Property-Preserving Hash Functions for Hamming Distance from Standard Assumptions

Nils Fleischhacker[1⋆][0000−0002−2770−5444], Kasper Green Larsen[2⋆⋆][0000−0001−8841−5929], and Mark Simkin[3][0000−0002−7325−5261]

[1] Ruhr University Bochum, Bochum, Germany
mail@nilsfleischhacker.de
[2] Aarhus University, Aarhus, Denmark
larsen@cs.au.dk
[3] Ethereum Foundation, Aarhus, Denmark
mark.simkin@ethereum.org

Abstract. Property-preserving hash functions allow for compressing long inputs x_0 and x_1 into short hashes $h(x_0)$ and $h(x_1)$ in a manner that allows for computing a predicate $P(x_0, x_1)$ given only the two hash values without having access to the original data. Such hash functions are said to be adversarially robust if an adversary that gets to pick x_0 and x_1 after the hash function has been sampled, cannot find inputs for which the predicate evaluated on the hash values outputs the incorrect result.

In this work we construct robust property-preserving hash functions for the hamming-distance predicate which distinguishes inputs with a hamming distance at least some threshold t from those with distance less than t. The security of the construction is based on standard lattice hardness assumptions.

Our construction has several advantages over the best known previous construction by Fleischhacker and Simkin (Eurocrypt 2021). Our construction relies on a single well-studied hardness assumption from lattice cryptography whereas the previous work relied on a newly introduced family of computational hardness assumptions. In terms of computational effort, our construction only requires a small number of modular additions per input bit, whereas the work of Fleischhacker and Simkin required several exponentiations per bit as well as the interpolation and evaluation of high-degree polynomials over large fields. An additional benefit of our construction is that the description of the hash function can be compressed to λ bits assuming a random oracle. Previous work has descriptions of length $\mathcal{O}(\ell\lambda)$ bits for input bit-length ℓ.

We prove a lower bound on the output size of any property-preserving hash function for the hamming distance predicate. The bound shows that the size of our hash value is not far from optimal.

⋆ Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

⋆⋆ Supported by Independent Research Fund Denmark (DFF) Sapere Aude Research Leader Grant No 9064-00068B.

1 Introduction

Efficient algorithms that compress large amounts of data into small digests that preserve certain properties of the original input data are ubiquitous in computer science and hardly need an introduction. Sketching algorithms [5], approximate membership data structures [8], locality-sensitive hash functions [15], streaming algorithms [20], and compressed sensing [11] are only a few among many examples.

Commonly, these algorithms are studied in benign settings where no adversarial parties are present. More concretely, these randomized algorithms usually state their (probabilistic) correctness guarantees by quantifying over all inputs and arguing that with high probability over the chosen random coins, the algorithm will behave as it should. Importantly, the inputs to the algorithm are considered to be *independent* of the random coins used.

In real world scenarios, however, the assumption of a benign environment may not be justified and an adversary may be incentivized to manipulate a given algorithm into outputting incorrect results by providing malicious inputs. Adversaries that choose their inputs adaptively *after* the random coins of the algorithm have been sampled, were previously studied in the context of sketching and streaming algorithms [19, 14, 21, 10, 9, 6, 7, 12]. These works show that algorithms which work well in benign environments are not guaranteed to work well in the presence of adaptive malicious inputs and several algorithms with security guarantees against malicious inputs were proposed.

The focus of this work are adversarially robust property-preserving hash (PPH) functions recently introduced by Boyle, LaVigne, and Vaikuntanathan [9], which allow for compressing long inputs x_0 and x_1 into short hashes $h(x_0)$ and $h(x_1)$ in a manner that allows for evaluating a predicate $P(x_0, x_1)$ given only the two hash values without having access to the original data. A bit more concretely, a PPH function for a predicate $P : X \times X \to \{0,1\}$ is composed of a deterministic compression function $h : X \to Y$ and an evaluation algorithm $\mathsf{Eval} : Y \times Y \to \{0,1\}$. Such a pair of functions is said to be adversarially robust if no computationally bounded adversary $\mathcal{A}$, who is given a random (h, Eval) from an appropriate family, can find inputs x_0 and x_1, such that $P(x_0, x_1) \neq \mathsf{Eval}(h(x_0), h(x_1))$.

BLV constructed PPH functions that compress inputs by a constant factor for the gap hamming predicate, which distinguishes inputs with very small hamming distance from those with a large distance[4]. For inputs that have neither a very small or very large distance, their construction provided no guarantees.

Subsequently Fleischhacker and Simkin [12] constructed PPH functions for the exact hamming distance predicate, which distinguishes inputs with distance at least t from those with distance less than t. Their construction compresses arbitrarily long inputs into hash values of size $\mathcal{O}(t\lambda)$, where λ is the computational security parameter. Unfortunately, their construction is based on a new family of

[4] We do not care about the exact size of their gap, since we will focus on a strictly stronger predicate in this work.

computational assumptions, which is introduced in their work, meaning that the security of their result is not well understood. From a computational efficiency point of view, their construction is rather expensive. It requires $\mathcal{O}(\ell)$ exponentiations for hashing a single ℓ-bit long input and evaluating the predicate on the hashes requires interpolating and evaluating high-degree polynomials over large fields.

1.1 Our Contribution

In this work we present a new approach for constructing PPH functions for the exact hamming distance predicate, which improves upon the result of Fleischhacker and Simkin in several ways.

The security of our construction relies on a well-studied hardness assumption from the domain of lattice-based cryptography. Both hashing an input and evaluating a predicate on hash values only involves fast operations, such as modular additions, xor, and evaluating a few t-wise independent hash functions. The size of our hash values is $\tilde{\mathcal{O}}(\lambda^2 t)$ bits. We present a lower bound of $\Omega(t \log(\ell/t))$ on the size of the hash value of any PPH function for the exact hamming distance predicate, showing that our result is not far from optimal.

Our hash functions can be described by a uniformly random bit string of sufficient length. This means that, assuming a random oracle, these descriptions can compressed into λ bits by replacing it with a short seed. This compression is not applicable to the work of Fleischhacker and Simkin, since their hash function descriptions are $\Theta(\ell\lambda)$-long bit strings with a secret structure that is only known to the sampling algorithm.

1.2 Technical Overview

Let x_0 and x_1 be two ℓ-bit strings, which we would like to compress using a hash function h in a manner that allows us to use $h(x_0)$ and $h(x_1)$ to check whether $d(x_0, x_1) < t$, where d is the hamming distance and t is some threshold. We start with a simple observation from the work of Fleischhacker and Simkin [12]. We can encode bit strings $x = x_1 x_2 \dots x_\ell$ into sets $X = \{2i - x_i \mid i = 1, \dots, \ell\}$ and for $x_0, x_1 \in \{0,1\}^\ell$ we have that $d(x_0, x_1) < t$, if and only if $|X_0 \triangle X_1| < 2t$. Thus, from now on we can focus on hashing sets and constructing a property-preserving hash function for the symmetric set difference, which turns out to be an easier task.

Conceptually, our construction is inspired by Invertible Bloom Lookup Tables (IBLTs), which were introduced by Goodrich and Mitzenmacher [13]. This data structure allows one to encode a set into an $\tilde{\mathcal{O}}(t)$ sketch with the following properties: Two sketches can be subtracted from each other, resulting in a new sketch that corresponds to an encoding of the symmetric set difference of the original sets. If a sketch contains at most $\mathcal{O}(t)$ many set elements, then it can be decoded with high probability, meaning that the elements within it can be fully recovered.

