \mathcal{C}$.
- $\texttt{Mult}_{\texttt{evk}}(c_1, c_2)$: Takes a pair of ciphertexts (c_1, c_2), $c_i \in \mathcal{C}$ and the evaluation key $\texttt{evk}$ as input, an outputs a ciphertext $c_{\texttt{mult}} \in \mathcal{C}$.
- $\texttt{Eval}_{\texttt{evk}}(f; c_1, ..., c_k)$: Takes an arithmetic circuit $f : \mathcal{M}^k \to \mathcal{M}$, ciphertexts $c_i \in \mathcal{C}$, and the evaluation key $\texttt{evk}$ as input, and outputs a ciphertext $c_{\texttt{eval}} \in \mathcal{C}$.

[4]The first vertex a is a special case, and technically kills off the (-1)-dimensional (reduced) homology class at index -1.

Here, $\texttt{params}$ generally consists of a security parameter λ and a multiplicative depth parameter L. The security parameter λ says that complexity of the best attack to break the security of the HE scheme is $\mathcal{O}$. The depth parameter L guarantees that the HE scheme can evaluate circuits of maximum multiplicative depth L. We frequently refer to multiplicative depth and computational complexity of circuits in our analysis and they are defined as follows.

Definition 2. *Let f be an arithmetic circuit. Multiplicative depth, or simply depth, of f is the maximum number of sequential multiplications required to compute f. Computational complexity, or simply complexity, of f is the number of multiplication and addition operations required to compute f.*

For example, $f(m_1, m_2, m_3, ..., m_n) = \sum_{i=1}^{n} m_i^{2^i}$ is a depth-n multiplicative circuit, where $m_i^{2^i}$ can be computed after i successive multiplications (squarings). A naive way to compute f would require $n(n+1)/2$ multiplications and $(n-1)$ additions and so we can say that f has computational complexity $\mathcal{O}(n^2)$.

A basic correctness requirement[5] for an HE scheme is that the decryption operation is the inverse of the encryption operation, that is

$$\texttt{Dec}_{\texttt{sk}}(\texttt{Enc}_{\texttt{pk}}(m)) = m$$

for all $m \in \mathcal{M}$. The homomorphic feature[5] of an HE scheme requires

$$\texttt{Dec}_{\texttt{sk}}(\texttt{Eval}_{\texttt{evk}}(f; c_1, ..., c_k)) = f(m_1, ..., m_k)$$

for all $c_i \in \mathcal{C}$ such that $c_i = \texttt{Enc}_{\texttt{pk}}(m_i)$. In other words, HE allows one to evaluate polynomials on encrypted data such that the decryption of the result is exactly the same as the value of that polynomial evaluated on plaintext messages. We should note that we presented here a limited overview of HE schemes so that our paper is self-contained. HE schemes are much more involved (e.g., consisting of other algorithms such as scaling, relinearization, bootstrapping, etc.) and their implementations require a great deal of detail (e.g., encoding and decoding algorithms so that the plaintext messages can be mapped into the message space of the scheme, batching operations, etc.). Moreover, most of these details depend on the choice of the HE scheme. For a survey of HE schemes and existing libraries, we refer the reader to [1, 5].

Some of the challenges of using HE in practice are:

– Increasing the depth of the arithmetic circuit significantly increases the complexity of the circuit's encrypted evaluation. Practical HE schemes can handle arithmetic circuits with relatively low depth. For example, [20] reports and compares some results for homomorphic evaluation of circuits up to depth 30. Bootstrapping is a viable option to reset the level of a ciphertext right before maximum tolerance is reached.

[5]The correctness and homomorphic features of HE may be violated with negligible probability.

– Algorithms in general require evaluation of functions that are not necessarily polynomials and approximation of functions through low-depth circuits is a challenge. Similarly, algorithms involve conditional statements and evaluating these statements while running an algorithm on ciphertext variables requires different ways of handling conditionals. As an example, given $m_1, m_2 \in \mathbb{Z}_{\text{ı}}$ for some prime p, the conditional statement that returns $m_1 + m_2$ if $m_1 = m_2$; and that returns m_1 if $m_1 \neq m_2$ can be implemented over ciphertexts as

$$(\texttt{Eval}_{\texttt{evk}}(f; c_1, c_2) \otimes c_1) \oplus ((1 - \texttt{Eval}_{\texttt{evk}}(f; c_1, c_2)) \otimes (c_1 \oplus c_2)),$$

where $c_i = \texttt{Enc}_{\texttt{pk}}(m_i)$, and $f(m_1, m_2) = (m_1 - m_2)^{p-1}$ can be implemented as an arithmetic circuit of depth $\mathcal{O}(\log_2 p)$ using a square-and-multiply type exponentiation algorithm.

Our objective is to adapt Algorithm 1 so that secure boundary matrix reduction operation can be performed based on encrypted boundary matrices using HE. In the light of our discussion above, there are three main challenges to address:

1. Develop an arithmetic circuit for *encrypted low* computations so that given a pair of ciphertexts c_1 and c_2 (representing the encryption of column vectors $\mathbf{v}_1$ and $\mathbf{v}_2$), $low(\mathbf{v}_1) = low(\mathbf{v}_2)$ can be verified; see line 3 in Algorithm 1.
2. Develop an arithmetic circuit so that the conditional modular addition operation (line 4 in Algorithm 1) can be performed in the ciphertext space.
3. Modify the logical structure of Algorithm 1 so that all of the modular vector additions in lines 2-4 in Algorithm 1 are correctly executed in the ciphertext space, until $low(\mathbf{R}_{j_0}) \neq low(\mathbf{R}_j)$ for all $j_0 < j$, and for all $j = 0, ..., (n-1)$.

3 HE-Compatible Matrix Reduction

3.1 `Low`: HE-compatible computation of *low*

The first obstacle to realizing an HE-compatible `Reduce` algorithm is computing the largest index of any 1 in an n-dimensional binary vector $\mathbf{v} \in \{0,1\}^n$, called $low(\mathbf{v})$ (see Section 2.1). For reasons that will become clear, it will be necessary for us to extend the usual definition of *low*—as defined in Section 2—to the n-dimensional 0-vector; we assign $low(\mathbf{0}) = n - 1$. By construction, a non-zero column in a boundary matrix of a valid filtration can never have a *low* of $n - 1$ before or during reduction by Algorithm 1. [6]

In [16], the authors introduce a method of locating the index of the maximum value of a vector (*maxidx*) of distinct numbers using HE. We adapt this method to obtain an approximation of the *low* value of a binary vector. First, in Lemma 1, we establish the correctness of our reimagining of the exact *low*

[6] If it did, that would imply the simplex that appeared latest is the boundary of a simplex that appeared earlier, which violates the condition that each step in the filtration gives a valid complex.

function obtained by monotonically scaling vector coordinates with respect to their index while ensuring all coordinates remain distinct and guaranteeing the *low* corresponds to the new largest coordinate.

Transformation 1. For $\mathbf{v} \in \mathbb{R}^n$, let $S(\mathbf{v}) := \left[\mathbf{v}[i] + \frac{i}{n}\right]_{i=0}^{n-1}$

Definition 3. *Let $\mathcal{D}^n = \{\mathbf{v} \in \mathbb{R}^n \mid \mathbf{v}[i] \neq \mathbf{v}[j], 0 \leq i \neq j < n\}$ be the collection of n-dimensional vectors with distinct coordinates. For a vector $\mathbf{v} \in \mathcal{D}^n$, define $maxidx : \mathcal{D}^n \to \mathbb{Z}$ by $maxidx(\mathbf{v}) = k$ if $\mathbf{v}[k] > \mathbf{v}[j]$ for all j different from k.*

Lemma 1. *For any binary vector $\mathbf{v} \in \{0,1\}^n$,*

$$low(\mathbf{v}) = maxidx(S(\mathbf{v}))$$

Proof. See Appendix A.

How does our argument about *maxidx* approximating *low* hold in our "approximate arithmetic" setting? The following generalization of Lemma 1 states that as long as our *approximate* binary vector $\mathbf{v}' \in \mathbb{R}^n$ isn't too far from an underlying, true binary vector $\mathbf{v} \in \{0,1\}^n$, then we may continue to extract $low(\mathbf{v})$ using $maxidx(\mathbf{v}')$.

Lemma 2. *Let $\mathbf{v} \in \{0,1\}^n$ and $\mathbf{v}' \in \mathbb{R}^n$ be given such that $|\mathbf{v}' - \mathbf{v}| < \frac{1}{2n}$. Then*

$$low(\mathbf{v}) = maxidx(S(\mathbf{v}')).$$

Proof. See Appendix A.

Remark 1. The proximity between $\mathbf{v}$ and $\mathbf{v}'$ cannot be relaxed for the above choice of Transformation 1, since it is possible to construct vectors, $\mathbf{v}$ and $\mathbf{v}'$ such that $|\mathbf{v} - \mathbf{v}'| = \frac{1}{2n} + c$, with $0 = low(\mathbf{v}) \neq maxidx(S(\mathbf{v}')) = n - 1$ for any $c > 0$.

Using this construction, it is then natural to apply the `MaxIdx` function presented in [16] (Algorithm 2), to develop the `Low` function (Algorithm 3). This `Low` function will estimate *low* with arbitrary accuracy, for real vectors that well-approximate binary vectors.

`MaxIdx` takes a vector $\mathbf{v} \in [1/2, 3/2)^n$ and returns a vector $\mathbf{b}$ with $\mathbf{b}[k] \approx 1$ if $maxidx(\mathbf{v}) = k$ and $\mathbf{b}[j] \approx 0$ for $j \neq k$. The component-wise accuracy in approximating the coordinates of the true maximum value indicator vector ($\mathbf{b}$ with $\mathbf{b}[maxidx(\mathbf{v})] = 1$ and 0 elsewhere) is controlled by a tuple of parameters $\mathcal{P}_{\mathtt{L}} = (d, d', m, t)$. In [16], the authors show that the error in each coordinate is bounded by $2^{-\alpha}$, for $\alpha > 0$ which can be made arbitrarily large with sufficiently large choices of d, d', m, and t.

To attain the actual index containing the maximum value of $\mathbf{v}$, as opposed to the maximum index indicator vector, $\mathbf{b}$, we compute the dot product between $\mathbf{b}$ and $[0, 1, ..., n-1]$. This is the approach we adopt in the `Low` function given in Algorithm 3. Since the `MaxIdx` algorithm requires the input vector to be in the interval $[\frac{1}{2}, \frac{3}{2})^n$, and our inputs $S(\mathbf{v}')$ will be in the interval $[0, 2)^n$, we apply a linear transformation that preserves the *maxidx* of its input.

Algorithm 2 $\texttt{MaxIdx}(\mathbf{v}; d, d', m, t)$ from [16]

Input: A vector $\mathbf{v} \in [\frac{1}{2}, \frac{3}{2})^n \cap \mathcal{D}^n$; $d, d', m, t \in \mathbb{N}$
Output: A vector $\mathbf{b} \in [0,1]^n$ such that $\mathbf{b}[k] \approx 1$, if $maxidx(\mathbf{v}) = k$, otherwise $\mathbf{b}[i] \approx 0$.
Depth: $d' + 1 + t(d + \log m + 2)$
Complexity: $O(n + d' + t(d + n \log m))$

$I \leftarrow \texttt{Inv}(\sum_{j=0}^{n-1} \mathbf{v}[j]/n; d')$
for $j \leftarrow 0$ **to** $n-2$ **do**
    $\mathbf{b}[j] \leftarrow \mathbf{v}[j]/n \cdot I$
end for
$\mathbf{b}[n-1] \leftarrow 1 - \sum_{j=0}^{n-2} \mathbf{b}[j]$
for $i \leftarrow 1$ **to** t **do**
    $I \leftarrow \texttt{Inv}(\sum_{j=0}^{n-1} \mathbf{b}[j]^m; d)$
    for $j \leftarrow 0$ **to** $n-2$ **do**
        $\mathbf{b}[j] \leftarrow \mathbf{b}[j]^m \cdot I$
    end for
    $\mathbf{b}[n-1] \leftarrow 1 - \sum_{j=0}^{n-2} \mathbf{b}[j]$
end for
return b

Transformation 2. $T_{\mathrm{L}}(\mathbf{v}) := \left[\frac{\mathbf{v}[i]+1}{2}\right]_{i=0}^{n-1}$

The error in `MaxIdx` propagates through the `Low` algorithm in the following manner:

Theorem 1. *Let $\alpha > 0$ and fix parameters d, d', m, t for the* `MaxIdx` *algorithm so that*

$$|\texttt{MaxIdx}(\mathbf{x}; d, d', m, t) - \mathbf{e}_{maxidx(\mathbf{x})}| < 2^{-\alpha},$$

for all $\mathbf{x} \in [\frac{1}{2}, \frac{3}{2})^n$. Further assume $\mathbf{v}' \in [0,1]^n$ and $\mathbf{v} \in \{0,1\}^n$ are such that $|\mathbf{v}' - \mathbf{v}| < \frac{1}{2n}$. Then

$$\left|\texttt{Low}(\mathbf{v}'; d, d', m, t) - low(\mathbf{v})\right| < \frac{3}{2}(n)(n-1)2^{-\alpha}.$$

Proof. The result follows from Lemmas 4 and 5 in Appendix A and the triangle inequality.

In the next section we establish choices of parameters ensuring a specified level of accuracy of the approximating `Low` function.

As a final remark, we note that the dependence of `Low`'s error on n^2 is a consequence of extracting the *low* of a vector using a dot product between the vector of indices, $[0, \ldots, n-1]$, and the max-index-indicator vector. This may be unavoidable when using the current implementation of the `MaxIdx` function, although it is conceivable that a fundamentally different approach to computing `Low` may yield a better error growth with the size of the boundary matrix.

