Adversarially Robust Property-Preserving Hash Functions

Elette Boyle* IDC Herzliya
Rio Lavigne† MIT
Vinod Vaikuntanathan‡ MIT

November 27, 2018

Abstract

Property-preserving hashing is a method of compressing a large input $\mathbf{x}$ into a short hash $h(\mathbf{x})$ in such a way that given $h(\mathbf{x})$ and $h(\mathbf{y})$, one can compute a property $P(\mathbf{x}, \mathbf{y})$ of the original inputs. The idea of property-preserving hash functions underlies sketching, compressed sensing and locality-sensitive hashing.

Property-preserving hash functions are usually probabilistic: they use the random choice of a hash function from a family to achieve compression, and as a consequence, err on some inputs. Traditionally, the notion of correctness for these hash functions requires that for every two inputs $\mathbf{x}$ and $\mathbf{y}$, the probability that $h(\mathbf{x})$ and $h(\mathbf{y})$ mislead us into a wrong prediction of $P(\mathbf{x}, \mathbf{y})$ is negligible. As observed in many recent works (incl. Mironov, Naor and Segev, STOC 2008; Hardt and Woodruff, STOC 2013; Naor and Yogev, CRYPTO 2015), such a correctness guarantee assumes that the adversary (who produces the offending inputs) has no information about the hash function, and is too weak in many scenarios.

We initiate the study of *adversarial robustness* for property-preserving hash functions, provide definitions, derive broad lower bounds due to a simple connection with communication complexity, and show the necessity of computational assumptions to construct such functions. Our main positive results are two candidate constructions of property-preserving hash functions (achieving different parameters) for the (promise) gap-Hamming property which checks if $\mathbf{x}$ and $\mathbf{y}$ are "too far" or "too close". Our first construction relies on generic collision-resistant hash functions, and our second on a variant of the syndrome decoding assumption on low-density parity check codes.

*E-mail: eboyle@alum.mit.edu
†E-mail: rio@mit.edu
‡E-mail: vinodv@csail.mit.edu. Research supported in part by NSF Grants CNS-1350619, CNS-1718161 and CNS-1414119, an MIT-IBM grant, and a DARPA Young Faculty Award.

1 Introduction

The problem of *property-preserving hashing*, namely how to compress a large input in a way that preserves a class of its properties, is an important one in the modern age of massive data. In particular, the idea of property-preserving hashing underlies sketching [MP80, MG82, AMS96, CM05, CCF04], compressed sensing [CDS01], locality-sensitive hashing [IM98], and in a broad sense, much of machine learning.

As two concrete examples in theoretical computer science, consider universal hash functions [CW77] which can be used to test the equality of data points, and locality-sensitive hash functions [IM98, Ind00] which can be used to test the ℓ_p-distance between vectors. In both cases, we trade off accuracy in exchange for compression. For example, in the use of universal hash functions to test for equality of data points, one stores the hash $h(x)$ of a point x together with the description of the hash function h. Later, upon obtaining a point y, one computes $h(y)$ and checks if $h(y) = h(x)$. The pigeonhole principle tells us that mistakes are inevitable; all one can guarantee is that they happen with an acceptably small probability. More precisely, universal hash functions tell us that

$$\forall x \neq y \in D, \Pr[h \leftarrow \mathcal{H} : h(x) \neq h(y)] \geq 1 - \epsilon$$

for some small ϵ. A cryptographer's way of looking at such a statement is that it asks the adversary to pick x and y first; and evaluates her success w.r.t. a hash function chosen randomly from the family $\mathcal{H}$. In particular, the adversary has no information about the hash function when she comes up with the (potentially) offending inputs x and y. Locality-sensitive hash functions have a similar flavor of correctness guarantee.

The starting point of this work is that this definition of correctness is too weak in the face of adversaries with access to the hash function (either the description of the function itself or perhaps simply oracle access to its evaluation). Indeed, in the context of equality testing, we have by now developed several notions of robustness against such adversaries, in the form of pseudorandom functions (PRF) [GGM86], universal one-way hash functions (UOWHF) [NY89] and collision-resistant hash functions (CRHF). Our goal in this work is to expand the reach of these notions beyond testing equality; that is, our aim is *to do unto property-preserving hashing what CRHFs did to universal hashing.*

Several works have observed the deficiency of the universal hash-type definition in adversarial settings, including a wide range of recent attacks within machine learning in adversarial environments (e.g., [MMS+17, KW17, SND17, RSL18, KKG18]). Such findings motivate a rigorous approach to combatting adversarial behavior in these settings, a direction in which significantly less progress has been made. Mironov, Naor and Segev [MNS08] showed *interactive protocols* for sketching in such an adversarial environment; in contrast, we focus on non-interactive hash functions. Hardt and Woodruff [HW13] showed negative results which say that linear functions cannot be robust (even against computationally bounded adversaries) for certain natural ℓ_p distance properties; our work will use non-linearity and computational assumptions to overcome the [HW13] attack. Finally, Naor and Yogev [NY15] study adversarial Bloom filters which compress a set in a way that supports checking set membership; we will use their lower bound techniques in Section 5.

Motivating Robustness: Facial Recognition. In the context of facial recognition, authorities A and B store the captured images x of suspects. At various points in time, say authority A wishes to look up B's database for a suspect with face x. A can do so by comparing $h(x)$ with $h(y)$ for all y in B's database.

This application scenario motivated prior notions of fuzzy extractors and secure sketching. As with secure sketches and fuzzy extractors, a locality-sensitive property-preserving hash guarantees that close inputs (facial images) remain close when hashed [DORS08]; this ensures that small changes in ones appearance do not affect whether or not that person is authenticated. However, neither fuzzy extractors nor secure sketching guarantees that *far* inputs remain far when hashed. Consider an adversarial setting, not where a person wishes to evade detection, but where she wishes to be mistaken for someone else. Her face x' will undoubtably be different (far) from her target x, but there is nothing preventing her from slightly altering her face and passing as a completely different person when using a system with such a one-sided guarantee. This is where our notion of robustness comes in (as well as the need for cryptography): not only will adversarially chosen close x and x' map to close $h(x)$ and $h(x')$, but if adversarially chosen x and x' are *far*, they will be mapped to far outputs, unless the adversary is able to break a cryptographic assumption.

Comparison to Secure Sketches and Fuzzy Extractors. It is worth explicitly comparing fuzzy extractors and secure sketching to this primitive [DORS08], as they aim to achieve similar goals. Both of these seek to preserve the privacy of their inputs. Secure sketches generate random-looking sketches that hide information about the original input so that the original input can be reconstructed when given something close to it. Fuzzy extractors generate uniform-looking keys based off of fuzzy (biometric) data also using entropy: as long as the input has enough entropy, so will the output. As stated above, both guarantee that if inputs are close, they will 'sketch' or 'extract' to the same object. Now, the entropy of the sketch or key guarantees that randomly generated far inputs will not collide, but there are no guarantees about adversarially generated far inputs. To use the example above, it could be that once an adversary sees a sketch or representation, she can generate two far inputs that will reconstruct to the correct input.

Robust Property-Preserving Hash Functions. We put forth several notions of *robustness* for property-preserving hash (PPH) functions which capture adversaries with increasing power and access to the hash function. We then ask which properties admit robust property-preserving hash functions, and show positive and negative results.

- On the negative side, using a connection to communication complexity, we show that most properties and even simple ones such as set disjointness, inner product and greater-than do not admit non-trivial property-preserving hash functions.
- On the positive side, we provide two constructions of robust property-preserving hash functions (satisfying the strongest of our notions). The first is based on the standard cryptographic assumption of collision-resistant hash functions, and the second achieves more aggressive parameters under a new assumption related to the hardness of syndrome decoding on low density parity-check (LDPC) codes.
- Finally, we show that for essentially any non-trivial predicate (which we call collision-sensitive), achieving even a mild form of robustness requires cryptographic assumptions.

We proceed to describe our contributions in more detail.

1.1 Our Results and Techniques

We explore two notions of properties. The first is that of property classes $\mathcal{P} = \{P : D \to \{0,1\}\}$, sets of single-input predicates. This notion is the most general, and is the one in which we

prove lower bounds. The second is that of two-input properties $P : D \times D \to \{0,1\}$, which compares two inputs. This second notion is more similar to standard notions of universal hashing and collision-resistance, stronger than the first, and where we get our constructions. We note that a two-input predicate has an analogous predicate-class $\mathcal{P} = \{P_x\}_{x \in D}$, where $P_{x_1}(x_2) = P(x_1, x_2)$.

The notion of a property can be generalized in many ways, allowing for promise properties which output 0, 1 or $\circledast$ (a don't care symbol), and allowing for more than 2 inputs. The simplest notion of correctness for property-preserving hash functions requires that, analogously to universal hash functions,

$$\forall x, y \in D \Pr[h \leftarrow \mathcal{H} : \mathcal{H}.\mathsf{Eval}(h, h(x), h(y)) \neq P(x, y)] = \mathrm{negl}(\lambda)$$

or for single-input predicate-classes

$$\forall x \in D \text{ and } P \in \mathcal{P} \Pr[h \leftarrow \mathcal{H} : \mathcal{H}.\mathsf{Eval}(h, h(x), P) \neq P(x)] = \mathrm{negl}(\lambda)$$

where λ is a security parameter.

For the sake of simplicity in our overview, we will focus on two-input predicates.

Defining Robust Property-Preserving Hashing. We define several notions of *robustness* for PPH, each one stronger than the last. Here, we describe the strongest of all, called *direct-access PPH*.

In a direct-access PPH, the (polynomial-time) adversary is given the hash function and is asked to find a pair of bad inputs, namely $x, y \in D$ such that $\mathcal{H}.\mathsf{Eval}(h, h(x), h(y)) \neq P(x, y)$. That is, we require that

$$\forall \text{ p.p.t. } \mathcal{A}, \Pr[h \leftarrow \mathcal{H}; (x, y) \leftarrow \mathcal{A}(h) : \mathcal{H}.\mathsf{Eval}(h, h(x), h(y)) \neq P(x, y)] = \mathrm{negl}(\lambda).$$

The direct-access definition is the analog of collision-resistant hashing for general properties.

Our other definitions vary by how much access the adversary is given to the hash function, and are motivated by different application scenarios. From the strong to weak, these include double-oracle PPH where the adversary is given access to a hash oracle and a hash evaluation oracle, and evaluation-oracle PPH where the adversary is given only a combined oracle. Definitions similar to double-oracle PPH have been proposed in the context of adversarial bloom filters [NY15], and ones similar to evaluation-oracle PPH have been proposed in the context of showing attacks against property-preserving hash functions [HW13]. For more details, we refer the reader to Section 2.

Connections to Communication Complexity and Negative Results. Property-preserving hash functions for a property P, even without robustness, imply communication-efficient protocols for P in several models. For example, any PPH for P implies a protocol for P in the simultaneous messages model of Babai, Gal, Kimmel and Lokam [BGKL03] wherein Alice and Bob share a common random string h, and hold inputs x and y respectively. Their goal is to send a single message to Charlie who should be able to compute $P(x, y)$ except with small error. Similarly, another formalization of PPH that we present, called PPH for single-input predicate classes (see Section 2) implies efficient protocols in the one-way communication model [Yao79].

We use known lower bounds in these communication models to rule out PPHs for several interesting predicates (even without robustness). There are two major differences between the PPH setting and the communication setting, however: (a) in the PPH setting, we demand an

error that is negligible (in a security parameter); and (b) we are happy with protocols that communicate $n-1$ bits (or the equivalent bound in the case of promise properties) whereas the communication lower bounds typically come in the form of $\Omega(n)$ bits. In other words, the communication lower bounds *as-is* do not rule out PPH.

At first thought, one might be tempted to think that the negligible-error setting is the same as the deterministic setting where there are typically lower bounds of n (and not just $\Omega(n)$); however, this is not the case. For example, the equality function which has a negligible error public-coin simultaneous messages protocol (simply using universal hashing) with communication complexity $CC = O(\lambda)$ and deterministic protocols require $CC \geq n$. Thus, deterministic lower bounds do not (indeed, cannot) do the job, and we must better analyze the randomized lower bounds. Our refined analysis shows the following lower bounds:

- PPH for the Gap-Hamming (promise) predicate with a gap of $\sqrt{n}/2$ is impossible by refining the analysis of a proof by Jayram, Kumar and Sivakumar [JKS08]. The Gap-Hamming predicate takes two vectors in $\{0,1\}^n$ as input, outputs 1 if the vectors are very far, 0 if they are very close, and we do not care what it outputs for inputs in the middle.

- We provide a framework for proving PPHs are impossible for some total predicates, characterizing these classes as *reconstructing*. A predicate-class is *reconstructing* if, when only given oracle access to the predicates of a certain value x, we can efficiently determine x with all but negligible probability.[1] With this framework, we show that PPH for the Greater-Than (GT) function is impossible. It was known that GT required $\Omega(n)$ bits (for constant error) [RS15], but we show a lower bound of exactly n if we want negligible error. Index and Exact-Hamming are also reconstructing predicates.

- We also obtain a lower bound for a variant of GT: the (promise) Gap-k GT predicate which on inputs $x, y \in [N = 2^n]$, outputs 1 if $x - y > k$, 0 if $y - x > k$, and we do not care what it outputs for inputs in between. Here, exactly $n - \log(k) - 1$ bits are required for a perfect PPH. This is tight: we show that with fewer bits, one cannot even have a non-robust PPH, whereas there is a perfect robust PPH that compresses to $n - \log(k) - 1$ bits.

New Constructions. Our positive results are two constructions of a direct-access PPH for gap-Hamming for n-length vectors for large gaps of the form $\sim O(n/\log n)$ (as opposed to an $O(\sqrt{n})$-gap for which we have a lower bound). Let us recall the setting: the gap Hamming predicate P_{ham}, parameterized by n, d and ϵ, takes as input two n-bit vectors x and y, and outputs 1 if the Hamming distance between x and y is greater than $d(1+\epsilon)$, 0 if it is smaller than $d(1-\epsilon)$ and a don't care symbol $\circledast$ otherwise. To construct a direct-access PPH for this (promise) predicate, one has to construct a compressing family of functions $\mathcal{H}$ such that

$$\begin{aligned} \forall \text{ p.p.t. } \mathcal{A}, \Pr[h \leftarrow \mathcal{H}; (x,y) \leftarrow \mathcal{A}(h) : P_{\mathsf{ham}}(x,y) \neq \circledast \\ \wedge \mathcal{H}.\mathsf{Eval}(h, h(x), h(y)) \neq P_{\mathsf{ham}}(x,y)] = \mathrm{negl}(\lambda). \end{aligned} \tag{1}$$

Our two constructions offer different benefits. The first provides a clean general approach, and relies on the standard cryptographic assumption of collision-resistant hash functions. The second builds atop an existing one-way communication protocol, supports a smaller gap and better efficiency, and ultimately relies on a (new) variant of the syndrome decoding assumption on low-density parity check codes.

[1] In the single-predicate language of above, the predicate class corresponds to $\mathcal{P} = \{P(x,\cdot)\}$.

Construction 1. The core idea of the first construction is to reduce the goal of robust Hamming PPH to the simpler one of robust equality testing; or, in a word, "subsampling." The intuition is to notice that if $\mathbf{x}_1 \in \{0,1\}^n$ and $\mathbf{x}_2 \in \{0,1\}^n$ are *close*, then *most small enough subsets* of indices of $\mathbf{x}_1$ and $\mathbf{x}_2$ will match identically. On the other hand, if $\mathbf{x}_1$ and $\mathbf{x}_2$ are *far*, then most *large enough subsets* of indices will differ.

The hash function construction will thus fix a collection of sets $\mathcal{S} = \{S_1, \ldots, S_k\}$, where each $S_i \subseteq [n]$ is a subset of appropriately chosen size s. The desired structure can be achieved by defining the subsets S_i as the neighbor sets of a bipartite expander. On input $\mathbf{x} \in \{0,1\}^n$, the hash function will consider the vector $\mathbf{y} = (\mathbf{x}|_{S_1}, \ldots, \mathbf{x}|_{S_k})$ where $\mathbf{x}|_S$ denotes the substring of $\mathbf{x}$ indexed by the set S. The observation above tells us that if $\mathbf{x}_1$ and $\mathbf{x}_2$ are close (resp. far), then so are $\mathbf{y}_1$ and $\mathbf{y}_2$.

Up to now, it is not clear that progress has been made: indeed, the vector $\mathbf{y}$ is not compressing (in which case, why not stick with $\mathbf{x}_1, \mathbf{x}_2$ themselves?). However, $\mathbf{y}_1, \mathbf{y}_2$ satisfy the desired Hamming distance properties with fewer symbols over a *large alphabet*, $\{0,1\}^s$. As a final step, we can then leverage (standard) collision-resistant hash functions (CRHF) to compress these symbols. Namely, the final output of our hash function $h(\mathbf{x})$ will be the vector $(g(\mathbf{x}|_{S_1}), \ldots, g(\mathbf{x}|_{S_k}))$, where each substring of $\mathbf{x}$ is individually compressed by a CRHF g.

The analysis of the combined hash construction then follows cleanly via two steps. The (computational) collision-resistence property of g guarantees that any efficiently found pair of inputs $\mathbf{x}_1, \mathbf{x}_2$ will satisfy that their hash outputs

$$h(\mathbf{x}_1) = (g(\mathbf{x}_1|_{S_1}), \ldots, g(\mathbf{x}_1|_{S_k})) \text{ and } h(\mathbf{x}_2) = (g(\mathbf{x}_2|_{S_1}), \ldots, g(\mathbf{x}_2|_{S_k}))$$

are close if and only if it holds that

$$(\mathbf{x}_1|_{S_1}, \ldots, \mathbf{x}_1|_{S_k}) \text{ and } (\mathbf{x}_2|_{S_1}, \ldots, \mathbf{x}_2|_{S_k})$$

are close as well; that is, $\mathbf{x}_1|_{S_i} = \mathbf{x}_2|_{S_i}$ for most S_i. (Anything to the contrary would imply finding a collision in g.) Then, the combinatorial properties of the chosen index subsets S_i ensures (unconditionally) that any such inputs $\mathbf{x}_1, \mathbf{x}_2$ must themselves be close. The remainder of the work is to specify appropriate parameter regimes for which the CRHF can be used and the necessary bipartite expander graphs exist.

Construction 2. The starting point for our second construction is a simple non-robust hash function derived from a one-way communication protocol for gap-Hamming due to Kushilevitz, Ostrovsky, and Rabani [KOR98]. In a nutshell, the hash function is parameterized by a random sparse $m \times n$ matrix A with 1's in $1/d$ of its entries and 0's elsewhere; multiplying this matrix by a vector $\mathbf{z}$ "captures" information about the Hamming weight of $\mathbf{z}$. However, this can be seen to be trivially *not robust* when the hash function is given to the adversary. The adversary simply performs Gaussian elimination, discovering a "random collision" (x, y) in the function, where, with high probability $x \oplus y$ will have large Hamming weight. This already breaks equation (1).

The situation is somewhat worse. Even in a very weak, oracle sense, corresponding to our evaluation-oracle-robustness definition, a result of Hardt and Woodruff [HW13] shows that there are no *linear functions* h that are robust for the gap-ℓ_2 predicate. While their result does not carry over *as-is* to the setting of ℓ_0 (Hamming), we conjecture it does, leaving us with two options: (a) make the domain sparse: both the Gaussian elimination attack and the Hardt-Woodruff attack use the fact that Gaussian elimination is easy on the domain of the hash function; however making the domain sparse (say, the set of all strings of weight at most βn for some constant $\beta < 1$) already rules it out; and (b) make the hash function non-linear: again,

both attacks crucially exploit linearity. We will pursue both options, and as we will see, they are related.

