Locality-Preserving Hashing for Shifts with Connections to Cryptography

Elette Boyle
IDC Herzliya, Israel
NTT Research, Sunnyvale, USA

Itai Dinur
Ben-Gurion University, Be'er Sheva, Israel

Niv Gilboa
Ben-Gurion University, Be'er Sheva, Israel

Yuval Ishai
Technion, Haifa, Israel

Nathan Keller
Bar-Ilan University, Ramat Gan, Israel

Ohad Klein
Bar-Ilan University, Ramat Gan, Israel

Abstract

Can we sense our location in an unfamiliar environment by taking a sublinear-size sample of our surroundings? Can we efficiently encrypt a message that only someone physically close to us can decrypt? To solve this kind of problems, we introduce and study a new type of hash functions for finding shifts in sublinear time. A function $h : \{0,1\}^n \to \mathbb{Z}_n$ is a (d, δ) *locality-preserving hash function for shifts* (LPHS) if: (1) h can be computed by (adaptively) querying d bits of its input, and (2) $\Pr[h(x) \neq h(x \lll 1) + 1] \leq \delta$, where x is random and $\lll 1$ denotes a cyclic shift by one bit to the left. We make the following contributions.

- **Near-optimal LPHS via Distributed Discrete Log.** We establish a general two-way connection between LPHS and algorithms for *distributed discrete logarithm* in the generic group model. Using such an algorithm of Dinur et al. (Crypto 2018), we get LPHS with near-optimal error of $\delta = \tilde{O}(1/d^2)$. This gives an unusual example for the usefulness of group-based cryptography in a post-quantum world. We extend the positive result to non-cyclic and worst-case variants of LPHS.
- **Multidimensional LPHS.** We obtain positive and negative results for a multidimensional extension of LPHS, making progress towards an optimal 2-dimensional LPHS.
- **Applications.** We demonstrate the usefulness of LPHS by presenting cryptographic and algorithmic applications. In particular, we apply multidimensional LPHS to obtain an efficient "packed" implementation of *homomorphic secret sharing* and a sublinear-time implementation of *location-sensitive encryption* whose decryption requires a significantly overlapping view.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Theory of computation → Sketching and sampling; Theory of computation → Nearest neighbor algorithms

Keywords and phrases Sublinear algorithms, metric embeddings, shift finding, discrete logarithm, homomorphic secret sharing

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.27

Related Version *Full Version*: `https://eprint.iacr.org/2022/028.pdf`

Funding *Elette Boyle*: AFOSR Award FA9550-21-1-0046, ERC Project HSS (852952), and a Google Research Scholar Award.
Itai Dinur: ISF grant 1903/20 and ERC starting grant 757731 (LightCrypt).
Niv Gilboa: ISF grant 2951/20, ERC grant 876110, and a grant by the BGU Cyber Center.

Yuval Ishai: ERC Project NTSC (742754), ISF grant 2774/20, and BSF grant 2018393.
Nathan Keller: ERC starting grant 757731 (LightCrypt) and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.
Ohad Klein: Supported by the Clore Scholarship Programme.

Acknowledgements We thank Piotr Indyk, Leo Reyzin, David Woodruff, and anonymous reviewers for helpful pointers and suggestions.

1 Introduction

A *locality-preserving hash function* [18, 16] is a distance-respecting mapping from a complex input space to a simpler output space. Inspired by recent results in cryptography, we study a new kind of locality-preserving hash functions that map strings to integers while respecting the *shift* distance between pairs of input strings with high probability. A distinctive feature of these hash functions is that they can be computed in *sublinear time* with low error probability.

Why shifts? Why sublinear? Our hash functions for shifts can be thought of as *sublinear-time location sensors* that measure a relative position in an unfamiliar environment by taking a sublinear-size sample of the surroundings. This can apply in a variety of settings. For instance, "surroundings" may refer to a local view of an unexplored territory, a long string such as a DNA sequence, an external signal such as a GPS synchronization sequence, a digital document such as big pdf file or a virtual world, or a huge mathematical object such as a cryptographic group. See [2, 19] for applications of shift finding to GPS synchronization, image alignment, motion estimation, and more.[1] We will discuss additional cryptographic and algorithmic applications in Section 1.1.3 below. We are motivated by scenarios in which the local view contains an enormous amount of relevant information that cannot be naively sub-sampled or compressed. This calls for *sublinear-time* solutions.

Simple shift-finding solutions. To motivate the new primitive, consider the following simple shift-finding problem. An n-bit string x is picked uniformly at random, and then cyclically shifted by s bits the left, for some $0 \leq s < n$. Let y be the resulting string. For instance, x, y may be obtained by measuring the same periodic signal at different phases. We write $y = x \ll s$. The shift-finding problem is to find the shift amount s given x and y.

In a centralized setting, where x and y are both given as inputs, it is easy to solve the problem in sublinear time (with small error probability), querying only $\tilde{O}(n^{1/2})$ bits of the input, by matching substrings of x of length $\ell = O(\log n)$ starting at positions $1, 2, \ldots, \sqrt{n}$ with length-ℓ substrings of y whose starting position is a multiple of $\sqrt{n}$. (This is a simplified version of a noise-resilient algorithm from [2].) This algorithm is nearly optimal, since any shift-finding algorithm for an unbounded shift amount s should read $\Omega(\sqrt{n})$ bits of the input [3].

In a distributed setting, a natural goal is to design a *sketching* algorithm that compresses a single input into a short sketch, such that given the sketches of x and y one can recover

[1] While previous related works study a *noise tolerant* variant of shift distance, which arises naturally in the applications they consider, in this work we focus on the simpler noiseless case. Beyond theoretical interest, the simpler notion is motivated by applications. For instance, a local view of a digital document or a mathematical object is noiseless. The noisy case is studied in a follow-up work [4], which obtains nearly tight bounds on the (sublinear) amount of random noise that can be tolerated.

s with high probability. Note that the previous centralized algorithm does imply such a sublinear-size sketch, but only with $\tilde{O}(\sqrt{n})$ output size, which is far from optimal. Instead, one could use the following classical approach [10]: let the sketch of x be an integer $0 \le z_x < n$ that minimizes $x \lll z_x$ (viewed as an n-bit integer), and similarly for y. It can be easily seen that $s = z_x - z_y \mod n$ whenever the minimum is uniquely defined.

The logarithmic sketch size of this simple solution is clearly optimal. Moreover, it realizes something even stronger than sketching: a hash function $h : \{0,1\}^n \to \mathbb{Z}_n$ that respects cyclic shifts in the sense that for a random input x, we have $h(x) = h(x \lll s) + s$ except with small probability. That is, shifting the input by s changes the output by s in the same direction. This is useful for applications. For instance, given t hashes $z_i = h(y_i)$, where $y_i = x \lll s_i$ for $i = 1, \ldots, t$, one can easily compute in time $\tilde{O}(t)$ the relative offsets of all y_i.

The main downside of the above hashing-based solution compared to the centralized algorithm is its linear running time. A natural question is whether one can enjoy the best of both worlds:

Can we combine the sublinear running time of the centralized algorithm with the optimal sketch size and locality-sensing features of the hashing-based solution?

1.1 Our Contribution

We initiate a study of hashing-based solutions to the shift-finding problem. We capture such solutions via the following notion of locality-preserving hash function for shifts.

▶ **Definition 1.** *A function $h : \{0,1\}^n \to \mathbb{Z}_n$ is a (d, δ)* locality-preserving hash function for shifts *(LPHS) if: (1) h can be computed by (adaptively) querying d bits of its input, and (2)* $\Pr[h(x) \neq h(x \lll 1) + 1] \le \delta$*, where x is random and $\lll 1$ denotes a cyclic shift by one bit to the left.*

Note that, by a union bound, an LPHS as above satisfies $\Pr[h(x) \neq h(x \lll s) + s] \le s \cdot \delta$ for any shift amount $0 \le s < n$. Thus, an LPHS has a better accuracy guarantee for smaller shifts. Intuitively, an LPHS can be thought of as a sublinear-time computable location identifier that suffices (with high probability) for determining the exact relative location with respect to adjacent identifiers.

Other LPHS flavors. The above notion of LPHS addresses the basic shift-finding problem as discussed above, but is limited in several important ways: it only considers cyclic shifts and 1-dimensional inputs, and it only guarantees average-case correctness for uniformly random inputs. To address these limitations, we additionally consider other flavors of the basic LPHS notion defined above that are more suitable for applications. These include a *non-cyclic* variant, where instead of $x \lll 1$ we remove the leftmost bit of x and add a random bit on the right; a *k-dimensional* variant, where the input is a k-dimensional matrix and the output is in $\mathbb{Z}_n^k$; and a *worst-case* variant where the quantification is over an arbitrary x that is "far from periodic" and the probability is over the choice of h. (The latter variant better corresponds to the typical notion of a randomized hash function.) The applications we present crucially depend on these extensions.

1.1.1 Near-Optimal LPHS via Distributed Discrete Log

We establish a general two-way connection between LPHS and algorithms for the *distributed discrete logarithm* (DDL) problem [5]. Before explaining this connection, we start with relevant background.

The traditional discrete logarithm (DL) problem is parameterized by a cyclic group $\mathbb{G}$ of order n with a generator g, where n is typically a large prime. The challenge is to recover a random $u \in \mathbb{Z}_n$ from g^u. Many cryptographic applications rely on the conjectured intractability of the DL problem in special types of groups, including subgroups of $\mathbb{Z}_p^*$ and certain families of elliptic curves.

The DDL problem is a distributed variant of the DL problem that was recently introduced in the context of group-based homomorphic secret sharing [5]. In DDL there are two parties, where the first party's input is g^u for a random u, and the second party's input is g^{u+s} where $s \in \{0, 1\}$ (more generally, s can be a small integer). The goal is for each party to locally output an integer, such that the difference between the two outputs is s. One can assume without loss of generality that the two parties run the same algorithm.

Note that a DL algorithm can be used to perfectly solve the DDL problem. However, this is computationally infeasible in a cryptographically hard group, where n is enormous. Instead, a DDL algorithm uses a bounded running time (typically polylogarithmic in the group order n) to obtain the correct difference except with error probability δ. For instance, the initial solution proposed in [5] uses a pseudorandom function to mark each group element as "distinguished" with probability δ, and makes each party, on input v, output the smallest $z \geq 0$ such that $v \cdot g^z$ is distinguished. The (expected) running time of this algorithm is roughly $1/\delta$, and the error probability is δ (corresponding to the case where $s = 1$ and g^u is distinguished).