Algorithm 3 $\texttt{Low}(\mathbf{v}'; d, d', m, t)$

Input: A vector $\mathbf{v}' \in [0,1]^n$; $d, d', m, t \in \mathbb{N}$ ▷ $\mathbf{v}' \approx \mathbf{v} \in \{0,1\}^n$
Output: A real number r ▷ $r \approx low(\mathbf{v})$
Depth: $d' + 2 + t(d + \log(m) + 2)$
Complexity: $O(n + d' + t(d + n \log m))$

1: $\mathbf{v}' \leftarrow S(\mathbf{v}')$ ▷ Lemma 2
2: $\mathbf{v}' \leftarrow T_{\text{L}}(\mathbf{v}')$ ▷ Maps $[0,2)^n$ to $[1/2, 3/2)^n$
3: $\mathbf{b} \leftarrow \texttt{MaxIdx}(\mathbf{v}'; d, d', m, t)$ ▷ Algorithm 2
4: $r \leftarrow \mathbf{b} \cdot [0, 1, ..., n-1]$ ▷ Extract *low* estimate
5: **return** r

3.2 Parameters for Low

Having established an approximation of the *low* function that is amenable to an HE framework, we next establish the prerequisite results needed to inform the choices of Low's parameters that will guarantee correctness. There are two results we create in order to help ease the proof of the theorem at the end of this section. The first of these is to establish a lower bound on this ratio over for all binary vector inputs to Low, as this value will directly affect the choice of parameters for the MaxIdx and, subsequently, the Low functions.

Let us borrow Theorem 5 from [16], which gives the parameter choices (d, d', m, t) to achieve any desired non-zero error

$$|\texttt{MaxIdx}(\mathbf{v}; d, d', m, t) - \mathbf{e}_{maxidx(\mathbf{v})}| < 2^{-\alpha}.$$

Theorem 2 (Theorem 5 in [16]). *Let $\mathbf{v} \in [\frac{1}{2}, \frac{3}{2})^n$ be a vector with n distinct entries. Define c to be the ratio of the maximum value over the second maximum value such that $c \in (1, 3)$. If*

$$t \geq \frac{1}{\log(m)}[\log(\alpha + \log(n) + 1) - \log\log(c)]$$
$$\min(d, d') \geq \log(\alpha + t + 2) + (m-1)\log(n) - 1$$

then the error (component-wise) of the $\texttt{MaxIdx}(\mathbf{v}; d, d', m, t)$ algorithm compared to $\mathbf{e}_{maxidx(\mathbf{v})}$ is bounded by $2^{-\alpha}$.

Of great importance to us is a lower bound on c, the ratio of the largest to the second largest coordinate values in the input to MaxIdx's parameters. As c approaches 1, MaxIdx and Low's parameters d, d', and t grow without limit. For this reason, we aim to obtain a larger lower bound on c across all possible (approximate binary) input vectors. We re-write the bound $|\mathbf{v} - \mathbf{v}'| < \frac{1}{2n}$ as $|\mathbf{v} - \mathbf{v}'| \leq \frac{\varepsilon}{2n}$ where $\varepsilon \in [0, 1)$ to fine-tune parameter c.

We compute that a lower bound on c is given by $c \geq 1 + \frac{2-2\varepsilon}{6n-4+\varepsilon}$ in Lemma 8 in Appendix A. Importantly, if $\varepsilon = 1$ (and so assume $\mathbf{v}'$ is approximately binary only within the bound $1/2n$ needed for Lemma 2 to compute *low* via *maxidx*)

then the ratio of the first to the second largest coordinates of the transformed $\mathbf{v}'$ can be arbitrarily close to 1. As a consequence, there will no longer exist a choice of finite parameters in the `Low` algorithm that guarantees correctness over all possible approximately-binary vectors $\mathbf{v}'$. On the other hand, as ε gets closer to 0, the lower bound on c increases away from 1, which will allow `Low` to be computed more efficiently. Thus there will be a trade-off between the computational cost of maintaining $\mathbf{v}'$ sufficiently close to binary throughout the boundary matrix reduction, and estimating *low* efficiently.

The variable α specifies the desired level of accuracy of `MaxIdx` (to $2^{-\alpha}$), and informs the minimum parameters needed to attain said accuracy. Lemma 6 recasts the accuracy parameter of `Low` to an arbitrary $\delta > 0$. With this, we can specify the choice of parameters needed to approximate $low(\mathbf{v})$ using $\texttt{Low}(\mathbf{v}'; d, d', m, t)$ to arbitrary accuracy.

Theorem 3. *Assume* $\mathbf{v} \in \{0,1\}^n$ *and* $\mathbf{v}' \in [0,1]^n$ *are such that* $|\mathbf{v} - \mathbf{v}'| \leq \frac{\varepsilon}{2n}$, *for some* $0 \leq \varepsilon < 1$. *Choose the parameters* $d, d', m,$ *and* t *for the* `MaxIdx` *function, along with a pre-determined* $\delta > 0$, *such that*

$$\begin{aligned} \alpha &> \log(3) + 2\log(n) - \log(\delta) - 1 \\ t &\geq \frac{\log\Big(\alpha + 1 + \log(n)\Big) - \log\log\Big(1 + \frac{2-2\varepsilon}{6n-4+\varepsilon}\Big)}{\log m} \\ \min(d, d') &\geq \log(\alpha + t + 2) + (m-1)\log(n) - 1 \end{aligned}$$

Then $\texttt{Low}(\mathbf{v}'; d, d', m, t)$ *has* δ*-error. That is,*

$$|\texttt{Low}(\mathbf{v}'; d, d', m, t) - low(\mathbf{v})| < \delta.$$

Proof. See Appendix B.

As these parameters are now well-established for the `Low` function, we now refer to this tuple of parameters $(d_{\texttt{L}}, d'_{\texttt{L}}, m_{\texttt{L}}, t_{\texttt{L}})$ as $\mathcal{P}_{\texttt{L}}$ to avoid confusion with the upcoming `Comp` function which will have a similar parameter naming convention. Furthermore, when $\mathcal{P}_{\texttt{L}}$ is clear from context, define

$$\texttt{L}_{\mathbf{v}} \coloneqq \texttt{Low}(\mathbf{v}; \mathcal{P}_{\texttt{L}})$$

for ease of notation in the upcoming sections.

3.3 `LowComp`: HE-compatible Equality Check

Theorem 3 approximates $low(\mathbf{x})$ and $low(\mathbf{y})$ via $\texttt{Low}(\mathbf{x}'; \mathcal{P}_{\texttt{L}})$ and $\texttt{Low}(\mathbf{y}'; \mathcal{P}_{\texttt{L}})$. One of the remaining challenges is to characterize the equality check $low(\mathbf{x}) = low(\mathbf{y})$ using $\texttt{Low}(\mathbf{x}'; \mathcal{P}_{\texttt{L}})$ and $\texttt{Low}(\mathbf{y}'; \mathcal{P}_{\texttt{L}})$. The second challenge is to rewrite (1) for $\mathbf{z}'$ so that it can be computed by avoiding the if statement and the mod 2 addition.

Suppose that $\mathbf{x}'$ and $\mathbf{y}'$ are two real valued vectors that are approximations of the binary vectors $\mathbf{x}$ and $\mathbf{y}$, respectively. We must now determine a method

that takes $\mathbf{x}'$ and $\mathbf{y}'$ as input, and outputs $\mathbf{z}'$ such that $\mathbf{z}'$ approximates the binary vector

$$\mathbf{z} = \begin{cases} \mathbf{x} + \mathbf{y} \mod 2 & \text{if } low(\mathbf{x}) = low(\mathbf{y}) \\ \mathbf{x} & \text{if } low(\mathbf{x}) \neq low(\mathbf{y}) \end{cases} \tag{1}$$

In Section 3.5, we show that $\mathbf{z}$ in (1) can be approximated by

$$\mathbf{z}' = \Omega(\mathbf{x}' - \mathbf{y}')^2 + (1 - \Omega)\mathbf{x}', \tag{2}$$

where the predicate Ω takes $\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L})$ and $\texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})$ as input, and approximates the boolean value $low(\mathbf{x}) == low(\mathbf{y})$. We establish the theory to calculate Ω in this section.

Lemma 3. *Let $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ and $\mathbf{x}', \mathbf{y}' \in [0,1]^n$ and assume that $\mathcal{P}_L$ is chosen such that $|\texttt{Low}(\mathbf{x}'; \mathcal{P}_L) - low(\mathbf{x})| < \delta$ and $|\texttt{Low}(\mathbf{y}'; \mathcal{P}_L) - low(\mathbf{y})| < \delta$ for some $0 < \delta < \frac{1}{4}$. Let ϕ be any value in the interval $(2\delta, 1 - 2\delta)$. Then*

$$|\texttt{Low}(\mathbf{x}'; \mathcal{P}_L) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_L)| \leq \phi \text{ iff } low(\mathbf{x}) = low(\mathbf{y})$$

Proof. Suppose that $|\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})| > \phi$. Then

$$\begin{aligned} \phi &< |\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})| \\ &\leq |\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - low(\mathbf{x})| + |\texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L}) - low(\mathbf{y})| + |low(\mathbf{x}) - low(\mathbf{y})| \\ &< 2\delta + |low(\mathbf{x}) - low(\mathbf{y})|. \end{aligned}$$

This implies that $|low(\mathbf{x}) - low(\mathbf{y})| > \phi - 2\delta > 0$ as $\phi > 2\delta$ by assumption. Both $low(\mathbf{x})$ and $low(\mathbf{y})$ are integer-valued functions, so it must be the case that $low(\mathbf{x}) \neq low(\mathbf{y})$.

Conversely, suppose that

$$|\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})| \leq \phi.$$

Then

$$\begin{aligned} |low(\mathbf{x}) - low(\mathbf{y})| \leq\ & |\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - low(\mathbf{x})| \\ & + |\texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L}) - low(\mathbf{y})| \\ & + |\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})| \\ & < \delta + \delta + \phi \end{aligned}$$

And so we have that $|low(\mathbf{x}) - low(\mathbf{y})| < 2\delta + \phi < 1$ as $\phi < 1 - 2\delta$. Again, as low is an integer-valued function, it must be the case that $low(\mathbf{x}) = low(\mathbf{y})$.

Remark 2. Tracing the proof of Lemma 3 also reveals that the intervals on which $|\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})|$ and ϕ live are disjoint, and so it will never be the case that

$$|\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})| = \phi,$$

despite the statement of the lemma.

The implications of LEMMA 3 is that one does not need to be very accurate in the calculation of $\texttt{Low}(\mathbf{x}';\mathcal{P}_{\texttt{L}})$, and in fact only needs to approximate $low(\mathbf{x})$ (using $\texttt{Low}(\mathbf{x}';\mathcal{P}_{\texttt{L}})$) to an accuracy of $\frac{1}{4}$. If that condition is guaranteed, then one may compare the value $|\texttt{Low}(\mathbf{x}';\mathcal{P}_{\texttt{L}}) - \texttt{Low}(\mathbf{y}';\mathcal{P}_{\texttt{L}})|$ to any $2\delta < \phi < 1 - 2\delta$ to check whether the underlying low values are equal or not.

With this lemma, our strategy to compare *low* values of two approximately binary vectors will be to exploit an approximation of the function that compares the relative size of its two inputs. First, we introduce the following function:

Definition 4. *For $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$, let $l_{\mathbf{x}} = low(\mathbf{x})$ and $l_{\mathbf{y}} = low(\mathbf{y})$. Define*

$$lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}}) = \begin{cases} 0, & \text{if } l_{\mathbf{x}} \neq l_{\mathbf{y}} \\ 1, & \text{if } l_{\mathbf{x}} = l_{\mathbf{y}} \end{cases}.$$

The function *lowcomp* will be used to gate the mod 2 addition of two columns in place of the conditional equality check in Algorithm 1. In particular, for a given $\mathbf{x}$ and $\mathbf{y} \in [0,1]^n$, the statement "update $\mathbf{x}$ to $\mathbf{x} + \mathbf{y} \mod 2$, if their lows are equal" may be reinterpreted as

$$\mathbf{x} = \mathbf{x} + lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}})\mathbf{y} \mod 2.$$

We now establish a `LowComp` algorithm to estimate the *lowcomp* function for approximately binary vectors. Our formulation is based on the `Comp` algorithm, which estimates the *comp* function given in Definition 5 (both introduced in [16]) that compares the relative size of its inputs.

Definition 5 ([16]). *For any non-zero real numbers a, b, define*

$$comp(a,b) = \lim_{k\to\infty} \frac{a^k}{a^k + b^k} = \begin{cases} 1, & \text{if } a > b \\ \frac{1}{2}, & \text{if } a = b \\ 0, & \text{if } a < b \end{cases}$$

The `Comp` algorithm (Algorithm 4), approximates the *comp* function by evaluating the expression $\dfrac{a^{m^t}}{a^{m^t} + b^{m^t}}$, for t a positive integer, and m often chosen to be a power to 2.

`Comp`, along with LEMMA 3, are the building blocks we need to build `LowComp`. Using LEMMA 3, we make the observation that

$$\begin{aligned} lowcomp(\mathbf{x},\mathbf{y}) = 1 &\Leftrightarrow low(\mathbf{x}) = low(\mathbf{y}) \\ &\Leftrightarrow \phi \geq |\texttt{Low}(\mathbf{x}';\mathcal{P}_{\texttt{L}}) - \texttt{Low}(\mathbf{y}';\mathcal{P}_{\texttt{L}})| \\ &\Leftrightarrow \phi^2 \geq (\texttt{Low}(\mathbf{x}') - \texttt{Low}(\mathbf{y}'))^2 \end{aligned}$$

and so we compare $(\texttt{Low}(\mathbf{x}';\mathcal{P}_{\texttt{L}}) - \texttt{Low}(\mathbf{y}';\mathcal{P}_{\texttt{L}}))^2$ to ϕ^2 to determine if the underlying *low* values are equal or not. This construction removes the need to implement an HE circuit to compute absolute value at the cost of two squarings.