Given this data structure, one could attempt the following construction of a PPH function for the symmetric set difference predicate. Given an input set, encode it as an IBLT. To evaluate the symmetric set difference predicate on two hash values, subtract the two given IBLTs and attempt to decode the resulting data structure. If decoding succeeds, then count the number of decoded elements and check, whether it's more or less than $2t$. If decoding fails, then conclude that the symmetric set difference is too large. The main issue with this construction is that IBLTs do not provide any correctness guarantees for inputs that are chosen adversarially. Thus, the main contribution of this work is to construct a robust set encoding similar to IBLTs that remains secure in the presence of an adversary.

Our robust set encoding is comprised of "random" functions $r_i : \{0,1\}^* \to \{1,\dots,2t\}$ for $i = 1,\dots,k$ and a "special" collision-resistant hash function A. To encode a set X, we generate an initially empty $k \times 2t$ matrix H. Each element $x \in X$ is then inserted by adding $A(x)$ in each row i to column $r_i(x)$ in H, i.e., $H[i, r_i(x)] = H[i, r_i(x)] + A(x)$ for $i = 1,\dots,k$. To subtract two encodings, we simply subtract the two matrices entry-wise. To decode a matrix back into a set, we repeatedly look for entries in H that contain a single hash value $A(x)$, i.e., for cells i, j with $|H[i,j]| = A(x)$ for some x, and peel them away. That is, whenever we find such an entry, we find x corresponding to $A(x)$ and then remove x from all positions, where it was originally inserted in H. Then we repeat the process until the matrix H is empty or until the process gets stuck, because no cell contains a single set element by itself.

To prove security of our construction, we will show two things. First, we will show that no adversary can find a pair of sets that have a small symmetric set difference, where the peeling process will get stuck. Actually, we will show something stronger, namely that such pairs do not exist with overwhelming probability over the random choices of $r_1,\dots,r_k$. Secondly, we will need to show that no (computationally bounded) adversary can find inputs, which decode incorrectly. In particular, we will have to argue that the peeling process never decodes an element that was not actually encoded, i.e., that the sum of several hash values in some cell $H[i,j]$ never looks like $A(x)$ for some single set element x. To argue that such a bad sum of hash values does not exist, one would need to pick the output length of A too big in the sense that our resulting PPH function would not be compressing. Instead, we will show that for an appropriate choice of A these sums may exist, but finding them is hard and can be reduced to the computational hardness of solving the Short Integer Solution Problem [4], a well-studied assumption from lattice-based cryptography.

2 Preliminaries

This section introduces notation, some basic definitions and lemmas that we will use throughout this work. We denote by $\lambda \in \mathbb{N}$ the security parameter and by $\mathsf{poly}(\lambda)$ any function that is bounded by a polynomial in λ. A function f in λ is negligible, if for every $c \in \mathbb{N}$, there exists some $N \in \mathbb{N}$, such that for all

$\lambda > N$ it holds that $f(\lambda) < 1/\lambda^c$. We denote by $\mathsf{negl}(\lambda)$ any negligible function. An algorithm is PPT if it is modeled by a probabilistic Turing machine with a running time bounded by $\mathsf{poly}(\lambda)$.

We write e_i to denote the i-th canonical unit vector, i.e. the vector of zeroes with a one in position i, and assume that the dimension of the vector is known from the context. For a row vector v, we write $v^\intercal$ to denote its transpose. Let $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. Let X, Y be sets, we denote by $|X|$ the size of X and by $X \triangle Y$ the symmetric set difference of X and Y, i.e., $X \triangle Y = (X \cup Y) \setminus (X \cap Y) = (X \setminus Y) \cup (Y \setminus X)$. We write $x \leftarrow X$ to denote the process of sampling an element of X uniformly at random. For $x, y \in \{0,1\}^n$, we write $w(x)$ to denote the Hamming weight of x and we write $d(x, y)$ to denote the Hamming distance between x and y, i.e., $d(x, y) = w(x \oplus y)$. We write x_i to denote the i-th bit of x.

2.1 Property-Preserving Hash Functions

The following definition of property-preserving hash functions is taken almost verbatim from [9]. In this work, we consider the strongest of several different security notions that were proposed in [9].

Definition 1 (Property-Preserving Hash). *For a $\lambda \in \mathbb{N}$ an η-compressing property-preserving hash function family $\mathcal{H}_\lambda = \{h : X \to Y\}$ for a two-input predicate requires the following three efficiently computable algorithms:*

$\mathsf{Sample}(1^\lambda) \to h$ *is an efficient randomized algorithm that samples an efficiently computable random hash function from $\mathcal{H}$ with security parameter λ.*

$\mathsf{Hash}(h, x) \to y$ *is an efficient deterministic algorithm that evaluates the hash function h on x.*

$\mathsf{Eval}(h, y_0, y_1) \to \{0,1\}$**:** *is an efficient deterministic algorithm that on input h, and $y_0, y_1 \in Y$ outputs a single bit.*

We require that $\mathcal{H}$ must be compressing, meaning that $\log |Y| \leq \eta \log |X|$ for $0 < \eta < 1$.

For notational convenience we write $h(x)$ for $\mathsf{Hash}(h, x)$.

Definition 2 (Direct-Access Robustness). *A family of PPH functions $\mathcal{H} = \{h : X \to Y\}$ for a two-input predicate $P : X \times X \to \{0,1\}$ is a family of direct-access robust PPH functions if, for any PPT adversary $\mathcal{A}$ it holds that,*

$$\Pr\left[\begin{matrix} h \leftarrow \mathsf{Sample}(1^\lambda); \\ (x_0, x_1) \leftarrow \mathcal{A}(h) \end{matrix} : \mathsf{Eval}(h, h(x_0), h(x_1)) \neq P(x_0, x_1)\right] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the internal random coins of Sample *and $\mathcal{A}$.*

Two-Input Predicates. We define the following two-input predicates, which will be the main focus of this work.

Definition 3 (Hamming Predicate). *For $x, y \in \{0,1\}^n$ and $t > 0$, the two-input predicate is defined as*

$$\mathsf{HAM}^t(x,y) = \begin{cases} 1 & \text{if } d(x,y) \geq t \\ 0 & \text{Otherwise} \end{cases}$$

2.2 Lattices

In the following we recall some lattice hardness assumptions and the relationships between them. We start by revisiting one of the most well-studied computational problems.

Definition 4 (Shortest Independent Vector Problem). *For an approximation factor of $\gamma := \gamma(n) \geq 1$, the (n,γ)-SIVP is defined as follows: Given a lattice $\mathcal{L} \subset \mathbb{R}^n$, output n linearly independent lattice vectors, which have all euclidean length at most $\gamma \cdot \lambda_n(\mathcal{L})$, where $\lambda_n(\mathcal{L})$ is the minimum possible.*

Starting with the celebrated work of Lenstra, Lenstra, and Lovász [16], a long line of research works [1, 3, 2] has been dedicated to finding fast algorithms for solving the exact and approximate shortest independent vector problem. All existing algorithms for finding any $\mathsf{poly}(n)$-approximation run in time $2^{\Omega(n)}$ and it is believed that one can not do better asymptotically as is captured in the following assumption.