The DDL problem can be related to the LPHS problem (over a non-binary alphabet) by associating each party's DDL input v with an LPHS input consisting of the sequence of group elements $x = (v, gv, g^2v, \ldots, g^{n-1}v)$. Indeed, multiplication of the DDL input by g corresponds to a cyclic shift of x by one symbol to the left. We formalize this intuition by proving a general two-way relation between LPHS and DDL algorithms in the *generic group model* [24], where group elements are assigned random labels and the algorithm is only given oracle access to the group operation.[2]

The applications we derive from the above connection give an unusual example for the usefulness of results on group-based cryptography in a post-quantum world. Indeed, all traditional applications of group-based cryptography are subject to quantum polynomial-time attacks using Shor's algorithm, and are thus useless in a post-quantum world. If scalable quantum computers become a reality, cryptosystems that are "quantum broken" will become obsolete. In contrast, sublinear-time classical algorithms will still be meaningful even in a post-quantum world.

LPHS constructions. The simple DDL algorithm from [5] corresponds to a (d, δ)-LPHS where $\delta = \tilde{O}(1/d)$. Another simple LPHS construction with similar parameters, implicit in a DDL algorithm from [6], makes a simple use of MinHash [8]: let $h(x)$ output the index i, $1 \leq i \leq d$, that minimizes the value of a MinHash applied to a polylogarithmic-length substring of x starting from x_i.

It is tempting to conjecture that the above simple LPHS constructions are near-optimal, in the sense that $\delta = o(1/d)$ is impossible. It turns out, however, that a quadratic improvement can be obtained from a recent optimal DDL algorithm due to Dinur et al. [11]. Their *Iterated Random Walk (IRW)* algorithm, whose self-contained description appears in the full version,

[2] Specifically, in Section 3 we prove that any LPHS gives a DDL algorithm (in the generic group model) with similar parameters, while any DDL algorithm gives an LPHS with a negligible cost in error probability assuming $d = O(n^{1/4})$ and n is prime.

is based on a carefully chosen sequence of random walks in the group. It can be viewed as a non-trivial extension of Pollard's classical "kangaroo" DL algorithm [22], which runs in time $\tilde{O}(\sqrt{n})$ and has low space complexity. Applying the LPHS vs. DDL connection to the positive and negative results on DDL from [11], we get the following theorem.

▶ **Theorem 2** (Near-optimal LPHS). *There exist (d, δ)-LPHS with: (1) $\delta = \tilde{O}(d^{-2})$ for $d \leq \sqrt{n}$, and (2) $\delta = n^{-\omega(1)}$ for $d = \tilde{O}(\sqrt{n})$. Furthermore, both "$\delta = \tilde{O}(d^{-2})$" in (1) and "$d = \tilde{O}(n^{1/2})$" in (2) are optimal up to polylogarithmic factors.*

Interestingly, any sublinear-time LPHS must inherently make adaptive queries to its input. Adaptive queries are unusual in the context of sublinear metric algorithms, but were previously used in sublinear algorithms for approximating edit distance [23, 9, 15]. A random walk technique was recently used in [17] to obtain a sublinear-time embedding of edit distance to Hamming distance.

Additional variants. We prove similar bounds for the *worst-case* and *non-cyclic variants* of LPHS, which are motivated by the applications we discuss in Section 1.1.3. The result for the worst-case variant is obtained via a general reduction, and inevitably excludes a small set of inputs that are close to being periodic. The result for the non-cyclic case does not follow generically from the cyclic case (except when $d < n^{1/3}$), and requires a special analysis of the IRW algorithm [11]. Also, in this case only (1) holds, since the error probability of a non-cyclic LPHS must satisfy $\delta = \Omega(1/n)$ regardless of d. In fact, whereas $d = \sqrt{n}$ is the hardest case for Theorem 2 in the non-cyclic case (in that (1) for $d = \sqrt{n}$ easily implies (1) for smaller d), it is the easiest for the cyclic case (in that it is implied by the simpler algorithm of Pollard [22]).

In the context of *sketching* for shifts, the above results imply solutions that simultaneously achieve near-optimal sketch size of at most polylog(n), near-optimal running time of $\tilde{O}(\sqrt{n})$, and negligible error probability, for both cyclic and non-cyclic shifts, and for arbitrary "far-from-periodic" inputs.

1.1.2 Multidimensional LPHS

Viewing LPHS as a location identifier, it is natural to consider a generalization to two dimensions and beyond. Indeed, a 2-dimensional (non-cyclic) LPHS can be useful for aligning or sequencing local views of a big 2-dimensional (digital or physical) object. A k-dimensional LPHS maps a k-dimensional matrix (with entries indexed by $\mathbb{Z}_n^k$) into a vector in $\mathbb{Z}_n^k$ so that (cyclically) shifting the input matrix by 1 in axis i changes the output vector by the unit vector e_i, except with δ error probability. As before, this guarantees recovering an arbitrary shift vector with error probability that scales with the ℓ_1 norm of the shift.

Upper bounds. In the 2-dimensional case, the algorithm can be viewed as allowing two non-communicating parties, who are given points (x, y) and $(x+\alpha, y+\beta)$ in the same random for unknown $\alpha, \beta \in \{0, 1\}$, to maximize the probability of synchronizing at the same point, where only d queries are allowed. A straightforward approach is to use a MinHash algorithm in which the parties take the minimal hash value computed on values of a $d^{1/2} \times d^{1/2}$ matrix of elements beginning at the location of each party, resulting in a (d, δ)-LPHS with $\delta = \tilde{O}(d^{-1/2})$. A better error bound of $\delta = \tilde{O}(d^{-2/3})$ can be obtained by combining the application of MinHash on one axis with the application of the aforementioned optimal IRW algorithm on the other axis.

We present three improved algorithms in Section 4. The simplest of those, with a bound of $\delta = \tilde{O}(d^{-4/5})$, is obtained by applying IRW on both axes. A natural idea is to first synchronize on the column; then, synchronizing on the row is easy, using the 1-dimensional IRW algorithm. To synchronize on the column, the parties perform the 1-dimensional IRW algorithm with d/d' "horizontal" steps, where the information used to determine each step is distilled from the column in which the current point resides by using an IRW algorithm with d' "vertical" steps. The analysis in Section 4 (Lemma 19) shows that the bound $\delta = \tilde{O}(d^{-4/5})$ is obtained for the parameter $d' = d^{3/5}$.

Our main upper bound is obtained by a more complex algorithm, which – perhaps, surprisingly – does not rely on the optimal 1-dimensional IRW algorithm at all. We prove:

▶ **Theorem 3.** *For $n = \tilde{\Omega}(d)$, there is a 2-dimensional (d, δ)-LPHS with $\delta = \tilde{O}(d^{-7/8})$. There is also a non-cyclic 2-dimensional LPHS with the same parameters.*

The algorithm we use to prove Theorem 3 consists of three stages. After each stage the parties either converge to the same location, in which case they stay synchronized to the end of the algorithm, or the two walks are within a bounded distance from each other. Stage 1 begins with a distance of at most 1 on each axis, and is a straightforward application of the 2-dimensional MinHash-based algorithm. Stage 2 begins with a distance of at most $\sqrt{d}$ on each axis and uses an asymmetric deterministic walk that consists of $\sqrt{d}$ horizontal steps of size $\sim d^{1/4}$, where each step is pseudo-randomly determined by information distilled from a vertical walk of length $\sqrt{d}$ and step sizes $\sim d^{1/4}$. Stage 3 begins with a distance of at most $d^{3/4}$ on each axis and uses a different deterministic walk. This time, the horizontal steps are of size 1, while the vertical steps are of size about $d^{3/8}$, and unlike all other steps, can be negative. The analysis of the algorithm relies on martingale techniques.

Finally, we present another 2-dimensional LPHS algorithm, which seems harder to analyze, but for which we conjecture that the error rate is at most $\tilde{O}(d^{-1})$. This bound is essentially the best one can hope for given the lower bound discussed below. The idea behind this algorithm is to not treat the axes separately but rather to perform a series of deterministic walks over $\mathbb{Z}^2$, with step sizes of about $d^{1/4}, d^{3/8}, d^{7/16}, \ldots, \sqrt{d}/2$. Our experiments suggest that the error rate of this algorithm is indeed $\tilde{O}(d^{-1})$. However, the analysis (and especially deterministic resolution of cycles in the random walk) is quite involved, and settling our conjecture is left open for future work. Nevertheless, the heuristic algorithm can be used in cryptographic applications (such as packed homomorphic secret sharing which is described next) without compromising their security. Moreover, the worst-case scenario in which the error is larger than predicted by our experiments can be easily detected by applications.

Lower bound. We complement our positive results by proving the following lower bound, extending in a nontrivial way the lower-bound for 1-dimensional LPHS obtained from [11] via the DDL connection.

▶ **Theorem 4.** *For $n = \Omega(d^{2/k})$, any k-dimensional (d, δ)-LPHS satisfies $\delta = \Omega(d^{-2/k})$.*

The intuition is related to the birthday bound. A k-dimensional box with edge length $d^{2/k}$ contains d^2 points. If the k-dimensional shift is uniform within this box, we expect the d queries of the two parties not to intersect with constant probability, implying that $\delta = \Omega(1)$. Given this, the proof for smaller shifts follows by a union bound. While the intuition is simple, is it not clear how to directly apply the birthday bound, and the formal proof is based on an argument involving Minkowski sums and differences of sets in $\mathbb{R}^k$.

1.1.3 Applications

We present several cryptographic and algorithmic applications that motivate different variants of LPHS, exploiting both the functionality and the sublinearity feature. All of the applications can benefit from our 2-dimensional LPHS constructions, and most require the non-cyclic, worst-case variant. See full version for a taxonomy of the LPHS variants required by different applications.

Packed homomorphic secret sharing. We demonstrate a cryptographic application of k-*dimensional* LPHS in trading computation for communication in group-based homomorphic secret sharing (HSS). In a nutshell, we use LPHS to further improve the succinctness of the most succinct approach for simple "homomorphic" computations on encrypted data, by packing 2 or more plaintexts into a single ciphertext. Compared to competing approaches (see, e.g., [1, 20] for recent examples), group-based packed HSS can provide much better succinctness and client efficiency. The key technical idea is to use a non-cyclic k-dimensional LPHS for implementing a k-dimensional variant of DDL, where k independent group generators are used for encoding k small integers by a single group element, and where multiplication by each generator is viewed as a (non-cyclic) shift in the corresponding direction. This k-dimensional generalization of DDL can be potentially useful for other recent applications of DDL that are unrelated to HSS [13, 14, 7]. See full version for a detailed discussion of this cryptographic application of LPHS along with the relevant background.

Location-sensitive encryption. We apply LPHS to obtain a sublinear-time solution for *location-sensitive encryption* (LSE), allowing one to generate a public ciphertext that can only be decrypted by someone in their (physical or virtual) neighborhood. Here proximity is defined as having significantly overlapping views, and security should be guaranteed as long as a non-overlapping view is sufficiently unpredictable. The above goal can be reduced to realizing a sublinear-time computable fuzzy extractor [12][3] for shift distance. Obtaining such fuzzy extractors from LPHS constructions requires an understanding of their behavior on entropic sources. It turns out that even for high-entropy sources, an LPHS provides no unpredictability guarantees. We get around this problem by defining a hash function that combines the output of an LPHS with a local function of the source. Using this approach, we obtain a sublinear-time LSE whose security holds for a broad class of mildly unpredictable sources. See full version for the LSE application of LPHS.