Algorithm 4 $\texttt{Comp}(a, b; d, d', m, t)$ from [16]

Input: distinct real numbers $a, b \in [1/2, 3/2)$; $d, d', m, t \in \mathbb{N}$
Output: a real number $r \in (0, 1)$ ▷ $r \approx 1$ if $a > b$; $r \approx 0$ if $a < b$
Depth: $d' + 1 + t(d + \log(m) + 2)$
Complexity: $O(d' + t(d + \log(m)))$

$I \leftarrow \texttt{Inv}(\frac{a+b}{2}; d')$
$a_0 \leftarrow \frac{a}{2} \cdot I$
$b_0 \leftarrow 1 - a_0$
for $n \leftarrow 0$ **to** $t - 1$ **do**
    $I \leftarrow \texttt{Inv}(a_n^m + b_n^m; d)$
    $a_{n+1} \leftarrow a_n^m \cdot I$
    $b_{n+1} \leftarrow 1 - a_{n+1}$
end for
return a_t

We make two important notes before we explicitly define LowComp. The first is that, by construction, $|\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})|$ and ϕ exist in disjoint intervals (refer to Lemma 3's remark), and so ϕ and $|\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})|$ will never be equal. Thus LowComp may be treated as an approximate binary indicator function for our application. The second is that the input $(\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L}) - \texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L}))^2$ is in the interval $[0, (n-1)^2]$. As the Comp function requires its inputs to be in the interval $[\frac{1}{2}, \frac{3}{2})$, we apply a linear transformation to bring values in the correct interval.

Transformation 3. $T_\texttt{C}(x) := \frac{1}{2} + \frac{x}{n^2}$

Since $T_\texttt{C}$ is a monotonic function, the relative order of the inputs are preserved. We now explicitly define LowComp by performing Comp on $T_\texttt{C}(\phi^2)$ and $T_\texttt{C}((\texttt{L}_{\mathbf{x}'} - \texttt{L}_{\mathbf{y}'})^2)$ as described in Algorithm 5.

LowComp inherits from Comp that its outputs live in (0,1) and that it can approximate *lowcomp* arbitrarily well given appropriately chosen parameters. We formalize this in the following theorem.

Theorem 4. *Let* $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ *and* $\mathbf{x}', \mathbf{y}' \in [0, 1]^n$ *and assume that* $\mathcal{P}_L$ *is chosen such that* $|\texttt{Low}(\mathbf{x}'; \mathcal{P}_L) - low(\mathbf{x})| < \delta$ *and* $|\texttt{Low}(\mathbf{y}'; \mathcal{P}_L) - low(\mathbf{y})| < \delta$ *for some* $0 < \delta < \frac{1}{4}$. *Let* ϕ *be any value in the interval* $(2\delta, 1 - 2\delta)$. *Define* LowComp *as in Algorithm 5.*

If the parameters in the Comp *function are chosen such that*

$$|\texttt{Comp}(a, b; d, d', m, t) - comp(a, b)| < \eta,$$

then we also have

$$|\texttt{LowComp}(\texttt{L}_{\mathbf{x}'}, \texttt{L}_{\mathbf{y}'}, \phi; d, d', m, t) - lowcomp(l_\mathbf{x}, l_\mathbf{y})| < \eta.$$

Proof. See Appendix C.

Algorithm 5 $\texttt{LowComp}(\mathtt{L}_{\mathbf{x}'}, \mathtt{L}_{\mathbf{y}'}, \phi; d, d', m, t)$

Input: $\mathtt{L}_{\mathbf{x}'}, \mathtt{L}_{\mathbf{y}'} \in [0, n-1], \phi \in (2\delta, 1-2\delta)$; $d, d', m, t \in \mathbb{N}$
Output: A real number $r \in (0,1)$ ▷ $r \approx lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}})$
Depth: $d' + 2 + t(d + \log(m) + 2)$
Complexity: $O(d' + t(d + \log m))$

1: $\mathtt{L}_d \leftarrow \mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'}$
2: $r \leftarrow \texttt{Comp}(T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}(\mathtt{L}_d^2); d, d', m, t)$ ▷ Algorithm 4
3: **return** r

3.4 Parameters for LowComp

We shall proceed with the analysis of LowComp's parameters in a similar fashion to Low's parameters in Section 3.2. Theorem 4 from [16] gives lower bounds for the parameters d, d', m, and t to achieve $2^{-\alpha}$ error in the Comp function.

Theorem 5 (Theorem 4 in [16]). *Let $x, y \in [1/2, 3/2)$ satisfy*

$$c \leq max(x,y)/min(x,y)$$

for a fixed $c \in (1,3)$. If

$$\begin{aligned} t &\geq \frac{1}{\log(m)}[\log(\alpha+1) - \log\log(c)] \\ d &\geq \log(\alpha + t + 2) + m - 2 \\ d' &\geq \log(\alpha+2) - 1 \end{aligned}$$

then $|\texttt{Comp}(x, y; d, d', m, t) - comp(x,y)| < 2^{-\alpha}$.

The role of c in Comp is similar to MaxIdx in the Section 3.2: the closer $c = \max(a,b)/\min(a,b)$ is to 1, the larger the value for all subsequent choice of parameters, thus increasing the "effort" needed for the Comp function to distinguish which of the two inputs is larger. For this reason, it is our goal to bound $c = \max(a,b)/\min(a,b)$ as far from 1 as possible.

Once a ϕ is fixed, the only guarantee is that $T_{\mathtt{C}}((\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{x}'})^2)$ is either strictly greater or less than said $T_{\mathtt{C}}(\phi^2)$ (see Lemma 3's remark). Since we are only concerned with whether $low(\mathbf{x})$ and $low(\mathbf{y})$ are equal or not, the ratio c may be reinterpreted as

$$c = \frac{\max\left\{T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}((\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{x}'})^2)\right\}}{\min\left\{T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}((\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{x}'})^2)\right\}} > \begin{cases} \dfrac{T_{\mathtt{C}}(\phi^2)}{T_{\mathtt{C}}((\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{x}'})^2)}, & \text{if } low(\mathbf{x}) = low(\mathbf{y}) \\ \dfrac{T_{\mathtt{C}}((\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{x}'})^2)}{T_{\mathtt{C}}(\phi^2)}, & \text{if } low(\mathbf{x}) \neq low(\mathbf{y}) \end{cases}$$

It follows that

$$c > \min\left\{\frac{T_{\mathtt{C}}(\phi^2)}{T_{\mathtt{C}}((2\delta)^2)}, \frac{T_{\mathtt{C}}((1-2\delta)^2)}{T_{\mathtt{C}}(\phi^2)}\right\} > 1$$

where the minimum changes depending on which case we are in. Thus, once a $\delta \in (0, 1/4)$ is chosen, this expression is variable with respect to the value of ϕ and thus $T_{\mathtt{C}}(\phi^2)$. The optimal choice of ϕ will ensure the minimum of these two ratios are as far away from 1 as possible. So, we aim to optimize the right side of this expression with respect to ϕ: that is, to determine what value of $T_{\mathtt{C}}(\phi^2)$ solves

$$\max\left(\min\left\{\frac{T_{\mathtt{C}}(\phi^2)}{T_{\mathtt{C}}((2\delta)^2)}, \frac{T_{\mathtt{C}}((1-2\delta)^2)}{T_{\mathtt{C}}(\phi^2)}\right\}\right), \tag{3}$$

where the max is taken over $T_{\mathtt{C}}(\phi^2)$ in the interval $\big(T_{\mathtt{C}}((2\delta)^2), T_{\mathtt{C}}((1-2\delta)^2)\big)$.

This solution to Eq. (3) comes from a general fact about positive real numbers, which we prove in PROPOSITION 9, and which establishes the following corollary:

Corollary 1. *The value of $T_{\mathcal{C}}(\phi^2)$ which solves Eq. (3) is*

$$T_{\mathcal{C}}(\phi^2) = \sqrt{\left(\frac{1}{2} + (\frac{2\delta}{n})^2\right)\left(\frac{1}{2} + (\frac{1-2\delta}{n})^2\right)}.$$

Thus, $c > \sqrt{\frac{n^2+2(1-2\delta)^2}{n^2+2(2\delta)^2}}$.

Proof. See Appendix D.

Having determined the bottleneck value c, we explicitly construct a choice of parameters for `LowComp` to achieve any desired level of accuracy (which has been re-contextualized from the $2^{-\alpha}$ error in `Comp` to an arbitrary η error in `LowComp`, see LEMMA 7).

Theorem 6. *Let $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ and $\mathbf{x}', \mathbf{y}' \in [0,1]^n$ and assume that $\mathcal{P}_L$ is chosen such that $|\mathtt{Low}(\mathbf{x}'; \mathcal{P}_L) - low(\mathbf{x})| < \delta$ and $|\mathtt{Low}(\mathbf{y}'; \mathcal{P}_L) - low(\mathbf{y})| < \delta$ for some $0 < \delta < \frac{1}{4}$. Define `LowComp` as in Algorithm 5, where we explicitly pick*

$$\phi = n\sqrt{\sqrt{\left(\frac{1}{2} + (\frac{2\delta}{n})^2\right)\left(\frac{1}{2} + (\frac{1-2\delta}{n})^2\right)} - \frac{1}{2}}.$$

If the parameters in the `Comp` function are chosen such that

$$\begin{aligned}
&\alpha > -\log(\eta)\\
&t \geq \frac{1}{\log(m)}\left[\log(\alpha+2) - \log\log\left(\sqrt{\frac{n^2+2(1-2\delta)^2}{n^2+2(2\delta)^2}}\right)\right]\\
&d \geq \log(\alpha+t+2) + m - 2\\
&d' \geq \log(\alpha+2) - 1
\end{aligned}$$

then `LowComp` has η-error. That is,

$$|\mathtt{LowComp}(\mathtt{L}_{\mathbf{x}'}, \mathtt{L}_{\mathbf{y}'}, \phi; d, d', m, t) - lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}})| < \eta$$

Proof. See Appendix D.

Remark 3. `LowComp` can now be thought of as a function of only two inputs ($\mathtt{L}_{\mathbf{x}'}$ and $\mathtt{L}_{\mathbf{y}'}$), as we will always choose this optimal value of ϕ. The theorem also implies a trade-off between δ and η. Indeed, estimating *low* using `Low` to a high degree requires less "effort" for `Comp` to distinguish the (in)equality of two `Low` values. Similarly, less accurate *low* estimates will require `Comp` to do more of the heavy-lifting. This intuition is confirmed by the dependence on δ of the lower bound on c. As δ approaches 0, the bound on c increases further away from 1, causing our choice of parameters for `LowComp` to get smaller. On the flip side, as δ approaches its upper limit of $1/4$, then c may get arbitrarily close to 1, causing `LowComp`'s parameters to get arbitrarily large. We refer to these parameters as $\mathcal{P}_{\mathtt{C}} = (d_{\mathtt{C}}, d'_{\mathtt{C}}, m_{\mathtt{C}}, t_{\mathtt{C}})$.

3.5 Conditional modular addition of vectors

For a given $\mathbf{x}$ and $\mathbf{y} \in [0,1]^n$, the statement "update $\mathbf{x}$ to $\mathbf{x}+\mathbf{y} \mod 2$, if their *low* values are equal" from Equation 1 may be reinterpreted as

$$\mathbf{x} = \mathbf{x} + lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}})\mathbf{y} \mod 2. \tag{4}$$

Furthermore, addition modulo 2 can be recast as a polynomial operation using the observation that for any two $a, b \in \{0,1\}$, the operation $(a-b)^2 = a+b \mod 2$. Thus, we may rewrite (1) as

$$\mathbf{x} = lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}})(\mathbf{x}-\mathbf{y})^2 + (1 - lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}}))\mathbf{x},$$

taking all operations component-wise, to remove mod 2 addition.

We may then approximate this operation using `Low` to esimate *low* and `LowComp` to estimate *lowcomp*. That is, the operation we will be performing on approximate binary vectors is

$$\mathbf{x}' = \mathtt{LowComp}(\mathtt{L}_{\mathbf{x}'}, \mathtt{L}_{\mathbf{y}'})(\mathbf{x}'-\mathbf{y}')^2 + (1 - \mathtt{LowComp}(\mathtt{L}_{\mathbf{x}'}, \mathtt{L}_{\mathbf{y}'}))\mathbf{x}', \tag{5}$$

as alluded to in Eq. (2) in Section 3.3.