Assumption 5. *For large enough n, there exists no $2^{o(n)}$-time algorithm for solving the (n,γ)-SIVP with $\gamma = \mathsf{poly}(n)$.*

A different computationally hard problem that has been studied extensively is the short integer solution problem.

Definition 6 (Short Integer Solution Problem). *For parameters n, m, q, $\beta_2, \beta_\infty \in \mathbb{N}$, the $(n,m,q,\beta_2,\beta_\infty)$-SIS problem is defined as follows: Given a uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$, find $s \in \mathbb{Z}^m$ with $\|s\|_2 \leq \beta_2$ and $\|s\|_\infty \leq \beta_\infty$, such that $As^\intercal = 0$.*

It was shown by Micciancio and Peikert that the difficulty of solving the SIS problem fast on average is related to the difficulty of solving the SIVP in the worst-case.

Theorem 1 (Worst-Case to Average-Case Reduction for SIS [17]). *Let n, $m := m(n)$, and $\beta_2 \geq \beta_\infty \geq 1$ be integers. Let $q \geq \beta_2 \cdot n^\delta$ for some constant $\delta > 0$. Solving the $(n,m,q,\beta_2,\beta_\infty)$-SIS problem on average with non-negligible probability in n is at least as hard as solving the (n,γ)-SIVP in the worst-case to within $\gamma = \max(1, \beta_2 \cdot \beta_\infty / q) \cdot \tilde{\mathcal{O}}(\beta_2 \sqrt{n})$.*

Combining the above result with Assumption 5, we get the following corollary.

Corollary 2. *Let $n \in \Theta(\lambda)$ and $m = \mathsf{poly}(\lambda)$ be integers, let $\beta_\infty = 2$, and let $\beta_2 = \sqrt{m+\nu}$ for some constant ν. Let $q > \beta_2 \cdot n^\delta$ for some constant $\delta > 0$. If Assumption 5 holds, then for large enough λ, there exists no PPT adversary that solves the $(n, m, q, \beta_2, \beta_\infty)$-SIS problem with non-negligible (in λ) probability.*

3 Robust Set Encodings

In this section, we define our notion of robust set encodings. The encoding transforms a possibly large set into a smaller sketch. Given two sketches of sets with a small enough symmetric set difference, one should be able to decode the symmetric set difference. The security of our encodings guarantees that no computationally bounded adversary can find a pair of sets where decoding either returns the incorrect result or fails even though the symmetric set difference between the encoded sets is small.

Definition 7 (Robust Set Encodings). *A robust set encoding for a universe U is comprised of the following algorithms:*

$\mathsf{Sample}(1^\lambda, t) \to f$ *is an efficient randomized algorithm that takes the security parameter λ and threshold t as input and returns an efficiently computable set encoding function f sampled from the family $\mathcal{E}$.*

$\mathsf{Encode}(f, X) \to y$ *is an efficient deterministic algorithm that takes set encoding function f and set $X \subset U$ as input and returns encoding y.*

$\mathsf{Decode}(f, y_0, y_1) \to X'/\perp$ *is an efficient deterministic algorithm that takes set encoding function f and two set encodings y_0, y_1 as input and returns set X' or $\perp$.*

We denote by $\mathsf{Len}_\mathcal{E} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ the function that describes the length of the encoding for a given security parameter λ and threshold t. For any two sets X_0, X_1 we use $X' \leftarrow \mathsf{Diff}(f, X_0, X_1)$ as a shorthand notation for

$$X' \leftarrow \mathsf{Decode}(f, \mathsf{Encode}(f, X_0), \mathsf{Encode}(f, X_1)).$$

We say a set encoding is robust, if for any PPT adversary $\mathcal{A}$ and any threshold $t \in \mathbb{N}$ it holds that,

$$\Pr\left[\begin{array}{l} f \leftarrow \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f, t); \\ X' \leftarrow \mathsf{Diff}(f, X_0, X_1) \end{array} : \begin{array}{c} X' \notin \{X_0 \triangle X_1, \perp\} \\ \vee\ (|X_0 \triangle X_1| < t \wedge X' = \perp) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins of the adversary $\mathcal{A}$ and Sample.

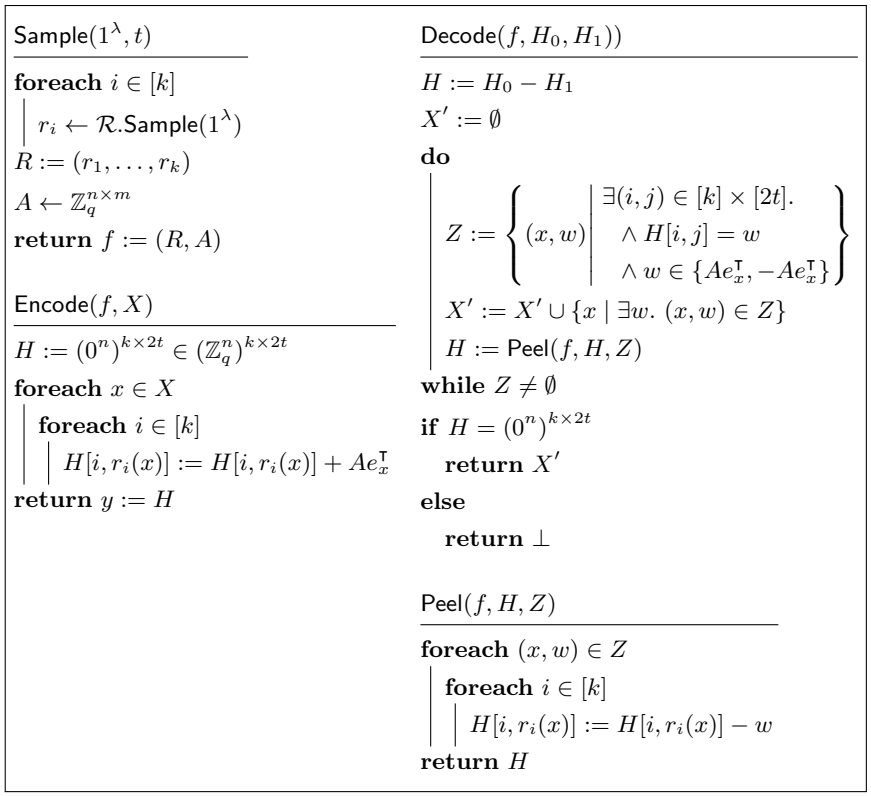

Fig. 1. Construction of a robust set encoding for universe $[m]$.

3.1 Instantiation

In this section we construct a set encoding for universe $[m]$ with $m = \mathsf{poly}(\lambda)$ by modifying Invertible Bloom Lookup Tables [13] to achieve security against adaptive malicious inputs. Since we are only encoding polynomially large sets and can leverage the cryptographic hardness of the SIS problem, we can get away with only maintaining a matrix of hash values in our sketch and we do not require the additional counter or value fields that were present in the original construction of Goodrich and Mitzenmacher. Refer to Figure 1 for a full description of the construction. Before we prove that the construction is a robust set encoding we will first prove a few of its properties that will be useful in the following.

The following lemma effectively states that given the difference of two encodings there will always be a least one element that can can be peeled if the symmetric set difference is small enough.