But before we get there, let us ask whether we even need computational assumptions to get such a PPH. Can there be information-theoretic constructions? The first observation is that by a packing argument, if the output domain of the hash function has size less than $2^{n-n\cdot H(\frac{d(1+\epsilon)}{n})} \approx 2^{n-d\log n(1+\epsilon)}$ (for small d), there are bound to be "collisions", namely, two far points (at distance more than $d(1+\epsilon)$) that hash to the same point. So, you really cannot compress much information-theoretically, especially as d becomes smaller. A similar bound holds when restricting the domain to strings of Hamming weight at most βn for constant $\beta < 1$.

With that bit of information, let us proceed to describe in a very high level our construction and the computational assumption. Our construction follows the line of thinking of Applebaum, Haramaty, Ishai, Kushilevitz and Vaikuntanathan [AHI+17] where they used the hardness of syndrome decoding problems to construct collision-resistant hash functions. Indeed, in a single sentence, our observation is that their collision-resistant hash functions are *locality-sensitive* by virtue of being *input-local*, and thus give us a robust gap-Hamming PPH (albeit under a different assumption).

In slightly more detail, our first step is to simply take the construction of Kushilevitz, Ostrovsky, and Rabani [KOR98], and restrict the domain of the function. We show that finding two close points that get mapped to far points under the hash function is simply impossible (for our setting of parameters). On the other hand, there exist two far points that get mapped to close points under the hash functions (in fact, they even collide). Thus, showing that it is hard to find such points requires a computational assumption.

In a nutshell, our assumption says that given a random matrix $\mathbf{A}$ where each entry is chosen from the Bernoulli distribution with $\mathsf{Ber}(1/d)$ with parameter $1/d$, it is hard to find a large Hamming weight vector $\mathbf{x}$ where $\mathbf{Ax} \pmod 2$ has small Hamming weight. Of course, "large" and "small" here have to be parameterized correctly (see Section 4.2 for more details), however we observe that this is a generalization of the syndrome decoding assumption for low-density parity check (LDPC) codes, made by [AHI+17].

In our second step, we remove the sparsity requirement on the input domain of the predicate. We show a sparsification transformation which takes arbitrary n-bit vectors and outputs $n' > n$-bit *sparse* vectors such that (a) the transformation is injective, and (b) the expansion introduced here does not cancel out the effect of compression achieved by the linear transformation $\mathbf{x} \to \mathbf{Ax}$. This requires careful tuning of parameters for which we refer the reader to Section 4.2.

The Necessity of Cryptographic Assumptions. The goal of robust PPH is to compress beyond the information theoretic limits, to a regime where incorrect hash outputs exist but are hard to find. If robustness is required even when the hash function is given, this inherently necessitates cryptographic hardness assumptions. A natural question is whether weaker forms of robustness (where the adversary sees only oracle access to the hash function) similarly require cryptographic assumptions, and what types of assumptions are required to build non-trivial PPHs of various kinds.

As a final contribution, we identify necessary assumptions for PPH for a kind of predicate we call *collision sensitive*. In particular, PPH for any such predicate in the double-oracle model implies the existence of one-way functions, and in the direct-access model implies existence of collision-resistant hash functions. In a nutshell, collision-sensitive means that finding a collision in the predicate breaks the property-preserving nature of any hash. The proof uses and expands on techniques from the work of Naor and Yogev on adversarially robust Bloom Filters [NY15]. The basic idea is the same: without OWFs, we can invert arbitrary

polynomially-computable functions with high probability in polynomial time, and using this we get a representation of the hash function/set, which can be used to find offending inputs.

2 Defining Property-Preserving Hash Functions

Our definition of property preserving hash functions (PPHs) comes in several flavors, depending on whether we support total or partial predicates; whether the predicates take a single input or multiple inputs; and depending on the information available to the adversary. We discuss each of these choices in turn.

Total vs. Partial Predicates. We consider *total predicates* that assign a 0 or 1 output to each element in the domain, and *promise (or partial) predicates* that assign a 0 or 1 to a subset of the domain and a wildcard (don't-care) symbol $\circledast$ to the rest. More formally, a *total predicate* P on a domain X is a function $P : X \to \{0,1\}$, well-defined as 0 or 1 for every input $x \in X$. A *promise predicate* P on a domain X is a function $P : X \to \{0,1,\circledast\}$. Promise predicates can be used to describe scenarios (such as gap problems) where we only care about providing an exact answer on a subset of the domain.

Our definitions below will deal with the more general case of promise predicates, but we will discuss the distinction between the two notions when warranted.

Single-Input vs Multi-Input Predicates. In the case of single-input predicates, we consider a class of properties $\mathcal{P}$ and hash a single input x into $h(x)$ in a way that given $h(x)$, one can compute $P(x)$ for any $P \in \mathcal{P}$. Here, h is a compressing function. In the multi-input setting, we think of a single fixed property P that acts on a tuple of inputs, and require that given $h(x_1), h(x_2), \ldots, h(x_k)$, one can compute $P(x_1, x_2, \ldots, x_k)$. The second syntax is more expressive than the first, and so we use the multi-input syntax for constructions and the single-input syntax for lower bounds[2].

Before we proceed to discuss robustness, we provide a working definition for a property-preserving hash function for the single-input syntax. For the multi-input predicate definition and further discussion, see appendix A.

Definition 1. *A (non-robust) η-compressing Property Preserving Hash (η-PPH) family $\mathcal{H} = \{h : X \to Y\}$ for a function η and a class of predicates $\mathcal{P}$ requires the following two efficiently computable algorithms:*

- $\mathcal{H}.\mathsf{Samp}(1^\lambda) \to h$ *is a randomized p.p.t. algorithm that samples a random hash function from $\mathcal{H}$ with security parameter λ.*
- $\mathcal{H}.\mathsf{Eval}(h, P, y)$ *is a deterministic polynomial-time algorithm that on input the hash function h, a predicate $P \in \mathcal{P}$ and $y \in Y$ (presumably $h(x)$ for some $x \in X$), outputs a single bit.*

Additionally, $\mathcal{H}$ must satisfy the following two properties:

- η-compressing, *namely,* $\log |Y| \leq \eta(\log |X|)$, *and*

[2]There is yet a third possibility, namely where there is a *fixed* predicate P that acts on a single input x, and we require that given $h(x)$, one can compute $P(x)$. This makes sense when the computational complexity of h is considerably less than that of P, say when P is the parity function and h is an AC^0 circuit, as in the work of Dubrov and Ishai [DI06]. We do not explore this third syntax further in this work.

For security parameter λ, fixed predicate class $\mathcal{P}$, and h sampled from $\mathcal{H}.\mathsf{Samp}$	
Non-Robust PPH	Adversary has no access to hash function or evaluation.
Evaluation-Oracle PPH	Access to the evaluation oracle $\mathcal{O}_h^{\mathsf{Eval}}(x, P) = \mathcal{H}.\mathsf{Eval}(h, P, h(x))$.
Double-Oracle PPH	Access to both $\mathcal{O}_h^{\mathsf{Eval}}$ (as above) and hash oracle $\mathcal{O}_h^{\mathsf{Hash}}(x) = h(x)$.
Robust PPH "Direct Access"	Direct access to the hash function, description of h.

Figure 1: A table comparing the adversary's access to the hash function within different robustness levels of PPHs.

- robust, *according to one of four definitions that we describe below, leading to four notions of PPH: definition 2 (non-robust PPH), 3 (evaluation-oracle-robust PPH or EO-PPH), 5 (double-oracle-robust PPH or DO-PPH), or 7 (direct-access robust PPH or DA-PPH). We will refer to the strongest form, namely direct-access robust PPH as simply robust PPH when the intent is clear from the context. See also figure 1 for a direct comparison between these.*

The Many Types of Robustness. We will next describe four definitions of robustness for PPHs, starting from the weakest to the strongest. Each of these definitions, when plugged into the last bullet of Definition 1, gives rise to a different type of property-preserving hash function. In each of these definitions, we will describe an adversary whose goal is to produce an input and a predicate such that the hashed predicate evaluation disagrees with the truth. The difference between the definitions is in what an adversary has access to, summarized in figure 1.

2.1 Non-Robust PPH

We will start by defining the weakest notion of robustness which we call non-robust PPH. Here, the adversary has no information at all on the hash function h, and is required to produce a predicate P and a valid input x, namely where $P(x) \neq \circledast$, such that $\mathcal{H}.\mathsf{Eval}(h, P, x) \neq P(x)$ with noticeable probability. When $\mathcal{P}$ is the family of point functions (or equality functions), this coincides with the notion of 2-universal hash families [CW77][3].

Here and in the following, we use the notation $\Pr[A_1; \ldots; A_m : E]$ to denote the probability that event E occurs following an experiment defined by executing the sequence $A_1, \ldots, A_m$ in order.

Definition 2. *A family of PPH functions $\mathcal{H} = \{h : X \to Y\}$ for a class of predicates $\mathcal{P}$ is a family of* non-robust *PPH functions if for any $P \in \mathcal{P}$ and $x \in X$ such that for $P(x) \neq \circledast$,*

$$\Pr[h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda) : \mathcal{H}.\mathsf{Eval}(h, P, h(x))) \neq P(x)] \leq \mathrm{negl}(\lambda).$$

2.2 Evaluation-Oracle Robust PPH

In this model, the adversary has slightly more power than in the non-robust setting. Namely, she can adaptively query an oracle that has $h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda)$ in its head, on inputs $P \in \mathcal{P}$ and $x \in X$, and obtain as output the hashed evaluation result $\mathcal{H}.\mathsf{Eval}(h, P, h(x))$. Let $\mathcal{O}_h(x, P) = \mathcal{H}.\mathsf{Eval}(h, P, h(x))$.

[3]While 2-universal hashing corresponds with a two-input predicate testing equality, the single-input version ($\{P_{x_1}\}$ where $P_{x_1}(x_2) = (x_1 == x_2)$) is more general, and so it is what we focus on.

Definition 3. *A family of PPH functions $\mathcal{H} = \{h : X \to Y\}$ for a class of predicates $\mathcal{P}$ is a family of* evaluation-oracle robust (EO-robust) *PPH functions if, for any PPT adversary $\mathcal{A}$,*

$$\Pr[h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda);\ (x, P) \leftarrow \mathcal{A}^{\mathcal{O}_h}(1^\lambda) : \\ P(x) \neq \circledast \wedge \mathcal{H}.\mathsf{Eval}(h, P, h(x)) \neq P(x)] \leq \mathrm{negl}(\lambda).$$

The reader might wonder if this definition is very weak, and may ask if it follows just from the definition of a non-robust PPH family. In fact, for *total predicates*, we show that the two definitions are the same. At a high level, simply querying the evaluation oracle on (even adaptively chosen) inputs cannot reveal information about the hash function since with all but negligible probability, the answer from the oracle will be correct and thus simulatable without oracle access. The proof of the following lemma is in Appendix B.1.

Lemma 4. *Let $\mathcal{P}$ be a class of total predicates on X. A non-robust PPH $\mathcal{H}$ for $\mathcal{P}$ is also an Evaluation-Oracle robust PPH for $\mathcal{P}$ for the same domain X and same codomain Y.*

However, when dealing with *promise* predicates, an EO-robustness adversary has the ability to make queries that do not satisfy the promise, and could get information about the hash function, perhaps even reverse-engineering the entire hash function itself. Indeed, Hardt and Woodruff [HW13] show that there are no EO-robust *linear* hash functions for a certain promise-ℓ_p distance property; whereas, non-robust linear hash functions for these properties follow from the work of Indyk [IM98, Ind00].

2.3 Double-Oracle PPH

We continue our line of thought, giving the adversary more power. Namely, she has access to two oracles, both have a hash function $h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda)$ in their head. The hash oracle $\mathcal{O}_h^{\mathsf{Hash}}$, parameterized by $h \in \mathcal{H}$, outputs $h(x)$ on input $x \in X$. The predicate evaluation oracle $\mathcal{O}_h^{\mathsf{Eval}}$, also parameterized by $h \in \mathcal{H}$, takes as input $P \in \mathcal{P}$ and $y \in Y$ and outputs $\mathcal{H}.\mathsf{Eval}(h, P, y)$. When $\mathcal{P}$ is the family of point functions (or equality functions), this coincides with the notion of psuedo-random functions.

Definition 5. *A family of PPH functions $\mathcal{H} = \{h : X \to Y\}$ for a class of predicates $\mathcal{P}$ is a family of* double-oracle-robust PPH (DO-PPH) *functions if, for any PPT adversary $\mathcal{A}$,*

$$\Pr[h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda);\ (x, P) \leftarrow \mathcal{A}^{\mathcal{O}_h^{\mathsf{Hash}}, \mathcal{O}_h^{\mathsf{Eval}}}(1^\lambda) : \\ P(x) \neq \circledast \wedge \mathcal{H}.\mathsf{Eval}(h, P, h(x)) \neq P(x)] \leq \mathrm{negl}(\lambda).$$

We show that any evaluation-oracle-robust PPH can be converted into a double-oracle-robust PPH at the cost of a computational assumption, namely, one-way functions. In a nutshell, the observation is that the output of the hash function can be encrypted using a symmetric key that is stored as part of the hash description, and the evaluation proceeds by first decrypting.

Lemma 6. *Let $\mathcal{P}$ be a class of (total or partial) predicates on X. Assume that one-way functions exist. Then, any EO-robust PPH for $\mathcal{P}$ can be converted into a DO-robust PPH for $\mathcal{P}$.*

See appendix B.2 for the full proof.

2.4 Direct-Access Robust PPH

Finally, we define the strongest notion of robustness where the adversary is given the description of the hash function itself. When $\mathcal{P}$ is the family of point functions (or equality functions), this coincides with the notion of collision-resistant hash families.

Definition 7. *A family of PPH functions $\mathcal{H} = \{h : X \to Y\}$ for a class of predicates $\mathcal{P}$ is a family of* direct-access robust *PPH functions if, for any PPT adversary $\mathcal{A}$,*

$$\Pr[h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda);\ (x, P) \leftarrow \mathcal{A}(h) : \\ P(x) \neq \circledast \wedge \mathcal{H}.\mathsf{Eval}(h, P, h(x))) \neq P(x)] \leq \mathrm{negl}(\lambda).$$

We will henceforth focus on *direct-access-robust* property-preserving hash functions and refer to them simply as robust PPHs.

3 Property Preserving Hashing and Communication Complexity

In this section, we identify and examine a relationship between property-preserving hash families (in the single-input syntax) and protocols in the one-way communication (OWC) model. A OWC protocol is a protocol between two players, Alice and Bob, with the goal of evaluating a certain predicate on their inputs and with the restriction that only Alice can send messages to Bob.

Our first observation is that non-robust property-preserving hash functions and OWC protocols [Yao79] are equivalent except for two changes. First, PPHs require the parties to be computationally efficient, and second, PPHs also require protocols that incur error negligible in a security parameter. It is also worth noting that while we can reference lower-bounds in the OWC setting, these lower bounds are typically of the form $\Omega(n)$ and are not exact. On the other hand, in the PPH setting, we are happy with getting a single bit of compression, and so an $\Omega(n)$ lower bound still does not tell us whether or not a PPH is possible. So, while we can use previously known lower bounds for some well-studied OWC predicates, we need to refine them to be exactly n in the presence of negligible error. We also propose a framework (for total predicates) that yields exactly n lower bounds for INDEX$_n$, GREATERTHAN,and EXACTHAMMING.

3.1 PPH Lower Bounds from One-Way Communication Lower Bounds

In this section, we will review the definition of OWC, and show how OWC lower bounds imply PPH impossibility results.

Definition 8. *[Yao79, KNR95] A δ-error public-coin OWC protocol Π for a two-input predicate $P : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ consists of a space R of randomness, and two functions $g_a : X_1 \times R \to Y$ and $g_b : Y \times X_2 \times R \to \{0,1\}$ so that for all $x_1 \in X_1$ and $x_2 \in X_2$,*

$$\Pr[r \leftarrow R; y = g_a(x_1; r) :\ g_b(y, x_2; r) \neq P(x_1, x_2)] \leq \delta.$$

A δ-error public-coin OWC protocol Π for a class of predicates *$\mathcal{P} = \{P : \{0,1\}^n \to \{0,1\}\}$, is defined much the same as above, with a function $g_a : X \times R \to Y$, and another function $g_b : Y \times \mathcal{P} \to \{0,1\}$, which instead of taking a second input, takes a predicate from the predicate class. We say Π has δ-error if*

$$\Pr[r \leftarrow R; y = g_a(x; r) : g_b(y, P; r) \neq P(x)] \leq \delta$$

Let $\mathsf{Protocols}_\delta(P)$ denote the set of OWC protocols with error at most δ for a predicate P, and for every $\Pi \in \mathsf{Protocols}_\delta(P)$, let Y_Π be the range of messages Alice sends to Bob (the range of g_a) for protocol Π.

Definition 9. *The randomized, public-coin* OWC complexity *of a predicate P with error δ, denoted $R_\delta^{A\to B}(P)$, is the minimum over all $\Pi \in \mathsf{Protocols}_\delta(P)$ of $\lceil \log |Y_\Pi| \rceil$.*

For a predicate class $\mathcal{P}$, we define the randomized, public-coin OWC complexity with error δ, denoted $R_\delta^{A\to B}(\mathcal{P})$, is the minimum over all $\Pi \in \mathsf{Protocols}_\delta(\mathcal{P})$ of $\lceil \log |Y_\Pi| \rceil$.

A PPH scheme for a two-input predicate[4] P yields a OWC protocol for P with communication comparable to a single hash output size.

Theorem 10. *Let P be any two-input predicate P and $\mathcal{P} = \{P_x\}_{x\in\{0,1\}^n}$ be the corresponding predicate class where $P_{x_2}(x_1) = P(x_1, x_2)$. Now, let $\mathcal{H}$ be a PPH in any model for $\mathcal{P}$ that compresses n bits to $m = \eta n$. Then, there exists a OWC protocol Π such that the communication of Π is m and with negligible error.*

Conversely, the amount of possible compression of any (robust or not) PPH family $\mathcal{H} : \{h : X \to Y\}$ is lower bounded by $R_{\mathrm{negl}(\lambda)}^{A\to B}(P)$. Namely, $\log |Y| \geq R_{\mathrm{negl}(\lambda)}^{A\to B}(\mathcal{P})$.

Essentially, the OWC protocol is obtained by using the public common randomness r to sample a hash function $h = \mathcal{H}.\mathsf{Samp}(1^\lambda; r)$, and then Alice simply sends the hash $h(x_1)$ of her input to Bob. See Appendix C.1 for the proof.

3.2 OWC and PPH lower bounds for Reconstructing Predicates

We next leverage this connection together with OWC lower bounds to obtain impossibility results for PPHs. First, we will discuss the total predicate case; we consider some partial predicates in section 3.3.

As discussed, to demonstrate the impossibility of a PPH, one must give an explicit n-bit communication complexity lower bound (not just $\Omega(n)$) for negligible error. We give such lower bounds for an assortment of predicate classes by a general approach framework we refer to as reconstructing. Intuitively, a predicate class is *reconstructing* if, when given only access to predicates evaluated on an input x, one can, in polynomial time, determine the exact value of x with all but negligible probability.