3.6 Modifying the Logical Structure of `Reduce`

The main operation in `Reduce` (Algorithm 1) is gated by a conditional **while** loop. As mentioned before, conditional statements cannot be explicitly implemented (or traced) over ciphertexts. Therefore, we need to rewrite lines 2-6 in Algorithm 1 so that they are HE-compatible. This is done by replacing the **while** loop with a double nested **for** loop, each of which run through all preceding column indices. Assume $low(\boldsymbol{\Delta}_i) \neq low(\boldsymbol{\Delta}_k)$ for all $0 \leq i \neq k < j$, as is the case when the `Reduce` algorithm, applied to a boundary matrix $\boldsymbol{\Delta}$, first encounters column j. If we loop through the preceding j columns once, comparing each $\boldsymbol{\Delta}_k, k = 0 \ldots, j-1$ to $\boldsymbol{\Delta}_j$, either $low(\boldsymbol{\Delta}_k) = low(\boldsymbol{\Delta}_j)$ for some $k < j$ or not. In

the latter case, we know $\mathbf{\Delta}_j$ is already in reduced form and will not change—no matter how many times one loops again through the preceding $j-1$ columns—since the (binary) addition only happens when the low's of two columns match. On the other hand, if some $low(\mathbf{\Delta}_k) = low(\mathbf{\Delta}_j)$ for some $k < j$, $\mathbf{\Delta}_j \leftarrow \mathbf{\Delta}_j + \mathbf{\Delta}_k \mod 2$ will change $\mathbf{\Delta}_j$, and in particular, this addition necessarily causes the $low(\mathbf{\Delta}_j)$ to decrease. Thus, after such an update this new $\mathbf{\Delta}_j$ will never again be equal to $\mathbf{\Delta}_k$. In other words each column vector will only update column j at most once. Without any assumptions about the order in which preceding columns update $\mathbf{\Delta}_j$, we simply loop over the preceding columns enough times to guarantee every vector which should have updated column j has done so. This requires exactly j loops over all preceding columns since each preceding column can only update $\mathbf{\Delta}_j$ at most once. For the base case, note that column $j = 0$ is trivially in reduced form and $\mathbf{\Delta}_1$ will certainly be in reduced form after a single comparison with $\mathbf{\Delta}_0$. This aligns with the worst case complexity for the original `Reduce` algorithm: $O(j^2)$ for column j, $O(n^3)$ overall [22]. In Section 3.1, we modified the existing `MaxIdx` from [16] to attain the `Low` algorithm to estimate low. We have already discussed how to check the equality of two low values using `Low` and `LowComp` in Section 3.3. Finally, the mod 2 addition over rational numbers was constructed in Section 3.5. With all of this combined, we may now rewrite the main block of the `Reduce` algorithm as written in lines 6-9 of Algorithm 6 in a way which makes it compatible with each approximate algorithm on approximate vectors framework.

3.7 An HE-Compatible `Reduce`

As the challenges as listed in Section 2.2 have now been addressed in Sections 3.1-3.6, we now present Algorithm 6, which is an HE-compatible version of `Reduce` and which can take an encrypted boundary matrix as input and reduce it using HE-operations in the ciphertext space. We note that the moment we do the very first column addition, vectors have moved from $\{0,1\}^n$ to $[0,1]^n$, requiring the need for all algorithms to be compatible with approximate binary vectors. For this reason, we must have a guarantee of correctness, which is a function of the controllable errors in our approximation variables: $\mathbf{v}'$ (Theorem 1), `Low` (Lemma 3) , and `LowComp` (Section 3.2).

As long as $|\mathbf{v}' - \mathbf{v}| < 1/2n$, we know that $\texttt{Low}(\mathbf{v}'; \mathcal{P}_\texttt{L})$ will approximate $low(\mathbf{v})$ as accurately as wanted. And as long as $\texttt{Low}(\mathbf{v}'; \mathcal{P}_\texttt{L})$ estimates $low(\mathbf{v})$ to within $1/4$, `LowComp` is able to distinguish between $low(\mathbf{x})$ and $low(\mathbf{y})$ using $\texttt{Low}(\mathbf{x}'; \mathcal{P}_\texttt{L})$ and $\texttt{Low}(\mathbf{y}'; \mathcal{P}_\texttt{L})$. `LowComp` directly defines an approximately binary indicator

$$\Omega \leftarrow \texttt{LowComp}(\texttt{L}_{j_0}, \texttt{L}_j; \mathcal{P}_\texttt{C})$$

which will be used to perform the "mod 2" addition, which will naturally have accumulating non-zero errors (determined by η). The finiteness of the algorithm guarantees the existence of an η such that the accumulation of errors never exceeds the maximum threshold of $1/2n$. In a strict sense, `HE-Reduce` only fails to produce the correct reduced boundary matrix if the maximum error in some

Algorithm 6 `HE-Reduce`$(\boldsymbol{\Delta}; \mathcal{P}_{\mathtt{L}}, \mathcal{P}_{\mathtt{C}})$

Input: A boundary matrix $\boldsymbol{\Delta} = [\, \boldsymbol{\Delta}_0 \mid \boldsymbol{\Delta}_1 \mid ... \mid \boldsymbol{\Delta}_{n-1} \,] \in \mathbb{Z}_2^{n \times n}$
Output: A matrix $\mathbf{R}' \in [0,1]^{n \times n}$ ▷ Approximates Algorithm 1

1: $\mathtt{L}_0 \leftarrow \mathtt{Low}(\boldsymbol{\Delta}_0; \mathcal{P}_{\mathtt{L}})$ ▷ Algorithm 3
2: $\boldsymbol{\Delta}'_0 \leftarrow \boldsymbol{\Delta}_0$
3: **for** $j \leftarrow 1$ **to** $n-1$ **do**
4: $\quad \mathtt{L}_j \leftarrow \mathtt{Low}(\boldsymbol{\Delta}_j; \mathcal{P}_{\mathtt{L}})$ ▷ Algorithm 3
5: $\quad \boldsymbol{\Delta}'_j \leftarrow \boldsymbol{\Delta}_j$
6: $\quad$ **for** $k \leftarrow 0$ **to** $j-1$ **do** ▷ Section 3.6
7: $\quad\quad$ **for** $j_0 \leftarrow 0$ **to** $j-1$ **do**
8: $\quad\quad\quad \Omega \leftarrow \mathtt{LowComp}(\mathtt{L}_{j_0}, \mathtt{L}_j; \mathcal{P}_{\mathtt{C}})$ ▷ Algorithm 5
9: $\quad\quad\quad \boldsymbol{\Delta}'_j \leftarrow \Omega(\boldsymbol{\Delta}'_j - \boldsymbol{\Delta}'_{j_0})^2 + (1-\Omega)\boldsymbol{\Delta}'_j$ ▷ Equation 5
10: $\quad\quad\quad \mathtt{L}_j \leftarrow \mathtt{Low}(\boldsymbol{\Delta}'_j; \mathcal{P}_{\mathtt{L}})$ ▷ Algorithm 3
11: $\quad\quad$ **end for**
12: $\quad$ **end for**
13: **end for**
14: **return** $\mathbf{R}' = [\, \boldsymbol{\Delta}'_0 \mid \boldsymbol{\Delta}'_1 \mid ... \mid \boldsymbol{\Delta}'_{n-1} \,]$

component is $1/2$ or larger. If $|\mathtt{Reduce}(\boldsymbol{\Delta}) - \mathbf{R}'| < 1/2$, then $\mathtt{Round}(\mathbf{R}') = \mathtt{Reduce}(\boldsymbol{\Delta})$, where `Round` casts entries to the nearest integer. This condition is guaranteed by the stricter requirement that errors are within $1/2n$.

4 Complexity and Implementation Analysis

As in all HE-compatible functions, there is particular interest in `HE-Reduce`'s complexity and depth to understand the noise growth that a ciphertext will accumulate as it passes through the algorithm. We will prove a more general statement that establishes the depth of our algorithm on an $n \times n$ boundary matrix. We note that while we establish the textbook version of the `Reduce` algorithm as `HE-Reduce`, an immediate improvement to the algorithm to make it even more HE-compatible is easily seen. We implement this verison, `HE-Reduce-Optimized`, and analyze its depth and complexity.

4.1 Analysis of `HE-Reduce-Optimized`

In our implementation, we use Algorithm 7, which is a slightly modified version of Algorithm 6. Here, the `Low` computation in line 10 in Algorithm 6 is now pushed out of the **for** loop (see line 14 in Algorithm 7) and the repetitive update operations

$$\boldsymbol{\Delta}'_j \leftarrow \Omega(\boldsymbol{\Delta}'_j - \boldsymbol{\Delta}'_{j_0})^2 + (1-\Omega)\boldsymbol{\Delta}'_j,\ j_0 = 0, ..., j-1$$

in line 9 in Algorithm 6 are now replaced by a single cumulative update operation (line 13 in Algorithm 7), which can be explicitly rewritten as

$$\boldsymbol{\Delta}'_j \leftarrow \sum_{j_0=0}^{j-1} \Omega_{j_0,j}((\boldsymbol{\Delta}'_j - \boldsymbol{\Delta}'_{j_0})^2) + (1 - \sum_{j_0=0}^{j-1} \Omega_{j_0,j})\boldsymbol{\Delta}'_j, \tag{6}$$

Algorithm 7 `HE-Reduce-Optimized`$(\mathbf{\Delta}; \mathcal{P}_{\mathtt{L}}, \mathcal{P}_{\mathtt{C}})$

Input: A boundary matrix $\mathbf{\Delta} = [\, \mathbf{\Delta}_0 \mid \mathbf{\Delta}_1 \mid ... \mid \mathbf{\Delta}_{n-1} \,] \in \mathbb{Z}_2^{n \times n}$
Output: A matrix $\mathbf{R}' \in [0,1]^{n \times n}$ ▷ Approximates Algorithm 1

$\mathtt{L}_0 \leftarrow \mathtt{Low}(\mathbf{\Delta}_0; \mathcal{P}_{\mathtt{L}})$ ▷ Algorithm 3
$\mathbf{\Delta}'_0 \leftarrow \mathbf{\Delta}_0$
for $j \leftarrow 1$ **to** $n-1$ **do**
  $\mathtt{L}_j \leftarrow \mathtt{Low}(\mathbf{\Delta}_j; \mathcal{P}_{\mathtt{L}})$ ▷ Algorithm 3
  $\mathbf{\Delta}'_j \leftarrow \mathbf{\Delta}_j$
  for $k \leftarrow 0$ **to** $j-1$ **do** ▷ Section 3.6
    $\mathtt{cumLeft} = 0,\ \mathtt{cumOmega} = 0$
    for $j_0 \leftarrow 0$ **to** $j-1$ **do**
      $\Omega \leftarrow \mathtt{LowComp}(\mathtt{L}_{j_0}, \mathtt{L}_j; \mathcal{P}_{\mathtt{C}})$ ▷ Algorithm 5
      $\mathtt{cumOmega} \leftarrow \mathtt{cumOmega} + \Omega$
      $\mathtt{cumLeft} \leftarrow \mathtt{cumLeft} + \Omega(\mathbf{\Delta}'_j - \mathbf{\Delta}'_{j_0})^2$
    end for
    $\mathbf{\Delta}'_j \leftarrow \mathtt{cumLeft} + (1 - \mathtt{cumOmega})\mathbf{\Delta}'_j$ ▷ Equation 6
    $\mathtt{L}_j \leftarrow \mathtt{Low}(\mathbf{\Delta}'_j; \mathcal{P}_{\mathtt{L}})$
  end for
end for
return $\mathbf{R}' = [\, \mathbf{\Delta}'_0 \mid \mathbf{\Delta}'_1 \mid ... \mid \mathbf{\Delta}'_{n-1} \,]$

where $\Omega_{j_0,j} = \mathtt{LowComp}(\mathtt{L}_{j_0}, \mathtt{L}_j; \mathcal{P}_{\mathtt{C}})$. The correctness follows from the fact that $\Omega_{j_0,j}$ is either approximately zero for all $j_0 = 0, ..., j-1$ except for at most one value of $j_0 = k$ (where it is approximately one), whence we have either $\mathbf{\Delta}'_j$ stays approximately the same or is updated to $\mathbf{\Delta}'_j \approx (\mathbf{\Delta}'_j - \mathbf{\Delta}'_k)^2$, as required.

Theorem 7. *Let $\mathbf{B} \in \mathbb{Z}_2^{n \times m}$ be a binary matrix with $n \geq m$. Furthermore suppose the tuples of parameters $\mathcal{P}_L = (d_L, d'_L, m_L, t_L)$ and $\mathcal{P}_C = (d_C, d'_C, m_C, t_C)$ are given which give depth $D_L = d_L + 1 + t_L(d'_L + \log(m_L) + 2)$ and $D_C = d_C + 1 + t_C(d'_C + \log(m_C) + 2)$ to the* `Low` *and* `Comp` *functions, respectively. Then, the depth of the* `HE-Reduce-Optimized` *(Algorithm 7) is $\frac{m(m-1)}{2}[D_L + D_C + 1]$ and its complexity is*

$$\mathcal{O}\big(m^3[1 + d'_C + t_C(d_C + \log(m_C))] + m^2[d'_L + t_L(d_L + m\log(m_L))]\big).$$

Proof. We proceed with an induction on m. For the base case, note that column $j = 0$ is trivially in reduced form and $\mathbf{\Delta}_1$ will certainly be in reduced form after a single comparison with $\mathbf{\Delta}_0$. This aligns with the worst case complexity for the original `Reduce` algorithm: $O(j^2)$ for column j, $O(n^3)$ overall [22].

For the inductive hypothesis, assume that for all $j \leq m-1$, that the depth of the `HE-Reduce-Optimized` algorithm, after termination on a $n \times j$ matrix, is $d(j) = \frac{j(j-1)}{2}D$.

Now consider an $n \times m$ matrix $\mathbf{B} = [\, \mathbf{x}_0 \mid ... \mid \mathbf{x}_{m-2} \mid \mathbf{x}_{m-1} \,] \in \mathbb{Z}_2^{n \times m}$. Then the sub-matrix $\mathbf{B}'$ obtained by excluding the last column $\mathbf{x}_{m-1}$ is an $n \times (m-1)$ matrix, and thus has depth $d(m-1) = \frac{(m-1)(m-2)}{2}D$ by the inductive hypothesis. Let us now focus on the last column, $\mathbf{x}_m$. Consider the

outer loop corresponding to $k = 0$ in the `HE-Reduce-Optimized` algorithm. After the first inner loop finishes, we have the depth of column $\mathbf{x}'_m$ is exactly $d(m-1) + D = \frac{(m-1)(m-2)}{2}D + D$, where the last D term is added from the very last update.

However, in `HE-Reduce-Optimized`, for all subsequent $k = 1, ..., m-1$, every k loop adds exactly D to the depth only one time. This is because every run of the inner j_0 **for** loop runs in parallel with ciphertexts of lower depth than the most recent update of $\mathbf{x}'_m$. A counting argument will yield that the depth of column $\mathbf{x'}_m$ after all loops are completed is $[\frac{(m-1)(m-2)}{2}D+D]+[(m-2)D] = \frac{m(m-1)}{2}D$, thus completing the induction.