Algorithmic applications. As discussed above, algorithmic applications of LPHS follow from the vast literature on sketching, locality-preserving and locality-sensitive hashing, and metric embeddings. Indeed, our different LPHS flavors can be roughly viewed as probabilistic isometric embeddings of certain shift metrics into a Euclidean space. Thus, for example, an LSH for the same shift metric can potentially follow by concatenating the LPHS with an LSH from the literature. However, some care should be taken in applying this high-level approach. One issue is the average-case nature of LPHS, which makes the failure probability input-dependent. We get around this via a worst-case to average-case reduction that restricts the input space to "non-pathological" inputs that are far from periodic. Another issue is that LPHS provides no explicit guarantees for inputs that are too far apart. We get around this

[3] There are two differences from the standard notion of fuzzy extractors: the "distance" is not a strict metric, and the notion of unpredictability needs to ensure that the source is far from periodic with high probability.

by using the fact that an LPHS must have a well-spread output distribution on a random input. As representative examples, we demonstrate how LPHS can be applied in the contexts of communication complexity and LSH-based near-neighbor data structures for shifts. The algorithmic applications of LPHS are discussed in the full version.

Open questions. Our work leaves several open questions. The main question, on which we make partial progress, is obtaining optimal parameters for k-dimensional LPHS. Other questions concern the optimality of the LPHS-based approach to sketching. A negative result from [11], which can be used to rule out sublinear-time LPHS with non-adaptive queries, in fact holds even for sketching. Do LPHS-based sketches also provide an optimal tradeoff between sketch size and error probability?

Organization. In Section 2 we introduce necessary preliminaries and notation, including the definition of Locality-Preserving Hash functions for Shifts (LPHS). Due to space limitations, this version of the paper spotlights the general two-way connection between LPHS and algorithms for distributed discrete logarithm in the generic group model (Section 3) and our results on multidimensional LPHS (Section 4). Additional results and applications are deferred to the full version of this paper.

2 Preliminaries

We denote by $\mathbb{Z}_n$ the additive group of integers modulo n. We will typically consider strings of length n over an alphabet $\Sigma_b = \{0,1\}^b$, indexing string entries by $i \in \mathbb{Z}_n$. We will use the notation $x^{(b)}$ when we want to make the alphabet size explicit. When the alphabet is binary or when b is clear from the context, we will typically omit the superscript and use the notation x. For $x^{(b)} \in \Sigma_b^n$ we denote by $x^{(b)}[i]$ the i'th symbol of $x^{(b)}$, for $i \in \mathbb{Z}_n$.

We denote by $x^{(b)} \lll r$ the cyclic rotation of $x^{(b)}$ by r symbols to the left, namely the string $y^{(b)}$ defined by $y^{(b)}[i] = x^{(b)}[i+r]$ with addition modulo n. We will also consider a *non-cyclic shift*, denoted by $x^{(b)} \lll\!\!\!< r$, where the r leftmost symbols of x are chopped and r *random* symbols are added on the right. Note that unlike the cyclic shift operator, which is deterministic, the non-cyclic version is randomized. We use $\Delta(x,y)$ to denote the Hamming distance between x and y, namely the number of symbols i in which $x[i]$ and $y[i]$ differ.

We use the notation $x^{(2,b)}$ to denote a 2-dimensional string (i.e., matrix) over alphabet Σ_b and denote by $x^{(2,b)}[i,j]$ its (i,j) entry. We denote by $x^{(2,b)} \lll (r_1,r_2)$ the cyclic rotation of $y^{(2,b)}$ by r_1 symbols to the left on the first axis and r_2 symbols to the left on the second axis. That is, $y = x^{(2,b)} \lll (r_1,r_2)$ is defined by $y[i_1,i_2] = x^{(2,b)}[i_1+r_1, i_2+r_2]$, where addition is modulo n. We will also consider the natural k-dimensional generalization $x^{(k,b)} \lll (r_1,r_2,\dots,r_k)$ and its non-cyclic variant $x^{(k,b)} \lll\!\!\!< (r_1,r_2,\dots,r_k)$.

2.1 Locality-Preserving Hash Functions for Shifts

We now define our main notion of LPHS and some of its useful variants.

▶ **Definition 5** (LPHS: main variants). *Let $h\colon \Sigma_b^n \to \mathbb{Z}_n$ be a function. We say that h is a (cyclic) (d,δ)-*LPHS *if h can be computed by making d* adaptive *queries (of the form $x[i]$) to an input $x \in \Sigma_b^n$ and moreover $\Pr_{x \in_R \Sigma_b^n}[h(x) \neq h(x \lll 1) + 1] \leq \delta$.*

We will consider the following modifiers (that can be combined in a natural way):

- **Non-cyclic** *LPHS: replace $\mathbb{Z}_n$ by $\mathbb{Z}$ and $x \lll 1$ by $x \lll\!\!\!< 1$;*

- k-**dimensional** *LPHS: let x be a random k-dimensional string $x \in \Sigma_b^{\mathbb{Z}_n^k}$ and $h\colon \Sigma_b^{\mathbb{Z}_n^k} \to \mathbb{Z}_n^k$. We require that $\Pr_x[h(x) \neq h(x \ll e_i) + e_i] \leq \delta$ for every* unit vector $e_i \in \mathbb{Z}_n^k$.

We will sometimes make more parameters explicit in the notation. For instance, an (n, b, d, δ)-LPHS is a (d, δ)-LPHS $h\colon \Sigma_b^n \to \mathbb{Z}_n$.

▶ Remark 6 (On computational complexity). A (d, δ)-LPHS $h\colon \Sigma_b^n \to \mathbb{Z}_n$ can be viewed as a *depth-d decision tree* over n input variables taking values from the alphabet Σ_b. In all of our positive results, h is *semi-explicit* in the sense that it can be realized by a *randomized* polynomial-time algorithm having oracle access to the input x. (In fact, our algorithms can be implemented in probabilistic $\tilde{O}(d)$ time.) Here the same randomness for h is used in the two invocations $h(x)$ and $h(x \ll 1)$. Alternatively, our positive results imply a deterministic h in a non-uniform setting. Our negative results apply to the existence of h with the given parameters, irrespective of the computational complexity of generating it.

▶ Remark 7 (Worst-case vs. average-case LPHS). Our default notions of LPHS assume a uniformly random input x. While this suffices for some applications, a worst-case notion of LPHS is more desirable for most applications. Since shift detection is impossible for highly periodic inputs (such as the all-0 string), or even for approximately periodic in the context of sublinear-time algorithms, the notion of worst-case LPHS is restricted to a set of "typical" inputs that are far from being periodic. Our notion of "typical" is very broad and arguably captures essentially all naturally occurring inputs in our motivating applications. In the full version we present a simple reduction of this worst-case flavor of LPHS to our default notion of LPHS for random inputs. This applies both to the cyclic and non-cyclic variants. The reduction only incurs a polylogarithmic loss in the parameters. Note that, unlike our main notion of LPHS, here it is inherent that the function h be randomized. A useful related byproduct of the worst-case variant is that the failure events of two independently chosen h_1 and h_2 are independent. This is useful for algorithmic applications of LPHS.

For some applications, we will be interested in the following additional LPHS variants.

▶ **Definition 8** (LPHS: additional variants). *We consider the following additional variants of the main notion of LPHS from Definition 5.*

- **Shift-bounded** *LPHS with shift bound R: requires that for every $1 \leq r \leq R$, we have*

$$\Pr_{x \in_R \Sigma_b^n}[h(x) \neq h(x \ll r) + r] \leq \delta,$$

and similarly for the non-cyclic case.

- **Las Vegas** *LPHS: allow h to output $\perp$ with probability $\leq \delta$, and require that h never fail in the event that neither of its two invocations outputs $\perp$.*

A generic way of obtaining a Las Vegas LPHS h' from an LPHS h is to invoke h on both x and $x' = x \ll 1$ and output $\perp$ if $h(x) \neq h(x') + 1$. However, an extension of this to a *shift-bounded* LPHS is inefficient, since it requires invoking h on $x \ll r$ for every $0 \leq r \leq R$. In the full version we show that an optimal shift-bounded LPHS admits a Las Vegas variant with better parameters.

3 LPHS and Distributed Discrete Log

We begin by presenting a general two-way connection between LPHS and algorithms for *distributed discrete logarithm* in the generic group model. We discuss and define the latter notion in Section 3.1, and we provide the correspondence with LPHS in Section 3.2.

3.1 Generic Group Model and Distributed Discrete Log

Our notion of LPHS is closely related to variants of the discrete logarithm problem: given a group generator $g \in \mathrm{G}$ and a group element g^v, find v. More concretely, we will be interested in algorithms for problems related to discrete logarithm in the so-called *generic group model* (GGM). The GGM assigns random labels to group elements and treats the group operation as an oracle. We formalize this below.

Let n be a positive integer parameter (corresponding to the group order) and $b \geq 3 \log n$ an integer (representation length of group elements). The GGM setting can be described as a game, where at the beginning, a string $x^{(b)} \in \Sigma_b^n$ is chosen uniformly at random.[4] In the discrete log problem for $\mathbb{Z}_n$, a value $v \in \mathbb{Z}_n$ is chosen uniformly at random. A generic algorithm A for the discrete log problem in $\mathbb{Z}_n$ is a probabilistic algorithm that issues d (adaptive) queries of the form $(i,j) \in \mathbb{Z}_n \times \mathbb{Z}_n$. The answer to query (i,j) is $x^{(b)}[\ell_v(i,j)]$, where $\ell : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$ is the affine query evaluation function defined by $\ell_v(i,j) = i \cdot v + j$ (where arithmetic operations are performed modulo n). Using the group notation, the query (i,j) corresponds to group element $(g^v)^i \cdot g^j$.

The algorithm A succeeds to solve the discrete log problem if $A^{\mathrm{G}}(x^{(b)}, v) = v$,[5] and its success probability is taken over the uniform choices of $x^{(b)}$ and v (and possibly additional randomness of its own coin-tosses).

The flavor of GGM we use in this paper is similar to the one of Shoup [24]. Besides differences in notations, there are two additional technical differences which are generally minor. First, in [24], strings are uniformly assigned to elements of $\mathbb{Z}_n$ without replacement, whereas in our model, we assign strings with replacement. However, a collision $x[i] = x[j]$ for some pair (i,j) is possible with probability $\leq 1/n$, which is negligible in our context. Second, in [24] queries of A are limited to linear combinations with coefficients of ± 1 to previously queried elements (where the initial queried elements consist of g and g^v). We note that any query (i,j) can be issued in Shoup's original GGM after at most $O(\log n)$ queries (using the double-and-add algorithm). Therefore, although our model is slightly stronger, any algorithm in our model can be simulated by an algorithm in the model of [24] by increasing the query complexity by a multiplicative factor of $O(\log n)$.

The following success probability upper bound was proved in [24].