Definition 11. *A class $\mathcal{P}$ of total predicates $P : \{0,1\}^n \to \{0,1\}$, is* reconstructing *if there exists a PPT algorithm L (a 'learner') such that for all $x \in \{0,1\}^n$, given randomness r and oracle access to predicates $\mathcal{P}$ on x, denoted $\mathcal{O}_x(P) = P(x)$,*

$$\Pr_r[L^{\mathcal{O}_x}(r) \to x] \geq 1 - \mathrm{negl}(n).$$

Theorem 12. *If $\mathcal{P}$ is a reconstructing class of predicates on input space $\{0,1\}^n$, then a PPH does not exist for $\mathcal{P}$.*

Proof. We will prove this by proving the following OWC lower bound:

$$R_{\mathrm{negl}(n)}^{A\to B}(\mathcal{P}) = n.$$

By Theorem 10, this implies a PPH cannot compress the input and still be correct.

[4]Or rather, for the induced class of single-input predicates $\mathcal{P} = \{P_{x_2}\}_{x_2\in\{0,1\}^n}$, where $P_{x_2}(x_1) = P(x_1, x_2)$; we will use these terminologies interchangeably.

We show that if Alice communicates any fewer than n bits to Bob, then there exists at least one pair of $(x, P) \in \{0,1\}^n \times \mathcal{P}$ such that the probability that the OWC protocol outputs $P(x)$ correctly is non-negligible. Our strategy is to generate pairs (x, P) over some distribution such that, for every fixed choice of randomness of the OWC protocol, the probability that the sampled (x, P) evaluates incorrectly will be $1/\mathrm{poly}(n)$. We first prove that such a distribution violates the negligible-error correctness of OWC.

A bad distribution violates correctness. Let $\mathcal{D}$ be the distribution producing (x, P), r_Π be the randomness of a OWC protocol Π, and $g_b(g_a(x), P) = g_b(g_a(x; r_\Pi), P; r_\Pi)$ for ease of notation. Suppose, for sake of contradiction, that our distribution had a non-negligible chance of producing an error (it is a "bad" distribution), but the correctness of the OWC protocol held. So, we have

$$\begin{aligned}\frac{1}{\mathrm{poly}} &= \Pr_{(x,P)\sim\mathcal{D},r_\Pi}[P(x) \neq g_b(g_a(x), P)] \\ &= \sum_{(x,P)} \Pr_{(x,P)}[\mathcal{D} = (x, P)] \Pr[P(x) \neq g_b(g_a(x), P) | (x, P) = \mathcal{D}],\end{aligned}$$

while the definition of negligible-error OWC protocol states that for every (x, P) pair, $\Pr[P(x) \neq g_b(g_a(x), P)] \leq \mathrm{negl}(n)$. If we plug in $\mathrm{negl}(n)$ for the value of $\Pr[P(x) \neq g_b(g_a(x), P) | (x, P) = \mathcal{D}]$, then we get

$$\Pr_{(x,P)\sim\mathcal{D},r_\Pi}[P(x) \neq g_b(g_a(x), P)] = \mathrm{negl}(n) \cdot \sum_{(x,P)} \Pr_{(x,P)}[\mathcal{D} = (x, P)] = \mathrm{negl}(n).$$

This is a contradiction, and therefore the existence of a distribution producing input-predicate pairs (x, P) that break the OWC protocol with $1/\mathrm{poly}$ probability, violates the (negligible error) correctness.

Generating a bad distribution. So, fix any randomness of the OWC protocol. We will now generate such a distribution $\mathcal{D}$, blind to the randomness of the protocol, that violates correctness of the protocol with $1/\mathrm{poly}(n)$ probability. Let L be the learner for $\mathcal{P}$. We generate this attack as follows:

1. $x \xleftarrow{\$} \{0,1\}^n$.
2. $r \xleftarrow{\$} \mathcal{U}_r$ (to fix the randomness for L).
3. Simulate $L(r)$, answering each query P to $\mathcal{O}_x$ correctly by computing $P(x)$, keeping a list $P_1, \ldots, P_t$ of each predicate query that was *not* answered with $\perp$.
4. $i \xleftarrow{\$} [t]$.
5. Output (x, P_i).

We will show that the probability this attack succeeds will be $\Omega(1/t)$, where t is the total number of queries L makes to $\mathcal{O}_x$. Since L is PPT, with all but negligible probability, $t = \mathrm{poly}(n)$, and therefore the attack succeeds with $1/\mathrm{poly}(n)$ probability.

Note that if Alice and Bob communicate fewer than n bits, for at least half of $x \in \{0,1\}^n$, there exists an x' that maps to the same communicated string: $g_a(x) = g_a(x')$. We analyze the attack success probability via a sequence of steps.

Chose x or x' in a pair. We will first compute the probability that we chose an x that was part of some pair hashing to the same string. Let $Pairs$ be a maximal set of non-overlapping pairs (x, x') that map to the same things. That is for (x, x') and (y, y') in $Pairs$, then none of x, x', y, y' can equal each other. The fraction of elements that show up in $Pairs$ is at least $1/4$. Therefore $\Pr_x[\text{choose } x \text{ or } x' \text{ in a pair}] \geq \frac{1}{4}$.

Chose x and x' that $L(r)$ reconstructs. Now assume that we have chosen either an x or x' in $Pairs$ (that is, fix x and x'). The probability that L distinguishes between x and x' is at least the probability that L correctly reconstructs *both* x and x'. Via a union bound, $\Pr_r[L^{\mathcal{O}_x}(r) = x \wedge L^{\mathcal{O}_{x'}}(r) = x'] \geq 1 - 2\text{negl}(n) = 1 - \text{negl}(n)$.

Chose i that distinguishes x and x'. Next, assume all previous points. Let $i^* \in [t]$ be the first query at which $P_{i^*}(x) \neq P_{i^*}(x')$. Because we fixed r, $L(r)$ now behaves deterministically, although adapts to query inputs, and so P_{i^*} will be the i^*'th query from L to both oracles $\mathcal{O}_x$ and $\mathcal{O}_{x'}$, and must be answered differently ($P_{i^*}(x) \neq P_{i^*}(x')$). Since $g_a(x) = g_a(x')$, we have that $g_b(g_a(x), P_{i^*}) = g_b(g_a(x'), P_{i^*})$ and so either $g_b(g_a(x), P_{i^*}) \neq P_{i^*}(x)$ or $g_b(g_a(x'), P_{i^*}) \neq P_{i^*}(x')$.

The probability we guess $i = i^*$ is $\frac{1}{t}$.

Chose the bad input from x or x'. Assuming all previous points in this list, we get that for one of x or x', the predicate P_i is evaluated incorrectly by g_b. Since we have assumed we chose one of x or x' (uniformly), the probability we chose the x or x' that evaluate incorrectly is $1/2$.

The probability the attack succeeds. Putting all of these points together, after fixing the hash function randomness (and sufficiently large n),

$$\Pr_L[g_b(g_a(x), P_i) \neq P_i(x)] \geq \frac{1}{4} \cdot (1 - \text{negl}(n)) \cdot \frac{1}{t} \cdot \frac{1}{2} \geq \frac{1}{10t}.$$

To recap: we have shown that for every randomness for a OWC protocol, we can produce an input and predicate such that the protocol fails with polynomial-chance. This implies that the OWC protocol does not have negligible error, and furthermore that no PPH can exist for such a predicate class. □

Reconstructing using INDEX$_n$, GREATERTHAN, or EXACTHAMMING

We turn to specific examples of predicate classes and sketch why they are reconstructing. For formal proofs, we refer the reader to Appendix C.2.

- The INDEX$_n$ class of predicates $\{P_1, \ldots, P_n\}$ is defined over $x \in \{0,1\}^n$ where $P_i(x) = x_i$, the i'th bit of x. INDEX$_n$ is reconstructing simply because the learner L can just query the each of the n indices of the input and exactly reconstruct: $x_i = P_i(x)$.
- The GREATERTHAN class of predicates $\{P_x\}_{x \in [2^n]}$ is defined over $x \in [2^n] = \{0,1\}^n$ where $P_{x_2}(x_1) = 1$ if $x_1 > x_2$ and 0 otherwise. GREATERTHAN is reconstructing because we can run a binary search on the input space, determining the exact value of x in n queries. GREATERTHAN is an excellent example for how an adaptive learner L can reconstruct.
- The EXACTHAMMING(α) class of predicates $\{P_x\}_{x \in \{0,1\}^n}$ is defined over $x \in \{0,1\}^n$ where $P_{x_2}(x_1) = 1$ if $||x_1 - x_2||_0 > \alpha$ and 0 otherwise. To show that EXACTHAMMING($n/2$) is reconstructing requires a little more work. The learner L resolves each

index of x independently. For each index, L makes polynomially many random-string queries $\mathbf{r}$ to $\mathcal{O}_x$; if the i'th bit of $\mathbf{r}$ equals x_i, then $\mathbf{r}$ is more likely to be within $n/2$ hamming distance of $\mathbf{x}$, and if the bits are different, $\mathbf{r}$ is more likely to not be within $n/2$ hamming distance of $\mathbf{x}$. The proof uses techniques from [JKS08], and is an example where the learner uses randomness to reconstruct.

We note that it was already known that INDEX$_n$ and EXACTHAMMING$(n/2)$ had OWC complexity of n-bits for any negligible error [KNR95], though no precise lower bound for randomized OWC protocols was known for GREATERTHAN. What is new here is our unified framework.

3.3 Lower bounds for some partial predicates

In the previous section, we showed how the ability to reconstruct an input using a class of total predicates implied that PPHs for the class cannot exist. This general framework, unfortunately, does not directly extend to the partial-predicate setting, since it is unclear how to define the behavior of an oracle for the predicate. Nevertheless, we can still take existing OWC lower bounds and their techniques to prove impossibility results in this case. We will show that GAPHAMMING$(n, n/2, 1/\sqrt{n})$ (the promise version of EXACTHAMMING) cannot admit a PPH, and that while Gap-k GREATERTHAN has a perfectly correct PPH compressing to $n - \log(k) - 1$ bits, compressing any further results in polynomial error (and thus no PPH with more compression).

First, we define these partial predicates.

Definition 13. *The definitions for* GAPHAMMING(n, d, ϵ) *and Gap-k* GREATERTHAN *are:*

- *The* GAPHAMMING(n, d, ϵ) *class of predicates* $\{P_x\}_{x\in\{0,1\}^n}$ *has* $P_{x_2}(x_1) = 1$ *if* $||x_1 - x_2||_0 \geq d(1+\epsilon)$, 0 *if* $||x_1 - x_2||_0 \leq d(1-\epsilon)$, *and* $\circledast$ *otherwise.*
- *The Gap-k* GREATERTHAN *class of predicates* $\{P_x\}_{x\in[2^n]}$ *has* $P_{x_2}(x_1) = 1$ *if* $x_1 > x_2 + k$, 0 *if* $x_1 < x_2 - k$, *and* $\circledast$ *otherwise.*

Now, we provide some intuition for why these lower bounds (and the upper bound) exist.

Gap-Hamming. Our lower bound will correspond to a refined OWC lower bound for the Gap-Hamming problem in the relevant parameter regime. Because we want to prove that we cannot even compress by a single bit, we need to be careful with our reduction: we want the specific parameters for which we have a lower bound, and we want to know just how the error changes in our reduction.

Theorem 14. *There does not exist a PPH for* GAPHAMMING$(n, n/2, 1/\sqrt{n})$.

To prove, we show the OWC complexity $R^{A\to B}_{\text{negl}(n)}$ (GAPHAMMING $(n, n/2, 1/\sqrt{n})) = n$. A $\Omega(n)$ OWC lower bound for Gap-Hamming in this regime has been proved in a few different ways [Woo04, Woo07, JKS08]. Our proof will be a refinement of [JKS08] and is detailed in appendix C.3.1.

The high-level structure of the proof is to reduce INDEX$_n$ to GAPHAMMING with the correct parameters. Very roughly, the ith coordinate of an input $x \in \{0,1\}^n$ can be inferred from the bias it induces on the Hamming distance between x and random public vectors. The reduction adds negligible error, but since we require n bits for negligible-error INDEX$_n$, we also require n bits for a OWC protocol for GAPHAMMING.

Notice that this style of proof looks morally as though we are "reconstructing" the input x using INDEX_n. However, the notion of getting a reduction from INDEX_n to another predicate-class in the OWC model is not the same as being able to query an oracle about the predicate and reconstruct based off of oracle queries. Being able to make a similar reconstructing characterization of partial-predicates as we have for total predicates would be useful and interesting in proving more lower bounds.

Gap-k GreaterThan. This predicate is a natural extension of GREATERTHAN: we only care about learning that $x_1 < x_2$ if $|x_1 - x_2|$ is larger than k (the gap). Intuitively, a hash function can maintain this information by simply removing the $\log(k)$ least significant bits from inputs and directly comparing: if $h(x_1) = h(x_2)$, they can be at most k apart. We can further remove one additional bit using the fact that we know x_2 when given $h(x_1)$ (considering Gap-k GreaterThan as the corresponding predicate class parameterized by x_2).

For the lower bound, we prove a OWC lower bound, showing $R^{A\to B}_{\text{negl}(n)}(\mathcal{P}) = n - \log(k) - 1$. This will be a proof by contradiction: if we compress to $n - \log(k) - 2$ bits, we obtain many collisions that are more than $3.5k$ apart. These far collisions imply the existence of inputs that the OWC protocol must fail on, even given the gap. We are able to find these inputs the OWC must fail on with polynomial probability, and this breaks the all-but-negligible correctness of the protocol. Our formal theorem statement is below.

Theorem 15. *There exists a PPH with perfect correctness for Gap-k GREATERTHAN compressing from n bits to $n - \log(k) - 1$. This is tight: no PPH for Gap-k GREATERTHAN can compress to fewer than $n - \log(k) - 1$ bits.*

For the proof, see appendix C.3.2.

4 Two Constructions for Gap-Hamming PPHs

In this section, we present two constructions of PPHs for GAPHAMMING. Recall from section 3.3 that the gap-Hamming property $P = \text{GAPHAMMING}(n, d, \epsilon)$ is parameterized by the input domain $\{0,1\}^n$, an integer $d \in [n]$ and a parameter $\epsilon \in \mathbb{R}^{\geq 0}$, so that $P(\mathbf{x}_1, \mathbf{x}_2) = 1$ if $||\mathbf{x}_1 \oplus \mathbf{x}_2||_0 \geq d(1+\epsilon)$ and 0 if $||\mathbf{x}_1 \oplus \mathbf{x}_2||_0 \leq d(1-\epsilon)$. Both of our constructions will distinguish between $d(1-\epsilon)$-close and $d(1+\epsilon)$-far vectors for $d \approx O(n/\log n)$. This means that the gap is quite large, approximately $O(n/\log n)$.

Our two constructions offer different benefits. The first provides a clean general approach, and relies on the standard cryptographic assumption of collision-resistant hash functions. The second takes a different approach, building on a one-way communication protocol for GAPHAMMING of [KOR98]. The latter construction supports a smaller gap and better efficiency, and ultimately relies on a (new) variant of the syndrome decoding assumption on low-density parity check codes. Both our constructions are two-input PPH construction, which are more general than single-input constructions that support a predicate class (see Lemma 33).

4.1 A Gap-Hamming PPH from Collision Resistance

Our first construction is a robust m/n-compressing $\text{GAPHAMMING}(n, d, \epsilon)$ PPH for any $m = n^{\Omega(1)}$, $d = o(n/\log\lambda)$ and any constant $\epsilon > 0$. Security of the construction holds under the (standard) assumption that collision-resistant hash function families (CRHFs) exist.

We now informally describe the idea of the construction which, in one word, is "subsampling". In slightly more detail, the intuition is to notice that if $\mathbf{x}_1 \in \{0,1\}^n$ and $\mathbf{x}_2 \in \{0,1\}^n$

are *close*, then *most small enough subsets* of indices of $\mathbf{x}_1$ and $\mathbf{x}_2$ will match identically. On the other hand, if $\mathbf{x}_1$ and $\mathbf{x}_2$ are *far*, then most *large enough subsets* of indices will differ. This leads us to the first idea for the construction, namely, fix a collection of sets $\mathcal{S} = \{S_1, \ldots, S_k\}$ where each $S_i \subseteq [n]$ is a subset of appropriately chosen size s. On input $\mathbf{x} \in \{0,1\}^n$, output $\mathbf{y} = (\mathbf{x}|_{S_1}, \ldots, \mathbf{x}|_{S_k})$ where $\mathbf{x}|_S$ denotes the substring of $\mathbf{x}$ indexed by the set S. The observation above tells us that if $\mathbf{x}_1$ and $\mathbf{x}_2$ are close (resp. far), so are $\mathbf{y}_1$ and $\mathbf{y}_2$.

However, this does not compress the vector $\mathbf{x}$. Since the union of all the sets $\bigcup_{i\in[k]} S_i$ has to be the universe $[n]$ (or else, finding a collision is easy), it turns out that we are just comparing the vectors index-by-index. Fortunately, it is not necessary to output $\mathbf{x}|_{S_i}$ by themselves; rather we can simply output the collision-resistant hashes. That is, we will let the PPH hash of $\mathbf{x}$, denoted $\mathbf{y}$, be $(g(\mathbf{x}|_{S_1}), \ldots, g(\mathbf{x}|_{S_k}))$ where g is a collision resistant hash function randomly drawn from a CRHF family.

This simple construction works as long as s, the size of the sets S_i, is $\Theta(n/d)$, and the collection $\mathcal{S}$ satisfies that any subset of disagreeing input indices $T \subseteq [n]$ has nonempty intersection with roughly the corresponding fraction of subsets S_i. The latter can be achieved by selecting the S_i of size $\Theta(n/d)$ at random, or alternatively as defined by the neighbor sets of a bipartite expander. We are additionally constrained by the fact that the CRHF must be secure against adversaries running in time $\mathrm{poly}(\lambda)$. So, let $t = t(\lambda)$ be the smallest output size of the CRHF such that it is $\mathrm{poly}(\lambda)$-secure. Since the input size s to the CRHF must be $\omega(t)$ so that g actually compresses, this forces $d = o(n/t)$.

Before presenting our construction more formally, we define our tools.

- We will use a family of CRHFs that take inputs of variable size and produce outputs of t bits and denote it by $\mathcal{H}_t = \{h : \{0,1\}^* \to \{0,1\}^t\}$. We implicitly assume a procedure for sampling a seed for the CRHF given a security parameter 1^λ. One could set $t = \omega(\log \lambda)$ and assume the exponential hardness of the CRHF, or set $t = \lambda^{O(1)}$ and assume polynomial hardness. These choices will result in different parameters of the PPH hash function.

- We will use an $(n, k, D, \gamma, \alpha)$-*bipartite expander* $G = (L \cup R, E)$ which is a D-left-regular bipartite graph, with $|L| = n$ and $|R| = k$ such that for every $S \subset L$ for which $|S| \leq \gamma n$, we have $|N(S)| \geq \alpha|S|$, where $N(S)$ is the set of neighbors of S. For technical reasons, we will need the expander to be δ-*balanced* on the right, meaning that for every $v \in R$, $|N(v)| \geq (1-\delta)nD/k$.