As for the complexity, the optimized algorithm calls `Low` (which has complexity $O(m+d'_L+t_L(d_L+m\log m_L))$ exactly $m(m-1)/2$ times but still calls `LowComp` (which has complexity $O(d'_C + t_C(d_C + \log m_C))$) exactly $m(m-1)(2m-1)/6$ times. Thus, the overall complexity is as stated.

Remark 4. This algorithm performed on a boundary matrix $\mathbf{\Delta} \in \mathbb{Z}_2^{n\times n}$ is depth $n(n-1)/2\,[D_{\mathtt{L}} + D_{\mathtt{C}} + 1]$ and cost $\mathcal{O}(n^3 + n^2[d'_{\mathtt{L}} + t_{\mathtt{C}}(d_{\mathtt{L}} + n)])$ for the choice of $m_{\mathtt{C}} = m_{\mathtt{L}} = 2$, and assuming that $d'_{\mathtt{L}} > d'_{\mathtt{C}}$, $d_{\mathtt{L}} > d_{\mathtt{C}}$, and $t_{\mathtt{C}} \approx t_{\mathtt{L}}$.

4.2 Implementation Notes

In this section, we discuss our implementation of Algorithm 6 using HE. We assume that a `Client` generates `pk`, `sk`, `evk`, for some suitable `params`; and the `Server` knows `pk` and `evk`. Note that the `Server` can evaluate circuits on ciphertexts but cannot decrypt; see Section 2.2. By construction, variables of Algorithm 6 deals with vectors over the set $\mathbb{R}$ of real numbers and the approximate arithmetic is performed over $\mathbb{R}$. Additionally, as comparisons feature heavily in our implementation, we note that CKKS comparison circuits are comparable in amortized time to both BFV/BGV and TFHE schemes [33]. Therefore, the HEAAN [15] HE scheme, also known as the CKKS scheme, would be a suitable choice for implementing Algorithm 6. In CKKS, we have $\mathcal{M} = \mathbb{Z}[X]/\langle X^N + 1\rangle$ and $\mathcal{C} = \mathbb{Z}_Q[X]/\langle X^N + 1\rangle \times \mathbb{Z}_Q[X]/\langle X^N + 1\rangle$. Moreover, CKKS allows one to encode and encrypt $N/2$ numbers $[x_0, ..., x_{N/2-1}]$, $x_i \in \mathbb{R}$, as a single ciphertext, where ciphertext operations can be performed component-wise and simultaneously. As a result, under the setting of above CKKS parameters, a `Client` can encode and encrypt an $n \times n$ boundary matrix $\mathbf{\Delta}$ in at least two different ways: as n ciphertexts $c_0, ..., c_{n-1}$, where $c_i \in \mathcal{C}$ represents the encryption of the i'th column of $\mathbf{\Delta}$, which requires $n \leq N/2$; or as a single ciphertext c, where $c \in \mathcal{C}$ represents the encryption of the "concatenated columns of $\mathbf{\Delta}$"-vector, which requires $n \leq \sqrt{N/2}$.

For simplicity, we assume that a `Client` encrypts $\mathbf{\Delta}$ using the first method; obtains and sends $c_i \in \mathcal{C}$ to the `Server`. The `Server` can use `evk` and compute $c'_0, ..., c'_{n-1} \leftarrow \mathtt{Eval}_{\mathtt{evk}}(f; c_0, ..., c_{n-1})$, using ciphertext addition and multiplication operations[7], where f is the arithmetic circuit induced by Algorithm 6. The

[7] In practice, one would have to utilize other ciphertext operations like `Rotate`.

`Server` sends c'_i, $i = 0, ..., n-1$, back to the `Client`, who can use `sk` and decrypt c'_i to x'_i. Note that, by our previous arguments following Algorithm 6, $\texttt{Round}(x'_i)$ would match the ith column of $\texttt{Reduce}(\mathbf{\Delta})$.

In order to get a more concrete sense of the implementation of Algorithm 6 using CKKS, we consider CKKS parameters at $\lambda = 128$-bit security level, and set $N = 2^{17}$ and $Q = P \cdot q_0 \cdot \prod_{i=1}^{50} q_i$, as a product of 52 primes with $\log_2 Q = 3300$, $\log_2 P = 660$, and $\log_2 q_i \approx \delta = 51 < \log_2 q_0 < 2\delta = 102$; see Table 6 in [14]. This choice maximizes the depth L of circuits that HEAAN can evaluate, without bootstrapping, to $L = 50$ and the precision digits of data during computations is kept at 10. Under this choice of parameters, a `Client` can encode and encrypt boundary matrices of size $(n \times n)$, where $n \leq N/2 = 2^{16} (\sqrt{N/2} = 2^8)$ using the first (second) encoding approach. CKKS can handle circuits of depth up to $L = 50$ and so one would have to bootstrap [15] once the depth limit is exhausted. In our implementation, we use Algorithm 7, which is a slightly modified and optimized version of Algorithm 6. Our implementation, using Intel(R) 16-Core(TM) i9-9900K 3.60GHz, can reduce a single encrypted 3x3 matrices in 4.5 seconds with 40 bootstrappings using the (non-cryptographic) CKKS parameters $N = 2^5$, $Q \approx 2^{3188}$; and $\mathcal{P}_{\mathtt{L}} = \mathcal{P}_{\mathtt{C}} = (5, 5, 2, 5)$, where the parameters are chosen such that the underlying `Comp` in our computations uses one of the optimal parameters as reported in [16]. Note that Reducing 3x3 matrices takes 225 minutes using 128-bit secure CKKS parameters with $N = 2^{17}$. Note that if ciphertext slots are fully utilized then the amortized times would be $4.5/(2^4/(3+1)) = 1.125$ and $225 \cdot 60/(2^{16}/(3+1)) = 0.82$ seconds, respectively.

4.3 Limitations and Potential Improvements

A major challenge in implementing `HE-Reduce` using HE is the cubic co-factor n^3 in the depth of the underlying arithmetic circuit (even `HE-Reduce-Optimized` has a quadratic co-factor n^2, see Theorem 7.) As pointed out in an implementation scenario in Section 4.2, HEAAN can handle circuits up to depth 50 but the depth of `HE-Reduce-Optimized` quickly reaches large numbers as n grows and exceed 50 even for small values of n. Therefore, the size of the boundary matrix may be too large in practice to be practically reducable. Indeed, the Vietoris-Rips and Čech filtrations have 2^m simplices in the worst case for a point cloud with m points [45] since they define scales for every simplex in the powerset of the vertices (although it would be unusual to compute with simplices of all dimensions). Another challenge is to encode and encrypt $(n \times n)$ boundary matrices for large n. As noted in Section 4.2, currently suggested HEAAN parameters [14] at 128-bit security level limits $n < N/2 = 2^{16}$ or $n < \sqrt{N/2} = 2^8$, depending on the choice of encoding. Therefore, substantial improvements would be required before an efficient implementation of `HE-Reduce-Optimized` can be realized.

A possible improvement would be to reduce the size of the boundary matrix by the choice of filtration, which is an active field of research. For example, for a point cloud of size m in dimension d, the (weighted) alpha [21, 22] and sparse Rips filtrations [51] create complexes of size $m^{\mathcal{O}(d/2)}$ and $\mathcal{O}(m)$ respectively [45]. Very recent theoretical results also justify computing the PH of small subsets

of a point cloud to construct a distribution of PDs representing the topology of the original cloud [54]. This approach has the potential to massively reduce the size of each boundary matrix, whose reductions can be carried out completely in parallel. Another improvement would come from relaxing our theoretical bounds for parameters to reduce the depth in Theorem 7. Section 4.4 provides some motivating evidence of the feasibility and potential consequences of such an approach.

4.4 Empirical Results

The output of Algorithm 7 is an approximately binary matrix

$$\mathbf{R}' = \texttt{HE-Reduce-Optimized}(\mathbf{\Delta}; \mathcal{P}_{\texttt{L}}, \mathcal{P}_{\texttt{C}}) \in [0,1]^{n \times n}$$

which approximates the output of $\texttt{Reduce}(\mathbf{\Delta})$. The key bound in parameter selection is that throughout `HE-Reduce-Optimized`, the approximate binary vectors must never disagree with the true underlying binary vectors by more than $1/2n$, to ensure the output of `HE-Reduce-Optimized` returns an approximately binary vector with the same implied birth-death pairings as the exact `Reduce`. How prevalent are the cases in which the maximum error between the approximate and the exact reduced matrix exceeds $1/2n$? This question focuses on the accumulation of error throughout `HE-Reduce-Optimized` due to approximating exact operations in plaintext, and is independent of the noise growth that is accumulated by HE operations.

We explored this question in a fashion similar to the parameter relaxation experiment conducted in [16] by systematically increasing the parameters $\mathcal{P}_{\texttt{L}}$ and $\mathcal{P}_{\texttt{C}}$ of `HE-Reduce-Optimized` with respect to their depth and complexity to determine a minimum depth cofactor $D = D_L + D_C + 1$ (as defined in Theorem 7) which resulted in 100% accuracy. Specifically, for each parameter choice, we randomly sampled the space of 10×10, upper-triangular, binary matrices and compared the results of exact and approximate reductions, recording when all entries were within $1/2n$ and/or $1/2$ of the exact-reduced binary matrix. We found that the minimum depth (119) and complexity (55300) parameter pair for which 100% of the approximately reduced matrices were within $1/2n$ of their exact counterparts was $\mathcal{P}_{\texttt{L}} = (3, 3, 2, 6)$ and $\mathcal{P}_{\texttt{C}} = (3, 3, 2, 12)$, as reported in Table 1. That said, it may be that some matrices will exhibit an error in excess of the $1/2n$ tolerance for these parameter choices, although we expect such examples to be rare if they exist. By reducing $t_{\texttt{C}}$ from 12 to 11, we found only 81.2% of approximately reduced matrices had errors less than $1/2n$ and only 91.2% of matrices had maximum error was less than $1/2$—and so would still yield the correct reduced matrix after rounding (Table 1). By additionally raising t_L from 6 to 7 (so the circuit depth is again 119 but complexity is 51800) we find 98.6% of approximately reduced matrices had errors less than $1/2n$ and 100% of matrices had maximum error was less than $1/2$ (Table 1). These results in expected accuracy suggests there is moderate sensitivity to the choice of some parameters.

Table 1. Empirically-determined parameters for accurate reduction of 10×10 matrices. $*$ represents the lowest depth and complexity parameters of `HE-Reduce-Optimized` that exhibited correct reduction (within $1/2n$ error) of 100% of randomly chosen matrices.

	$\mathcal{P}_{\mathtt{L}}$	$\mathcal{P}_{\mathtt{C}}$	D	Complexity	Within $1/2n$	Within $1/2$
	(3, 3, 2, 6)	(3, 3, 2, 11)	113	51300	81.2%	91.2%
	(3, 3, 2, 7)	(3, 3, 2, 11)	119	51800	98.6%	100%
$*$	(3, 3, 2, 6)	(3, 3, 2, 12)	119	55300	100%	100%

We found that the same parameters for `HE-Reduce-Optimized` shown to correctly reduce random 10×10 matrices, also correctly reduces the 12×12 example boundary matrix given in Figure 2. Indeed, the maximum error in any component of the approximate reduced boundary matrix is 2.04e−3, well within the required $1/2n = 1/24 \approx 0.041$ tolerance to guarantee correct computation of column low 1s (Figure 3 (A)).

By relaxing some choices of accuracy parameters we observe failure cases where `HE-Reduce-Optimized` produces approximate binary matrices that do not cast to the exact reduced matrix. For instance, relaxing $t_{\mathtt{L}}$ from 5 to 6 returns a matrix that fails to be in reduced form, as both columns ab and bc have the same low 1s (Figure 3 (B)). By increasing $t_{\mathtt{C}}$ substantially, this issue is remedied, however, the approximately reduced matrix does not agree with the exact reduction (Figure 3 (C)). It is interesting to note that, in this case, the low 1s are all correct, and so the correct persistence diagram is computed.

Relaxing `LowComp` parameters also leads to failure, as shown in (Figure 3 (D)), where large errors accumulate during reduction leading to values that fall far outside the allowed range of $[0, 1]$.

5 Concluding Remarks and Future Research

We developed a new algorithm that enables key TDA computations in the ciphertext space using HE. We proved the correctness of our proposed algorithm, provided detailed correctness and complexity analysis, and an implementation of our algorithms using CKKS from the OpenFHE library [5]. We also presented some concrete directions for improvement and provided experimental results. To our knowledge, this is the first attempt to introduce secure computing for TDA. It would be interesting to extend and improve our results, and to implement secure TDA algorithms on realistic data sets.