Lemma 3. *Let $\mathcal{R}$ be a family of t-wise independent hash functions $r : [m] \to [2t]$ and let $k \geq 2 \log_{3/e} m$. With probability at least $1 - 2^{-\Omega(k)}$, it simultaneously holds for all sets $T \subseteq [m]$ with $0 < |T| \leq t$ that there is at least one $x \in T$ and one*

index $i \in [k]$ such that $r_i(x) \neq r_i(y)$ for all $y \in T \setminus \{x\}$. Here the probability is taken over the random choice of the r_i's.

Proof. Let E denote the event that there is a set T with $0 < |T| \leq t$ such that for all $x \in T$ and all $i \in [k]$, there is a $y \in T \setminus \{x\}$ with $r_i(x) = r_i(y)$. We show that $\Pr[E]$ is small. The proof follows from a union bound over all $T \subseteq [m]$ with $2 \leq |T| \leq t$. So fix one such T. Let E_T denote the event that there is no $i \in [k]$ and $x \in T$ such that $r_i(x) \neq r_i(y)$ for all $y \in T \setminus \{x\}$. Then by a union bound, we have

$$\Pr[E] \leq \Pr\Big[\bigcup_{T \subseteq [m]} E_T\Big] \leq \sum_{T \subseteq [m]} \Pr[E_T].$$

To bound $\Pr[E_T]$, notice that conditioned on E_T, the number of distinct hash values $|\{r_i(x) \mid x \in T\}|$ for the ith hash function is at most $|T|/2$, as every hash value is *hit* by either 0 or at least 2 elements from T. Now define an event $E_{T,S}$ for every k-tuple $S = (S_1, \ldots, S_k)$ where S_i is a subset of $|T|/2$ values in $[k]$. The event $E_{T,S}$ occurs if $r_i(x) \in S_i$ for every $x \in T$ and every $i \in [k]$. If E_T happens then at least one event $E_{T,S}$ happens. Thus

$$\Pr[E_T] \leq \Pr\Big[\bigcup_{S} E_{T,S}] \leq \sum_{S} \Pr[E_{T,S}].$$

To bound $\Pr[E_{T,S}]$, notice that by t-wise independence, the values $r_i(x)$ are independent and fall in S_i with probability exactly $|T|/(2 \cdot 2t)$. Since this must happen for every i and every $x \in T$, we get that $\Pr[E_{T,S}] \leq (|T|/(4t))^{|T|k}$ and $\Pr[E_T] \leq \binom{2t}{|T|/2}^k (|T|/(4t))^{|T|k}$. A union bound over all T gives us $\Pr[E] \leq \sum_{j=2}^{t} \binom{m}{j} \binom{2t}{j/2}^k (j/(4t))^{jk}$. Using the bound $\binom{n}{k} \leq (en/k)^k$ for all $0 \leq k \leq n$ and the bound $\binom{m}{j} \leq m^j$, we finally conclude:

$$\begin{aligned}
\Pr[E] &\leq \sum_{j=2}^{t} \binom{m}{j} \binom{2t}{j/2}^k (j/(4t))^{jk} \\
&\leq \sum_{j=2}^{t} m^j (4et/j)^{jk/2} (j/(4t))^{jk} \\
&= \sum_{j=2}^{t} m^j (e/3)^{jk/2} (3j/(4t))^{jk/2}
\end{aligned}$$

For $k \geq 2 \log_{3/e} m$ we have $(e/3)^{k/2} \leq 1/m$. The above is thus bounded by

$$\begin{aligned}
\Pr[E] &\leq \sum_{j=2}^{t} (3j/(4t))^{jk/2} \\
&\leq \sum_{j=2}^{t} (3/4)^{jk/2}
\end{aligned}$$

For any $k \geq 2$, the terms in this sum go down by a factor at least $4/3$ and thus is bounded by $2^{-\Omega(k)}$. □

In the next lemma we show that correctly peeling one layer of elements during decoding leads to a state that is equivalent to never having inserted those elements in the first place.

Lemma 4. *For any security parameter λ, any threshold t, any encoding function $f \leftarrow \mathsf{Sample}(1^\lambda, t)$, any pair of subsets $X_0, X_1 \subseteq [m]$ and any set*

$$Z \subseteq \{(x, Ae_x^\intercal) \mid x \in X_0 \setminus X_1\} \cup \{(x, -Ae_x^\intercal) \mid x \in X_1 \setminus X_0\}$$

and $X := \{x \mid \exists w.\ (x,w) \in Z\}$ it holds that

$$\mathsf{Peel}(\mathsf{Encode}(f, X_0) - \mathsf{Encode}(f, X_1), Z) = \mathsf{Encode}(f, X_0 \setminus X) - \mathsf{Encode}(f, X_1 \setminus X).$$

Proof. Let $H_b := \mathsf{Encode}(f, X_b)$, $H'_b := \mathsf{Encode}(f, X'_b \setminus X)$ and $H := \mathsf{Peel}(f, H_0 - H_1, Z)$ For any $(i,j) \in [k] \times [2t]$, let $S_{i,j} = \{x \in [m] \mid r_i(x) = j\}$. Then for each $(i,j) \in [k] \times [2t]$ we have

$$\begin{aligned}
H[i,j] =& H_0[i,j] - H_1[i,j] - \sum_{x \in X \cap S_{i,j}} Z(x) && (1)\\
=& \sum_{x \in X_0 \cap S_{i,j}} Ae_x^\intercal - \sum_{x \in X_1 \cap S_{i,j}} Ae_x^\intercal - \sum_{x \in X \cap S_{i,j}} Z(x) && (2)\\
=& \sum_{x \in X_0 \cap S_{i,j}} Ae_x^\intercal - \sum_{x \in X_1 \cap S_{i,j}} Ae_x^\intercal - \sum_{x \in X \cap X_0 \cap S_{i,j}} Z(x) - \sum_{x \in X \cap X_1 \cap S_{i,j}} Z(x) && (3)\\
=& \sum_{x \in X_0 \cap S_{i,j}} Ae_x^\intercal - \sum_{x \in X_1 \cap S_{i,j}} Ae_x^\intercal - \sum_{x \in X \cap X_0 \cap S_{i,j}} Ae_x^\intercal + \sum_{x \in X \cap X_1 \cap S_{i,j}} Ae_x^\intercal && (4)\\
=& \sum_{x \in (X_0 \setminus X) \cap S_{i,j}} Ae_x^\intercal - \sum_{x \in (X_1 \setminus X) \cap S_{i,j}} Ae_x^\intercal && (5)\\
=& H'_0[i,j] - H'_1[i,j], && (6)
\end{aligned}$$

where we denote by $Z(x)$ the unique value w such that $(x,w) \in Z$. Equations 1 and 2 follow from the definitions of Peel and Encode respectively. Equations 3 and 5 follow from the fact that X is a subset of the symmetric set difference of X_0 and X_1. Equation 4 follows from the fact that $w = (-1)^b Ae_x^\intercal$ iff $x \in X_b$. Finally, Equation 6 follows again from the definition of Encode. □

The following lemma essentially states that during the decoding process we will never peel an element that is not in the symmetric set difference *and* all elements will be peeled correctly, i.e., the decoding algorithm correctly identifies whether an element is from X_0 or from X_1.