▶ **Theorem 9** ([24], Theorem 1 (adapted)). *For a generic discrete log algorithm A with d queries and prime n, we have $\Pr_{x^{(b)},v}[A^{\mathrm{G}}(x^{(b)}, v) = v] = O(d^2/n)$.*

Although our model is slightly different than the one of [24], this result holds in our model as well (by a straightforward adaptation of the proof of [24]). The assumption that n is prime ensures that $\mathbb{Z}_n$ does not contain any non-trivial subgroup. It is necessary in general, since for composite n, the Pohlig-Hellman algorithm [21] breaks the discrete log problem into smaller problems in subgroups of $\mathbb{Z}_n$, beating the bound of Theorem 9.

We now define a restricted class of GGM algorithms that better correspond to LPHS.

▶ **Definition 10.** *A GGM algorithm A is called query-restricted if it only issues queries of the form $(i,j) \in \mathbb{Z}_n \times \mathbb{Z}_n$ with $i = 1$.*

[4] We require $b \geq 3 \log n$ to ensure that for each $i \neq j$, $x[i] \neq x[j]$ (except with $\leq 1/n$ probability).

[5] We use the notation $A^{\mathrm{G}}(x^{(b)}, v)$ to indicate that A is a generic algorithm with no direct access to the parameters $x^{(b)}, v$.

Thus, A is restricted to query group elements with a known shift j from v, analogously to the way an LPHS algorithm queries elements at a known offset. Query-restricted algorithms cannot exploit the subgroup structure of composite groups, and thus Theorem 9 holds for them regardless of whether n is prime. For a similar reason, the factorization of n will not play any role in our results on LPHS.

LPHS is closely related to query-restricted GGM algorithms for a variant of discrete log called *distributed discrete log* (DDL) [5, 11] that we describe next. The syntax is identical to that of discrete log. However, the goal here is different: rather than output v when the (implicit) input is v, the goal here is to maintain the *difference* between the outputs on v and $v+1$, except with error probability δ. More formally:

▶ **Definition 11.** *A GGM algorithm A is an (n,b,d,δ)-DDL algorithm if it makes d (potentially adaptive) queries to $x^{(b)}$ and $\Pr_{x^{(b)},v}[A^{\mathrm{G}}(x^{(b)},v) - A^{\mathrm{G}}(x^{(b)},v+1) \neq 1] \leq \delta$.*

▶ Remark 12. The original definition of DDL in [5] involves two parties A and B that may potentially run two different algorithms. The parties are placed within an unknown distance $r \in \{-1,0,1\}$ from each other and their goal is to minimize the error probability defined as $\Pr_{x^{(b)},v}[A^{\mathrm{G}}(x^{(b)},v) - B^{\mathrm{G}}(x^{(b)},v+r) \neq r]$. However, it was shown in [11, Lemma 9], that if both parties use A's algorithm, then the multiplicative loss in error probability is bounded by a constant. Hence, the above restricted definition of DDL is essentially equivalent to the original one of [5].

While LPHS is only directly related to query-restricted GGM algorithms for DDL, Lemma 16 asserts that when n is prime and d is sufficiently small compared to n, any unrestricted GGM algorithm for DDL can be converted to a query-restricted one at a negligible cost in error probability. This gives rise to Corollary 17 that establishes a reduction which converts any unrestricted GGM algorithm for DDL to an LPHS with a negligible cost in error probability.

3.2 Reductions Between LPHS Variants and DDL

In this section we show that a query-restricted DDL algorithm in the GGM is equivalent to our basic notion of (cyclic) LPHS, and then describe consequences of this equivalence. This is formally captured by the following two-way relation.

▶ **Lemma 13.** *There exist reductions that convert an (n,b,d,δ)-LPHS to a query-restricted (n,b,d,δ)-DDLA and vice versa.*

Proof. Given access to an (n,b,d,δ)-LPHS denoted by h, we construct a query-restricted (n,b,d,δ)-DDLA, denoted by A as follows. We run h and translate query j into query $(1,j)$ for $A^{\mathrm{G}}(x^{(b)},v)$. We then feed the answer $x^{(b)}[v+j]$ to h. Finally, we output the same value as h. Since $x^{(b)}[v+j] = (x^{(b)} \ll v)[j]$, we have $A^{\mathrm{G}}(x^{(b)},v) = h(x^{(b)} \ll v)$, where $x^{(b)} \ll v$ is a uniform string. Therefore,

$$\Pr_{x^{(b)},v}[A^{\mathrm{G}}(x^{(b)},v) - A^{\mathrm{G}}(x^{(b)},v+1) \neq 1] =$$
$$\Pr_{x^{(b)},v}[h(x^{(b)} \ll v) - h(x^{(b)} \ll v+1) \neq 1] =$$
$$\Pr_{x^{(b)}}[h(x^{(b)}) - h(x^{(b)} \ll 1) \neq 1] = \delta.$$

In a similar way, a query-restricted (n,b,d,δ)-DDLA can be used to construct a (n,b,d,δ)-LPHS. ◀

The DDL algorithm based on the Iterated Random Walk (IRW) from [11] is query-restricted. Therefore, combining Lemma 13 with the parameters of IRW, we get the positive result below for cyclic LPHS. The result for non-cyclic LPHS follows from the fact that the random walk makes queries within an interval of size bounded by $O(d^2)$, hence if $n = \Omega(d^2)$ is large enough, the LPHS gives both cyclic and non-cyclic LPHS with the same parameters.

▶ **Theorem 14.** *For $n = \Omega(d^2)$ and $b \geq 3 \log n$ there is an (n, b, d, δ)-LPHS such that $\delta = O(1/d^2)$. Moreover, for $n = \Omega(d^2)$ and any $b \geq 1$ there is an (n, b, d, δ)-LPHS with $\delta = \tilde{O}(1/d^2)$. There are also non-cyclic LPHS with the same parameters.*

We can similarly convert the main negative result for DDLA from [11, Theorem 5] to a nearly tight lower bound on the error probability of LPHS.

▶ **Theorem 15.** *For $n = \Omega(d^2)$, any (cyclic or non-cyclic) $(n, 1, d, \delta)$-LPHS satisfies $\delta \geq \Omega(1/d^2)$.*

From GGM to query-restricted GGM. We show in the full version of this work that, when n is prime and d is sufficiently small compared to n, any DDL algorithm can be converted into a query-restricted one with similar parameters.

▶ **Lemma 16.** *For $b \geq 3 \log b$, there exists a reduction that converts any (n, b, d, δ)-DDLA, for prime n, to a query-restricted $(n, b, d, \delta + O(d^2/n))$-DDLA.*

If $d = O(n^{1/4})$, then since $\delta = \Omega(d^{-2})$ by Theorem 15, we have $\delta + O(d^2/n) = \delta + O(d^{-2}) = O(\delta)$. Hence the reduction is almost without loss.

Combined with Lemma 13, we obtain the following corollary.

▶ **Corollary 17.** *For $b \geq 3 \log b$, there exists a reduction that converts any (n, b, d, δ)-DDLA, for prime n, to an $(n, b, d, \delta + O(d^2/n))$-LPHS.*

4 Multidimensional LPHS

In this section we study the k–dimensional generalization of LPHS, focusing mainly on the case $k = 2$ (2D-LPHS). First, in Section 4.1, we consider the upper bound side. We start with simple constructions achieving error $\delta = \tilde{O}(d^{-1/2})$ (Section 4.1.1) and $\delta = \tilde{O}(d^{-4/5})$ (Section 4.1.2). The latter makes a black-box use of the 1-dimensional IRW algorithm of [11]. More concretely, for $n = \Omega(d^{6/5})$, any $b \geq 3 \log n$ and any $(r_1, r_2) \in \{0, 1\} \times \{0, 1\}$, we have

$$\Pr_{x \in_R \Sigma_b^{\mathbb{Z}_n \times \mathbb{Z}_n}} [h(x) - h(x \ll (r_1, r_2)) \neq (r_1, r_2)] = O(d^{-4/5}). \tag{1}$$

Since our construction only makes queries in a limited box of dimensions $O(d^{6/5}) \times O(d^{4/5})$ while $n = \Omega(d^{6/5})$, it gives both a cyclic and a non-cyclic LPHS with the same parameters.

This proves a weak version of Theorem 3. In Section 4.1.3 we obtain the improved upper bound of Theorem 3 by presenting a more intricate algorithm that achieves error rate of $\delta = \tilde{O}(d^{-7/8})$. Finally, in Section 4.1.4 we present a heuristic algorithm that we *conjecture* to achieve the near-optimal error probability of $\delta = \tilde{O}(d^{-1})$. This conjecture is supported by experimental evidence.

In Section 4.2 we study limitations of k-dimensional LPHS. We prove Theorem 4, which for $k = 2$ implies that the error probability of a 2D-LPHS must satisfy $\delta = \tilde{\Omega}(d^{-1})$.

4.1 Upper bounds on 2D-LPHS algorithms

We present and analyze 2D-LPHS algorithms achieving error $\delta = \tilde{O}(d^{-1/2}), \tilde{O}(d^{-4/5}), \tilde{O}(d^{-7/8})$, respectively, as well as a heuristic algorithm conjectured to achieve optimal error $\delta = \tilde{O}(d^{-1})$.

4.1.1 A simple 2D-LPHS with error rate $\delta = O(d^{-1/2})$

We begin by describing a very simple 2D-LPHS algorithm called Min-Hash.

Notation. All integer operations in algorithms within this section are assumed to be floored. For example, we write x/y for $\lfloor x/y \rfloor$ and $\sqrt{n}$ for $\lfloor \sqrt{n} \rfloor$.

Algorithm 1 Min-Hash($x \in \Sigma_b^{\mathbb{Z}_n^2}, d \in \mathbb{N}$).

```
1 begin
2     return $\arg\min_{i,j \in [0,\sqrt{d}]} \{x[i,j]\}$
3 end
```

The following lemma captures the performance of Min-Hash.

Lemma 18. *For $n = \Omega(d^{1/2})$ and any $b \geq 3 \log n$ and $(r_1, r_2) \in \{0,1\} \times \{0,1\}$,*

$$\Pr\left[\textit{Min-Hash}(x,d) - \textit{Min-Hash}(x \ll (r_1,r_2), d) \neq (r_1,r_2)\right] = O(d^{-1/2}). \tag{2}$$

Proof. Notice that no matter what the values of $r_1, r_2 \in \{0,1\}$ are, both applications of Min-Hash on x and y query the values $G = \{x[i,j]\}_{i,j \in [1,\sqrt{d}]}$, together with some other $2\sqrt{d} - 1$ values. Hence Min-Hash(x,d) and Min-Hash(y,d) together read at most $4\sqrt{d}$ values outside of G. Under the uniformity assumption of x, the probability the minimum of all the symbols read by the two applications of Min-Hash is not in G, is bounded by $4\sqrt{d}/d = O(1/\sqrt{d})$. Hence, assuming that the minimum $x[i_0,j_0]$ is in G, and that this minimum is unique, we have

$$\text{Min-Hash}(x,d) = (i_0, j_0).$$
$$\text{Min-Hash}(y,d) = \arg\min_{i,j \in [0,\sqrt{d}]} \{x[i+r_1, j+r_2)\} = (i_0 - r_1, j_0 - r_2),$$

in which case, Min-Hash(x,d) − Min-Hash$(y,d) = (r_1, r_2)$.