The `Reduce` algorithm represents one of several fundamental components of TDA machinery which challenge existing technologies in the HE space. Another is the calculation of distances between PDs, which rely on combinatorial optimization algorithms to minimize the cost of matchings between persistence pairs in pairs of PDs [38]. Others include the numerous methods being broadly deployed to vectorize PDs for use with downstream ML models [2, 7, 9, 18, 47, 49]. HE-compatible implementations could allow remote processing of encrypted PDs and would immediately enable the use of existing implementations of encrypted

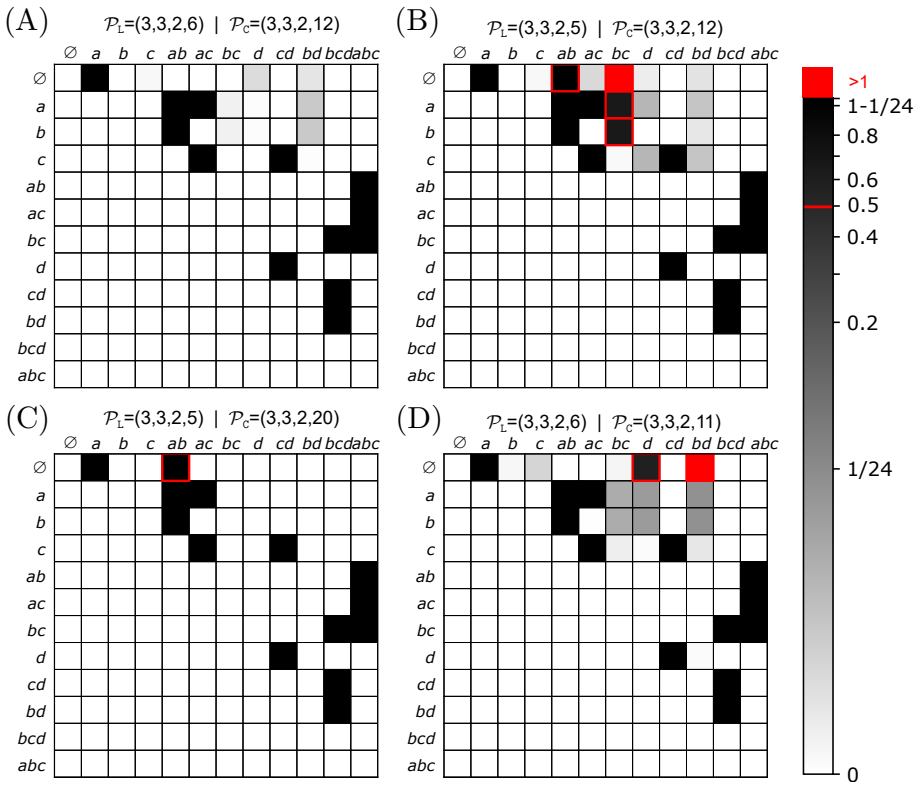

Fig. 3. Approximate reductions of the boundary matrix in Figure 2 using arithmetic circuits for various choices of algorithm parameters. Matrix entries are colored according their magnitude after approximate reduction by `HE-Reduce-Optimized`. Errors in excess of 1/2 are bordered in red. Entries with values in excess of 1 are colored red.

Euclidean distance calculations [13] and encrypted ML models that take as input finite-dimensional feature vectors [12,20,36]. We are hopeful these challenges will have implications beyond TDA-ML use cases by soliciting contributions from the broader HE community, and that the constraints imposed by HE will motivate new TDA approaches.

References

1. Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys **51**, 1–35 (2019)
2. Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., Ziegelmeier, L.: Persistence images: A stable vector representation of persistent homology. J. Mach. Learn. Res. **18**(1), 218–252 (January 2017)

3. Al Badawi, A., Chen, L., Vig, S.: Fast homomorphic svm inference on encrypted data. Neural Computing and Applications **34**(18), 15555–15573 (Sep 2022), https://doi.org/10.1007/s00521-022-07202-8
4. Aloufi, A., Hu, P., Wong, H.W.H., Chow, S.S.M.: Blindfolded evaluation of random forests with multi-key homomorphic encryption. IEEE Transactions on Dependable and Secure Computing **18**(4), 1821–1835 (2021). https://doi.org/10.1109/TDSC.2019.2940020
5. Badawi, A., Bates, J., Bergamaschi, F., Cousins, D., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Quah, I., Polyakov, Y., Saraswathy, R., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: OpenFHE: Open-Source Fully Homomorphic Encryption Library. Cryptology ePrint Archive, Paper 2022/915 (2022), https://eprint.iacr.org/2022/915
6. Braun, T., Fung, B.C.M., Iqbal, F., Shah, B.: Security and privacy challenges in smart cities. Sustainable Cities and Society **39**, 499–507 (2018), https://www.sciencedirect.com/science/article/pii/S2210670717310272
7. Bubenik, P.: Statistical topological data analysis using persistence landscapes. Journal of Machine Learning Research **16**, 77–102 (01 2015)
8. Bukkuri, A., Andor, N., Darcy, I.K.: Applications of topological data analysis in oncology. Frontiers in Artificial Intelligence **4** (2021). https://doi.org/10.3389/frai.2021.659037, https://www.frontiersin.org/articles/10.3389/frai.2021.659037
9. Carriere, M., Chazal, F., Ike, Y., Lacombe, T., Royer, M., Umeda, Y.: PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures. Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) **108**, 2786–2796 (2020), https://proceedings.mlr.press/v108/carriere20a.html
10. Chazal, F., Michel, B.: An introduction to topological data analysis: Fundamental and practical aspects for data scientists. Frontiers in Artificial Intelligence **4** (2021). https://doi.org/10.3389/frai.2021.667963, https://www.frontiersin.org/article/10.3389/frai.2021.667963
11. Chazal, F., de Silva, V., Oudot, S.: Persistence stability for geometric complexes. Geometriae Dedicata **173**, 193–214 (2012)
12. Chen, H., Gilad-Bachrach, R., Han, K., Huang, Z., Jalali, A., Laine, K., Lauter, K.: Logistic regression over encrypted data from fully homomorphic encryption. BMC Medical Genomics **11**, 1–12 (2018)
13. Chen, H., Laine, K., Player, R.: Simple encrypted arithmetic library - seal v2.1. In: Brenner, M., Rohloff, K., Bonneau, J., Miller, A., Ryan, P.Y., Teague, V., Bracciali, A., Sala, M., Pintore, F., Jakobsson, M. (eds.) Financial Cryptography and Data Security. pp. 3–18. Springer International Publishing, Cham (2017)
14. Cheon, H., Son, Y., Yhee, D.: Practical FHE parameters against lattice attacks. Journal of the Korean Mathematical Society **59**, 35–51 (2022)
15. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. Advances in Cryptology – ASIACRYPT 2017, Lecture Notes in Computer Science **10624**, 409–437 (2017)
16. Cheon, J.H., Kim, D., Kim, D., Lee, H.H., Lee, K.: Numerical method for comparison on homomorphically encrypted numbers. Advances in Cryptology – ASIACRYPT 2019, Lecture Notes in Computer Science **11922**, 415–445 (2019)
17. Chialva, D., Dooms, A.: Conditionals in homomorphic encryption and machine learning applications. CoRR **abs/1810.12380** (2018), http://arxiv.org/abs/1810.12380

18. Chung, Y.M., Lawson, A.: Persistence Curves: A canonical framework for summarizing persistence diagrams. Advances in Computational Mathematics **48**(1), 6 (Jan 2022), https://doi.org/10.1007/s10444-021-09893-4
19. Cohen-Steiner, D., Edelsbrunner, H., Harer, J., Mileyko, Y.: Lipschitz functions have lp-stable persistence. Found. Comput. Math. **10**(2), 127–139 (2010), http://dblp.uni-trier.de/db/journals/focm/focm10.html#Cohen-SteinerEHM10
20. Crockett, E.: A low-depth homomorphic circuit for logistic regression model training. In: WAHC 2020 Workshop on Encrypted Computing and Applied Homomorphic Cryptography (2020)
21. Edelsbrunner, H.: The union of balls and its dual shape. Discrete & Computational Geometry **13**(3), 415–440 (Jun 1995), https://doi.org/10.1007/BF02574053
22. Edelsbrunner, H., Harer, J.L.: Computational Topology - an Introduction. American Mathematical Society (2010)
23. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science. p. 454. FOCS '00, IEEE Computer Society, USA (2000)
24. Fang, H., Qian, Q.: Privacy preserving machine learning with homomorphic encryption and federated learning. Future Internet **13**(4), 94 (2021)
25. Giacomelli, I., Jha, S., Joye, M., Page, C.D., Yoon, K.: Privacy-preserving ridge regression with only linearly-homomorphic encryption. In: Preneel, B., Vercauteren, F. (eds.) Applied Cryptography and Network Security. pp. 243–261. Springer International Publishing, Cham (2018)
26. Gidea, M., Katz, Y.: Topological data analysis of financial time series: Landscapes of crashes. Physica A: Statistical Mechanics and its Applications **491**, 820–834 (2018). https://doi.org/https://doi.org/10.1016/j.physa.2017.09.028, https://www.sciencedirect.com/science/article/pii/S0378437117309202
27. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of The 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 201–210. PMLR, New York, New York, USA (20–22 Jun 2016), https://proceedings.mlr.press/v48/gilad-bachrach16.html
28. Grubbs, P., Ristenpart, T., Shmatikov, V.: Why your encrypted database is not secure. Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS) pp. 162–168 (2017)
29. Hacigümüs, H., Iyer, B.R., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD). pp. 216–227 (2002)
30. Huynh, D.: Cryptotree: fast and accurate predictions on encrypted structured data. CoRR **abs/2006.08299** (2020), https://arxiv.org/abs/2006.08299
31. Iliashenko, I., Zucca, V.: Faster homomorphic comparison operations for BGV and BFV. Proceedings of Privacy Enhancing Technologies **2021**(3), 246–264 (2021)
32. Iliashenko, I., Zucca, V.: Faster homomorphic comparison operations for bgv and bfv. Proceedings on Privacy Enhancing Technologies **2021**, 246 – 264 (2021)
33. Iliashenko, I., Zucca, V.: Faster homomorphic comparison operations for BGV and BFV. Proceedings on Privacy Enhancing Technologies **2021**(3), 246–264 (2021). https://doi.org/10.2478/popets-2021-0046, https://hal.science/hal-03506798
34. Jäschke, A., Armknecht, F.: Unsupervised machine learning on encrypted data. In: Cid, C., Jacobson Jr., M.J. (eds.) Selected Areas in Cryptography – SAC 2018, Lecture Notes in Computer Science. pp. 453–478 (2019)

35. Jiang, X., Kim, M., Lauter, K., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. p. 1209–1222. CCS '18, Association for Computing Machinery, New York, NY, USA (2018), https://doi.org/10.1145/3243734.3243837
36. Kaissis, G., Makowski, M.R., Rueckert, D., Braren, R.: Secure, privacy-preserving and federated machine learning in medical imaging. Nature Machine Intelligence **2**(6), 305–311 (2020)
37. Karn, R.R., Elfadel, I.A.M.: Confidential inference in decision trees: Fpga design and implementation. In: 2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC). pp. 1–6 (2022). https://doi.org/10.1109/VLSI-SoC54400.2022.9939567
38. Kerber, M., Morozov, D., Nigmetov, A.: Geometry helps to compare persistence diagrams. Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX) pp. 1–10 (2016)
39. Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X.: Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR Medical Informatics **6** (2018)
40. Konečný, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. CoRR **abs/1610.02527** (2016), http://arxiv.org/abs/1610.02527
41. Lee, J.W., Kang, H., Lee, Y., Choi, W., Eom, J., Deryabin, M., Lee, E., Lee, J., Yoo, ., Kim, Y.S., No, J.S.: Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. IEEE Access **10**, 30039–30054 (2022)
42. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR **54**, 1273–1282 (2017)
43. Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., Taft, N.: Privacy-preserving ridge regression on hundreds of millions of records. In: 2013 IEEE Symposium on Security and Privacy. pp. 334–348 (2013). https://doi.org/10.1109/SP.2013.30
44. Otter, N., Porter, M.A., Tillmann, U., Grindrod, P., Harrington, H.A.: A roadmap for the computation of persistent homology. EPJ Data Science **6**(1), 17 (2017), https://doi.org/10.1140/epjds/s13688-017-0109-5
45. Otter, N., Porter, M.A., Tillmann, U., Grindrod, P., Harrington, H.A.: A roadmap for the computation of persistent homology. EPJ Data Science **6** (2017), https://doi.org/10.1140/epjds/s13688-017-0109-5
46. Park, S., Byun, J., Lee, J., Cheon, J.H., Lee, J.: He-friendly algorithm for privacy-preserving svm training. IEEE Access **8**, 57414–57425 (2020). https://doi.org/10.1109/ACCESS.2020.2981818
47. Perea, J.A., Munch, E., Khasawneh, F.A.: Approximating Continuous Functions on Persistence Diagrams Using Template Functions. Foundations of Computational Mathematics (Jun 2022). https://doi.org/10.1007/s10208-022-09567-7, https://doi.org/10.1007/s10208-022-09567-7
48. Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H.: CryptDB: protecting confidentiality with encrypted query processing. Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP), Association for Computing Machinery pp. 85–100 (2011), https://doi.org/10.1145/2043556.2043566