Lemma 5. *For an encoding function $f \leftarrow \mathsf{Sample}(1^\lambda, t)$ and two sets X_0, X_1, let $Z_1, Z_2, \ldots$ denote the sequence of sets peeled during the execution of*

$$\mathsf{Decode}(f, \mathsf{Encode}(f, X_0), \mathsf{Encode}(f, X_1)).$$

Let further $X_b^c = X_b \backslash \{y \mid \exists w.\, (y,w) \in Z_1 \cup \cdots \cup Z_{c-1}\}$. *If the* $(n, m, q, \sqrt{m+3}, 2)$-*SIS problem is hard, then for any PPT algorithm* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{matrix} f := \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f) \end{matrix} \exists c.\, Z_c \not\subseteq \begin{matrix} \{(x, Ae_x^\intercal) \mid x \in X_0^c \setminus X_1^c\} \\ \cup \{(x, -Ae_x^\intercal) \mid x \in X_1^c \setminus X_0^c\} \end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

Proof. Let $\mathcal{A}$ be an arbitrary PPT algorithm with

$$\Pr\left[\begin{matrix} f := \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f) \end{matrix} \exists c.\, Z_c \not\subseteq \begin{matrix} \{(x, Ae_x^\intercal) \mid x \in X_0^c \setminus X_1^c\} \\ \cup \{(x, -Ae_x^\intercal) \mid x \in X_1^c \setminus X_0^c\} \end{matrix}\right] = \epsilon(\lambda).$$

We construct an algorithm $\mathcal{B}$ that solves $(n, m, q, \sqrt{m+3}, 2)$-SIS as follows. $\mathcal{B}$ receives as input a random matrix $A \in \mathbb{Z}_q^{n \times m}$, samples $r_i \leftarrow \mathcal{R}$ for $i \in [k]$ and invokes $\mathcal{A}$ on $f = (A, (r_1, \ldots, r_k))$. Once $\mathcal{A}$ outputs X_0, X_1, $\mathcal{B}$ runs $H_0 := \mathsf{Encode}(f, X_0)$ and $H_1 := \mathsf{Encode}(f, X_1)$ and then starts to execute $\mathsf{Decode}(f, H_0, H_1)$. Let Z_c denote the set Z in the c-th iteration of the main loop of Decode. In each iteration, if

$$Z_c \not\subseteq \{(x, Ae_x^\intercal) \mid x \in X_0^c \setminus X_1^c\} \cup \{(x, -Ae_x^\intercal) \mid x \in X_1^c \setminus X_0^c\},$$

then $\mathcal{B}$ stops the decoding process and proceeds as follows.

Let $S_{i,j} = \{x \in [m] \mid r_i(x) = j\}$. By definition of Z, there must exists at least one element $(x, w) \in Z_c$, such that

$$H[i,j] = (-1)^b Ae_x^\intercal \quad \text{and} \quad x \notin X_b^c \setminus X_{1-b}^c \tag{7}$$

for some cell (i, j) and some bit b. $\mathcal{B}$ identifies one such cell by exhaustive search and outputs the vector

$$s := \sum_{y \in X_0^c \cap S_{i,j}} e_y \; - \sum_{y \in X_1^c \cap S_{i,j}} e_y - (-1)^b e_x.$$

If the decoding procedure terminates without such a Z_c occurring, $\mathcal{B}$ outputs $\perp$.

To analyze the success probability of $\mathcal{B}$, consider that by Lemma 4 and since Z_c is the *first* set in which an element as specified above exists, we have that $H = \mathsf{Encode}(f, X_0') - \mathsf{Encode}(f, X_1')$, i.e.

$$(-1)^b Ae_x^\intercal = H[i,j] = \sum_{y \in X_0^c \cap S_{i,j}} Ae_y^\intercal \; - \sum_{y \in X_1^c \cap S_{i,j}} Ae_y^\intercal$$

Thus, whenever $\mathcal{B}$ outputs a vector s, it holds that $As^\intercal = 0$. Furthermore, this vector consists of the sum of at most m unique canonical unit vectors and one additional canonical unit vector. This implies that $\|s\|_2 \leq \sqrt{m+3}$ and $\|s\|_\infty \leq 2$. It remains to argue that s is non-zero. The vector s is zero, iff

$$\sum_{y \in X_0^c \cap S_{i,j}} e_y \; - \sum_{y \in X_1^c \cap S_{i,j}} e_y = (-1)^b e_x.$$

Observe that, since we are summing up canonical unit vectors, this can hold only if $x \in X_b^c \setminus X_{1-b}^c$. However, by Equation 7 this does not occur, therefore s is non-zero.

We can conclude that $\mathcal{B}$ solves $(n, m, q, \sqrt{m+3}, 2)$-SIS, with probability $\epsilon(\lambda)$. Since $(n, m, q, \sqrt{m+3}, 2)$-SIS is assumed to be hard, $\epsilon(\lambda)$ must be negligible. □

The following lemma states that with overwhelming probability the decoding process will output either $\perp$ or a subset of the symmetric set difference, even for maliciously chosen sets X_0, X_1.

Lemma 6. *If the $(n, m, q, \sqrt{m+3}, 2)$-SIS problem is hard, then for any PPT adversary $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{l} f := \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f); \ : \ X' \neq \perp \wedge X' \not\subseteq X_0 \triangle X_1 \\ X' := \mathsf{Diff}(f, X_0, X_1) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Proof. Let $Z_1, Z_2, \ldots$ denote the sequence of sets peeled during the execution of

$$\mathsf{Decode}(f, \mathsf{Encode}(f, X_0), \mathsf{Encode}(f, X_0)).$$

If an algorithm outputs X_0, X_1, such that $X' \not\subseteq X_0 \triangle X_1$, there must exist an $x \in X'$ such that

$$x' \notin X_0 \triangle X_1 = (X_0 \setminus X_1) \cup (X_1 \setminus X_0).$$

Since $X' := \{x \mid \exists w.\, (x, w) \in Z_1 \cup \ldots\}$, this can only happen with negligible probability by Lemma 5. □

The following lemma states that with overwhelming probability the decoding process will never output a *strict* subset of the symmetric set difference, even for maliciously chosen sets X_0, X_1.

Lemma 7. *If the $(n, m, q, \sqrt{m+3}, 2)$-SIS problem is hard, then for any PPT adversary $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{l} f := \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f); \ : \ X' \subsetneq X_0 \triangle X_1 \\ X' := \mathsf{Diff}(f, X_0, X_1) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Proof. Let $\mathcal{A}$ be a PPT an adversary for the above experiment. We construct an adversary $\mathcal{B}$ against $(n, m, q, \sqrt{m+3}, 2)$-SIS as follows. $\mathcal{B}$ is given matrix A, samples $r_i \leftarrow \mathcal{R}$ for $i \in [k]$ and invokes $\mathcal{A}$ on $f = (A, (r_1, \ldots, r_k))$. Adversary $\mathcal{A}$ returns X_0 and X_1 and $\mathcal{B}$ computes $X' := \mathsf{Diff}(f, X_0, X_1)$. If $X' \subsetneq X_0 \triangle X_1$, then $\mathcal{B}$ computes $X_b' = X_b \setminus X'$ for $b \in \{0, 1\}$ and finds an index i, j such that there exists an $x \in X_0' \triangle X_1'$ with $r_i(x) = j$. $\mathcal{B}$ returns

$$s := \sum_{y \in X_0' \cap S_{i,j}} e_y \ - \sum_{y \in X_1' \cap S_{i,j}} e_y.$$

Since every canonical unit vector appears at most once in the sum above, it follows that $\|s\|_2 \leq \sqrt{m}$ and $\|s\|_\infty = 1$. Further, since, by construction, there exists at least one $y \in (X'_0 \cap S_{i,j}) \triangle (X'_1 \cap S_{i,j})$ it follows that $s \neq 0$.