  A simple probabilistic construction shows that for every $n \in \mathbb{N}$, $k = o(n)$ and constant $a \in (0,1)$, and any $\gamma = o(\frac{k}{n\log(n/k)})$ and $D = \Theta(\log(1/\gamma))$ so that for every $\delta > 0$, δ-balanced $(n, k, D, \gamma, \alpha)$-bipartite expanders exist. In fact, there are even explicit efficient constructions that match these parameters [CRVW02]. The formal lemma statement and proof are given in Appendix D.1.

We next describe the general construction, and then discuss explicit parameter settings and state our formal theorem.

Setting the Parameters. The parameters required for this construction to be secure and constructible are as follows.

- Let $n \in \mathbb{N}$ and constant $\epsilon > 0$.

- We require two building blocks: a CRHF and an expander. So, let $\mathcal{H}_t = \{g : \{0,1\}^* \to \{0,1\}^t\}$ be a family of CRHFs secure against $\mathrm{poly}(\lambda)$-time adversaries. Let G be a δ-balanced $(n, k, D, \gamma, \alpha)$-expander for a constant δ bounding the degree of the right-nodes,

Robust GAPHAMMING(n, d, ϵ) PPH family $\mathcal{H}$ from any CRHF

Our (n, m, d, ϵ)-robust PPH family $\mathcal{H} = (\mathcal{H}.\mathsf{Samp}, \mathcal{H}.\mathsf{Eval})$ is defined as follows.

- $\mathcal{H}.\mathsf{Samp}(1^\lambda, n)$. Fix a δ-balanced $(n, k, D, \gamma, \alpha)$-bipartite expander $G = (L \cup R, E)$ (either deterministically or probabilistically). Sample a CRHF $g \leftarrow \mathcal{H}_t$. Output $h = (G, g)$.
- $\mathcal{H}.\mathsf{Hash}(h = (G, g), \mathbf{x})$. For every $i \in [k]$, compute the (ordered) set of neighbors of the i-th right vertex in G, denoted $N(i)$. Let $\hat{\mathbf{x}}^{(i)} := \mathbf{x}|_{N(i)}$ be $\mathbf{x}$ restricted to the set $N(i)$. Output
$$h(\mathbf{x}) := \left(g(\hat{\mathbf{x}}^{(1)}), \ldots, g(\hat{\mathbf{x}}^{(k)})\right)$$
as the hash of $\mathbf{x}$.
- $\mathcal{H}.\mathsf{Eval}(h = (G, g), \mathbf{y}_1, \mathbf{y}_2)$. Compute the threshold $\tau = D \cdot d \cdot (1 - \epsilon)$. Parse $\mathbf{y}_1 = (\hat{\mathbf{y}}_1^{(1)}, \ldots, \hat{\mathbf{y}}_1^{(k)})$ and $\mathbf{y}_2 = (\hat{\mathbf{y}}_2^{(1)}, \ldots, \hat{\mathbf{y}}_2^{(k)})$. Compute
$$\Delta' = \sum_{i=1}^{k} \mathbb{1}(\hat{\mathbf{y}}_1^{(i)} \neq \hat{\mathbf{y}}_2^{(i)}),$$
where $\mathbb{1}$ denotes the indicator predicate. If $\Delta' \leq \tau$, output ***CLOSE***. Otherwise, output ***FAR***.

Table 1: Construction of a robust GAPHAMMING(n, d, ϵ) PPH family from CRHFs.

where n is the size of the left side of the graph, k is the size of the right, D is the left-degree, γn is the upper bound for an expanding set on the right that expands to a set of size α times the original size.

- These building blocks yield the following parameters for the construction: compression is $\eta = kt/n$ and our center for the Gap-Hamming problem is bounded by $\frac{\gamma n}{(1+\epsilon)} \leq d < \frac{k}{D(1-\epsilon)}$.

If we consider what parameter settings yield secure CRHFs with output size t and for what parameters we have expanders, we have a PPH construction where given any n and ϵ, there exists a $d = o(n)$ and $\eta = O(1)$ such that Construction 1 is a robust PPH for gap-Hamming. We will see that the smaller t is, the stronger the CRHF security assumption, but the better our compression.

Now, given these settings of parameters, we will formally prove Construction 1 is a robust Gap-Hamming PPH.

Theorem 16. *Let λ be a security parameter. Assuming that exponentially secure CRHFs exist, for any polynomial $n = n(\lambda)$, and any constants $\epsilon, \eta > 0$, Construction 1 is an η-compressing robust property preserving hash family for $\text{GAPHAMMING}(n, d, \epsilon)$ where $d = o(n/\log\lambda \log\log\lambda)$. Assuming only that polynomially secure CRHFs exist, for any constant $c > 0$, we achieve $d = o(n/\lambda^c)$.*

Proof. Before getting into the proof, we more explicitly define the parameters ti include parameters associated with the expander in our construction. Once we have defined these parameters, we will show how each of these parameters is used to prove the construction is correct and robust. So, in total, we have the following paramters and implied constraints for our construction (including the graph):

1. Let $n \in \mathbb{N}$ be the input size and let $\epsilon > 0$ be any constant.
2. Our CRHF is $\mathcal{H}_t = \{g : \{0,1\}^* \to \{0,1\}^t\}$.
3. Our expander will be a δ-balanced $(n, k, D, \gamma, \alpha)$-expander, where $k < n/t$, $\gamma = o(\frac{k}{n\log(n/k)})$, $D = \Theta(\log(1/\gamma))$, and $\alpha > D \cdot \frac{d(1-\epsilon)}{\gamma n}$.
4. Our center for the gap-hamming problem is d, and is constrained by $\frac{\gamma n}{1+\epsilon} \leq d < \frac{k}{D(1-\epsilon)}$.
5. Constraint 4 implies that $k = n^{\Omega(1)}$, since $\frac{\gamma n \cdot D(1-\epsilon)}{1+\epsilon} < k$.

Now, we prove our construction is well-defined and efficient. Fix any $\delta, a \in (0,1)$. Using lemma 47, proved in the appendix, we know that δ-balanced $(n, k, D, \gamma, \alpha = an)$-bipartite expanders exist and can be efficiently sampled for $k = o(n/\log n)$, $D = \Theta(\log\log n)$, and $\gamma = \tilde{\Theta}(1/\log n)$. Thus, sampling G before running the construction is efficient. Once we have a G, sampling and running a CRHF $k = O(n)$ times is efficient. Comparing k outputs of the hash function is also efficient. Therefore, each of $\mathcal{H}$.Samp, $\mathcal{H}$.Hash, and $\mathcal{H}$.Eval is efficient in $\lambda = \text{poly}(n)$.

Now, we prove that Construction 1 is compressing. Points 2 and 3 mean that $m = k \cdot t < n/t \cdot t = n$, as required.

Lastly, we will prove our construction is robust. Let $\mathcal{A}$ be a PPT adversary. We will show that $\mathcal{A}$ (in fact, even an unbounded adversary) cannot find $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $||\mathbf{x}_1 - \mathbf{x}_2|| \leq d(1-\epsilon)$ but $\mathcal{H}.\text{Eval}(h, h(\mathbf{x}_1), h(\mathbf{x}_2))$ evaluates to ***FAR***, and that $\mathcal{A}$ must break the collision-resistance of $\mathcal{H}_t$ in order to find $\mathbf{x}_1$ and $\mathbf{x}_2$ where $||\mathbf{x}_1 - \mathbf{x}_2|| \geq d(1+\epsilon)$ but $\mathcal{H}.\text{Eval}(h, h(\mathbf{x}_1), h(\mathbf{x}_2))$ evaluates to ***CLOSE***.

- First, consider any $\mathbf{x}_1, \mathbf{x}_2 \in \{0,1\}^n$ where $||\mathbf{x}_1 - \mathbf{x}_2||_0 \leq d(1-\epsilon)$. Let $\Delta = ||\mathbf{x}_1 - \mathbf{x}_2||_0$. So, consider the set $S \subset L$ corresponding to the indices that are different between $\mathbf{x}_1$ and $\mathbf{x}_2$, and $T = N(S) \subset R$. The maximum size of T is $|S| \cdot D$, the degree of the graph.

  For every $i \in T$, we get that the intermediate computation has $\hat{\mathbf{x}}_1^{(i)} \neq \hat{\mathbf{x}}_2^{(i)}$, but for every $j \notin T$, we have $\hat{\mathbf{x}}_1^{(j)} = \hat{\mathbf{x}}_2^{(j)}$ which implies $\hat{\mathbf{y}}_1^{(j)} = \hat{\mathbf{y}}_2^{(j)}$ after applying g. Therefore $\sum_{i=1}^{k} \mathbb{1}(\hat{\mathbf{y}}_1^{(i)} \neq \hat{\mathbf{y}}_2^{(i)}) \leq \sum_{i \in S} \mathbb{1}(\hat{\mathbf{y}}_1^{(i)} \neq \hat{\mathbf{y}}_2^{(i)}) + \sum_{j \notin S} \mathbb{1}(\hat{\mathbf{y}}_1^{(j)} \neq \hat{\mathbf{y}}_2^{(j)}) \leq \Delta \cdot D$.
  We set the threshold $\tau = D \cdot d \cdot (1-\epsilon)$ in the evaluation. Point 4 guarantees that $\tau < k$ (and implicitly implies $k > D(1-\epsilon)$), so because $D \cdot \Delta \leq D \cdot d(1-\epsilon) = \tau < k$, $\mathcal{H}$.Eval will evaluate $\Delta' \leq \tau$. Thus, $\mathcal{H}$.Eval will always evaluate to ***CLOSE*** in this case, regardless of the choice of CRHF.

- Now consider $||\mathbf{x}_1 - \mathbf{x}_2||_0 \geq d(1+\epsilon)$, and again, let $\Delta = ||\mathbf{x}_1 - \mathbf{x}_2||_0$ and define $S \subset L$ and $T \subset R$ as before.

  By point 4 again ($\gamma n \leq d(1+\epsilon)$), we can restrict S to S' where $|S'| = \gamma n$, and by the properties of expanders $|N(S')| \geq \gamma n \cdot \alpha$. Now, point 3 guarantees that $\tau = D \cdot d \cdot (1-\epsilon) < \alpha \cdot \gamma n$. So, for every $i \in T'$, $\hat{\mathbf{x}}_1^{(i)} \neq \hat{\mathbf{x}}_2^{(i)}$, and $|T'| \geq \alpha \cdot \gamma n > \tau$. Now we want to argue that with all but negligible probability over our choice of g, g will preserve this equality relation, and so $\Delta' = |T'|$. Given that our expander is δ-balanced for some constant $\delta > 0$, we have that $|\hat{\mathbf{x}}_1^{(i)}| = |\hat{\mathbf{x}}_2^{(i)}| = |N(r_i)| \geq (1-\delta)nD/k$. Now, point 3 states that the constraints have $k < n/t$, implying $n/k > t$. So, $(1-\delta)D \cdot n/k > (1-\delta)D \cdot t$.

  This means that every input to g will be larger than the output ($(1-\delta)$ is a constant and $D = \omega(1)$), and so if $g(\hat{\mathbf{x}}_1^{(i)}) = g(\hat{\mathbf{x}}_2^{(i)})$ but $\hat{\mathbf{x}}_1^{(i)} \neq \hat{\mathbf{x}}_2^{(i)}$ for any i, then our adversary has found a collision, which happens with all but negligible probability for adversaries running in time $\mathrm{poly}(\lambda)$.

  Therefore, with all but negligible probability over the choice of g and adversarially chosen $\mathbf{x}_1$ and $\mathbf{x}_2$ in this case, $\Delta' = \sum_{i=1}^{m'} \mathbb{1}(\hat{\mathbf{y}}_1^{(i)} \neq \hat{\mathbf{y}}_2^{(i)}) \geq \alpha \cdot \gamma n = \tau$, and $\mathcal{H}$.Eval outputs ***FAR***.

□

4.2 A Gap-Hamming PPH from Sparse Short Vectors

In this section, we present our second family of robust property-preserving hash (PPH) functions for gap Hamming distance. The construction proceeds in three steps: in Section 4.2.1, we start with an (unconditionally) secure non-robust PPH; in Section 4.2.2, we build on this to construct a robust PPH with a restricted input domain; and finally, in Section 4.2.3, we show how to remove the restriction on the input domain.

The construction is the same as the collision-resistant hash function construction in the work of [AHI+17]. In a single sentence, our observation is that their *input-local* hash functions are *locality-sensitive* and thus give us a robust gap-Hamming PPH (albeit under a different assumption). We proceed to describe the construction in full for completeness.

4.2.1 Non-Robust Gap-Hamming PPH

We first describe our starting point, a non-robust PPH for GapHamming, derived from the locality sensitive hash of Kushilevitz, Ostrovsky, and Rabani [KOR98]. In a nutshell, the hash function is parameterized by a random sparse $m \times n$ matrix $\mathbf{A}$ with 1s in a $1/d$ fraction of its entries and 0s elsewhere; multiplying this matrix by a vector $\mathbf{z}$ "captures" some information

Non-robust $\text{GAPHAMMING}(n, d, \epsilon)$ PPH family $\mathcal{H}$

- $\mathcal{H}.\mathsf{Samp}(1^\lambda, 1^n)$. Pick a constant c appropriately such that $m := \frac{c\lambda}{\epsilon^2} < n$. Let

$$\mu_1 = \frac{m}{2}(1 - e^{-2(1-\epsilon)}); \;\; \mu_2 = \frac{m}{2}(1 - e^{-2(1+\epsilon)}) \text{ and } \tau = (\mu_1 + \mu_2)/2$$

  Generate an $m \times n$ matrix $\mathbf{A}$ by choosing each entry from the Bernoulli distribution $\mathsf{Ber}(1/d)$. Output $(\mathbf{A}, \tau)$ as the description of the hash function.

- $\mathcal{H}.\mathsf{Hash}((\mathbf{A}, \tau), \mathbf{x})$. Output $\mathbf{Ax} \in \mathbb{Z}_2^m$.

- $\mathcal{H}.\mathsf{Eval}((\mathbf{A}, \tau), \mathbf{y}_1, \mathbf{y}_2)$. If $||\mathbf{y}_1 \oplus \mathbf{y}_2||_0 \leq \tau$, output ***CLOSE***, otherwise output ***FAR***.

Table 2: Construction of a non-robust $\text{GAPHAMMING}(n, d, \epsilon)$ PPH family.

about the Hamming weight of $\mathbf{z}$; in particular, it distinguishes between the cases that the Hamming weight is much larger than d versus much smaller. Furthermore, since this hash function is linear, it can be used to compress two inputs $\mathbf{x}$ and $\mathbf{y}$ independently and later compute their Hamming *distance*. The construction is described in Table 2.

Lemma 17 ([KOR98]). *Let λ be a security parameter. For every $n \in \mathbb{N}$, $d \in [n]$ and $\epsilon = \Omega(\sqrt{\lambda/n})$, Construction 2 is a non-robust PPH for $\text{GAPHAMMING}(n, d, \epsilon)$.*

Proof. First, we note that $\mathcal{H}$ is compressing. We have $m = \frac{c\lambda}{\epsilon^2}$ and $\epsilon = \Omega(\sqrt{\lambda/n})$. Therefore, if we have chose c appropriately, there exists another constant $c' < 1$ such that $m \leq c'n$. Compression is why we require a lower bound on epsilon.

Now, we show that $\mathcal{H}$ satisfies the non-robust notion of correctness. For any $\mathbf{x}_1$ and $\mathbf{x}_2 \in \{0,1\}^n$, let $\mathbf{z} = \mathbf{x}_1 \oplus \mathbf{x}_2$. $\mathcal{H}.\mathsf{Eval}(\mathbf{Ax}_1, \mathbf{Ax}_2)$ tests if $||\mathbf{Az}||_0 \leq \tau$. We will show that for all $\mathbf{z}$, with all but negligible probability over our choice of $\mathbf{A}$, this threshold test will evaluate correctly.

To do this, we will invoke the (information theoretic) XOR lemma. That is, if $||\mathbf{z}||_0 = k$, then for $\mathbf{a}_i \xleftarrow{\$} \mathsf{Ber}(p)^n$, $\Pr[\mathbf{a}_i \cdot \mathbf{z} = 1] = \frac{1}{2}(1 - (1-2p)^k)$. Our hash function has $p = 1/d$, so now, we will apply this to our two cases for $\mathbf{z}$:

- $||\mathbf{z}||_0 \leq d(1-\epsilon)$. We have that $\Pr[\mathbf{a}_i \cdot \mathbf{z} = 1] = \frac{1}{2}(1 - (1 - \frac{2}{d})^{d(1-\epsilon)}) \sim \frac{1}{2}(1 - \frac{1}{e^{2(1-\epsilon)}})$ by the XOR lemma. We get that for all $\mathbf{z}$ such that $||\mathbf{z}||_0 \leq d(1-\epsilon)$,

$$\mathop{\mathbb{E}}_{\mathbf{A}}[||\mathbf{Az}||_0] = m \cdot \Pr_{\mathbf{a}_i}[\mathbf{a}_i \cdot \mathbf{z} = 1] \leq \frac{m}{2}(1 - \frac{1}{e^{2(1-\epsilon)}}) := \mu_1$$

- $||\mathbf{z}||_0 \geq d(1+\epsilon)$. Again, using the XOR lemma, and plugging in $k = d(1+\epsilon)$, we have that,

$$\mathop{\mathbb{E}}_{\mathbf{A}}[||\mathbf{Az}||_0] = m \cdot \Pr_{\mathbf{a}_i}[\mathbf{a}_i \cdot \mathbf{z} = 1] \geq \frac{m}{2}(1 - \frac{1}{e^{2(1+\epsilon)}}) := \mu_2$$

Now, a routine calculation using Chernoff bounds shows that with overwhelming probability over our choice of $\mathbf{A}$, we do not miscategorize the Hamming weight of $\mathbf{z}$. We will go through both cases again.

- $||\mathbf{z}||_0 \leq d(1-\epsilon)$. Recall that $\mu_1 = \frac{m}{2}(1 - \frac{1}{e^{2(1-\epsilon)}})$. Let $\tau = (1+\delta)\mu_1$. A Chernoff bound states that

$$\Pr_{\mathbf{A}}[||\mathbf{Az}||_0 \geq \tau] \leq e^{-\delta\mu_1/3}$$

Now we will ensure that $e^{-\delta\mu_1/3} = \text{negl}(\lambda)$. Using τ, we can compute δ exactly to be $\frac{1}{2}(\frac{e^{-2(1-\epsilon)}-e^{-2(1+\epsilon)}}{(1-e^{-2(1-\epsilon)})})$. We will use the fact that $e^{2\epsilon} - e^{-2\epsilon} > 4\epsilon$ for all $\epsilon > 0$.