49. Reininghaus, J., Huber, S., Bauer, U., Kwitt, R.: A stable multi-scale kernel for topological machine learning. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 4741–4748 (2015)
50. Shafieinejad, M., Gupta, S., Liu, J.Y., Karabina, K., Kerschbaum, F.: Equi-joins over encrypted data for series of queries. IEEE 38th International Conference on Data Engineering (ICDE) pp. 1635–1648 (2022)
51. Sheehy, D.R.: Linear-size approximations to the vietoris-rips filtration. In: Proceedings of the Twenty-Eighth Annual Symposium on Computational Geometry. p. 239–248. SoCG '12, Association for Computing Machinery, New York, NY, USA (2012), https://doi.org/10.1145/2261250.2261286
52. Shnier, D., Voineagu, M.A., Voineagu, I.: Persistent homology analysis of brain transcriptome data in autism. Journal of The Royal Society Interface **16**(158), 20190531 (2019). https://doi.org/10.1098/rsif.2019.0531, https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2019.0531
53. Skraba, P., Turner, K.: Wasserstein stability for persistence diagrams (2021)
54. Solomon, E., Wagner, A., Bendich, P.: From Geometry to Topology: Inverse Theorems for Distributed Persistence. In: Goaoc, X., Kerber, M. (eds.) 38th International Symposium on Computational Geometry (SoCG 2022). Leibniz International Proceedings in Informatics (LIPIcs), vol. 224, pp. 61:1–61:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022). https://doi.org/10.4230/LIPIcs.SoCG.2022.61, https://drops.dagstuhl.de/opus/volltexte/2022/16069
55. Togan, M., Pleşca, C.: Comparison-based computations over fully homomorphic encrypted data. In: 2014 10th International Conference on Communications (COMM). pp. 1–6 (2014). https://doi.org/10.1109/ICComm.2014.6866760
56. Upmanyu, M., Namboodiri, A.M., Srinathan, K., Jawahar, C.V.: Efficient privacy preserving video surveillance. 2009 IEEE 12th International Conference on Computer Vision pp. 1639–1646 (2009)
57. Wagner, H., Chen, C., Vuçini, E.: Efficient Computation of Persistent Homology for Cubical Data, pp. 91–106. Springer Berlin Heidelberg, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-23175-9_7
58. Wu, D.J., Feng, T., Naehrig, M., Lauter, K.E.: Privately evaluating decision trees and random forests. Proceedings on Privacy Enhancing Technologies **2016**, 335 – 355 (2016)
59. Xu, R., Joshi, J.D., Li, C.: Cryptonn: Training neural networks over encrypted data. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). pp. 1199–1209. IEEE Computer Society, Los Alamitos, CA, USA (jul 2019), https://doi.ieeecomputersociety.org/10.1109/ICDCS.2019.00121
60. Xu, R., Joshi, J., Li, C.: Nn-emd: Efficiently training neural networks using encrypted multi-sourced datasets. IEEE Transactions on Dependable and Secure Computing **19**(4), 2807–2820 (2022). https://doi.org/10.1109/TDSC.2021.3074439
61. Yao, A.: How to generate and exchange secrets. Proceeedings of the 27th Annual Symposium on Foundations of Computer Science pp. 162–167 (1986)
62. Yu, J., Lu, X., Gao, Z., Li, S., Li, L., Li, C., Tong, H.: Topology control of wsns based on persistent homology. Journal of Physics: Conference Series **2333**(1), 012003 (aug 2022), https://dx.doi.org/10.1088/1742-6596/2333/1/012003
63. Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C., Li, H., Tan, Y.: Secure multiparty computation: Theory, practice and applications. Information Sciences **476**, 357–372 (2019)

64. Zhou, Y., Shen, H., Zhang, M.: A distributed and privacy-preserving random forest evaluation scheme with fine grained access control. Symmetry **14**(2), 415 (Feb 2022), http://dx.doi.org/10.3390/sym14020415
65. Zomorodian, A.: Fast construction of the vietoris-rips complex. Computers & Graphics **34**(3), 263–271 (2010). https://doi.org/https://doi.org/10.1016/j.cag.2010.03.007, https://www.sciencedirect.com/science/article/pii/S0097849310000464, shape Modelling International (SMI) Conference 2010

Appendix

A `Low` Correctness Proofs

(**Proof of Lemma 1**). First note that $S(\mathbf{v}) \in \mathcal{D}^n$, since all entries are necessarily distinct by definition of Transformation 1, and so $maxidx(S(\mathbf{v}))$ is defined.

Suppose that $low(\mathbf{v}) = k$ for some $0 \leq k \leq n-1$. We have to show $S(\mathbf{v})[k] > S(\mathbf{v})[j]$ for all $0 \leq j \leq n-1$ and $j \neq k$.

Case 1: $0 \leq j < k$. Note that $\mathbf{v}[k] = 1$ and that $\mathbf{v}[j] \in \{0, 1\}$. Therefore,

$$S(\mathbf{v})[k] = \mathbf{v}[k] + k/n = 1 + k/n > \mathbf{v}[j] + j/n = S(\mathbf{v})[j].$$

Case 2: $0 \leq k < j \leq n-1$. Note that $\mathbf{v}[j] = 0$ because $low(\mathbf{v}) = k$ is the largest index with $\mathbf{v}[k] = 1$. Therefore, $S(\mathbf{v})[j] = \mathbf{v}[j] + j/n = j/n$ and that

$$S(\mathbf{v})[k] = 1 + k/n > (n-1)/n \geq j/n = S(\mathbf{v})[j].$$

Remark 5. The same argument in the proof of Lemma 1 would work for any transformation S that is strictly monotonically increasing on $\{0, 1, \ldots, n-1\}$ and strictly bounded by 1. However, in the implementation of `MaxIdx`, which we use to approximate *maxidx* (and thus *low*), the rate of convergence is increasing with the distance between distinct values in $\mathbf{v}$, and so a nonlinear choice of S may have the effect of decreasing the rate of convergence, at least for some input vectors. Further analysis would be required to find an optimal choice for S.

(**Proof of Lemma 2**). Suppose that $low(\mathbf{v}) = k$ for some $0 \leq k \leq n-1$. We have to show $S(\mathbf{v}')[k] > S(\mathbf{v}')[j]$ for all $0 \leq j \leq n-1$ and $j \neq k$.

Case 1: $0 \leq j < k$. Note that $\mathbf{v}[k] = 1$, $\mathbf{v}'[k] > 1 - 1/2n$, $\mathbf{v}[j] \in \{0, 1\}$, and $\mathbf{v}'[j] < 1 + 1/2n$. Therefore,

$$\begin{aligned} S(\mathbf{v}')[k] &= \mathbf{v}'[k] + k/n > (1 - 1/2n) + k/n \\ &> (1 + 1/2n) + (k-1)/n > \mathbf{v}'[j] + j/n = S(\mathbf{v}')[j]. \end{aligned}$$

Case 2: $0 \leq k < j \leq n-1$. Note that $\mathbf{v}[j] = 0$ because $low(\mathbf{v}) = k$ is the largest index with $\mathbf{v}[k] = 1$. Moreover, $\mathbf{v}'[k] > 1 - 1/2n$ and $\mathbf{v}'[j] < 1/2n$. Therefore,

$$\begin{aligned} S(\mathbf{v}')[k] &= \mathbf{v}'[k] + k/n > (1 - 1/2n) + k/n \geq 1 - 1/2n \\ &= 1/2n + (n-1)/n > \mathbf{v}'[j] + j/n = S(\mathbf{v}')[j]. \end{aligned}$$

Lemma 4. *Let $\varepsilon > 0$ and let $\mathbf{e}_j$ denote the standard n-dimensional basis vector, with $\mathbf{e}_j[j] = 1$ and $\mathbf{e}_j[i] = 0$ for all $i \neq j$. Fix parameters d, d', m, t for the* `MaxIdx` *algorithm so that $|\texttt{MaxIdx}(\mathbf{x}; d, d', m, t) - \mathbf{e}_{maxidx(\mathbf{x})}| < 2^{-\alpha}$, for all $\mathbf{x} \in [\frac{1}{2}, \frac{3}{2})^n$. Then, for any binary vector $\mathbf{v} \in \{0, 1\}^n$,*

$$\left|\texttt{Low}(\mathbf{v}; d, d', m, t) - low(\mathbf{v})\right| < \frac{n(n-1)}{2}(2^{-\alpha}),$$

where `Low`*, low, and* `MaxIdx` *are computed as described in* [16].

Proof. To ease notation let $\mathbf{b} = \texttt{MaxIdx}(\mathbf{v}; d, d', m, t)$. Assume the paramaters d, d', m and t have been chosen so that

$$\left|\mathbf{b}[i] - \mathbf{e}_k[i]\right| < 2^{-\alpha}$$

for all $0 \leq i < n-1$, if $k = maxidx(\mathbf{v})$. Then

$$\begin{aligned}
\left|\texttt{Low}(\mathbf{v}; d, d', m, t) - low(\mathbf{v})\right| &= \left|\sum_{i=0}^{n-1} i\mathbf{b}[i] - \sum_{i=0}^{n-1} i\mathbf{e}_j[i]\right| \\
&= \left|\sum_{i=0}^{n-1} i(\mathbf{b}[i] - \mathbf{e}_j[i])\right| \\
&\leq \sum_{i=0}^{n-1} \left|i(\mathbf{b}[i] - \mathbf{e}_j[i])\right| \\
&< 2^{-\alpha} \sum_{i=0}^{n-1} i \\
&= \frac{n(n-1)}{2}\left(2^{-\alpha}\right).
\end{aligned}$$

Lemma 5. *Let $\alpha > 0$ and fix parameters d, d', m, t for the* `MaxIdx` *algorithm so that $|\texttt{MaxIdx}(\mathbf{x}; d, d', m, t) - \mathbf{e}_{maxidx(\mathbf{x})}| < 2^{-\alpha}$, for all $\mathbf{x} \in [\frac{1}{2}, \frac{3}{2})^n$. Further assume $\mathbf{v}' \in [0,1]^n$ and $\mathbf{v} \in \{0,1\}^n$ are such that $|\mathbf{v}' - \mathbf{v}| < \frac{1}{2n}$. Then*

$$\left|\texttt{Low}(\mathbf{v}'; d, d', m, t) - \texttt{Low}(\mathbf{v}; d, d', m, t)\right| < n(n-1)2^{-\alpha}$$

Proof. Recall that the `Low` algorithm presented in Algorithm 3 requires the input vectors to undergo two transformations, S and $T_{\texttt{L}}$, before being fed into the `MaxIdx` algorithm. Let $\mathbf{x}' = T_{\texttt{L}}(S(\mathbf{v}'))$ and $\mathbf{x} = T_{\texttt{L}}(S(\mathbf{v}))$. The problem now reduces to bounding

$$\left|\texttt{MaxIdx}(\mathbf{x}'; d, d', m, t) - \texttt{MaxIdx}(\mathbf{x}; d, d', m, t)\right|$$

where $|\mathbf{x}' - \mathbf{x}| < \frac{1}{4n} < \frac{1}{2n}$. By Lemmas 1 and 2

$$\begin{aligned}
maxidx(\mathbf{x}') &= maxidx(T_{\texttt{L}}(S(\mathbf{v}'))) = low(\mathbf{v}) \\
&= maxidx(T_{\texttt{L}}(S(\mathbf{v}))) = maxidx(\mathbf{x}),
\end{aligned}$$

and so a triangle inequality obtained by adding and subtracting $\mathbf{e}_{maxidx(\mathbf{x}')}$ and $\mathbf{e}_{maxidx(\mathbf{x})}$ yields

$$\begin{aligned}
&\left|\texttt{MaxIdx}(\mathbf{x}'; d, d', m, t) - \texttt{MaxIdx}(\mathbf{x}; d, d', m, t)\right| \\
&\qquad < 2^{-\alpha} + 0 + 2^{-\alpha} = 2^{-\alpha+1}.
\end{aligned}$$

Letting

$$\begin{aligned}
\mathbf{b} &= \texttt{MaxIdx}(\mathbf{x}; d, d', m, t) \\
\mathbf{b}' &= \texttt{MaxIdx}(\mathbf{x}'; d, d', m, t),
\end{aligned}$$

we have

$$\begin{aligned}\left|\texttt{Low}(\mathbf{v}';d,d',m,t)-\texttt{Low}(\mathbf{v};d,d',m,t)\right| &= \left|\sum_{i=0}^{n-1} i\mathbf{b}'[i]-\sum_{i=0}^{n-1} i\mathbf{b}[i]\right| \\ &= \left|\sum_{i=0}^{n-1} i(\mathbf{b}'[i]-\mathbf{b}[i])\right| \\ &\leq \sum_{i=0}^{n-1}\left|i(\mathbf{b}'[i]-\mathbf{b}[i])\right| < 2^{-\alpha+1}\sum_{i=0}^{n-1} i \\ &= \frac{n(n-1)}{2}(2^{-\alpha+1}) = n(n-1)2^{-\alpha}.\end{aligned}$$

B `Low` Parameters Proofs

Proposition 8. *Fix $n \geq 2$ and assume $\mathbf{v} \in \{0,1\}^n$ and $\mathbf{v}' \in [0,1]^n$ satisfy $|\mathbf{v}-\mathbf{v}'| \leq \frac{\varepsilon}{2n}$, for some $0 \leq \varepsilon < 1$. Let $\mathbf{x}' = T_L(S(\mathbf{v}'))$ and denote the $(i+1)$-th smallest value of $\mathbf{x}'$ by $\mathbf{x}'_{(i)}$, so $\min\{\mathbf{x}'[i] \mid 0 \leq i \leq n-1\} =: \mathbf{x}'_{(0)} < \mathbf{x}'_{(1)} < \ldots < \mathbf{x}'_{(n-1)} := \max\{\mathbf{x}'[i] \mid 0 \leq i \leq n-1\}$. Let*

$$c = \mathbf{x}'_{(n-1)}/\mathbf{x}'_{(n-2)}$$

be the ratio of the largest coordinate value of $\mathbf{x}'$ over the second largest value. Then

$$c \geq 1 + \frac{2-2\varepsilon}{6n-4+\varepsilon}.$$

Proof. Suppose $low(\mathbf{v}) = k$. By Lemma 2, $\mathbf{x}'[k]$ has the highest value in the vector. Define the two sets

$$\begin{aligned}M_1 &= \{j \mid \mathbf{v}[j] = 1 \text{ and } j < k\} \\ &= \left\{j \mid \mathbf{v}'[j] \geq 1 - \frac{\varepsilon}{2n} \text{ and } j < k\right\}\end{aligned}$$

and

$$M_0 = \{j \mid \mathbf{v}[j] = 0\} = \left\{j \mid \mathbf{v}'[j] \leq \frac{\varepsilon}{2n}\right\}.$$

There are two cases to consider, either M_1 is empty or not.