The only thing left for the proof is showing that with a very high probability, the minimum of Algorithm 1 is uniquely attained. It can be easily verified (e.g., by induction on d) that the probability the minimum is not unique, is upper bounded by $d/2^b$. Under the assumption $b \geq 3/2 \lg(d)$, this probability is dominated by the $O(1/\sqrt{d})$ error in (2). ◀

4.1.2 An IRW-based 2D-LPHS with error rate $\delta = O(d^{-4/5})$

In this subsection we demonstrate how an 1D-LPHS may be used in order to construct a 2D-LPHS with $\delta = O(d^{-4/5})$. Let us recall the functionality of an optimal 1D-LPHS (see Theorem 14).

Optimal 1D-LPHS, rephrased. For $b \geq 3\log n$, there exists an algorithm $\textsc{optimal1D}\colon \Sigma_b^{\mathbb{Z}_n} \times \mathbb{Z} \to \mathbb{Z}_n$ with the following properties. If $n = \Omega(d^2)$, then

$$\Pr_x[\textsc{optimal1D}(x,d) - \textsc{optimal1D}(x \ll 1, d) \neq 1] < O(1/d^2).$$

The 2D-LPHS is described in the Recursive-Hash algorithm (Algorithm 3). The algorithm works in two stages, beginning from an initial input (j,i), first returning a column i_1 and then a row j_1. The column i_1 is located by a walk along the row j which uses the $\textsc{optimal1D}$ algorithm with $d/d' - 1$ queries (for a parameter d'). A query in this algorithm on column i_0 is answered by the $\textsc{rec1D}$ algorithm, which executes $\textsc{optimal1D}$ with d' queries on column i_0. After returning column i_1, the textscRecursive-Hash algorithm runs the $\textsc{optimal1D}$ algorithm along this column, from input row j to return the output row j_1.

Sketching the analysis observe that there are three possible sources of error in the algorithm. First, the values that $\textsc{rec1D}$ returns on shared columns i_0 may not be identical. However, since the parties start at row distance of at most 1, the executions of $\textsc{optimal1D}$ from $x[j,i_0]$ and $x[j+1,i_0]$ agree on the same symbol in column i_0, except with probability $O((d')^{-2})$. Thus, with probability $1 - O((d')^{-2}) \cdot d/d'$, the column values defined by the $\textsc{rec1D}$ algorithm agree for all columns visited by both parties. The second source of error is that although the column values of shared columns are identical, the two parties do not converge to the same column. Since each party queries d/d' columns, the probability for this event is $O((d/d')^{-2})$. The third source of error is that the execution of $\textsc{optimal1D}$ on the agreed column fails on d' queries. The probability for this event is $O(d'^{-2})$. Hence the total error probability is $\delta = O((d')^{-2} \cdot d/d' + (d/d')^{-2})$. Choosing $d' = d^{3/5}$ gives $\delta = O(d^{-4/5})$.

Algorithm 2 $\textsc{rec1D}(z \in \Sigma_b^{\mathbb{Z}_n^2}, d' \in \mathbb{N}, i_0 \in \mathbb{Z}_n)$.

```
1 begin
2     Define $u \in \Sigma_b^{\mathbb{Z}_n}$ by $u[j] \leftarrow z[j, i_0]$
3     $j_0 \leftarrow \textsc{optimal1D}(u, d')$
4     return $z[j_0 + 10d'^2, i_0]$
5 end
```

Algorithm 3 $\textsc{Recursive-Hash}(z \in \Sigma_b^{\mathbb{Z}_n^2}, d \in \mathbb{N})$.

```
1 begin
2     $i_1 \leftarrow \textsc{optimal1D}(i \mapsto \textsc{rec1D}(z, d^{3/5} - 1, i), d^{2/5})$
3     $j_1 \leftarrow \textsc{optimal1D}(j \mapsto z[j, i_1], d^{2/5})$
4     return $(j_1, i_1)$
5 end
```

▶ **Lemma 19.** *For $n = \Omega(d^{6/5})$ and any $b \geq 3\log n$ and $(r_1, r_2) \in \{0,1\} \times \{0,1\}$,*

$$\Pr\left[\textit{Rec2D}(x,d) - \textit{Rec2D}(x \ll (r_1,r_2), d) \neq (r_1,r_2)\right] = O(d^{-4/5}).$$

4.1.3 3-Stage-Hash: a 2D-LPHS algorithm with $\delta = \tilde{O}(d^{-7/8})$

In this subsection we prove Theorem 3, by presenting the algorithm 3-Stage-Hash which achieves error rate of $\delta = \tilde{O}(d^{-7/8})$. 3-Stage-Hash is composed of 3 stages. The first is Min-Hash and we refer to the other two as $\textsc{stage2}$ and $\textsc{stage3}$.

Algorithm 4 3-Stage-Hash($z \in \Sigma_b^{\mathbb{Z}_n^2}, d \in \mathbb{N}$).

```
begin
    $(i_0, j_0) \leftarrow$ Min-Hash$(z,\ d/3)$
    $(i_1, j_1) \leftarrow$ STAGE2$(z,\ d/3,\ i_0 + 2\sqrt{d},\ j_0 + 2\sqrt{d})$
    $(i_2, j_2) \leftarrow$ STAGE3$(z,\ d/3,\ i_1 + 2d^{3/4},\ j_1 + 2d^{3/4})$
    return $(i_2, j_2)$
end
```

Algorithm 5 STAGE2($z \in \Sigma_b^{\mathbb{Z}_n^2}, d' \in \mathbb{N}, i_0 \in \mathbb{Z}_n, j_0 \in \mathbb{Z}_n$).

```
begin
    $L \leftarrow \sqrt[4]{d'}$
    Let $\psi_1, \psi_2 \colon \Sigma_b \to \mathbb{Z}_L$ be $\psi_1(t) = t \bmod L$ and $\psi_2(t) = \lfloor t/L \rfloor \bmod L$
    $P, P' \leftarrow \emptyset, \emptyset$
    $i \leftarrow i_0$
    for $s = 1$ to $\sqrt{d'}$ do
        $j \leftarrow j_0$
        for $t = 1$ to $\sqrt{d'}$ do
            $P' \leftarrow P' \cup \{(i, j)\}$
            $j \leftarrow j + 1 + \psi_1(z[i, j])$
        end
        $i \leftarrow i + 1 + \psi_2(\min_{p' \in P'}\{z[p']\})$
        $P, P' \leftarrow P \cup P', \emptyset$
    end
end
return $\arg\min_{p \in P}\{z[p]\}$
```

▶ **Lemma 20.** *Let $2^b \geq d^4$, $n \geq \Omega(d)$, and $r_1, r_2 \in \{0, 1\}$. Then,*

$$\Pr\left[\text{3-Stage-Hash}(x, d) - \text{3-Stage-Hash}(y, d) \neq (r_1, r_2)\right] < O(d^{-7/8}), \tag{3}$$

In order to prove Lemma 20, we need the following facts, which we prove later.

▶ **Lemma 21.** *Let $S_1, S_2, \ldots$ be a sequence of i.i.d. geometric random variables: $S_i \sim \mathrm{Geom}(p)$. If K is the minimal integer with $\sum_{k=1}^{K} S_k \geq r$, then $\mathbb{E}[K] \leq rp + 1$.*

▶ **Lemma 22.** *Let $I_0, I_1, I_2, \ldots$ be a random walk with $I_{k+1} - I_k$ being i.i.d. variables distributed as the difference of two uniform $U(0, m)$ random variables, with $m \in \mathbb{N}$. If T is the minimal time with $I_T = 0$, then $\mathbb{E}[\min(T, r)] \leq O\left((m + I_0/m)\sqrt{r}\right)$.*

Given these facts, we turn to Lemma 20.

Proof of Lemma 20, sketch. Let (i_k^z, j_k^z) be the values computed as (i_k, j_k) at 3-Stage-Hash(z, d), for $k \in \{0, 1, 2\}$ and $z \in \{x, y\}$. We say that (i_k^x, j_k^x) and (i_k^y, j_k^y) are synchronized if $(i_k^x, j_k^x) - (i_k^y, j_k^y) = (r_1, r_2)$. Observe that if (i_k^x, j_k^x) and (i_k^y, j_k^y) are synchronized, then so are (i_{k+1}^x, j_{k+1}^x) and (i_{k+1}^y, j_{k+1}^y). This is because each stage $k+1$ of 3-Stage-Hash(z, d) deterministically depends on values queried from z with offset (i_k^z, j_k^z), so that evaluations keep being aligned. Overall,

$$\begin{aligned}
\delta \doteqdot{} & \Pr_{x,y,r_1,r_2}\left[\text{3-Stage-Hash}(x, d) - \text{3-Stage-Hash}(y, d) \neq (r_1, r_2)\right] \\
& \leq \Pr[(i_0^x, j_0^x) - (i_0^y, j_0^y) \neq (r_1, r_2)] \cdot \\
& \cdot \prod_{k=1}^{2} \Pr\left[(i_k^x, j_k^x) - (i_k^y, j_k^y) \neq (r_1, r_2) \,\middle|\, (i_{k-1}^x, j_{k-1}^x) - (i_{k-1}^y, j_{k-1}^y) \neq (r_1, r_2)\right].
\end{aligned} \tag{4}$$

Algorithm 6 STAGE3($z \in \Sigma_b^{\mathbb{Z}_n^2}, d' \in \mathbb{N}, i_0 \in \mathbb{Z}_n, j_0 \in \mathbb{Z}_n$).

```
begin
    Let $\psi\colon \Sigma_b \to \{-d'^{3/8}, \dots, d'^{3/8}\}$ be $\psi(t) = (t \bmod (2d'^{3/8}+1)) - d'^{3/8}$
    $P \leftarrow \emptyset$
    $(i, j) \leftarrow (i_0, j_0)$
    for $s = 1$ to $d'$ do
        $P \leftarrow P \cup \{(i, j)\}$
        $(i, j) \leftarrow (i + 1, j + \psi(z[i, j]))$
    end
    return $\arg\min_{p \in P}\{z[p]\}$
end
```

Hence, to bound δ it is sufficient to verify the following three claims:
1. $\Pr[(i_0^x, j_0^x) - (i_0^y, j_0^y) \neq (r_1, r_2)] \leq O(1/\sqrt{d})$.
2. $\Pr\left[(i_1^x, j_1^x) - (i_1^y, j_1^y) \neq (r_1, r_2) \,\middle|\, (i_0^x, j_0^x) - (i_0^y, j_0^y) \neq (r_1, r_2)\right] \leq O(1/\sqrt[4]{d})$.
3. $\Pr\left[(i_2^x, j_2^x) - (i_2^y, j_2^y) \neq (r_1, r_2) \,\middle|\, (i_1^x, j_1^x) - (i_1^y, j_1^y) \neq (r_1, r_2)\right] \leq O(1/\sqrt[8]{d})$.