Now, using our expression for μ_1, we get that

$$\begin{aligned}
e^{-\delta\mu_1/3} &= \exp[-\delta \cdot \frac{m}{2}(1 - e^{-2(1-\epsilon)})] \\
&= \exp[-\frac{1}{2} \cdot \left(\frac{e^{-2(1-\epsilon)} - e^{-2(1+\epsilon)}}{1 - e^{-2(1-\epsilon)}}\right) \cdot \frac{c\lambda}{2\epsilon^2}(1 - e^{-2(1-\epsilon)})] \\
&= \exp[-\frac{1}{2}(e^{-2(1-\epsilon)} - e^{-2(1+\epsilon)}) \cdot \frac{c\lambda}{2\epsilon^2}] \\
&\sim \exp[-\frac{1}{2} \cdot \frac{4\epsilon}{e^2} \cdot \frac{c\lambda}{2\epsilon^2}] = \exp[\frac{-c\lambda}{e^2\epsilon}] = \text{negl}(\lambda)
\end{aligned}$$

- $||\mathbf{z}||_0 \geq d(1+\epsilon)$. Recall that $\mu_2 = \frac{m}{2}(1 - \frac{1}{e^{2(1+\epsilon)}})$. Given our expression of τ, we have that $\tau = (1-\delta)\mu_2$. We will use the same strategy as in the previous case to show correctness, showing

$$e^{-\delta^2\mu_2/2} \leq e^{-\lambda}$$

So, recall that $\delta = \frac{1}{2}(\frac{e^{-2(1-\epsilon)}-e^{-2(1+\epsilon)}}{(1-e^{-2(1-\epsilon)})})$, and notice that $(1 - e^{-2(1+\epsilon)}) \geq 1 - e^{-2} > \frac{4}{5}$ for all $\epsilon > 0$. We compute

$$\begin{aligned}
e^{-\delta^2\mu_2/2} &= \exp[-\delta^2\frac{m}{2}(1 - e^{-2(1+\epsilon)})] \\
&\leq \exp[-\delta^2\frac{c\lambda}{2\epsilon^2} \cdot \frac{4}{5}] = \exp[-\frac{2}{5} \cdot \delta^2\frac{c\lambda}{\epsilon^2}] \\
&\leq \exp[-\frac{2}{5} \cdot \frac{1}{4}\left(\frac{e^{-2(1-\epsilon)} - e^{-2(1+\epsilon)}}{(1 - e^{-2(1-\epsilon)})}\right)^2 \cdot \frac{c\lambda}{\epsilon^2}] \\
&\leq \exp[-\frac{1}{10} \cdot \left(\frac{e^{-2(1-\epsilon)} - e^{-2(1+\epsilon)}}{1}\right)^2 \cdot \frac{c\lambda}{\epsilon^2}] \\
&\leq \exp[-\frac{1}{10} \cdot (e^{-2}(e^{2\epsilon} - e^{-2\epsilon}))^2 \cdot \frac{c\lambda}{\epsilon^2}] \\
&\leq \exp[-\frac{1}{10} \cdot \frac{16\epsilon^2}{e^4} \cdot \frac{c\lambda}{\epsilon^2}] = \exp[-\frac{8c\lambda}{5e^4}] = \text{negl}(\lambda)
\end{aligned}$$

In both cases, the probability that we choose a bad matrix $\mathbf{A}$ for a fixed input $\mathbf{z}$ is negligible in the security parameter, proving the lemma. □

4.2.2 Robust Gap-Hamming PPH with a Sparse Domain

To make the construction robust, we need to protect against two directions of attack: finding a low-weight vector that gets mapped to a high-weight vector, and finding a high-weight vector that maps to a low-weight one. To address the first line of attack, we will use an information-theoretic argument identical to the one in the proof of lemma 17. In short, in the proof of

lemma 17, we computed the probability that a *fixed* low-weight vector maps to a high-weight vector (on multiplication by the sparse matrix $\mathbf{A}$). The number of low-weight vectors is small enough that by a union bound, the probability that *there exists* a low-weight vector that maps to a high-weight vector is small as well.

To address the second line of attack, we unfortunately cannot make an information-theoretic argument (to be expected, as we compress beyond the information theoretic limits). Indeed, one possible attack is simply to use Gaussian elimination on $\mathbf{A}$ to come up with non-zero (probably high-weight) vector that maps to 0. Because $\mathbf{A}$ has a non-trivial null-space, this attack is likely to succeed.

To thwart such attacks, we leverage one of the following two loopholes that circumvent Gaussian elimination: (1) our first approach is to consider linear functions with *sparse* domains, restricting the input to be vectors of weight $\leq \beta n$ for constant $\beta < 1/2$, and so Gaussian elimination no longer works; and (2) building on this, we extend this to a *non-linear* construction where the domain is the set of all strings of a certain length. Our construction relies on the following hardness assumption that we refer to as the "sparse short vector" (SSV) assumption. The SSV assumption is a variant of the syndrome decoding problem on low-density parity-check codes, and roughly states that it is hard to find a preimage of a low-weight syndrome vector for which the preimage has "medium" weight.

Definition 18. *Let $n \in \mathbb{N}$. Let $\hat{\beta}, \alpha, \eta \in [0,1]$ and $\omega, \tau \in [n]$. The $(\hat{\beta}, \alpha, \omega, \tau, \eta)$-SSV (Sparse Short Vector) assumption states that for any PPT adversary $\mathcal{A}$ given an $\eta n \times n$ matrix $\mathbf{A}$ with entries sampled from $\mathsf{Ber}(\alpha)$,*

$$\Pr_{\mathbf{A} \sim \mathsf{Ber}(\alpha)^{\eta n \times n}} \left[\mathcal{A}(\mathbf{A}) \to \mathbf{z} \in \{0,1\}^n : \omega \leq ||\mathbf{z}||_0 \leq \hat{\beta} n \text{ and } ||\mathbf{Az}||_0 \leq \tau \right] = \mathrm{negl}(n).$$

On the Assumption. We now consider attacks on the SSV assumption which help us refine the parameter settings.

One way to attack the assumption is to solve the syndrome decoding problem for sparse parity-check matrices (also called the binary short vector problem or bSVP in [AHI+17]). In particular, find a $\hat{\beta}$-sparse $\mathbf{z}$ such that $\mathbf{Az} = 0$. To thwart these attacks, and at the same time have a compressing PPH construction, we need at the very minimum that $H(\hat{\beta}/2) > \eta > 2\hat{\beta}$.

$\eta < H(\hat{\beta}/2)$ ensures compression. Recall that we hash elements $\mathbf{x}_1$ and $\mathbf{x}_2$ in the hopes of being able to approximate their hamming distance. We have $\mathbf{z} = \mathbf{x}_1 \oplus \mathbf{x}_2$ is the vector we want to compute gap-hamming on, and so to guarantee $\mathbf{z}$ has sparsity $\hat{\beta}$, $\mathbf{x}_1$ and $\mathbf{x}_2$ need sparsity $\hat{\beta}/2 = \beta$. Vector $\mathbf{x}_1, \mathbf{x}_2 \in \{0,1\}^n$ of weight at most $\hat{\beta} n/2$ require (asymptotically) $H(\hat{\beta}/2)n$ bits each to describe, and so ηn needs to be less than that.

$\eta > 2\hat{\beta}$ is for security. If $\eta n \leq 2\hat{\beta} n$, then we are able to use Gaussian elimination to find a $\hat{\beta}$-sparse vector. Consider an $\eta n \times n$ matrix $\mathbf{A}$, and the first $\eta n \times \eta n$ square of it, call it $\mathbf{A}'$. Use Gaussian elimination to compute an ηn-length vector $\mathbf{z}'$ such that $\mathbf{A}'\mathbf{z}' = \mathbf{0}$. Padding $\mathbf{z}'$ with 0's, we get $\mathbf{z}$ where $\mathbf{Az} = \mathbf{0}$. We expect $||\mathbf{z}||_0 = \eta n/2$, and since $\eta n/2 \leq \hat{\beta} n$, we have broken the assumption.

Thus, for $\beta = \hat{\beta}/2$, we would like η to be as close to $H(\beta)$ as possible to give us non-trivial compression and at the same time, security from as conservative an assumption as possible. The reader might wonder about efficient unique decoding algorithms for LDPC codes. It turns out that the noise level ($\hat{\beta}$) for which the efficient decoding algorithms for LDPC imply a solution to bSVP is only a subset of the entire range $(H^{-1}(\eta), \eta/2)$. The range where efficient algorithms do not work (the "gap") grows with the locality parameter α [GB16, DKP16], and as the sparsity $\hat{\beta}$ tends towards 0, LDPC becomes similar to random linear code both combina-

Robust PPH for Sparse GAPHAMMING(n, d, ϵ)

- $\mathcal{H}.\mathsf{Samp}(1^\lambda, n, d, \beta, \epsilon)$:
  - Choose appropriate constants $c_1, c_2 > 0$ such that

$$m := \max\left\{\frac{c_1\lambda}{\epsilon^2}, \frac{n \cdot 3e^2 \ln(2) H(d(1-\epsilon)/n) + c_2\lambda}{\epsilon}, 4\beta n + 1\right\} < H(\beta)n\ .$$

  - Compute $\mu_1 = \frac{m}{2}(1 - e^{-2(1-\epsilon)})$ and $\mu_2 = \frac{m}{2}(1 - e^{-2(1+\epsilon)})$. Let $\tau = (\mu_1 + \mu_2)/2$.
  - Generate an $m \times n$ matrix $\mathbf{A}$ by choosing each entry from $\mathsf{Ber}(1/d)$.
  - Output $\mathbf{A}$ and the threshold τ.
- $\mathcal{H}.\mathsf{Hash}(\mathbf{A}, \mathbf{x})$: If $||\mathbf{x}||_0 \leq \beta n$, output $\mathbf{A}\mathbf{x} \in \mathbb{Z}_2^m$. Otherwise, output failure.
- $\mathcal{H}.\mathsf{Eval}(\mathbf{y}_1, \mathbf{y}_2)$: if $||\mathbf{y}_1 \oplus \mathbf{y}_2||_0 \leq \tau$, output ***CLOSE***, otherwise output ***FAR***.

Table 3: Construction of a robust PPH for sparse-domain GAPHAMMING(n, d, ϵ).

torially [Gal63, LS02], and, presumably, computationally. For a more detailed discussion, we refer the reader to [AHI+17].

We will set the sparsity parameter $\alpha \geq c/n$ for a large enough constant c, or to be conservative $\alpha \geq \log n/n$ to ensure that w.h.p. there are no all-0 columns.

Finally, we point out that when the matrix $\mathbf{A}$ is uniformly random and not sparse, SSV (where the adversary has to map small vectors to tiny vectors) is equivalent to bSVP (where the adversary has to map small vectors to 0). We briefly sketch how to reduce bSVP to SSV for a uniformly random $\mathbf{A}$. The reduction takes an instance of bSVP, a matrix $\mathbf{B} = [\mathbf{B}_1||\mathbf{B}_2]$ where $\mathbf{B}_2$ is square, and generates the matrix $\mathbf{A} := \mathbf{B}_2^{-1}\mathbf{B}_1$ as an instance of SSV. If the adversary finds $\mathbf{x}_1, \mathbf{x}_2$ such that $\mathbf{B}_2^{-1}\mathbf{B}_1\mathbf{x}_1 = \mathbf{x}_2$ solving SSV, then we have $[\mathbf{B}_1||\mathbf{B}_2] \cdot [\mathbf{x}_1^T||\mathbf{x}_2^T] = 0$, solving bSVP. However, this reduction does not work when $\mathbf{A}$ is sparse, though this connection indicates that the SSV problem is also hard.

Robust Hashing Construction for Gap-Hamming. Let the problem *β-Sparse Gap-Hamming* be the same as GAPHAMMING, except we restrict the domain to be over $\mathbf{x} \in \{0,1\}^n$ with sparsity $||\mathbf{x}||_0 \leq \beta n$. Our construction of a robust PPH for *Sparse Gap-Hamming* is as follows.

Settings Parameters from SSV Assumption to the Sparse Domain Construction Our main tool in this construction is the SSV assumption. So, here we consider a very conservative parameter setting for the SSV assumption, and show what parameters we acheive for our β-Sparse Gap-Hamming construction.

- $n \in \mathbb{N}$ and $\epsilon = \Omega(\sqrt{\lambda/n})$.
- Let the $(\beta, \alpha, \omega = (1+\epsilon)/\alpha, \tau, \eta)$-SSV be true for the following parameters: $0 < \beta \leq 0.04$ is a constant (this needs to be true in order to have $4\beta < H(\beta)$), $\alpha \geq \log n/n$, $\tau = \frac{\eta n}{4}(e - e^{-2(1-\epsilon)} - e^{-2(1+\epsilon)})$, and $\eta = H(\beta) \cdot (1 - \zeta)$ for a small constant ζ. These parameters come from believing a relatively conservative parameter settings for the SSV.

- Notice that the η in the assumption is just the number of output bits over the number of input bits. The actual compression of our construction is actually (at most) the number of output bits over $H(\beta)n$. So, with this assumption, we get compression $(\eta H(\beta)n)/(H(\beta)n) = (1-\zeta) = \Omega(1)$, and a center for our Gap-Hamming problem to be at any $d \leq \frac{n}{\log n}$.

We formally show why assuming the SSVunder the correct parameter settings yields a β-Sparse Gap-Hamming PPH.

Theorem 19. *Let λ be a security parameter, let $n = \mathsf{poly}(\lambda) \in \mathbb{N}$ and take any $\epsilon = \Omega(\sqrt{\lambda/n})$. Let $\beta, \alpha, \eta \in [0,1]$ and $\omega, \tau \in [n]$. Under the $(\beta, \alpha, \omega = (1+\epsilon)/\alpha, \tau, \eta)$-SSV assumption, the construction in Table 3 is a direct-access secure PPH for $\beta/2$-sparse $\text{GAPHAMMING}(n, d, \epsilon)$ where $d = 1/\alpha$.*

Proof. This proof follows the same structure as the proof for lemma 17, with the same computations for μ_1 and μ_2, but we will require a different m to get an information-theoretic argument for the case when $||\mathbf{z}||_0 \leq d(1-\epsilon)$ and rely on the assumption to show robustness for the other case.

So, let $\mu_1 = \frac{m}{2}(1 - \frac{1}{e^{2(1-\epsilon)}})$ and $\mu_2 = \frac{m}{2}(1 - \frac{1}{e^{2(1+\epsilon)}})$. δ is also computed as before to be $\delta = \frac{1}{2}(\frac{e^{-2(1-\epsilon)} - e^{-2(1+\epsilon)}}{(1-e^{-2(1-\epsilon)})})$. We analyze both cases with two claims.

Claim 20. *Given any adversary $\mathcal{A}$, it is impossible for $\mathcal{A}$ to output a vector $\mathbf{z}$ such that $||\mathbf{z}||_0 < d(1-\epsilon)$ and $||\mathbf{Az}||_0 \geq \tau$ with all but negligible probability over our choice of $\mathbf{A}$.*

Proof. We get, via a union bound and then Chernoff bound. Recall that $m = \max\left\{\frac{c_1\lambda}{\epsilon^2}, \frac{1}{\epsilon}\left(n \cdot 3e^2 \ln(2) H(d(1-\epsilon)/n) + c_2\lambda\right)\right\}$, and so $m \geq \frac{n \cdot 3e^2 \ln(2) H(d(1-\epsilon)/n) + c_2\lambda}{\epsilon}$. Also, notice that $\delta \geq \frac{2\epsilon}{e^2(1-e^{-2(1-\epsilon)})}$.

$$
\begin{aligned}
&\Pr_{\mathbf{A}}[\exists \mathbf{z} \text{ s.t. } ||\mathbf{z}||_0 \leq d(1-\epsilon) \wedge ||\mathbf{Az}||_0 \geq \tau] \leq 2^{nH(d(1-\epsilon)/n)} \Pr_{\mathbf{A}}[||\mathbf{Az}||_0 \geq \tau] \\
&\quad \leq \exp[\ln(2) n H(d(1-\epsilon)/n) - \delta\mu_1/3] \\
&\quad \leq \exp[\ln(2) H(d(1-\epsilon)/n) n - \frac{2\epsilon}{e^2(1-e^{-2(1-\epsilon)})} \cdot \frac{m}{2}\left(1 - e^{-2(1-\epsilon)}\right) \cdot \frac{1}{3}] \\
&\quad = \exp[\ln(2) H(d(1-\epsilon)/n) n - \frac{\epsilon}{3e^2} \cdot m] \\
&\quad \leq \exp[\ln(2) H(d(1-\epsilon)/n) n - \frac{\epsilon}{3e^2} \cdot \left(\frac{3e^2}{\epsilon} \cdot (\ln(2) H(d(1-\epsilon)/n) n) + \frac{c_2\lambda}{\epsilon}\right)] \\
&\quad = \exp[-\frac{\epsilon}{3e^2} \cdot \frac{c_2\lambda}{\epsilon}] = \exp[-\frac{c_2}{3e^2}] = \text{negl}(\lambda)
\end{aligned}
$$

□

Claim 21. *Let $\eta = m/n$, β be the parameter input into $\mathcal{H}$.Samp, and τ the threshold computed in $\mathcal{H}$.Samp. Assuming the $(2\beta, 1/d, d(1+\epsilon), \tau, \eta)$-SSV assumption, any PPT adversary $\mathcal{A}$ cannot find $\mathbf{z}$ such that $||\mathbf{z}||_0 \geq d(1+\epsilon)$ and $||\mathbf{Az}||_0 \leq \tau$.*

Proof. We need to use the assumption to bound the probability an adversary $\mathcal{A}$ is able to produce two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\{0,1\}^n$ such that $||\mathbf{x}_1||_0, ||\mathbf{x}_2||_0 \leq \beta n$, $||\mathbf{x}_1 - \mathbf{x}_2|| \geq d(1+\epsilon)$, and $||\mathbf{Ax}_1 \oplus \mathbf{Ax}_2||_0 \leq \tau$, when given $\mathbf{A}$. That is, equivalently, it can produce a vector $\mathbf{z} \in \{0,1\}^n$ where $d(1+\epsilon) \leq ||\mathbf{z}||_0 \leq 2\beta n$ and $||\mathbf{Az}||_0 \leq \tau$. So, the statement becomes exactly the definition of the $(2\beta, 1/d, d(1+\epsilon), \tau, \eta)$-SSV assumption:

$$\Pr_{\mathbf{A}}\left[\mathcal{A}(\mathbf{A}) \to \mathbf{z} \in \{0,1\}^n : \tau \leq ||\mathbf{z}||_0 \leq 2\beta n \wedge ||\mathbf{Az}||_0 \leq \tau\right] = \text{negl}(n)$$

□

The claims work together to show that no PPT adversary can find low weight vectors that map to high weight ones, and vice-versa, even if she has access to the code of hash function, $\mathbf{A}$. Therefore, the construction is robust in the direct-access model. □

4.2.3 From the Full Domain to a Sparse Domain

Now that we have a gap-Hamming preserving hash for sparse vectors ($||\mathbf{x}||_0 \leq \beta n$), we can extend this to work for the full domain.

One might consider a trivial way of converting any vector into a sparse vector via padding with 0's; we can take any vector $\mathbf{x} \in \{0,1\}^n$ and convert it into a 'sparse' vector $\mathbf{x}' \in \{0,1\}^{n'}$ with density at most n/n' by padding it with $n' - n$ zeros. However, this transformation is expensive in the length of the vector; we need to more than quadruple the length of $\mathbf{x}$ to get the density to be $\beta < .04$. Unfortunately, this transformation is also *linear*, and if we believe that the non-sparse version (construction 2) is not robust, then this combined linear version cannot be robust with the same parameters.

Instead of using padding, we will use a non-linear transformation that is more efficient in sparsifying a vector in terms of length, but incurs some bounded error in measuring gap-hamming distance. As long as we are liberal enough with the gap, ϵ, this will be a robust PPH that gets around attacks on a linear sketches, despite incurring some error.