To analyze the probability that $As^\intercal = 0$ we consider the following. Let H' be the value of the matrix H when the decoding procedure terminates. By Lemma 5 and Lemma 4 it holds with overwhelming probability that $H' = H'_0 - H'_1 = \mathsf{Encode}(f, X'_0) - \mathsf{Encode}(f, X'_1)$. However, since the decoding terminates successfully, it must also hold that $H' = (0^n)^{k \times 2t}$. It follows that for all i, j, we have $H'_0[i,j] - H'_1[i,j] = 0$ and therefore $As = 0$ with overwhelming probability. Since $(n, m, q, \sqrt{m+3}, 2)$-SIS is assumed to be hard the lemma follows. □

By combining Lemma 6 and Lemma 7 we obtain the following corollary stating that with overwhelming probability the decoding process will output *either* the correct symmetric set difference *or* the error symbol $\perp$.

Corollary 8. *If the $(n, m, q, \sqrt{m+3}, 2)$-SIS problem is hard, then for any PPT adversary $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{l} f := \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f); \\ X' := \mathsf{Diff}(f, X_0, X_1) \end{array} : X' \notin \{X_0 \triangle X_1, \perp\}\right] \leq \mathsf{negl}(\lambda)$$

The following lemma states that with overwhelming probability the decoding process will not output $\perp$ if the symmetric set difference is small.

Lemma 9. *If the $(n, m, q, \sqrt{m+3}, 2)$-SIS problem is hard, then for any PPT adversary $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{l} f \leftarrow \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f, t); \\ X' \leftarrow \mathsf{Diff}(f, X_0, X_1) \end{array} : |X_0 \triangle X_1| < t \wedge X' = \perp\right] \leq \mathsf{negl}(\lambda)$$

Proof. Let $\mathcal{A}$ be an arbitrary PPT algorithm. By Lemma 5 and Lemma 4 it holds that in each iteration c we have $H = H_{c,0} - H_{c,1}$, where $H_{c,b} = \mathsf{Encode}(f, X_{c,0}, X_{c,1})$ and $X_{c,b} = X_b \setminus \{x \mid \exists w.\ (x,w) \in Z_1 \cup \cdots \cup Z_{c-1}\}$. Since it must hold that $|X_0 \triangle X_1| < t$ it in particular holds that $|X_{c,0} \triangle X_{c,1}| < t$ in each iteration. By Lemma 3, in each iteration where $X_{c,1} \triangle X_{c,2} \neq \emptyset$ it holds that $Z_c \neq \emptyset$ with overwhelming probability. Therefore, the decoding process terminates after at most t steps, with $X' = X_0 \triangle X_1$. Since each peeling step was correct with overwhelming probability it must hold that $H = (0^n)^{k \times 2t}$. □

Given the above lemmas, we can now easily prove the following theorem.

Theorem 10. *Let $\mathcal{R}$ be a family of t-wise independent hash functions $r : [m] \to [2t]$ and let $k \geq \max\{\lambda, 2\log_{3/e} m\}$. Then the construction in Figure 1 is a robust set encoding for universe $[m]$ if the $(n = n(\lambda), m, q, \sqrt{m+3}, 2)$-SIS problem is hard.*

$\mathsf{Sample}(1^\lambda)$	$\mathsf{Hash}(h, x)$	$\mathsf{Eval}(h, y_0, y_1)$
$f \leftarrow \mathcal{E}.\mathsf{Sample}(1^\lambda, 2t)$	$X := \{2i - x_i \mid i \in [\ell]\}$	$X' := \mathcal{E}.\mathsf{Decode}(h, y_0, y_1)$
return $h := f$	$y := \mathcal{E}.\mathsf{Encode}(h, X)$	**if** $X' = \perp$ **or** $\lvert X' \rvert \geq 2t$
	return y	**return** 1
		else
		return 0

Fig. 2. A family of direct-access robust PPHs for the predicate HAM^t over the domain $\{0,1\}^\ell$ for any $\ell \in \mathbb{N}$.

Proof. Let $\mathcal{A}$ be an arbitrary PPT algorithm, using Corollary 8, Lemma 9 and a simple union bound we can conclude that

$$\begin{aligned}
&\Pr\left[\begin{array}{l} f \leftarrow \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f, t); \\ X' \leftarrow \mathsf{Diff}(f, X_0, X_1) \end{array} : \begin{array}{c} X' \notin \{X_0 \triangle X_1, \perp\} \\ \vee\ (|X_0 \triangle X_1| < t \wedge X' = \perp) \end{array}\right] \\
\leq\ &\Pr\left[\begin{array}{l} f \leftarrow \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f, t); \\ X' \leftarrow \mathsf{Diff}(f, X_0, X_1) \end{array} : X' \notin \{X_0 \triangle X_1, \perp\}\right] \\
&+ \Pr\left[\begin{array}{l} f \leftarrow \mathsf{Sample}(1^\lambda, t); \\ (X_0, X_1) \leftarrow \mathcal{A}(f, t); \\ X' \leftarrow \mathsf{Diff}(f, X_0, X_1) \end{array} : |X_0 \triangle X_1| < t \wedge X' = \perp\right] \\
\leq\ &\mathsf{negl}(\lambda).
\end{aligned}$$

Remark 1. Instantiated as specified, the construction has keys that consist of k many t-wise independent hash functions and a matrix $A \in \mathbb{Z}_q^{m \times n}$, leading to a key length of $kt \cdot \log m + mn \cdot \log q$. Note that the entire key can be represented by a public uniformly random $kt \cdot \log m + mn \cdot \log q$ bit string. Assuming the existence of a random oracle, this string can be replaced by a short λ bit seed.

4 Construction

In this section we construct property-preserving hash functions for the exact hamming distance predicate based on robust set encodings.

4.1 PPH for the Hamming Distance Predicate

Theorem 11. *Let $\ell = \mathsf{poly}(\lambda)$ and $t \leq \ell$. Let $\mathcal{E}$ be a robust set encoding for universe $[2\ell]$ with encoding length $\mathsf{Len}_\mathcal{E}$. Then, the construction in Figure 2 is a $\mathsf{Len}_\mathcal{E}(\lambda, 2t)/\ell$-compressing direct-access robust property-preserving hash function family for the two-input predicate HAM^t and domain $\{0,1\}^\ell$.*