Claim 1. Follows from Lemma 18.

Claim 2. Since MIN-HASH(z, d) scans a $\sqrt{d} \times \sqrt{d}$ area and outputs a point inside it, we are guaranteed that $|i_0^x - i_0^y| \leq \sqrt{d} + 1$ and $|j_0^x - j_0^y| \leq \sqrt{d} + 1$. Because STAGE2 is fed with the output point of MIN-HASH shifted by $2\sqrt{d}$ in each axis, its queries do not overlap these of MIN-HASH, and its performance is independent of the MIN-HASH phase. Moreover, STAGE2 can be modeled as a random walk on the i axis, whose steps are integers uniformly distributed in $[1, \sqrt{d/3}]$, which are determined by some random walk on the j axis. Denote by $I_1^x, \dots, I_{\sqrt{d'}}^x$ and $I_1^y, \dots, I_{\sqrt{d'}}^y$ the sequences of i's observed by STAGE2 applied on x and on y.

In order to compute the probability that the outputs of STAGE2(x, d') and STAGE2(y, d') are not synchronized, it is sufficient (following the proof of Lemma 18) to count the number of queries that the two processes make, which are not shared. These queries can be classified into two categories: queries with non-shared i, and queries with shared i and non-shared j. Our goal is to show each class contains on average $O(d'^{3/4})$ such queries, implying that the probability of the outputs not being synchronized is $O(d'^{3/4}/d')$ (similarly to Lemma 18). We start by reasoning about the first class of queries, and then proceed to the second.

Let U_1 denote the total number of i-steps until i^x and i^y are synchronized (i.e. $U_1 = k + k'$ when k, k' are minimal with $I_k^x - I_{k'}^y = r_1$). Up to this point, the two STAGE2 applications act independently, as their queries do not overlap. Using [11, Lemmas 3,5] with $b \leq \sqrt{d} + 1$ and $L = \sqrt[4]{d'}$ we see $\mathbb{E}[U_1] \leq O(d^{1/4})$. Next, we note that once $I_k^x - I_{k'}^y = r_1$, it is likely that $I_{k+1}^x - I_{k'+1}^y = r_1$. Specifically, we will show

$$\Pr\left[I_{k+1}^x - I_{k'+1}^y \neq r_1 \,\middle|\, I_k^x - I_{k'}^y = r_1\right] \leq O(1/\sqrt{d}).$$

Assuming this, the two walks make U_1 unsynchronized steps, then S_1 synchronized steps with $S_1 \sim \mathrm{Geom}(O(1/\sqrt{d}))$ distributed geometrically. The walks then make another U_2 unsynchronized steps, with [11, Lemma 5] yielding $\mathbb{E}[U_2] \leq O(d^{1/4})$, followed by S_2 synchronized steps with $S_2 \sim \mathrm{Geom}(O(1/\sqrt{d}))$. The process continues this way until one of the walks has completed its $\sqrt{d'}$ steps. Using Lemma 21, the expected number of such phases of synchronization-unsychronization is $\leq O(d'\sqrt{1/d} + 1) = O(1)$. Combining this with the fact that $\mathbb{E}[U_k] = O(d^{1/4})$, we deduce that the expected number of unsynchronized i-steps is $O(d^{1/4})$. Each such step involves $\sqrt{d'}$ j-steps, so the total number of queries with non-shared i is $O(d^{3/4})$.

Regarding the steps with shared i and non-shared j, random-walk arguments similar to the above argument imply that since on each shared i, the two j-walks start with distance $O(\sqrt{d})$, and have steps of size $\Theta(d^{1/4})$, they are expected to meet after $O(d^{1/4})$ queries (follows from [11, Lemmas 3,5]). Since there are $O(\sqrt{d})$ i-steps, the total number of such non-shared queries is $O(d^{3/4})$ as well.

Claim 3. Similarly to the previous claim, the queries made by stages before STAGE3 are confined to a square area of size $(d^{3/4} + \sqrt{d}) \times (d^{3/4} + \sqrt{d})$, and since the input is shifted by $2d^{3/4}$, the queries of STAGE3 do not overlap previous stages. Note that the queries made by STAGE3 are confined to a $2d \times 2d$ square except for a negligibly small error prbobability $(\exp(-\Omega(d^{1/4})))$ due to Hoeffding's inequality, and since $n \geq \Omega(d)$ (recall $x, y \in \Sigma_b^{\mathbb{Z}_n^2}$), the queries of the different stages do not overlap (with high probability) even though the index space of x, y is cyclic.

It is clear that once the two walks of STAGE3 on x and on y are synchronized, they remain synchronized. Let T be the total number of steps until the two walks share a point. There are at most $\min(2T, d')$ steps which are not shared, and the failure probability is $\leq \min(2T, d')/d'$, similarly to the proof of Lemma 18. Thus, it is sufficient to verify $\mathbb{E}[\min(T, d)] \leq d^{7/8}$. Clearly, after $|i_1^x - i_1^y|$ steps, the walks are being synchronized with respect to the i-axis. Let J denote the random variable measuring their distance on the j-axis, once the walks first share this same i. Since each of the advances of j are independent of the other steps,

$$\mathbb{E}[J^2] = |j_1^x - j_1^y|^2 + \sum_{t=0}^{|i_1^x - i_1^y|} \mathbb{E}[S_t^2],$$

where S_t is the jump on the j-axis on the t-step of the runner-up walk. In particular $\mathbb{E}[S_t^2] \leq ((d/3)^{3/8})^2 \leq d^{3/4}$. Since $|i_1^x - i_1^y| \leq 2d^{3/4}$, we overall deduce $\mathbb{E}[J^2] \leq O(d^{3/2})$.

From this point on, the walks of STAGE3(x, d') and STAGE3(y, d') keep being aligned with respect to the i-axis. Once they meet on the j-axis, they will remain synchronized. The distance on the j-axis between the walks can be modeled as a one dimensional random walk, starting at J, and having independent steps whose lengths are a difference of two independent uniform $U(0, 2d'^{3/8})$ variables. Once this difference walk hits 0, the walks keep being synchronized. Lemma 22 then immediately yields

$$\mathbb{E}[\min(T, d')] \leq |i_1^x - i_1^y| + O\left((d'^{3/8} + \mathbb{E}[J]/d'^{3/8})\sqrt{d'}\right).$$

Substituting $\mathbb{E}[J]^2 \leq \mathbb{E}[J^2] = O(d^{3/2})$, we obtain $\mathbb{E}[\min(T, d')] \leq d^{7/8}$, as required. ◀

We now fill in the proofs of the above-stated facts.

Proof of Lemma 21. Since each S_i counts the number of Bernoulli-p variables until success, K distributes as 1+ the number of successful Bernoulli-p variables, out of r. This interpretation immediately gives $\mathbb{E}[K] = p(r-1) + 1$. ◀

Proof of Lemma 22. Let $T_0, T_1, T_2, \ldots$ be the sequence of times $t \geq 0$ with $|I_t| \leq \frac{m+1}{2}$ (in increasing order). Observe that for all i, the event $I_{T_i+1} = 0$ happens with probability $\geq \frac{1}{2(m+1)}$, even when conditioning on $T_0, \ldots, T_i$. Let K be minimal with $I_{T_K+1} = 0$. This observation means that (for all values of $T_0, \ldots, T_k$)

$$\Pr\left[K \geq k \,|\, T_0, \ldots, T_k\right] \leq (1 - 1/(2m+2))^k.$$

When combined with

$$\mathbb{E}[\min(T_K, r)] \leq \mathbb{E}[\min(T_0, r)] + \sum_{k=1}^{\infty} \mathbb{E}\left[\mathbb{1}_{\{K \geq k-1\}} \cdot \min(T_k - T_{k-1}, r)\right],$$

we deduce

$$\mathbb{E}[\min(T_K, r)] \leq \mathbb{E}[\min(T_0, r)] + \sum_{k=1}^{\infty} \left(\frac{2m+1}{2m+2}\right)^{k-1} \mathbb{E}\left[\min(T_k - T_{k-1}, r) \,|\, K \geq k-1\right]. \quad (5)$$

Clearly, upper bounding $\mathbb{E}[\min(T_K, r)]$ is relevant, as if T is the minimal time with $I_T = 0$, then $T \leq T_K + 1$, and in particular, $\min(T, r) \leq \min(T_K, r) + 1$. We claim the following:

1. $\mathbb{E}[\min(T_0, r)] \leq O(1 + I_0\sqrt{r}/m)$.
2. For all i (and all values of $I_0, \ldots, I_{T_i}$), $\mathbb{E}\left[\min(T_{i+1} - T_i, r) \,|\, I_0, \ldots, I_{T_i}\right] \leq O(\sqrt{r})$.

These claims together with (5) and $\min(T, r) \leq \min(T_K, r) + 1$ give

$$\mathbb{E}[\min(T, r)] \leq O(1 + I_0\sqrt{r}/m) + \sum_{k=1}^{\infty} \left(\frac{2m+1}{2m+2}\right)^{k-1} O(\sqrt{r}) \leq O(m\sqrt{r} + I_0^2/m^2),$$

as required. Let us verify the above claims.

Claim 2. This is a specialization of Claim 1 to the walk $I_{T_i+1}, I_{T_i+2}, \ldots$, satisfying $|I_{T_i+1}| \leq \frac{3m+1}{2}$.

Claim 1. Without loss of generality assume $I_0 \geq 0$ (due to symmetry). Write $L = m\sqrt{r}$. Instead of the stopping time $\min(T_0, r)$, consider the stopping time T' which is the minimal (time) $t \geq 0$ with $I_t \leq (m+1)/2$ or $I_t > L$. It is standard to show $\Pr[T' > k]$ decreases exponentially fast with k (albeit with deficient constants), and so all quantities presented in the proof will turn out to be finite (in particular, $\mathbb{E}[T']$).

Since the definition of T_0 is similar to that of T', except that the latter allows to stop also when $I_t > L$, we may upper bound $\mathbb{E}[\min(T_0, r)]$ by $\mathbb{E}[T'] + r\Pr[I_{T'} > L]$, i.e., we compensate by r in all cases when T' is not identical to T_0.