Algorithm 1: Sparsify(x, k)

Input: $x \in \{0,1\}^n$ and parameter k

Output: $x' \in \{0,1\}^{2^k n/k}$ with density $1/2^k$.

```
Let x' =" (the empty string);
for i = 1 to n/k do
    Let y_i ← x_{ki}, x_{ki+1}, ..., x_{k(i+1)-1};
    Let t_i ← Σ_{j=0}^{k-1} 2^j y_{i,j};
    Let y'_i = e_{t_i}, the t_i'th basis vector in 2^k dimensions;
    x' ← x'||y'_i;
end
return x'
```

Algorithm 1 takes a dense bit vector of length n and turns it into a $1/2^k$-sparse bit vector of length $2^k n/k$. This is done by breaking the vector $x \in \{0,1\}^n$ into n/k blocks of k bits, and replacing each k-bit value with its corresponding (unit) indicator vector in $\{0,1\}^{2^k}$. Given two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \{0,1\}^n$ with $||\mathbf{x}_1 - \mathbf{x}_2||_0 = \Delta$, the sparsified versions $\mathbf{x}'_1$ and $\mathbf{x}'_2$ have $2\Delta/k \leq ||\mathbf{x}_1 - \mathbf{x}_2|| \leq 2\Delta$.

Recall that the trivial sparsifying method, simple padding, goes from n bits to n/β. If we let $k = \log(1/\beta)$, then using Algorithm 1, we go from n bits to $\frac{n2^k}{k} = \frac{n}{\log(1/\beta)\beta}$, saving a log factor of $1/\beta$. The construction in Table 4 is for dense gap-hamming.

Parameter settings for the full-domain construction Just as in the sparse case, we will propose a parameter setting compatible with a conservative instantiation of the SSV assumption.

- Let $n \in \mathbb{N}$, $\beta < 0.01$, and $\epsilon' \geq \frac{1}{\log(1/\beta)+1} \approx 0.13$.
- We will be using the same parameter setting as for the sparse case (notice that ϵ' is larger than in the sparse setting). So, let the $(\beta, \alpha, \omega = (1+\epsilon')/\alpha, \tau, \eta')$-SSV be true for the

Robust GAPHAMMING(n, d, ϵ) PPH family $\mathcal{H}$

- $\mathcal{H}.\mathsf{Samp}(1^\lambda, n, d, \beta, \epsilon)$.
  - If $\frac{1-1/\log(1/\beta)}{1+1/\log(1/\beta)} \geq \epsilon$, output failure.
  - Let $n' = \frac{n}{\log(1/\beta)\beta}$, $d' = \left((1-\epsilon)d + (1+\epsilon)\frac{d}{\log(1/\beta)}\right)$, and $\epsilon' = 1 - \frac{(1-\epsilon)d}{d'}$.
  - Pick constants c_1, c_2 such that

$$m := \max\left\{\frac{c_1\lambda}{\epsilon'^2}, \frac{n' \cdot 3e^2 \ln(2) H(d'(1-\epsilon')/n') + c_2\lambda}{\epsilon'}, 4\beta n' + 1\right\} < n$$

  - Compute $\mu_1 = \frac{m}{2}(1 - e^{-2(1-\epsilon')})$ and $\mu_2 = \frac{m}{2}(1 - e^{-2(1+\epsilon')})$. Let $\tau = (\mu_1 + \mu_2)/2$.
  - Generate an $m \times n'$ matrix $\mathbf{A}$ by choosing each entry from $\mathsf{Ber}(1/d')$.
  - Output $\mathbf{A}$ and the threshold τ.
- $\mathcal{H}.\mathsf{Hash}(\mathbf{A}, \mathbf{x})$: let $x' \leftarrow \mathsf{Sparsify}(x', \log(1/\beta))$, output $\mathbf{A}\mathbf{x}' \in \mathbb{Z}_2^m$.
- $\mathcal{H}.\mathsf{Eval}(\mathbf{y}_1, \mathbf{y}_2)$: if $||\mathbf{y}_1 - \mathbf{y}_2||_0 \leq \tau$, output ***CLOSE***, otherwise output ***FAR***.

Table 4: Construction of a robust GAPHAMMING(n, d, ϵ) PPH family.

following parameters from the sparse case: $\beta \leq 0.01$, $\alpha \geq \log n/n$, and $\tau = \frac{\eta' n}{4}(e - e^{-2(1-\epsilon')} - e^{-2(1+\epsilon')})$. Now, we will need a better compression term than before. Let $z > 0$ be a constant (close to 0), and $\eta' = \beta\log(1/\beta)(1-z) \approx 0.066(1-z)$. These parameters come from believing a relatively conservative parameter settings for the SSV assumption, with the parameters tuned just right to imply a Gap-Hamming PPH for the full domain.

- In total, Construction 4 is an η-compressing GAPHAMMING(n, d, ϵ) PPH, where $\eta = \frac{\eta'}{\beta\log(1/\beta)} = (1-z)$, $d \leq \frac{1}{2\alpha}\left((1-\epsilon') + \log(1/\beta)\epsilon'\right) \approx \frac{0.87n}{\log n}$, and gap $\epsilon \geq \frac{1-1/\log(1/\beta)}{1+1/\log(1/\beta)} \approx 0.74$. These parameters are formally proved to hold due to the security of the sparse construction, Construction 3, in Lemma 23.

Next, we will go into the details for why certain settings of the SSV assumption imply a Gap-Hamming PPH.

Theorem 22. *Let λ be a security parameter, let $n = \mathsf{poly}(\lambda) \in \mathbb{N}$. Let $\beta, \eta \in [0,1]$ and $\tau \in [n]$.*

Assuming the $(2\beta, 1/d', d'(1+\epsilon'), \tau, \eta)$-SSV assumption, where β is sparsity, $\eta = m/n'$, and τ is computed as in Table 4, then the construction in Table 4 is a Direct-Access secure PPH.

Proof. This is a simple application of theorem 19 with the correctness of algorithm Sparsify.

First, some properties about Sparsify. When given the input of an n-bit vector $\mathbf{x}$, and parameter $\log(1/\beta)$, the inner loop executes $n/\log(1/\beta)$ times. Each loop adds $2^{\log(1/\beta)} = 1/\beta$ coordinates to $\mathbf{x}'$, and so the output vector $\mathbf{x}'$ is $\frac{n}{\log(1/\beta)\beta}$ bits. Second, $\mathbf{x}'$ is β-sparse. This is because each loop adds at most one coordinate with a 1 in it (a standard basis vector), and the loop executes $n/\log(1/\beta)$ times, meaning the density is at most $\frac{n/\log(1/\beta)}{n/\log(1/\beta)\cdot 1/\beta} = \beta$.

In order for this to be a PPH, we first need compression: sparsification expands inputs and so we need to be sure that we shrink it enough afterwards. Of course the construction fails

any time $m \geq n$, but we need to argue such an m even exists. As per earlier analysis, we need $m > 4\beta n'$. Given $n' = n/(\log(1/\beta)\beta)$, this means $m > 4n/\log(1/\beta)$. If $\log(1/\beta) \geq 5$, then there exists an m such that $4n/\log(1/\beta) < m < n$, and so there exists a compressing m in this context for sufficiently large n.

Now note that when given $\mathbf{x}_1$, and $\mathbf{x}_2$ where $||\mathbf{x}_1 - \mathbf{x}_2||_0 = \Delta$, we have that $\frac{2\Delta}{\log(1/\beta)} \leq ||\mathsf{Sparsify}(\mathbf{x}_1) - \mathsf{Sparsify}(\mathbf{x}_2)||_0 \leq 2\Delta$; so Sparsify introduces some error.

So, assume $(2\beta, 1/d', d'(1+\epsilon'), \tau, \eta)$-SSV holds for $\tau, \beta, \eta, d', \epsilon'$ computed as in the construction and for a contradiction assume there exists an adversary $\mathcal{A}$ that can break Direct-Access robustness of construction 4. We will show that $\mathcal{A}$ must break the $(2\beta, 1/d', d(1+\epsilon), \tau, \eta)$-SSV assumption. So, $\mathcal{A}$ will output two vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$ that with noticeable probability will fit into one of two cases breaking Direct-Access robustness:

- $||\mathbf{x}_1 \oplus \mathbf{x}_2||_0 < d(1-\epsilon)$ such that $||\mathbf{A}(\mathsf{Sparsify}(\mathbf{x}_1) - \mathsf{Sparsify}(\mathbf{x}_2))||_0 > \tau$. Let $\hat{\mathbf{z}} \leftarrow \mathsf{Sparsify}(\mathbf{x}_1) - \mathsf{Sparsify}(\mathbf{x}_2)$. We have that $||\hat{\mathbf{z}}||_0 < d(1-\epsilon) \leq 2d'(1-\epsilon')$. Now because of how we computed m, the same proof as in the proof of construction 3 will show that there exists such a $\hat{\mathbf{z}}$ with negligible probability in λ. Therefore, with all-but-negligible probability, $\mathcal{A}$ cannot take this line of attack.

- $||\mathbf{x}_1 \oplus \mathbf{x}_2||_0 > d(1+\epsilon)$ such that $||\mathbf{A}(\mathsf{Sparsify}(\mathbf{x}_1) - \mathsf{Sparsify}(\mathbf{x}_2))||_0 < \tau$. Let $\hat{\mathbf{z}} \leftarrow \mathsf{Sparsify}(\mathbf{x}_1) - \mathsf{Sparsify}(\mathbf{x}_2)$. Recall that Sparsify introduces bounded error, so we know that $||\hat{\mathbf{z}}||_0 > \frac{2d(1+\epsilon)}{\log(1/\beta)} \geq d'(1+\epsilon')$ given how we have computed our parameters.

  Therefore, we have computed a 2β-sparse vector $\hat{\mathbf{z}} \in \{0,1\}^{n'}$, with τ computed as in construction 3 for parameters n', d', ϵ', which exactly violates the $(2\beta, 1/d', d'(1+\epsilon'), \tau, \eta)$-SSV.

□

On Feasibility Settings for the Full-Domain Construction. Here we will prove that if there exists a parameter setting for the sparse construction, Construction 3, with good-enough compression, then there exists a parameter setting for the full domain, Construction 4. It is important to note, however, that we get a worse lower bound for our resulting gap ϵ: η' is constant, so β is also constant, and since we require $\epsilon' > 1/(\log(1/\beta)+1)$ our resulting ϵ is also constant.

Lemma 23. *Assume that Construction 3 is an η'-compressing robust PPH for* GapHamming(n', d', ϵ') *where $\eta' < \beta\log(1/\beta)$ and $\epsilon' > \frac{1}{\log(1/\beta)+1}$. Then, Construction 4 is an η-compressing robust PPH for* GapHamming(n, d, ϵ) *for the following parameters:*

- $\eta = \frac{\eta'}{\beta\log(1/\beta)}$,
- $n = \beta\log(1/\beta)n'$,
- $d = \frac{d'}{2}\left((1-\epsilon') + \log(1/\beta)\epsilon'\right)$,
- $\epsilon = \frac{\epsilon'\log(1/\beta) - (1-\epsilon')}{\epsilon'\log(1/\beta) + (1-\epsilon')}$.

Proof. Note that we can allways assume the gap is bigger, so if $\epsilon' \leq \frac{1}{\log(1/\beta)+1}$, then we can use our PPH for GapHamming(n', d', ϵ') as a PPH for GapHamming$(n', d', \frac{1}{\log(1/\beta)+1} + .0001)$. So, without loss of generality, assume $\epsilon' > \frac{1}{\log(1/\beta)+1}$.

We will show that Construction 4 is a robust PPH for GapHamming(n, d, ϵ) by showing that we can take any dense input in $\{0,1\}^n$, turn it into a sparse input in $\{0,1\}^{n'}$, and

then using Construction 3, that hash functions and evaluations will produce correct results for GAPHAMMING(n, d, ϵ). Robustness follows from the robustness of Construction 3.

First, compression is guaranteed since $\eta n = \eta' n' < \beta \log(1/\beta) n' = n$, implying $\eta < 1$.

Next, take any $\mathbf{x} \in \{0,1\}^n$ and let $\mathbf{x}' = \mathsf{Sparsify}(\mathbf{x}, \log(1/\beta))$. Notice that $\mathbf{x}'$ has length $n' = n/(\beta \log(1/\beta))$ and sparsity at least β by the correctness of $\mathsf{Sparsify}$. Therefore, $\mathbf{x}$ is a valid input to the hash functions from Construction 3.

Now, we need to show that the resulting hash function is correct. For any $\mathbf{x}_1, \mathbf{x}_2 \in \{0,1\}^n$, where $||\mathbf{x}_1 - \mathbf{x}_2|| = \Delta$, we have $\frac{2\Delta}{\log(1/\beta)} \leq ||\mathsf{Sparsify}(\mathbf{x}_1) - \mathsf{Sparsify}(\mathbf{x}_2)|| \leq 2\Delta$. We have two cases to consider to ensure correctness.

- If $\Delta < d(1-\epsilon)$, then $||\mathbf{x}_1 - \mathbf{x}_2|| < 2d(1-\epsilon) = d'(1-\epsilon')$. Then, by the robustness of Construction 3, the hash function will output ***CLOSE*** with all but negligible probability over our choice of hash, even with adversarially chosen inputs.
- If $\Delta > d(1+\epsilon)$, then $||\mathbf{x}_1 - \mathbf{x}_2|| > 2d(1+\epsilon)/\log(1/\beta) = d'(1+\epsilon')$. Then, by the robustness of Construction 3, the hash function will output ***FAR*** with all but negligible probability over our choice of hash, even with adversarially chosen inputs.

□

Notice that setting n,d, and ϵ to these values exactly translates into having n', d', and ϵ' be the intermediate values in Construction 4. With this lemma we can explicitly characterize the valid parameter settings for the full domain as follows.

Full-Domain Parameter Settings. Fix our input size n, and assume that the Sparse-Domain construction works for any constant compression η', any $\epsilon = \Omega(1)$, for some constant sparsity β.

- We can compress by any constant $\eta = O(1)$,
- we can handle any constant gap $\epsilon = \Omega(1)$,
- and we can let our center be any $d \leq \frac{n}{2\log n}((1-\epsilon) + (1+\epsilon))$.

5 Necessity of Cryptographic Assumptions

Recall the goal of robust PPH is to compress beyond the information theoretic limits, to a regime where incorrect hash outputs exist but are computationally hard to find. When the hash function is given, this inherently means such constructions necessitate cryptographic hardness assumptions. A natural question is what types of assumptions are required to build non-trivial PPHs of various kinds.

In this section, we address the computational assumptions implied by PPH for classes of predicates which satisfy a notion we refer to as *collision sensitivity*. As the name suggests, a class of predicates is collision sensitive if finding a collision in a given hash function breaks the soundness of the hash.

Definition 24. *A class of predicates $\mathcal{P}$ is* collision sensitive *if there exists a PPT algorithm $\mathcal{A}$ such that for any pair x, x', $\Pr[\mathcal{A}(x, x') \to P : P(x) \neq P(x')] \geq 1 - \mathrm{negl}(n)$.*

Notice that a class of predicates being *reconstructing* (as per Section 3.2) automatically implies collision-sensitivity. Indeed, it is a stronger characteristic: Since the reconstructing learner can use a series of predicates P to determine x from all other x' with negligible probability, this

already implies we can use that same series P to distinguish x from any other x' (also with negligible probability).

We show two lower bounds for achieving a PPH for any class of collision-sensitive predicates $\mathcal{P}$:

1. Direct-Access robust PPHs for $\mathcal{P}$ implies the existence of collision resistant hash functions (using the definition of equality PPHs).

2. Double-Oracle robust PPHs for $\mathcal{P}$ implies one-way functions (using techniques from [NY15]).

These results follow from characterizations of PPH for the specific case of the *equality* predicate in the respective models.

On the other hand, we demonstrate an unconditional construction for the weaker notion of Evaluation-Oracle PPHs for equality, using pairwise independence. Note that existence of an unconditional construction is to be expected, as Evaluation-Oracle PPHs align with non-robust PPHs for the case of total predicates (such as equality).

5.1 The Equality Predicate and Collision-Sensitivity

Denote $Q_{x_2}(x_1) := [x_1 == x_2]$ the x_2-parameterized equality predicate, and denote $Q'_{x_2}(y) := \mathcal{H}.\mathsf{Eval}(h, Q_{x_2}, y)$ for a given hash function h sampled from PPH $\mathcal{H}$. One thing to notice is that finding any collision with respect to h, i.e. $h(x_1) = h(x_2)$ but $x_1 \neq x_2$, means that $Q'_{x_2}(h(x_1)) = Q'_{x_2}(h(x_2))$, and so either $Q'_{x_2}(h(x_1)) \neq Q_{x_2}(x_1)$ or $Q'_{x_2}(h(x_2)) \neq Q_{x_2}(x_2)$. This necessarily means that no matter what Q' actually computes with respect to x_2 and $h(x_1) = h(x_2)$, its output at least one of the inputs x_1, x_2 is incorrect. We leverage this together with the following reduction from PPH for any collision-sensitive predicate class to PPH for equality, in order to prove lower bounds.

Theorem 25. *If there exists a (direct-access / double-oracle / evaluation-oracle) robust PPH for any collision-sensitive predicate $\mathcal{P}$ with compression η, then there exists a (direct-access / double-oracle / evaluation-oracle, resp.) robust equality PPH with compression η.*

Proof. We will prove the contrapositive. Assume in any of our robust models that there does not exist an equality PPH with compression η. Then, for any η-compressing hash h, there exists an adversary $\mathcal{B}$ that, when playing the game corresponding to the model, can output an x_1 and x_2 such that $Q_{x_2}(x_1) \neq Q'_{x_2}(h(x_1))$.

Now, for sake of contradiction, also assume that there exists a PPH $\mathcal{H}$ for a class of collision-sensitive predicates $\mathcal{P}$. Consider the following PPH $\mathcal{H}'$ for the class of equality predicates, where $\mathcal{H}'.\mathsf{Samp} = \mathcal{H}.\mathsf{Samp}$, and where we define $\mathcal{H}'.\mathsf{Eval}(h, Q_{x_2}, h(x_1)) = 1$ if and only if $h(x_1) = h(x_2)$. Note that $\mathcal{H}'$ also has compression factor η. Because equality PPH does not exist by assumption, there exists an efficient adversary $\mathcal{B}$ who can output x_1 and x_2 such that $Q_{x_2}(x_1) \neq \mathcal{H}'.\mathsf{Eval}(h, Q_{x_2}, h(x_1))$ with non-negligible probability. By construction, it must be that $x_1 \neq x_2$. This implies $Q_{x_2}(x_1) = 0$ and therefore $\mathcal{H}'.\mathsf{Eval}(h, Q_{x_2}, h(x_1)) = 1$. From our definition of Q', this means that $h(x_1) = h(x_2)$. Now because $\mathcal{P}$ is collision-sensitive, we can use algorithm $\mathcal{A}$ on x_1, x_2 to generate a predicate $P_{cs} \in \mathcal{P}$ such that $P_{cs}(x_1) \neq P_{cs}(x_2)$. However, because $h(x_1) = h(x_2)$, $\mathcal{H}.\mathsf{Eval}(h, P_{cs}, h(x_1)) = \mathcal{H}.\mathsf{Eval}(h, P_{cs}, h(x_2))$. One of these evaluations *must* be incorrect. Therefore, any attack against the corresponding equality version of the PPH is also an attack against the collision-sensitive predicate class PPH (in any model). □

In the following subsections, we focus on characterizations of PPH for equality within our respective levels of robustness. Then in Section 5.5 we return to the corresponding implications for PPH for any class of collision-sensitive predicates.