Proof. Let $\mathcal{A}$ be an arbitrary PPT adversary against the direct-access robustness of $\mathcal{H}$. We construct an adversary $\mathcal{B}$ against the robustness of $\mathcal{E}$ as follows. Upon input e, $\mathcal{B}$ invokes $\mathcal{A}$ on input $h := f$. When $\mathcal{A}$ outputs x_0, x_1, $\mathcal{B}$ outputs $X_0 := \{2i - x_{0,i} \mid i \in [\ell]\}$ and $X_1 := \{2i - x_{1,i} \mid i \in [\ell]\}$. We note that it holds that

$$\Pr\left[\begin{matrix} h \leftarrow \mathsf{Sample}(1^\lambda); \\ (x_0, x_1) \leftarrow \mathcal{A}(h) \end{matrix} : \mathsf{Eval}(h, h(x_0), h(x_1)) \neq \mathsf{HAM}^t(x_0, x_1)\right] \tag{8}$$

$$= \Pr\left[\begin{matrix} f \leftarrow \mathcal{E}.\mathsf{Sample}(1^\lambda, 2t); \\ (X_0, X_1) \leftarrow \mathcal{B}(f); \\ y_0 := \mathcal{E}.\mathsf{Encode}(f, X_0); \\ y_1 := \mathcal{E}.\mathsf{Encode}(f, X_1) \end{matrix} : \mathsf{Eval}(f, y_0, y_1) \neq \mathsf{HAM}^t(x_0, x_1)\right] \tag{9}$$

$$= \Pr\left[\begin{matrix} f \leftarrow \mathcal{E}.\mathsf{Sample}(1^\lambda, 2t); \\ (X_0, X_1) \leftarrow \mathcal{B}(f); \\ X' := \mathsf{Diff}(f, X_0, X_1) \end{matrix} : \begin{matrix} (d(x_0, x_1) \geq t \wedge X' \neq \perp \wedge |X'| < 2t) \\ \vee (d(x_0, x_1) < t \wedge (X' = \perp \vee |X'| \geq 2t)) \end{matrix}\right] \tag{10}$$

$$= \Pr\left[\begin{matrix} f \leftarrow \mathcal{E}.\mathsf{Sample}(1^\lambda, 2t); \\ (X_0, X_1) \leftarrow \mathcal{B}(f); \\ X' := \mathsf{Diff}(f, X_0, X_1) \end{matrix} : \begin{matrix} (|X_0 \triangle X_1| \geq 2t \wedge X' \neq \perp \wedge |X'| < 2t) \\ \vee (|X_0 \triangle X_1| < 2t \wedge (X' = \perp \vee |X'| \geq 2t)) \end{matrix}\right] \tag{11}$$

$$= \Pr\left[\begin{matrix} f \leftarrow \mathcal{E}.\mathsf{Sample}(1^\lambda, 2t); \\ (X_0, X_1) \leftarrow \mathcal{B}(f); \\ X' := \mathsf{Diff}(f, X_0, X_1) \end{matrix} : \begin{matrix} (|X_0 \triangle X_1| \geq 2t \wedge X' \neq \perp \wedge |X'| < 2t) \\ \vee (|X_0 \triangle X_1| < 2t \wedge X' \neq \perp \wedge |X'| \geq 2t) \\ \vee (|X_0 \triangle X_1| < 2t \wedge X' = \perp) \end{matrix}\right] \tag{12}$$

$$= \Pr\left[\begin{matrix} f \leftarrow \mathcal{E}.\mathsf{Sample}(1^\lambda, 2t); \\ (X_0, X_1) \leftarrow \mathcal{B}(f); \\ X' := \mathsf{Diff}(f, X_0, X_1) \end{matrix} : \begin{matrix} (X' \neq \perp \wedge |X_0 \triangle X_1| \neq |X'|) \\ \vee (|X_0 \triangle X_1| < 2t \wedge X' = \perp) \end{matrix}\right] \tag{13}$$

$$\leq \Pr\left[\begin{matrix} f \leftarrow \mathcal{E}.\mathsf{Sample}(1^\lambda, 2t); \\ (X_0, X_1) \leftarrow \mathcal{B}(f); \\ X' := \mathsf{Diff}(f, X_0, X_1) \end{matrix} : \begin{matrix} X' \notin \{X_0 \triangle X_1, \perp\} \\ \vee (|X_0 \triangle X_1| < 2t \wedge X' = \perp) \end{matrix}\right]. \tag{14}$$

$$\tag{15}$$

Here Equation 9 follows from the definition of Sample and Hash and Equation 10 follows from the definition of Eval as well as the exact hamming distance predicate. Equation 11 follows from the definition of the sets X_0, X_1: for each position i where the $x_{0,i} = x_{1,i}$, the sets share an element, whereas for every position where $x_{0,i} \neq x_{1,i}$, one of them contains the element $2i$ and the other $2i-1$, thus $d(x_0, x_1) = t \iff |X_0 \triangle X_1| = 2t$. Equations 12 and 13 follow by first splitting the bottom clause and then rewriting the top two clauses.

Finally, since $\mathcal{E}$ is a robust set encoding it holds by assumption that the probability in Equation 14 is negligible and the theorem thus follows.

Corollary 12. *Instantiating the construction from Figure 2 using the robust set encoding from Section 3 with* $k = n = \lambda$ *and* $q = \sqrt{\lambda(2\ell+3)}$ *leads to a* $\frac{2tkn\log q}{\ell} = \frac{t\lambda^2\log(\lambda(2\ell+3))}{\ell}$ *compressing PPH for exact hamming distance.*

5 Lower Bound

In this section, we show a lower bound on the output length of a PPH for exact Hamming distance. We prove the lower bound by reduction from indexing. In the indexing problem, there are two parameters k and m. The first player Alice is given a string $x = (x_1, \ldots, x_m) \in [k]^m$, while the second player Bob is given an integer $i \in [m]$. Alice sends a single message to Bob and Bob should output x_i. The following lower bound holds:

Lemma 13 ([18]). *In any one-way protocol for indexing in the joint random source model with success probability at least* $1-\delta$ *over a uniform random string* x *and uniform random index* i*, Alice must send a message of size* $\Omega((1-\delta)m\log k - m)$ *in expectation.*

Here the joint random source model means that Alice and Bob have shared randomness that is drawn independently of their inputs. Note that we have strengthened the lemma a bit over the original result, to allow the failure probability to be "on average" over a uniform random index. The proof of the above lemma is very short using modern techniques:

Proof. Let $X = (X_1, \ldots, X_m)$ be a uniform random string over $[k]^m$ and let I be a uniform random index in $[m]$. Let R be a random variable giving the shared randomness between Alice and Bob (independent of their inputs) drawn from some universe $\mathcal{R}$ of finite bit strings. Let $\pi : [k]^m \times \mathcal{R} \to \{0,1\}^*$ give Alice's message in a protocol and let $\tau : \{0,1\}^* \times [m] \times \mathcal{R} \to [k]$ be Bob's decoding. That is, $\pi(X,R)$ is Alice's message and $\tau(\pi(X,R), I, R)$ is Bob's output. Assume $\Pr_{X,I,R}[\tau(\pi(X,R), I, R) = X_I] \geq 1-\delta$. For every $i \in [m]$, let $\delta_i = \Pr_{X,R}[\tau(\pi(X,R), i, R) \neq X_i]$. Then $\sum_{i=1}^m \delta_i/m \leq \delta$. Thus given Alice's message $\pi(X,R)$, Bob may reconstruct X_i except with probability δ_i by computing $\tau(\pi(X,R), i, R)$. By Fano's inequality, this implies that $H(X_i \mid \pi(X,R), R) \leq H_b(\delta_i) + \delta_i \log k \leq 1 + \delta_i \log k$ (here $H_b(\cdot)$ denotes binary entropy). Therefore, we have $H(X \mid \pi(X,R), R) \leq \sum_{i=1}^m 1 + \delta_i \log k \leq m + \delta m \log k$. But $H(X \mid R) = m \log k$. Thus $H(\pi(X,R)) \geq H(\pi(X,R) \mid R) \geq I(X; \pi(X,R) \mid R) = H(X \mid R) - H(X \mid R, \pi(X,R)) \geq (1-\delta)m\log k - m$. Since the entropy of a bit string is no more than its expected length, the lower bound follows.