Since $I_{k+1} - I_k$ is a symmetric random variable, and is independent of $I_0, \ldots, I_k$, the sequence $I_0, I_1, \ldots$ constitutes a martingale. In particular, by the *Optional stopping theorem*, $\mathbb{E}[I'_T] = I_0$. This implies

$$I_0 = \mathbb{E}[I_{T'}] \geq -(m+1)/2 \cdot \Pr[I_{T'} \leq (m+1)/2] + L\Pr[I_{T'} > L] \geq L\Pr[I_{T'} > L] - m,$$

and in particular, $\Pr[I_{T'} > L] \leq (m + I_0)/L$. Recall this claim is non-trivial only when $I_0 > (m+1)/2$, and so we may assume $\Pr[I_{T'} > L] \leq O(I_0/L)$. In particular,

$$\mathbb{E}[\min(T_0, r)] \leq \mathbb{E}[T'] + r \cdot O(I_0/L) \leq \mathbb{E}[T'] + O(I_0\sqrt{r}/m).$$

Thus the claim is implied from $\mathbb{E}[T'] \leq O(I_0\sqrt{r}/m)$ which we now prove.

Write $s = \mathbb{E}[(I_{k+1} - I_k)^2] = (m^2 + 2m)/6$, and consider the sequence of random variables $\left(I_k^2 - sk\right)_{k=0}^{\infty}$.

We claim it is a martingale. Indeed:

$$\begin{aligned}\mathbb{E}\left[I_{k+1}^2 - s(k+1) \,\middle|\, I_0, \ldots, I_k\right] &= \mathbb{E}\left[I_k^2 + I_k(I_{k+1} - I_k) + ((I_{k+1} - I_k)^2 - s) - sk \,\middle|\, I_0, \ldots, I_k\right] \\ &= I_k^2 - sk\end{aligned}$$

The first equality is just $(a+b)^2 = a^2 + 2ab + b^2$, and the second uses the fact that $I_{k+1} - I_k$ is a symmetric random variable independent of $I_0, \ldots, I_k$, having variance s. The *Optional stopping theorem* thus implies $I_0^2 = \mathbb{E}[I_{T'}^2 - sT']$, yielding $\mathbb{E}[T'] \le \mathbb{E}[I_{T'}^2]/s$. To bound $\mathbb{E}[I_{T'}^2]$, we recall that $I_{T'}$ is has absolute value $\le (m+1)/2$ with probability ≤ 1 (trivially), and is between L and $L+m$ with probability $\le O(I_0/L)$. Hence $\mathbb{E}[I_{T'}^2] \le m^2 + (L+m)^2 O(I_0/L)$. Since $L \ge m$, we deduce $\mathbb{E}[I_{T'}^2] \le O(s + I_0 L)$. Overall,

$$\mathbb{E}[T'] \le O(1 + I_0 L/s) \le O(I_0^2/m^2 + I_0\sqrt{r}/m).$$

Notice we may assume $I_0 \le m\sqrt{r}$, for otherwise the claim is trivial: $\min(T_0, r) \le r \le I_0\sqrt{r}/m$. Hence $\mathbb{E}[T'] \le O(I_0\sqrt{r}/m)$, as required. ◀

4.1.4 Conjectured optimal algorithm

The 2D-LPHS algorithms presented in the previous sections have the property of not treating both axes symmetrically. For example, Recursive-Hash iterates over several i_0's, and for each of them it makes many queries of the form $x[i_0, j]$ for different j's. Except for not being aesthetic, this asymmetry has other disadvantages. For example, it is not obvious how to generalize these algorithms to higher dimensions. More importantly, these algorithms (that we considered) do not have optimal dependence of δ on d.

We conjecture that the following symmetric algorithm (Random-Walk-Hash) has the optimal performance of $\delta = \widetilde{O}(1/d)$. However, we were not able to rigorously analyze it.

Algorithm 7 RW-STAGE$(z \in \Sigma_b^{\mathbb{Z}_n^2}, d \in \mathbb{N}, L \in \mathbb{N}, i \in \mathbb{Z}_n, j \in \mathbb{Z}_n)$.

```
Let $\psi_1, \psi_2 \colon \Sigma_b \to \{-L, \ldots, L\}$ be independent random functions
begin
    $P \leftarrow$ list()
    for $s \leftarrow 0 \ldots d-1$ do
        $P[s] \leftarrow (i, j)$
        $v \leftarrow z[i, j]$
        $(i, j) \leftarrow (i + \psi_1(v), j + \psi_2(v))$
        if $(i, j) \in P$ then
            Let $t$ be the only index satisfying $P[t] = (i, j)$
            $k \leftarrow \arg\min_{u \in [t,s]}\{z[P[u]]\}$
            $(i, j) \leftarrow P[k]$
            while $(i, j) \in P$ do
                $j \leftarrow j + 1$
            end
        end
    end
    return $\arg\min_{(i',j') \in P}\{z[i', j']\}$
end
```

Algorithm 8 Random-Walk-Hash($z \in \Sigma_b^{\mathbb{Z}_n^2}, d \in \mathbb{N}$).

```
begin
    $I \leftarrow \lg\lg(d)$
    $d' \leftarrow d/I$
    $(i_0, j_0) \leftarrow$ Min-Hash$(z,\ d')$
    $(i_1, j_1) \leftarrow$ rw-stage$(z,\ d',\ d'^{1/4},\ i_0,\ j_0)$
    $(i_2, j_2) \leftarrow$ rw-stage$(z,\ d',\ d'^{3/8},\ i_1,\ j_1)$
    $(i_3, j_3) \leftarrow$ rw-stage$(z,\ d',\ d'^{7/16},\ i_2,\ j_2)$
    $\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$
    $(i_I, j_I) \leftarrow$ rw-stage$(z,\ d',\ \sqrt{d'/2},\ i_{I-1},\ j_{I-1})$
    return $(i_I, j_I)$
end
```

Heuristic performance. Here we heuristically describe why we expect the algorithm Random-Walk-Hash to achieve $\delta = \widetilde{O}(1/n)$.

The main heuristic assumption we make is that each rw-stage(x, d, L, i, j) can be modeled by a random walk on $\mathbb{Z}^2$, starting at (i, j) and having independent steps which are uniformly distributed on each axis as $\sim U(-L, L)$. We further assume that once the two walks of rw-stage(x) and rw-stage(y) are synchronized (collided), they remain synchronized. Moreover, we recall that the output location of a rw-stage(x) is the point visited in this walk having the minimal x-value.

▶ Remark 23. These assumptions are not precise mainly because we need the steps to be both deterministic and independent (with respect to the input's randomness) of the previous steps. In practice we cannot guarantee independence, which makes us run into loops. We try to break these in a canonical way, which complicates the analysis. If the algorithm would make monotone queries along (at least) one axis (as the one-dimensional algorithm), then it would avoid loops and its analysis would be much simpler. Unfortunately, we do not know how to design such an algorithm with similar performance.

Based on the heuristic assumptions above, an analysis of Random-Walk-Hash would follow from the following two claims:

- Let T be (a random variable measuring) the time that two random walks on $\mathbb{Z}^2$, starting at distance D (in L_1 norm), and making steps distributed as $U(-L, L)$, meet. Then $\mathbb{E}[\min(d', T)] \le O(\sqrt{d'}(L + D/L))$.[6] We say that walks A, B "meet" in time t if t is minimal so that $\exists i, j \le t$ with location$_i(A) =$ location$_j(B)$.
- The expected distance (in L_1) of a 2D-walk with d' steps distributed as $U(-L, L)$, between the starting point and the final one, is $O(L \cdot \sqrt{d'})$.

Let us analyze the first few stages of Random-Walk-Hash using these claims (which we do not prove here).

Just after the first stage, which is Min-Hash, the walks of Random-Walk-Hash(x) and Random- Walk-Hash(y) are synchronized except for probability $O(1/\sqrt{d'})$ (Lemma 18), and in case of this failure event, the distance of the two walks has expected value $O(\sqrt{d'})$.

At the second stage, rw-stage$(x, d', d'^{1/4})$, the initial distance between the walks is $D = O(d'^{1/2})$ and $L = d'^{1/4}$. Using the above claims, and a Markov inequality, the random walks would synchronize except for probability $O(D/L + L)\sqrt{d'}/d' = O(d'^{-1/4})$, and in case of failure, the expected distance is $O(d'^{3/4})$.

[6] The parameters are chosen so that the parties meet within an expected number of $O(\sqrt{d'}(L + D/L))$ steps, while they do not meet within d' steps with probability $O((L + D/L)/\sqrt{d'})$.

At the third stage, RW-STAGE$(x, d', d'^{3/8})$, $D = O(d'^{3/4}), L = d'^{3/8}$ and the failure probability becomes $O(d'^{3/8})/\sqrt{d'} = O(d'^{-1/8})$, and the distance upon failure is $= O(d'^{7/8})$.

Continuing this heuristic to later stages we get that the total failure probability, which is the product of failure probabilities of all the stages, is $2^{O(I)}/d' = \widetilde{O}(d^{-1})$.

4.2 Lower bounds on 2D-LPHS algorithms

In this section we prove Theorem 4 for $k = 2$ (i.e., 2D-LPHS). The proof for any other value of $k > 1$ is similar. In particular, we show that any 2D-LPHS algorithm satisfies $\delta \geq \Omega(1/d)$, given $n \geq 2d$.

▶ **Lemma 24** (Bigger shifts). *Let h be a 2D (n, b, d, δ)-LPHS. Let $r'_1, r'_2 \in \mathbb{N}$, and $y' = x' \ll (r'_1, r'_2)$ where $x' \in \Sigma_b^{\mathbb{Z}_n^2}$ is a uniformly random string. Then,*

$$\Pr\left[h(x', d) - h(y', d) \neq (r'_1, r'_2)\right] \leq \max\{r'_1, r'_2\}\delta.$$

Proof. Write $I = \max\{r'_1, r'_2\}$ and set $r'^{(i)}_1 = \min(r'_1, i), r'^{(i)}_2 = \min(r'_2, i)$. Note $\forall i$: $(r'^{(i+1)}_1 - r'^{(i)}_1), (r'^{(i+1)}_2 - r'^{(i)}_2) \in \{0, 1\}$. Define the strings $x'_i \in \Sigma_b^{\mathbb{Z}_n^2}$ by $x'_i = x' \ll (r'^{(i)}_1, r'^{(i)}_2)$ for $i = 0, \ldots, I$. Since each x'_i is a uniformly random function (because x' is), we may use (1) to deduce

$$\delta_i \doteqdot \Pr\left(A(x'_i, d) - A(x'_{i+1}, d) \neq (r'^{(i+1)}_1 - r^{(i)}_1, r'^{(i+1)}_2 - r'^{(i)}_2)\right) \leq \delta.$$

Notice $x' = x'_0$ and $y' = x'_I$. Using a union-bound argument, we conclude

$$\Pr\left[A(x', d) - A(y', d) \neq (r'_1, r'_2)\right] \leq \sum_{i=0}^{I-1} \delta_i \leq I\delta$$

as required. ◀

The following lemma implies Theorem 4 for $k = 2$.

▶ **Lemma 25.** *For $n > 2d$ and any $b > 0$, every 2D (cyclic or non-cyclic) (n, b, d, δ)-LPHS satisfies $\delta \geq 1/(3d)$.*

Proof. Consider the set of queries P made to a uniformly random string $x \in \Sigma_b^{\mathbb{Z}_n^2}$ by $A(x, d)$. That is, P is a random variable whose values are sets of sizes $\leq d$ of (i, j) pairs.