5.2 Direct-Access Equality PPHs if and only if CRHFs

We observe that Direct-Access robust PPHs for equality are equivalent to collision-resistant hash functions (CRHFs):

Definition 26. *A family of functions $\mathcal{H} = \{h : X \to Y\}$ is a family of CRHFs if it is efficiently sampleable, efficiently evaluatable, compressing, and for any PPT adversary $\mathcal{A}$,*

$$\Pr_{h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda)} [\mathcal{A}(h) \to (x_1, x_2) : h(x_1) = h(x_2) \wedge x_1 \neq x_2] \leq \mathrm{negl}(\lambda)$$

Notice that a CRHF family satisfies the definition of an equality PPH, $\mathcal{H}.\mathsf{Eval}(h, P_{x_2}, y) = (h(x_2) == y)$: an adversary who finds $\mathcal{H}.\mathsf{Eval}(h, P_{x_2}, h(x_1)) \neq (x_1 == x_2)$, violating correctness of the PPH, must have found $h(x_1) = h(x_2)$, violating the collision-resistance of the CRHF. Notice also that an equality PPH must satisfy collision-resistance: an adversary finding a collision between x_1 and x_2 can break the correctness of the PPH with either $\mathcal{H}.\mathsf{Eval}(h, P_{x_2}, h(x_1))$ or $\mathcal{H}.\mathsf{Eval}(h, P_{x_2}, h(x_2))$. Therefore, the two definitions are equivalent.

5.3 Double-oracle Equality PPHs if and only if OWFs

We will prove that such a hash family existing is equivalent to OWFs. This is significantly less obvious than the previous characterization of the equality PPHs using CRHFs. First we will show the obvious direction, that OWFs imply Double-oracle Equality PPHs.

Claim 27. *Suppose one-way functions exist. Then for any polynomial p, there exist Double-Oracle robust PPH families $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ for equality also exist.*

Proof. OWFs imply the existence of (compressing) PRFs. Let $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a family of PRFs. We will prove that $\mathcal{H}$ is also an equality-preserving hash robust in the Double-Oracle model.

Consider an adversary $\mathcal{A}$ that has a non-negligible advantage at finding a collision. That is, $\Pr_h \left[\mathcal{A}^{h(\cdot)} \to (x, y) : h(x) = h(y)\right] \geq \epsilon + \delta$, where δ is non-negligible. Now, let $R : \{0,1\}^n \to \{0,1\}^m$ be a truly random function. Clearly, for any $\mathcal{A}$, $\Pr_h \left[\mathcal{A}^{R(\cdot)} \to (x, y) : h(x) = h(y)\right] = \epsilon$ — no PPT algorithm can have any advantage over finding a collision in a random function beyond a guaranteed collision probability for random guessing.

Since $\mathcal{A}$ has a noticeable advantage, we can distinguish when h is a PRF and when we have a random oracle. This contradicts the definition of a PRF. Therefore, no PPT $\mathcal{A}$ should have more than a negligible advantage in producing a collision. □

5.3.1 Double-Oracle PPHs for Equality imply OWFs.

We will show that without OWFs, given *any* compressing family of functions $\mathcal{H}$, we can find a collision given only an oracle to $h \in \mathcal{H}$ with noticeable probability. Finding a collision is equivalent to finding a pair of inputs breaking the guarantees of the PPH.

Theorem 28. *Double-Oracle robust PPHs for equality imply OWFs.*

The proof of this theorem is in appendix E and is an adaptation and generalization of the proof that adversarially robust Bloom Filters require OWFs from [NY15]. The basic idea is to use the fact that we can invert any poly-time evaluatable function in polynomial time to reverse-engineer the randomness used in generating that specific hash function from $\mathcal{H}.\mathsf{Samp}$. Once we are able to do this, we can augment that inversion algorithm to also return a nearly random preimage. This will cause us to find a collision with noticeable probability.

5.4 Evaluation-Oracle PPHs for Equality with Pairwise Independence.

We note that we do not need any computational assumptions to obtain an Equality PPH in the Evaluation-Oracle model. Let $\mathcal{F}$ be a pairwise-independent hash family from m bits to n bits, with $n < m$. We will prove that $\mathcal{F}$ is secure in the Evaluation-Oracle model.

Theorem 29. *Let $\mathcal{F} = \{f : \{0,1\}^m \to \{0,1\}^n\}$ be any compressing pairwise-independent hash family. $\mathcal{F}$ is an equality-preserving hash that is robust in the Evaluation Oracle Model.*

Proof. First, note that $\mathcal{F}$ is compressing by definition. So, we will move on to proving it is secure. We will show that we can replace every query answered by the evaluation oracle with an oracle that just returns the correct answer using a series of hybrids. Let $\mathcal{O}_{eq}$ be an oracle that returns 1 if two strings are different and 0 if they are equal.

So, let $\mathcal{A}$ be a PPT adversary and let $T = \mathrm{poly}(n)$ be the maximum number of queries $\mathcal{A}$ can make. We will prove that $\mathcal{A}$ cannot distinguish whether he is receiving actual predicate evaluations or correct evaluations. In Hybrid 0, $\mathcal{A}$ is using $\mathcal{O}_{h.\mathsf{Eval}'}$ for all of the queries. In Hybrid t, $\mathcal{A}$ is getting answers from $\mathcal{O}_{eq}$ for the first t queries, and then gets answers from $\mathcal{O}_{h.\mathsf{Eval}'}$ for the last $T - t$ queries.

We will now show that statistically $\mathcal{A}$ cannot distinguish between Hybrid t and Hybrid $t-1$. So, $\mathcal{A}$ has made $t-1$ queries and gotten correct responses for each of them. $\mathcal{A}$'s tth query can be x_1, x_2 where $x_1 = x_2$ or $x_1 \neq x_2$. Both oracles are guaranteed to answer the same way if $x_2 = x_2$, so let's examine the case where $x_1 \neq x_2$. The probability over our choice of $f \in \mathcal{F}$ that $h(x_1) = h(x_2)$ is 2^{-n} because $\mathcal{F}$ is pairwise independent. Therefore, the probability that $\mathcal{O}_{h.\mathsf{Eval}'}$ answers differently from $\mathcal{O}_{h.\mathsf{Eval}'}$ is 2^{-n}, and $\mathcal{A}$ can only distinguish Hybrid t and $t-1$ with probability 2^{-n}.

This means that $\mathcal{A}$ can distinguish Hybrid 0 from Hybrid T with probability at most $\mathrm{poly}(n) \cdot 2^{-n} = \mathrm{negl}(n)$ (union bound). □

5.5 Collision-Sensitivity, OWFs, and CRHFs

As shown in Theorem 25, any lower bound for equality PPHs implies a lower bound for all PPHs for collision sensitive predicate classes. Therefore, we get the following two corollaries.

Corollary 30. *Let $\mathcal{P}$ be any collision-sensitive predicate class. Then any PPH for $\mathcal{P}$ in the Direct-Access model implies that CRHFs exist.*

Corollary 31. *Let $\mathcal{P}$ be any collision-sensitive predicate class. Then any PPH for $\mathcal{P}$ in the Evaluation-Oracle model implies that OWFs exist.*

Acknowledgments. Many thanks to Daniel Wichs for suggesting the construction in Section 4.1 and for useful conversations regarding the connection to secure sketches and fuzzy extractors.

References

[AHI+17] Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 7:1–7:31, 2017.

[AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 20–29, 1996.

[BGKL03] László Babai, Anna Gál, Peter G. Kimmel, and Satyanarayana V. Lokam. Communication complexity of simultaneous messages. *SIAM J. Comput.*, 33(1):137–166, 2003.

[CCF04] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[CDS01] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):129–159, 2001.

[CM05] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[CRVW02] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the Thiryfourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 659–668, New York, NY, USA, 2002. ACM.

[CW77] Larry Carter and Mark N. Wegman. Universal classes of hash functions (extended abstract). In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 106–112, 1977.

[DI06] Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 711–720, 2006.

[DKP16] I. Dumer, A. A. Kovalev, and L. P. Pryadko. Distance verification for ldpc codes. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2529–2533, July 2016.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, March 2008.

[Gal63] Robert Gallagher. Low-density parity-check codes, 1963.

[GB16] Leonid Geller and David Burshtein. Bounds on the belief propagation threshold of non-binary LDPC codes. *IEEE Trans. Information Theory*, 62(5):2639–2657, 2016.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GIL+90] O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesan, and D. Zuckerman. Security preserving amplification of hardness. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 318–326 vol.1, Oct 1990.

[HW13] Moritz Hardt and David P. Woodruff. How robust are linear sketches to adaptive inputs? In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 121–130. ACM, 2013.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 604–613, 1998.

[Ind00] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 189–197, 2000.

[JKS08] T. S. Jayram, Ravi Kumar, and D. Sivakumar. The one-way communication complexity of hamming distance. *Theory of Computing*, 4(1):129–135, 2008.

[KKG18] Harini Kannan, Alexey Kurakin, and Ian J. Goodfellow. Adversarial logit pairing. *CoRR*, abs/1803.06373, 2018.

[KNR95] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 596–605, New York, NY, USA, 1995. ACM.

[KOR98] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 614–623, New York, NY, USA, 1998. ACM.

[KW17] J. Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017.

[LS02] S. Litsyn and V. Shevelev. On ensembles of low-density parity-check codes: asymptotic distance distributions. *IEEE Transactions on Information Theory*, 48(4):887–908, Apr 2002.

[MG82] Jayadev Misra and David Gries. Finding repeated elements. *Sci. Comput. Program.*, 2(2):143–152, 1982.

[MMS+17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017.

[MNS08] Ilya Mironov, Moni Naor, and Gil Segev. Sketching in adversarial environments. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 651–660, 2008.

[MP80] J. Ian Munro and Mike Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 33–43, 1989.

[NY15] Moni Naor and Eylon Yogev. Bloom filters in adversarial environments. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 565–584, 2015.

[RS15] Sivaramakrishnan Natarajan Ramamoorthy and Makrand Sinha. On the communication complexity of greater-than. In *53rd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2015, Allerton Park & Retreat Center, Monticello, IL, USA, September 29 - October 2, 2015*, pages 442–444, 2015.

[RSL18] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *CoRR*, abs/1801.09344, 2018.

[SND17] Aman Sinha, Hongseok Namkoong, and John C. Duchi. Certifiable distributional robustness with principled adversarial training. *CoRR*, abs/1710.10571, 2017.

[Woo04] David Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 167–175, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[Woo07] David P. Woodruff. *Efficient and private distance approximation in the communication and streaming models*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2007.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979.

A Multi-Input vs Single-Input Predicates

We discuss the differences between definitions of property-preserving hashes with a family of single-input predicates versus a fixed multi-input predicate. For an example, consider the two-input equality predicate, namely $P(x_1, x_2) = 1$ if $x_1 = x_2$ and 0 otherwise. However, we can also define a class of predicates $\mathcal{P} = \{P_x\}_{x \in X}$ where $P_{x_1}(x_2) = 1$ if $x_1 = x_2$ and 0 otherwise. This exponential-size predicate class $\mathcal{P}$ accomplishes the same task as the two-input single predicate. In general, we can take any two-input (or multi-input) predicate and convert it into a predicate class in the same manner.

We give below the definition of (direct access) PPH for a two-input property P. The other definitions follow a similar vein, and are omitted.

Definition 32. *A (direct-access-robust) property-preserving hash family $\mathcal{H} = \{h : X \to Y\}$ for a two-input predicate $P : X \times X \to \{0, 1\}$ consists of two efficiently computable algorithms:*

- $\mathcal{H}.\mathsf{Samp}(1^\lambda) \to h$ *is a randomized p.p.t. algorithm that samples a random hash function from $\mathcal{H}$ with security parameter λ.*
- $\mathcal{H}.\mathsf{Eval}(h, y_1, y_2)$ *is a deterministic polynomial-time algorithm that that on input the hash function h and values $y_1, y_2 \in Y$ (presumably $h(x_1)$ and $h(x_2)$ for some $x_1, x_2 \in X$), outputs a single bit.*

Additionally, $h \in \mathcal{H}$ must satisfy the following two properties:

- compressing: $\lceil \log |Y| \rceil < \lceil \log |X| \rceil$, *and*
- direct-access robust: *for any PPT adversary $\mathcal{A}$,*

$$\begin{aligned}\Pr[h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda);\ (x_1, x_2) \leftarrow \mathcal{A}(h) :\\ P(x_1, x_2) \neq \circledast \wedge \mathcal{H}.\mathsf{Eval}(h(x_1), h(x_2)) \neq P(x_1, x_2)] \leq \mathrm{negl}(\lambda)\end{aligned}$$

Any multi-input family of PPH can be converted into a PPH for the corresponding predicate-class with the following simple transformation: $P_{x_1}(x_2) := P(x_1, x_2)$ is transformed into the class of predicates $\{P_x\}_{x \in X}$. In general, single-input PPHs look easier to construct, since the transformed predicate has more information to work with. In fact, we can show an explicit example where the lower bound for the single-predicate version is smaller than the multi-predicate version (see the Gap GREATERTHANproofs in appendix section C.3.2).

Lemma 33. *Let $\mathcal{H}$ be a robust PPH in any model for a two-input predicate P on X. Then, there exists a PPH secure in the same model for the predicate class $\{P_x\}_{x \in X}$ where $P_{x_1}(x_2) = P(x_1, x_2)$.*

Proof. Assume we have a PPH $\mathcal{H}$ for a two-input predicate P and the corresponding predicate class is $\mathcal{P} = \{P_{x_2}\}_{x_2 \in X}$. We will define $\mathcal{H}'$ as follows.

- $\mathcal{H}'.\mathsf{Samp}(1^\lambda) = \mathcal{H}.\mathsf{Samp}(1^\lambda)$.
- $\mathcal{H}'.\mathsf{Eval}(h, y, P'_{x_2}) = \mathcal{H}.\mathsf{Eval}(h, y, h(x_2))$.

Our goal is now to show that an adversary breaking $\mathcal{H}'$ in the security model $\mathcal{H}$ could also break $\mathcal{H}$ in that model.

- Consider the Evaluation-Oracle model. Any evaluation query will be of the form P_{x_1} and x_2, where P_{x_1} has enough information to extract x_1. So, we just query the 2-input Evaluation-Oracle on x_1 and x_2 and pass on the result.

- Consider the Double-Oracle model. Again, any evaluation query will be handled in a similar way, although we get P_{x_1} and y_2, and first need to query for the hash for x_1 and then query the evaluation oracle for $h(x_1), y_2$. Any hash query carries over directly.
- Consider the Direct-Access model. If we are given the code for $\mathcal{H}$, we can easily construct code for $\mathcal{H}'$ and hand an adversary that code with the same distribution as if we were to sample $\mathcal{H}'$ without first sampling $\mathcal{H}$. Thus, if the adversary can break $\mathcal{H}'$ with non-negligible probability, the adversary will break our construction of $\mathcal{H}'$ with the same probability.

□

B Proofs of Section 2: Relationships between definitions

Here we will write the proofs for the lemmas described in section 2. First, lemma 4 states that for total predicates, non-robustness and evaluation-oracle PPHs are equivalent. Then, lemma 6 states that with OWFs, we can take a EO-robust PPH to get a DO-robust PPH, simply by pairing a hash function with an invertible PRF.

B.1 Proof of lemma 4: From a Non-Robust to Robust PPH for Total Predicates

Here is the proof that a non-robust PPH for a total predicate implies an Evaluation-Oracle PPH.

Lemma 34. *Let $\mathcal{P}$ be a class of total predicates on X. A non-robust PPH $\mathcal{H}$ for $\mathcal{P}$ is also an Evaluation-Oracle robust PPH for $\mathcal{P}$ for the same domain X and same codomain Y.*

Proof. Let $\mathcal{H} = \{h : X \to Y\}$ be a non-robust PPH for a class of total predicates $\mathcal{P}$ on X. Without any access to the hash function itself, any adversary (not even computationally bounded) has a negligible chance of coming up with an x and P that violate correctness because the adversary has no idea which h was sampled from $\mathcal{H}$. We will show that even given an Evaluation Oracle, $\mathcal{A}$ still cannot learn anything about which h was sampled, and so has the same advantage as blindly guessing.

Let $\mathcal{A}$ make at most T queries to $\mathcal{O}_h^{\mathsf{Eval}}$. Let $\mathcal{O}_{\mathcal{P}}$ just be the trivial predicate evaluation oracle, so $\mathcal{O}_{\mathcal{P}}(x, P) = P(x)$. We will now construct a series of t hybrids.

- Hybrid 0. $\mathcal{A}$ queries $\mathcal{O}_h^{\mathsf{Eval}}$.
- Hybrid t. For the first t queries, $\mathcal{A}$ gets answers from $\mathcal{O}_{\mathcal{P}}$. For the last $T - t$ queries, $\mathcal{A}$ gets answers from $\mathcal{O}_h^{\mathsf{Eval}}$.
- Hybrid T. $\mathcal{A}$ makes all T queries to $\mathcal{O}_{\mathcal{P}}$.

Note that $\mathcal{A}$'s first query to $\mathcal{O}_h^{\mathsf{Eval}}$ has a negligible chance of being answered incorrectly due to the correctness of the PPH (i.e. has a negligible chance of being distinguishable from $\mathcal{O}_{\mathcal{P}}$). The only way for $\mathcal{A}$ to distinguish hybrids $t-1$ and t is if query t was answered incorrectly. Since query t is $\mathcal{A}$'s first query to the $\mathcal{O}_h^{\mathsf{Eval}}$ in Hybrid t, $\mathcal{A}$ will be able to detect this difference with negligible probability in λ.

Since $T = \mathrm{poly}(\lambda)$, a union bound yields that the maximum possible probability $\mathcal{A}$ can distinguish Hybrid 0 from Hybrid T is $\mathrm{poly}(\lambda) \cdot \mathrm{negl}(\lambda) = \mathrm{negl}(\lambda)$.

Given $\mathcal{H}$ with algorithms $(\mathsf{Samp}, \mathsf{Transf})$ a no-function access, oracle-predicate PPH family and a CCA2-secure symmetric encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, we can construct $\mathcal{H}^*$ with algorithms $(\mathsf{Samp}^*, \mathsf{Transf}^*)$ as follows.

$\mathsf{Samp}^*(1^\lambda)$:

1. $h \leftarrow \mathsf{Samp}(1^\lambda)$.
2. $(f_k, k) \xleftarrow{\$} \mathcal{F}$.
3. Output $h^* = (h, k)$ where $h^*(x) = f_k(h(x))$.

$\mathsf{Eval}^*(h^*, P, y^*)$

1. Parse $h^* = (h, k)$.
2. $y \leftarrow f_k^{-1}(y^*)$.
3. Output $\mathsf{Eval}(h, P, y)$.

Figure 2: Tranforming a PPH that is secure against adversaries that do not have access to the hash function and only oracle access to predicates to a PPH secure against adversaries with oracle access to the hash functions using CCA2-secure symmetric encryption.