Case 1: If M_1 is empty, let $m = \max M_0$. Since S is an increasing function with respect to index, and T_{L} is strictly increasing we have that $T_{\mathrm{L}}(S(\mathbf{v}'))[m] = \mathbf{x}'[m]$ is necessarily the second highest value. This is because

$$\mathbf{x}'[m] = \frac{\mathbf{v}'[m]+\frac{m}{n}+1}{2} \geq \frac{\frac{m}{n}+1}{2} > \frac{\frac{\varepsilon}{2n}+\frac{j}{n}+1}{2} > \frac{\mathbf{v}'[j]+\frac{j}{n}+1}{2} = \mathbf{x}'[j]$$

for $0 \le j < m \le n-1$. The middle inequality follows because $m, n \in \mathbb{Z}$ and $m > j \implies m - j \ge 1 \implies \frac{m}{n} \ge \frac{1}{n} + \frac{j}{n} > \frac{\varepsilon}{2n} + \frac{j}{n}$. Thus,

$$\begin{aligned}\frac{\mathbf{x}'_{(n-1)}}{\mathbf{x}'_{(n-2)}} = \frac{\mathbf{x}'[k]}{\mathbf{x}'[m]} &> \frac{1 - \frac{\varepsilon}{2n} + \frac{k}{n} + 1}{\frac{\varepsilon}{2n} + \frac{n-1}{n} + 1} \\ &= 1 + \frac{2k + 2 - 2\varepsilon}{4n - 2 + \varepsilon} \ge 1 + \frac{2 - 2\varepsilon}{4n - 2 + \varepsilon}.\end{aligned}$$

Case 2: If M_1 is not empty, let $m = \max M_1$. We first show that $\mathbf{x}'[i] > \mathbf{x}'[j]$ for all $i \in M_1, j \in M_0$; that is, all transformed approximate 1's are larger than all transformed approximate 0's. Let $j \in M_0$ and $i \in M_1$ be arbitrary. Then

$$\begin{aligned}\mathbf{v}'[j] + \frac{j}{n} < \frac{\varepsilon}{2n} + \frac{j}{n} &\le \frac{\varepsilon}{2n} + \frac{n-1}{n} \\ &\le 1 - \frac{1}{2n} \le 1 - \frac{\varepsilon}{2n} + \frac{i}{n} < \mathbf{v}'[i] + \frac{i}{n}\end{aligned}$$

for $i \ge 0$. Since $T_{\mathtt{L}}$ is an increasing function, we have that $\mathbf{x}'[j] < \mathbf{x}'[i]$ as desired. Thus, the second largest coordinate of $\mathbf{x}'$ is $\mathbf{x}'[m] > (1 - \frac{\varepsilon}{2n} + \frac{m}{n} + 1)/2$, where $m = \max M_1$. Necessarily $m \le k - 1$, and so we have that

$$\begin{aligned}\frac{\mathbf{x}'_{(n-1)}}{\mathbf{x}'_{(n-2)}} = \frac{\mathbf{x}'[k]}{\mathbf{x}'[m]} &> \frac{1 - \frac{\varepsilon}{2n} + \frac{k}{n} + 1}{1 + \frac{k-1}{n} + 1} \\ &= 1 + \frac{2 - \varepsilon}{4n + 2k - 2} \ge 1 + \frac{2 - \varepsilon}{6n - 4}.\end{aligned}$$

Thus, as n and ε were chosen from the beginning, we know that the ratio between the largest and second largest value for any "approximate" binary vector is bounded below by $\min\left\{1 + \frac{2-2\varepsilon}{4n-2+\varepsilon}, 1 + \frac{2-\varepsilon}{6n-4}\right\}$. However, both $\frac{2-2\varepsilon}{4n-2+\varepsilon}$ and $\frac{2-\varepsilon}{6n-4}$ are greater than or equal to $\frac{2-2\varepsilon}{6n-4+\varepsilon}$, with the former being immediate for $n \ge 1$ and the latter is true as

$$\frac{2 - \varepsilon}{6n - 4} \ge \frac{2 - 2\varepsilon}{6n - 4} \ge \frac{2 - 2\varepsilon}{6n - 4 + \varepsilon}, \text{ for } 0 \le \varepsilon < 1.$$

And so, it follows that

$$c \ge \min\left\{1 + \frac{2 - 2\varepsilon}{4n - 2 + \varepsilon}, 1 + \frac{2 - \varepsilon}{6n - 4}\right\} \ge 1 + \frac{2 - 2\varepsilon}{6n - 4 + \varepsilon}.$$

Lemma 6. *Assume $\mathbf{v} \in \{0,1\}^n$ and $\mathbf{v}' \in [0,1]^n$ satisfy $|\mathbf{v} - \mathbf{v}'| \le \frac{\varepsilon}{2n}$, for some $0 \le \varepsilon < 1$. Consider Theorem 5 in [16], which gives the parameters (d, d', m, t) for the* `MaxIdx` *function for a given desired accuracy of $2^{-\alpha}$. If α is chosen such that*

$$\alpha > \log(3) + 2\log(n) - \log(\delta) - 1$$

then $|\mathtt{Low}(\mathbf{v}'; d, d', m, t) - low(\mathbf{v})| < \delta$.

Proof. We note that from THEOREM 1 it suffices to have

$$\begin{aligned}
&\frac{3}{2}n(n-1)2^{-\alpha} < \frac{3n^2}{2^{\alpha+1}} < \delta \\
&\Leftrightarrow 2^{\alpha+1} > \frac{3n^2}{\delta} \\
&\Leftrightarrow \alpha > \log(3) + 2\log(n) - \log(\delta) - 1
\end{aligned}$$

(**Proof of Theorem 3**). LEMMA 6 provides the corresponding α needed for a desired δ-error in `Low`. That is, if one desires error δ in $|\texttt{Low}(\mathbf{v}') - low(\mathbf{v})|$, then one may choose $\alpha > \log(3) + 2\log(n) - \log(\delta) - 1$.

c's lower bound has been found in PROPOSITION 8. In particular, by PROPOSITION 8, we know that $c \geq 1 + \frac{2-2\varepsilon}{6n-4+\varepsilon}$. But this is equivalent to saying that

$$\begin{aligned}
&-\log(\log(c)) \leq -\log\left(\log\left(1 + \frac{2-2\varepsilon}{6n-4+\varepsilon}\right)\right) \\
\Longrightarrow\ &\log(\alpha + 1 + \log(n)) - \log(\log(c)) \\
&\leq \log(\alpha + 1 + \log(n)) - \log\left(\log\left(1 + \frac{2-2\varepsilon}{6n-4+\varepsilon}\right)\right)
\end{aligned}$$

Therefore, if we choose

$$t \geq \frac{\log(\alpha + 1 + \log(n)) - \log\left(\log\left(1 + \frac{2-2\varepsilon}{6n-4+\varepsilon}\right)\right)}{\log m}$$

we fulfill the former inequality in Theorem 5 (from [16]).

At this point, both α and t are determined. As $\min\{d, d'\}$ is dependent upon these choice of parameters, our choice of $\min\{d, d'\}$ is also determined.

C `LowComp` Correctness Proofs

(**Proof of Theorem 4**). Recall LEMMA 3 and the following chain of if and only if implications:

$$\begin{aligned}
&lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}}) = 1 \Leftrightarrow low(\mathbf{x}) = low(\mathbf{y}) \Leftrightarrow \phi \geq |\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'}| \\
&\Leftrightarrow \phi^2 \geq (\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2 \Leftrightarrow T_{\mathtt{C}}(\phi^2) \geq T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2
\end{aligned}$$

There are two cases: $lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}}) = 1$ or $lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}}) = 0$. Case 1: $lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}}) = 1$. Then $T_{\mathtt{C}}(\phi^2) \geq T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2$ which implies that

$$\begin{aligned}
&|\mathtt{Comp}(T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2; d, d', m, t) \\
&\quad - comp(T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2)| < \eta \\
\Longrightarrow\ &|\mathtt{Comp}(T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2; d, d', m, t) - 1| < \eta \\
\Longrightarrow\ &1 - \eta < \mathtt{Comp}(T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2; d, d', m, t) < 1 \\
\Longrightarrow\ &1 - \eta < \mathtt{LowComp}\mathtt{L}_{\mathbf{x}'}, \mathtt{L}_{\mathbf{y}'}; d, d', m, t) < 1.
\end{aligned}$$

Case 2: $lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}}) = 0$. Then $T_{\mathtt{C}}(\phi^2) < T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2$ which implies that

$$\begin{aligned}
&|\mathtt{Comp}(T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2; d, d', m, t) \\
&\quad - comp(T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2)| < \eta \\
\implies &|\mathtt{Comp}(T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2; d, d', m, t) - 0| < \eta \\
\implies &0 < \mathtt{Comp}(T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}(\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{y}'})^2; d, d', m, t) < \eta \\
\implies &0 < \mathtt{LowComp}(\mathtt{L}_{\mathbf{x}'}, \mathtt{L}_{\mathbf{y}'}; d, d', m, t) < \eta.
\end{aligned}$$

D LowComp Parameters Proofs

Proposition 9. *Consider positive $a < b < c$. Then the value of b which optimizes the problem*

$$\max_{b \in (a,c)} \left(\min \left\{ \frac{b}{a}, \frac{c}{b} \right\} \right)$$

is $b = \sqrt{ac}$. In other words, b is equal to the geometric mean of the endpoints. Furthermore, for this choice of b, we have that

$$\max_{b \in (a,c)} \left(\min \left\{ \frac{b}{a}, \frac{c}{b} \right\} \right) = \sqrt{\frac{c}{a}}.$$

Proof. Suppose $b = \sqrt{ac}$. Then $\frac{b}{a} = \frac{c}{b} = \sqrt{\frac{c}{a}}$. If $b > \sqrt{ac}$, then $\frac{b}{a} > \frac{c}{b}$, implying that the minimum of the two ratios is $\frac{c}{b}$. But then $\frac{c}{b} < \frac{c}{\sqrt{ac}} = \sqrt{\frac{c}{a}}$. Similarly, if $b < \sqrt{ac}$, then $\frac{b}{a} < \frac{c}{b}$, so that the minimum of the two ratios is now $\frac{b}{a}$. It follows that $\frac{b}{a} < \frac{\sqrt{ac}}{a} = \sqrt{\frac{c}{a}}$. Thus the value of b over (a, c) which maximizes the $\min \left\{ \frac{b}{a}, \frac{c}{b} \right\}$ is $b = \sqrt{ac}$ and the maximum value is $\sqrt{\frac{c}{a}}$.

(**Proof of Corollary 1**). The value of $T_{\mathtt{C}}(\phi^2)$ comes immediately from applying the preceding PROPOSITION 9. This proposition also establishes c's lower bound, as

$$\begin{aligned}
c &= \frac{\max \left\{ T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}((\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{x}'})^2) \right\}}{\min \left\{ T_{\mathtt{C}}(\phi^2), T_{\mathtt{C}}((\mathtt{L}_{\mathbf{x}'} - \mathtt{L}_{\mathbf{x}'})^2) \right\}} \\
&> \min \left\{ \frac{T_{\mathtt{C}}(\phi^2)}{T_{\mathtt{C}}((2\delta)^2)}, \frac{T_{\mathtt{C}}((1-2\delta)^2)}{T_{\mathtt{C}}(\phi^2)} \right\} = \sqrt{\frac{T_{\mathtt{C}}((1-2\delta)^2)}{T_{\mathtt{C}}((2\delta)^2)}} \\
&= \sqrt{\frac{\frac{1}{2} + (\frac{1-2\delta}{n})^2}{\frac{1}{2} + (\frac{2\delta}{n})^2}} = \sqrt{\frac{n^2 + 2(1-2\delta)^2}{n^2 + 2(2\delta)^2}}.
\end{aligned}$$

Lemma 7. *Consider* THEOREM *5, which gives the parameters (d, d', m, t) for the* Comp *function for a given desired accuracy of $2^{-\alpha}$. If α is chosen such that $\alpha > -\log(\eta)$ then* Comp *(and by extension,* LowComp*) has η-error. That is,*

$$|\mathtt{LowComp}(\mathtt{L}_{\mathbf{x}'}, \mathtt{L}_{\mathbf{y}'}, \phi; d, d', m, t) - lowcomp(l_{\mathbf{x}}, l_{\mathbf{y}})| < \eta$$

Proof. If $\alpha > -\log(\eta)$, then $2^{-\alpha} < \eta$. The result follows from THEOREM 4.

(**Proof of Theorem 6**). For

$$\phi = n\sqrt{\sqrt{\left(\frac{1}{2} + (\frac{2\delta}{n})^2\right)\left(\frac{1}{2} + (\frac{1-2\delta}{n})^2\right)} - \frac{1}{2}},$$

one may calculate

$$T_{\mathtt{C}}(\phi^2) = \sqrt{\left(\frac{1}{2} + (\frac{2\delta}{n})^2\right)\left(\frac{1}{2} + (\frac{1-2\delta}{n})^2\right)}.$$

By COROLLARY 1, c is bounded below by $\sqrt{\frac{n^2+2(1-2\delta)^2}{n^2+2(2\delta)^2}}$. But this is equivalent to saying that

$$\begin{aligned} & -\log(\log(c)) \leq -\log\left(\log\left(\sqrt{\frac{n^2+2(1-2\delta)^2}{n^2+2(2\delta)^2}}\right)\right) \\ \implies & \log(\alpha+2) - \log(\log(c)) \\ & \leq \log(\alpha+2) - \log\left(\log\left(\sqrt{\frac{n^2+2(1-2\delta)^2}{n^2+2(2\delta)^2}}\right)\right). \end{aligned}$$

Therefore, if we choose

$$t \geq \frac{1}{\log m}\Bigg[\log(\alpha+1+\log(n)) - \log\left(\log\left(\sqrt{\frac{n^2+2(1-2\delta)^2}{n^2+2(2\delta)^2}}\right)\right)\Bigg],$$

we fulfill the former inequality in Theorem 4.

Furthermore, LEMMA 7 determines what α must be to achieve the desired η-error in `LowComp`. At this point, with a fixed choice of m, Theorem 4 from [16] establishes the remaining parameters d, d', and t.