Let $x', y' \in \Sigma_b^{\mathbb{Z}_n^2}$ be two uniformly random strings related by $y' = x' \ll (r'_1, r'_2)$ for uniform independent variables $r'_1, r'_2 \sim U(0, 2d)$. Using Lemma 24,

$$\begin{aligned}\Pr[h(x', d) - h(y', d) \neq (r'_1, r'_2)] &= \mathop{\mathbb{E}}_{r'_1, r'_2}\left[\mathbb{E}\left[h(y', d) - h(x', d) \neq (r'_1, r'_2) \,|\, r'_1, r'_2\right]\right] \\ &\leq \mathop{\mathbb{E}}_{r'_1, r'_2}\left[\max(r'_1, r'_2)\delta\right] \leq 2d\delta. \qquad (6)\end{aligned}$$

Let $P_{x'}, P_{y'}$ be copies of P which are the set of queries issued by $h(x', d), h(y', d)$, respectively. Generally, the random variables $P_{x'}, P_{y'}$ are dependent. However, we are going to see they are only slightly dependent. Indeed, suppose P_1, P_2 are two values of P. We claim that given specific values of r'_1, r'_2 (call these r''_1, r''_2) so that P_1 and (the Minkowski sum) $P_2 + \{(r''_1, r''_2)\}$ are disjoint, we have

$$\Pr\left[P_{x'} = P_1 \wedge P_{y'} = P_2 \,|\, (r'_1, r'_2) = (r''_1, r''_2)\right] = \Pr\left[P = P_1\right] \cdot \Pr\left[P = P_2\right]. \qquad (7)$$

This is because the event $P_{x'} = P_1$ is in the σ-algebra generated by the random variable $x'\big|_{P_1}$ (that is, events depending only on $x'\big|_{P_1}$)[7], which is independent of the σ-algebra generated by $y'\big|_{P_2}$ (or $x'\big|_{P_2+\{(r''_1,r''_2)\}}$), as different entries of x' are independent. Consider the conditional random variable

$$X_{r''_1,r''_2,P_1,P_2} \doteqdot [h(x',d) - h(y',d) \,|\, r'_1 = r''_1, r'_2 = r''_2, P_{x'} = P_1, P_{y'} = P_2],$$

and compute

$$\begin{aligned}
&\Pr[h(x',d) - h(y',d) = (r'_1, r'_2)] \\
&\quad = \mathop{\mathbb{E}}_{r'_1,r'_2,P_{x'},P_{y'}} [\Pr[h(x',d) - h(y',d) = (r'_1,r'_2) \,|\, r'_1, r'_2, P_{x'}, P_{y'}]] \\
&\quad = \mathop{\mathbb{E}}_{r'_1,r'_2} \left[\sum_{P_1,P_2} \Pr\left[X_{r'_1,r'_2,P_1,P_2} = (r'_1,r'_2)\right] \cdot \Pr\left[P_{x'} = P_1, P_{y'} = P_2 \,|\, r'_1, r'_2\right] \right] \\
&\quad \overset{(a)}{\leq} \underbrace{\mathop{\mathbb{E}}_{r'_1,r'_2} \left[\sum_{\substack{P_1,P_2: \\ ((r'_1,r'_2)\in P_1 - P_2)}} \Pr\left[P_{x'} = P_1, P_{y'} = P_2 \,|\, r'_1, r'_2\right] \right]}_{Q_1} + \\
&\quad + \underbrace{\mathop{\mathbb{E}}_{r'_1,r'_2} \left[\sum_{\substack{P_1,P_2: \\ ((r'_1,r'_2)\notin P_1 - P_2)}} \Pr\left[X_{r'_1,r'_2,P_1,P_2} = (r'_1,r'_2)\right] \cdot \Pr[P = P_1]\Pr[P = P_2] \right]}_{Q_2} \\
&\quad \overset{(b)}{\leq} \frac{d^2}{(2d+1)^2} + \frac{1}{(2d+1)^2} = \frac{d^2+1}{(2d+1)^2},
\end{aligned}$$

where $P_1 - P_2$ is a Minkowski difference. Inequality (a) follows from (7) and the fact that probabilities are upper bounded by 1. Inequality (b) is the key argument. To bound Q_1 (by $d^2/(2d+1)^2$) we use

$$Q_1 + \underbrace{\mathop{\mathbb{E}}_{r'_1,r'_2} \left[\sum_{\substack{P_1,P_2: \\ ((r'_1,r'_2)\notin P_1 - P_2)}} \Pr[P = P_1]\Pr[P = P_2] \right]}_{Q_3} = 1.$$

Exchanging summation order and using $|P_1 - P_2| \leq |P_1| \cdot |P_2| \leq d^2$, which holds since $A(x,d)$ makes at most d queries, we see that $Q_3 \geq 1 - d^2/(2d+1)^2$. Notice we use here $n > 2d$. This proves $Q_1 \leq d^2/(2d+1)^2$.

To bound Q_2, we note that every (P_1, P_2) contribute at most

$$\Pr[P = P_1]\Pr[P = P_2]/(2d+1)^2$$

to Q_2. To see this, observe that the distribution of the random variable $X_{r''_1,r''_2,P_1,P_2}$ does not depend on the particular value of r''_1, r''_2 (given $P_1 + (r''_1, r''_2)$ is disjoint from P_2), and

[7] If the LPHS is probabilistic, then we should add the algorithm's randomness into the σ-algebra. Since this randomness is independent of all other random variables, it doesn't change the proof.

it attains at most one value (that is, a random variable X and a set E always satisfy $\sum_{e \in E} \Pr[X = e] \leq 1$). Hence

$$Q_2 \leq \sum_{P_1, P_2} \frac{\Pr[P = P_1] \Pr[P = P_2]}{(2d+1)^2} = \frac{1}{(2d+1)^2}.$$

Overall, we deduce $2d\delta + (d^2+1)/(2d+1)^2 \geq 1$, implying $\delta \geq 3/(8d)$. ◀

▶ Remark 26 (Extending Lemma 25 to higher dimensions). The proof of Lemma 25 for $k = 2$ readily extends to a lower bound on the error probability of any k-dimensional LPHS with $k > 2$ (the case $k = 1$ follows from the lower bound in [11] and our generic model equivalence with LPHS). Concretely, for a k-dimensional LPHS we have $\delta \geq 1/(3d^{2/k})$ whenever $n > (2d)^{2/k}$, implying Theorem 4.

The extension to a general dimension k requires the following modifications. First, we use a distance-extension lemma, analogous to Lemma 24, in a way similar to (6). This step reduces our task to showing that no algorithm can synchronize on random inputs $x, y \in \Sigma_b^{\mathbb{Z}_n^k}$ with probability higher than, say $1/2$, where y is a random k-dimensional shift of x by about $(2d)^{2/k}$ in every axis.

Then, we observe that in the event that the LPHS applied on x and y queries disjoint input cells (we think of x and y as inlaid in a common landscape), synchronization is unlikely, as expressed by the bound on Q_2 in the proof of Lemma 25. Hence, the synchronization probability is dominated by the probability that the LPHS queries a shared input cell. To bound this latter probability we use a birthday-paradox argument similar to the bound on Q_1 in the proof of Lemma 25: there are at least $((2d)^{2/k})^k = 4d^2$ possible shifts, while there are only $d \times d$ pairs of queries that may collide – any of the d queries made to x may collide with any of the d queries made to y. It follows that there is a probability of at most $1/4$ to have a shared query, concluding the argument.

References

1 Adi Akavia, Hayim Shaul, Mor Weiss, and Zohar Yakhini. Linear-regression on packed encrypted data in the two-server model. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, WAHC@CCS 2019*, pages 21–32. ACM, 2019.

2 Alexandr Andoni, Piotr Indyk, Dina Katabi, and Haitham Hassanieh. Shift finding in sub-linear time. In *SODA 2013*, pages 457–465, 2013.

3 Tugkan Batu, Funda Ergün, Joe Kilian, Avner Magen, Sofya Raskhodnikova, Ronitt Rubinfeld, and Rahul Sami. A sublinear algorithm for weakly approximating edit distance. In *STOC 2003*, pages 316–324, 2003.

4 Elette Boyle, Itai Dinur, Niv Gilboa, Yuval Ishai, Nathan Keller, and Ohad Klein. On the noise sensitivity of locality-preserving hashing for shifts. Manuscript in preparation, 2021.

5 Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO 2016, Part I*, pages 509–539, 2016. Full version: IACR Cryptology ePrint Archive 2016: 585 (2016).

6 Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *EUROCRYPT 2017, Part II*, pages 163–193, 2017.

7 Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In *CRYPTO 2020, Part III*, pages 738–767, 2020.

8 Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *J. Comput. Syst. Sci.*, 60(3):630–659, 2000.

9 Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *STOC 2016*, pages 712–725, 2016.

10 Thomas M. Cover and B. Gopinath. *Open Problems in Communication and Computation.* Springer-Verlag, 1987.

11 Itai Dinur, Nathan Keller, and Ohad Klein. An optimal distributed discrete log protocol with applications to homomorphic secret sharing. In *CRYPTO 2018, Part III*, pages 213–242, 2018.

12 Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

13 Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO 2019*, pages 3–32, 2019.

14 Sanjam Garg, Mohammad Hajiabadi, and Rafail Ostrovsky. Efficient range-trapdoor functions and applications: Rate-1 OT and more. In *TCC 2020, Proceedings, Part I*, pages 88–116, 2020.

15 Elazar Goldenberg, Robert Krauthgamer, and Barna Saha. Sublinear algorithms for gap edit distance. In David Zuckerman, editor, *FOCS 2019*, pages 1101–1120, 2019.

16 Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh S. Vempala. Locality-preserving hashing in multidimensional spaces. In *STOC 1997*, pages 618–625, 1997.

17 Tomasz Kociumaka and Barna Saha. Sublinear-time algorithms for computing & embedding gap edit distance. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1168–1179. IEEE, 2020.

18 Nathan Linial and Ori Sasson. Non-expansive hashing. In *STOC 1996*, pages 509–518, 1996.

19 Henrik Ohlsson, Yonina C Eldar, Allen Y Yang, and S Shankar Sastry. Compressive shift retrieval. *IEEE Transactions on Signal Processing*, 62(16):4105–4113, 2014.

20 Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of Paillier: Homomorphic secret sharing and public-key silent OT. *IACR Cryptol. ePrint Arch.*, 2021:262, 2021. To appear in Eurocrypt 2021.

21 Stephen C. Pohlig and Martin E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (corresp.). *IEEE Trans. Information Theory*, 24(1):106–110, 1978.

22 John M Pollard. Monte carlo methods for index computation mod p. *Mathematics of computation*, 32(143):918–924, 1978.

23 Barna Saha. The Dyck language edit distance problem in near-linear time. In *FOCS 2014*, pages 611–620, 2014.

24 Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT 97*, pages 256–266, 1997.