Using the above lemma, we prove the following lower bound:

Theorem 14. *Any PPH for the exact Hamming distance predicate on* ℓ*-bit strings with threshold* t *and success probability at least* $1-\delta$ *(This means that the direct access robustness error is at most* δ*.), must have an output length of* $\Omega((1-\delta)(t-1)\log(\ell/t) - t)$ *bits.*

Proof. Assume that there exists a PPH-family $\mathcal{H}$ for the predicate HAM^t and input length ℓ with $t \leq \ell$ and direct robustness error at most δ. Let s denote the output length of $\mathcal{H}$. We then use $\mathcal{H}$ to solve indexing with parameters $k = \lfloor \ell/t \rfloor$ and $m = t - 1$. When Alice receives a string $x \in [k]^m$, she constructs a binary string y consisting of m chunks of k bits. If $mk < \ell$, she pads this string with 0's. Each chunk in y has a single 1 in position x_i and 0's elsewhere. She then computes the hash value $h(y)$, where h is sampled from $\mathcal{H}$ using joint randomness, and sends it to Bob, costing s bits.

From his index $i \in [m]$, Bob constructs k bit strings $z_1, \ldots, z_k$ of length ℓ, such that z_j has a 1 in the position corresponding to the j'th position of the i'th chunk of y, and 0 everywhere else. He then computes the hash values $h(z_1), \ldots, h(z_k)$ (using the joint randomness to sample h) and runs $\mathsf{Eval}(h, h(y), h(z_j))$. Bob outputs as his guess for x_i, an index j, such that $\mathsf{Eval}(h, h(y), h(z_j)) = 0$. Notice that the Hamming distance between z_j and y is $m + 1 \geq t$ if $j \neq x_i$ and it is $m - 1 < t$ otherwise. Thus if all k evaluations are correct, Bob succeeds in reporting x_i. The probability that all evaluations are correct is at least $1 - \delta$, since otherwise an adversary could break the direct access robustness of $\mathcal{H}$ with probability greater than δ by sampling x and i uniformly at random, simulating the above protocol, checking for which z_j the evaluation is correct and outputting y, z_j. Thus, Bob is correct with probability at least $1 - \delta$. By Lemma 13, we conclude $s = \Omega((1 - \delta)(t - 1)\log(\ell/t) - t)$.

Remark 2. We note that for $\delta = \mathsf{negl}(\lambda)$, $t > 2$, and $\ell > 4t$ the lower bound from Theorem 14 simplifies to $\Omega(t \log(\ell/t))$.

References

1. Aggarwal, D., Dadush, D., Regev, O., Stephens-Davidowitz, N.: Solving the shortest vector problem in 2^n time using discrete Gaussian sampling: Extended abstract. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th Annual ACM Symposium on Theory of Computing. pp. 733–742. ACM Press, Portland, OR, USA (Jun 14–17, 2015). https://doi.org/10.1145/2746539.2746606
2. Aggarwal, D., Li, J., Nguyen, P.Q., Stephens-Davidowitz, N.: Slide reduction, revisited - filling the gaps in SVP approximation. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 274–295. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56880-1_10
3. Aggarwal, D., Stephens-Davidowitz, N.: Just Take the Average! An Embarrassingly Simple 2^n-Time Algorithm for SVP (and CVP). In: Seidel, R. (ed.) 1st Symposium on Simplicity in Algorithms (SOSA 2018). OpenAccess Series in Informatics (OASIcs), vol. 61, pp. 12:1–12:19. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). https://doi.org/10.4230/OASIcs.SOSA.2018.12, http://drops.dagstuhl.de/opus/volltexte/2018/8306
4. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: 28th Annual ACM Symposium on Theory of Computing. pp. 99–108. ACM Press, Philadephia, PA, USA (May 22–24, 1996). https://doi.org/10.1145/237814.237838

5. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. In: 28th Annual ACM Symposium on Theory of Computing. pp. 20–29. ACM Press, Philadephia, PA, USA (May 22–24, 1996). https://doi.org/10.1145/237814.237823
6. Ben-Eliezer, O., Jayaram, R., Woodruff, D.P., Yogev, E.: A framework for adversarially robust streaming algorithms. In: Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. pp. 63–80 (2020)
7. Ben-Eliezer, O., Yogev, E.: The adversarial robustness of sampling. In: Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. pp. 49–62 (2020)
8. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM **13**(7), 422–426 (1970)
9. Boyle, E., LaVigne, R., Vaikuntanathan, V.: Adversarially robust property-preserving hash functions. In: Blum, A. (ed.) ITCS 2019: 10th Innovations in Theoretical Computer Science Conference. vol. 124, pp. 16:1–16:20. LIPIcs, San Diego, CA, USA (Jan 10–12, 2019). https://doi.org/10.4230/LIPIcs.ITCS.2019.16
10. Clayton, D., Patton, C., Shrimpton, T.: Probabilistic data structures in adversarial environments. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019: 26th Conference on Computer and Communications Security. pp. 1317–1334. ACM Press (Nov 11–15, 2019). https://doi.org/10.1145/3319535.3354235
11. Donoho, D.L.: Compressed sensing. IEEE Transactions on information theory **52**(4), 1289–1306 (2006)
12. Fleischhacker, N., Simkin, M.: Robust property-preserving hash functions for hamming distance and more. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021, Part III. Lecture Notes in Computer Science, vol. 12698, pp. 311–337. Springer, Heidelberg, Germany, Zagreb, Croatia (Oct 17–21, 2021). https://doi.org/10.1007/978-3-030-77883-5_11
13. Goodrich, M.T., Mitzenmacher, M.: Invertible bloom lookup tables. In: 2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton). pp. 792–799. IEEE (2011)
14. Hardt, M., Woodruff, D.P.: How robust are linear sketches to adaptive inputs? In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th Annual ACM Symposium on Theory of Computing. pp. 121–130. ACM Press, Palo Alto, CA, USA (Jun 1–4, 2013). https://doi.org/10.1145/2488608.2488624
15. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: 30th Annual ACM Symposium on Theory of Computing. pp. 604–613. ACM Press, Dallas, TX, USA (May 23–26, 1998). https://doi.org/10.1145/276698.276876
16. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**, 515–534 (1982)
17. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 21–39. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40041-4_2
18. Miltersen, P.B., Nisan, N., Safra, S., Wigderson, A.: On data structures and asymmetric communication complexity. Journal of Computer and System Sciences **57**(1), 37–49 (1998)
19. Mironov, I., Naor, M., Segev, G.: Sketching in adversarial environments. In: Ladner, R.E., Dwork, C. (eds.) 40th Annual ACM Symposium on Theory of Computing.

pp. 651–660. ACM Press, Victoria, BC, Canada (May 17–20, 2008). https://doi.org/10.1145/1374376.1374471

20. Muthukrishnan, S.: Data streams: algorithms and applications. In: 14th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 413–413. ACM-SIAM, Baltimore, MD, USA (Jan 12–14, 2003)
21. Naor, M., Yogev, E.: Bloom filters in adversarial environments. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part II. Lecture Notes in Computer Science, vol. 9216, pp. 565–584. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015). https://doi.org/10.1007/978-3-662-48000-7_28