So, in Hybrid T, $\mathcal{A}$ is making no queries to $\mathcal{O}_h^{\mathsf{Eval}}$. In fact, $\mathcal{A}$ can simulate every response from $\mathcal{O}_{\mathcal{P}}$ just by evaluating $P(x)$ on its own.

$$\Pr_{h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda)}[\mathcal{A}^{\mathcal{O}_h^{\mathsf{Eval}}}(1^\lambda) \to (x, P) : P'(h(x)) \neq P(x)]$$
$$\Pr_{h \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda)}[\mathcal{A}(1^\lambda) \to (x, P) : P'(h(x)) \neq P(x)] + \mathrm{negl}(\lambda) = \mathrm{negl}(\lambda)$$

Therefore, $\mathcal{H}$ is secure in the Evaluation-Oracle model. □

B.2 Proof of lemma 6: Amplifying an EO-robust PPH to a DO-robust PPH

Here we will restate the lemma.

Lemma 35. *Let $\mathcal{P}$ be a class of (total or partial) predicates on X. Assume that one-way functions exist. Then, any EO-robust PPH for $\mathcal{P}$ can be converted into a DO-robust PPH for $\mathcal{P}$.*

Proof. First, let OWFs exist. Then, PRP's also exist. So, let $m = \eta n$, and $\mathcal{F} = \{f : \{0,1\}^n \to \{0,1\}^m\}$ be a family of *strong* PRP's where each f_k is efficiently invertible with the key k. The characterization of strong here means that f_k^{-1} is also a PRP. Figure 2 details how to take an EO-robust PPH $\mathcal{H}$ and get a DO-robust PPH $\mathcal{H}^*$. It is easy to see that $\mathcal{H}^*$ satisfies the efficiency properties of the sampling algorithm, and since $\mathcal{F}$ is a PRP, $\mathcal{H}^*$ is also η-compressing. We still need to prove that this is robust. We will do this with a series of hybrids. Let $\mathcal{A}$ be an adversary against $\mathcal{H}^*$. Let $\mathcal{B}$ run $\mathcal{A}$ as a subroutine and have access to $\mathcal{O}_h$. Let $T = \mathrm{poly}(n)$ be the maximum number of queries $\mathcal{A}$ makes to $\mathcal{O}_{h^*}^{\mathsf{Hash}}$ and $\mathcal{O}_{h^*}^{\mathsf{Eval}}$ to break the correctness $\mathcal{H}^*$ with non-negligible probability.

- Hybrid 0. In this game, $\mathcal{A}$ makes all T hash and evaluation queries to $\mathcal{O}_{h^*}^{\mathsf{Hash}}$ and $\mathcal{O}_{h^*}^{\mathsf{Eval}}$ respectively. $\mathcal{B}$ outputs the (x, P) that $\mathcal{A}$ outputs at the end of its queries.
- Hybrid t. In this game, $\mathcal{A}$ makes the first $t-1$ queries to $\mathcal{O}_{h^*}^{\mathsf{Hash}}$ and $\mathcal{O}_{h^*}^{\mathsf{Eval}}$ appropriately. But, then $\mathcal{B}$ simulates every query from t to T as follows:

– For every hash query x, if x had already been queried before, $\mathcal{B}$ just sends the same answer as given before by $\mathcal{O}_{h^*}^{\mathsf{Hash}}$. If x has not been queried before, $\mathcal{B}$ chooses a random element y in the image of h^* that has not been seen before. $\mathcal{B}$ saves the pair (x, y) in memory.

– For every evaluation query y, if y is associated with some x as $h^*(x)$, then $\mathcal{B}$ queries $\mathcal{O}_h$ with the pair (x, P). $\mathcal{O}_h$ correctly returns $\mathcal{H}.\mathsf{Eval}(h, P, h(x)) = \mathcal{H}^*.\mathsf{Eval}(h^*, P, h^*(x))$. If y has not been associated with an x, $\mathcal{B}$ chooses a random element $x \in \{0,1\}$ that has not been queried/seen before, saves the pair (x, y). Then, $\mathcal{B}$ queries $\mathcal{O}_h(x, P)$.

$\mathcal{B}$ outputs the (x, P) that $\mathcal{A}$ outputs at the end of its queries.

- Hybrid T. $\mathcal{B}$ simulates the answer to every single query $\mathcal{A}$ makes as follows just as above. $\mathcal{B}$ outputs the (x, P) that $\mathcal{A}$ outputs at the end of its queries.

If $\mathcal{B}$ has a non-negligible probability of outputting (x, P) breaking the correctness of $\mathcal{H}^*$ in Hybrid T, then either $\mathcal{A}$ has non-negligible probability of outputting some (x, P) in Hybrid 0, or there exists a $t \in [T]$ where $\mathcal{A}$ has a noticeable gap in winning Hybrid t versus winning Hybrid $t-1$. Therefore, we can create an adversary $\mathcal{A}^*$ that can distinguish, with non-negligible probability, between Hybrids t and $t-1$. Moreover, the query made at that point must be a query x or y that has not been asked about before (otherwise there is no difference between the Hybrids). If the query is a hash query, then this implies $\mathcal{A}^*$ can distinguish between the PRP $f_k(h(x))$ and a truly random permutation. This cannot happen because of the pseudorandomness of f_k. If the query is an evaluation query on a y that we have not yet seen, then we assume it was associated with a random x not yet queried; $\mathcal{A}^*$ is distinguishing between $f_k^{-1}(y)$ and random. Because f_k is a strong PRP, f_k^{-1} is also a PRP, and therefore, distinguishing $f_k^{-1}(y)$ from random should also be impossible for PPT adversaries.

So, since any PPT algorithm in finding (x, P) such that $\mathcal{H}.\mathsf{Eval}(h, h(x), P) \neq P(x)$, $\mathcal{B}$ must also have negligible advantage, and therefore $\mathcal{A}$ also has negligible advantage. □

C Proofs for Section 3: PPH Lower Bounds from One-Way Communication Lower Bounds

C.1 Proof of theorem 10: OWC Lower Bounds Imply PPH Lower Bounds

Here is our proof that a lower bound in OWC complexity implies a lower bound for PPHs.

Theorem 36. *Let P be any two-input predicate P and $\mathcal{P} = \{P_x\}_{x\in\{0,1\}^n}$ be the corresponding predicate class where $P_{x_2}(x_1) = P(x_1, x_2)$. Now, let $\mathcal{H}$ be a PPH in any model for $\mathcal{P}$ that compresses n bits to $m = \eta n$. Then, there exists a OWC protocol Π such that the communication of Π is m and with negligible error.*

Proof. Let P'_x be the transformed predicate for P_x, so $P'_x(y_1) = \mathcal{H}.\mathsf{Eval}(h_r, y_1, P)$. Π will operate as follows:

- Alice computes $g_a(x_1; r) = h_r(x_1)$ where $h_r = \mathcal{H}.samp(1^\lambda; r)$ (runs the sampling algortihm with public randomness r).

- Bob computes $g_b(y, x_2; r) = \mathcal{H}.\mathsf{Eval}(h_r, y_1, P_{x_2})$ where Bob can also evaluate $h_r = \mathcal{H}.\mathsf{Samp}(1^\lambda; r)$ with the public randomness and can compute $P'_{x_2}(y_1) = \mathcal{H}.\mathsf{Eval}(h_r, y_1, P_{x_2})$.

First, the communication of Π is clearly m bits since Alice only sends a single hashed value of x_1 during the protocol.

Second, Π is correct with all but negligible probability. This follows directly from the soundness or correctness of the PPH — even a non-robust PPH has correctness with overwhelming probability. Formally, for any two inputs from Alice and Bob, x_1 and x_2 respectively,

$$\begin{aligned}&\Pr_r[g_b(g_a(x_1;r),x_2;r) = P(x_1,x_2)]\\ &\quad= \Pr_{h_r \leftarrow \mathcal{H}.\mathsf{Samp}(1^\lambda)}[P'_{x_2}(h_r(x_1)) = P(x_1,x_2)] \geq 1 - \mathrm{negl}(n).\end{aligned}$$

□

C.2 Proofs that INDEX$_n$, GREATERTHAN, and EXACTHAMMINGare Reconstructing

First, we will go over INDEX$_n$. It was already known that INDEX$_n$ had OWC complexity of n-bits for any negligible error [KNR95]. While the methods of Kremer et. al. give a lower bound relative to the error, we care about negligible error from our definition of PPHs.

Lemma 37. INDEX$_n$ *is reconstructing.*

Proof. The learning algorithm L is straightforward: for every $\mathbf{x} \in \{0,1\}^n$, $L^{O_\mathbf{x}}$ makes n static queries $P_1, \ldots, P_n$ where $P_j(x) = x_j$. After n queries, L has $(x_1, \ldots, x_n) = \mathbf{x}$. Note that L does not require adaptivity or randomness. □

Corollary 38. *There does not exist a PPH for* INDEX$_n$.

Now we will examine GREATERTHAN. GREATERTHAN is a problem where the deterministic lower bound is known to be exactly n, but no precise lower bound for randomized OWC protocols is known. Recall that for equality, we have the same deterministic lower bound, but a randomized protocol with negligible error can have significantly smaller OWC complexity $O(\lambda)$. The same will *not* be true of GREATERTHAN.

Lemma 39. GREATERTHAN *is reconstructing.*

Proof-sketch. For every $x \in [2^n]$, $L^{\mathcal{O}_x}$ is simply binary searching for x using the greater-than predicate. So, the first query is $\mathcal{O}_x(2^{n-1})$ and depending on the answer, the next query is either 2^{n-2} or $2^{n-1} + 2^{n-2}$, and so forth. There are a total of n queries, and from those queries L can exactly reconstruct x. □

Corollary 40. *There does not exist a PPH for* GREATERTHAN

Next, we turn to EXACTHAMMING, with parameter α.

Definition 41. *The* EXACTHAMMING$_\alpha$ *two-input predicate is defined as*

$$\text{EXACTHAMMING}_\alpha(x_1,x_2) = \begin{cases} 0 & \text{if } ||x_1 - x_2||_0 \leq \alpha \\ 1 & \text{if } ||x_1 - x_2||_0 > \alpha \end{cases}$$

While making the claim that EXACTHAMMING has OWC complexity of n bits follows from Theorem 14 in the following section, we are able to demonstrate the flexibility of reconstructing predicates; the proof of this lemma uses an L that is randomized.

Lemma 42. EXACTHAMMING$(n/2)$ *is reconstructing.*

Proof. This proof borrows techniques from [JKS08], where they showed that GAPHAMMING $(n/2, c\sqrt{n})$ required $\Omega(n)$ bits of communication, by reducing INDEX_n to an instance of this problem. We will have L use $\mathcal{O}_{\mathbf{x}}$ to create this same GAPHAMMING instance just as Alice and Bob separately computed it.

$L^{\mathcal{O}_{\mathbf{x}}}$ will use the following algorithm:

1. For every $i \in [n]$ and $j \in [m]$:
   (a) Use the randomness to generate a new random vector $\mathbf{r}_{i,j} \xleftarrow{\$} \{0,1\}^n$.
   (b) Let $b_{i,j} \leftarrow \mathbf{r}_{i,j}[i]$ and $a_{i,j} = 1 - \mathcal{O}_{\mathbf{x}}(\mathbf{r}_{i,j})$.
2. For every $i \in [n]$, let $\mathbf{x}'_i = (a_{i,1}, \ldots, a_{i,m})$ and $\mathbf{y}'_i = (b_{i,1}, \ldots, b_{i,m})$.
3. For every $i \in [n]$, let $\hat{x}_i = 1$ if $||\mathbf{y}'_i - \mathbf{x}'_i||_0 \leq n/2$ and $\hat{x}_i = 0$ otherwise.
4. Return $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n)$.

This algorithm is exactly the algorithm Alice and Bob use in the proof that Gap-Hamming requires n bits of communication in appendix C.3.1: L acts out Alice's part by using $\mathcal{O}_{\mathbf{x}}$ to compute exact-hamming between $\mathbf{r}_{i,j}$ and $\mathbf{x}$, and acts out Bob's part by just taking the i'th coordinate from that random vector as the guess for the i'th bit of $\mathbf{x}$. Now, without generality assume n is odd, and the analysis is then the same:

- Assume $x_i = 1$. Then, $\mathbb{E}_{\mathbf{r}_{i,j}}[||\mathbf{x}'_i - \mathbf{y}'_i||_0] \leq \frac{n}{2} - \frac{\sqrt{2\pi}}{e^2}\sqrt{n}$, and so as long as $m = O(n^2)$, a Chernoff bound yields $\Pr[||\mathbf{x}'_i - \mathbf{y}'_i||_0 \geq \frac{n}{2}] \leq e^{-O(n)} = \text{negl}(n)$. And so, the probability that we guess x_i is 0 when $x_i = 1$ is negligible.
- Assume $x_i = 0$. We have $\mathbb{E}_{\mathbf{r}_{i,j}}[||\mathbf{x}'_i - \mathbf{y}'_i||_0] \leq \frac{n}{2} + \frac{\sqrt{2\pi}}{e^2}\sqrt{n}$. Again, as long as $m = O(n^2)$, a Chernoff bound yields $\Pr[||\mathbf{x}'_i - \mathbf{y}'_i||_0 \leq \frac{n}{2}] \leq e^{-O(n)} = \text{negl}(n)$.

And with that, the chance that we guess x_i incorrectly is negligible. □

Corollary 43. *There does not exist a PPH for* $\text{EXACTHAMMING}(n/2)$.

C.3 Proofs of Lower bounds for Gap-Hamming and Gap-GreaterThan

C.3.1 Proof that Gap-Hamming Requires n bits of Communication

Here is the full proof that Gap-Hamming Requires n bits of Communication. Recall the definition of the GAPHAMMING problem.

Definition 44. *The* $\text{GAPHAMMING}(n, d, \epsilon)$ *promise predicate acts on a pair of vectors from* $\{0,1\}^n$ *and is defined as follows.*

$$P(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} 0 & \text{if } ||\mathbf{x}_1 \oplus \mathbf{x}_2||_0 \leq d(1-\epsilon) \\ 1 & \text{if } ||\mathbf{x}_1 \oplus \mathbf{x}_2||_0 \geq d(1+\epsilon) \\ \circledast & \text{otherwise} \end{cases}$$

where $||\cdot||_0$ *denotes the* ℓ_0 *norm, or equivalently Hamming weight.*

Theorem 45. *The randomized OWC complexity of* $\text{GAPHAMMING}_n(n/2, \sqrt{n}/2)$ *with negligible error is exactly* n*;*

$$R^{A \to B}_{\text{negl}(n)}(\text{GAPHAMMING}(n, n/2, 1/\sqrt{n})) = n.$$

Proof. This will be a randomized reduction of INDEX_n to GAPHAMMING for an arbitrary error δ; we will show that this randomized reduction introduces negligible error, and so if we want negligible error for GAPHAMMING on these parameters, we require $\delta = \text{negl}(n)$, too. We will take Alice's input $\mathbf{x}$ and Bob's index $i \in [n]$ and create two new vectors, $\mathbf{a}$ and $\mathbf{b}$ n-bit vectors, correlated using the public randomness so that if $x_i = 1$, $\mathbf{a}$ and $\mathbf{b}$ will be within $n/2 - \sqrt{n}/2$ distance from each other and if $x_i = 0$, the vectors will be at least $n/2 + \sqrt{n}/2$ distance with all but negligible probability (over n).

Without loss of generality, assume n is odd.

- For each coordinate b_j, Bob samples the same public randomness $\mathbf{r}_j \leftarrow \{0,1\}^n$ and sets $b_j \leftarrow r_j$. Bob is essentially pretending $\mathbf{r}_j$ is Alice's vector.
- For each coordinate a_j, Alice samples the public randomness $\mathbf{r}_j \xleftarrow{\$} \{0,1\}^n$. If $||\mathbf{x} - \mathbf{r}_j||_0 < n/2$, she sets $a_j \leftarrow 1$, and if $||\mathbf{x} - \mathbf{r}_j||_0 > n/2$, she sets $a_j \leftarrow 0$. Alice is marking if $\mathbf{r}_j$ is a good proxy for $\mathbf{x}$.

We now need to argue that $\mathbf{a}$ and $\mathbf{b}$ are close if $x_i = 1$ and far otherwise. We will do this by computing the expected hamming distance between $\mathbf{a}$ and $\mathbf{b}$, and then applying a Chernoff bound. Let's go through both cases.

- Assume $x_i = 1$. Then, $\mathbb{E}_{\mathbf{r}}[||\mathbf{a} - \mathbf{b}||_0] = \sum_{j=1}^{n} \Pr_{\mathbf{r}}[a_j \neq b_j]$. Now, looking at $\Pr_{\mathbf{r}}[a_j \neq b_j]$, we have the two more cases. Either $\mathbf{x}$ and $\mathbf{r}_j$ agree on *strictly* less than or greater than $(n-1)/2$ bits (meaning r_i is not used in determining a_j); or, $\mathbf{x}$ and $\mathbf{r}_j$ agree on exactly $(n-1)/2$ bits. In the second case, since $x_i = 1$, the probability that $b_j = a_j$ is 1. So,

$$\Pr_{\mathbf{r}}[a_j \neq b_j] = \Pr[\text{Case 1}] \cdot \frac{1}{2} - \Pr[\text{Case 2}] \cdot 0$$

  Using Stirling's approximation, we get that $\Pr[\text{Case 2}] = \frac{c\sqrt{2}}{\sqrt{n-1}}$, where $\frac{2\sqrt{\pi}}{e^2} \leq c \leq \frac{e}{\pi\sqrt{2}}$. And so,

$$\begin{aligned}\Pr_{\mathbf{r}}[a_j = b_j] &= \left(1 - \frac{c\sqrt{2}}{\sqrt{n-1}}\right) \cdot \frac{1}{2} \\ &\leq \frac{1}{2} - \frac{c}{\sqrt{2n}}\end{aligned}$$

  Now, when we commute the expected hamming distance if $x_i = 1$, we get

$$\mathop{\mathbb{E}}_{\mathbf{r}}[||\mathbf{a} - \mathbf{b}||_0] = \sum_{j=1}^{n} \Pr_{\mathbf{r}}[a_j \neq b_j] \leq n \cdot (\frac{1}{2} - \frac{c}{\sqrt{2n}}) = \frac{n}{2} - \frac{c\sqrt{n}}{\sqrt{2}}$$

  Plugging in the lower bound for c we computed with Stirling's approximation, we have

$$\mathop{\mathbb{E}}_{\mathbf{r}}[||\mathbf{a} - \mathbf{b}||_0] \leq \frac{n}{2} - \frac{\sqrt{2\pi}}{e^2} \cdot \sqrt{n}$$

  Now, using a Chernoff bound, we get that $\Pr[||\mathbf{a} - \mathbf{b}||_0 > \frac{n}{2} + \frac{\sqrt{n}}{4}] \leq e^{-O(n)} = \text{negl}(n)$.

- Assume $x_i = 0$. We will use the same analysis as before, but now in the second case, we have $x_i = 0$, so the probability that $r_i = a_j$ is 0. And hence,

$$\mathop{\mathbb{E}}_{\mathbf{r}}[||\mathbf{a} - \mathbf{b}||_0] \geq \frac{n}{2} + \frac{\sqrt{2\pi}}{e^2} \cdot \sqrt{n}$$

  Again, using a Chernoff bound, we get that $\Pr[||\mathbf{a} - \mathbf{b}||_0 < \frac{n}{2} + \frac{\sqrt{n}}{4}] \leq e^{-O(n)} = \text{negl}(n)$.

Therefore, with all but negligible probability in n, this randomized reduction is correct. □

C.3.2 Proof that Gap-k Greater-Than requires $n - \log(k) - 1$ bits

First, let us recall the definition of the Gap-k greater-than predicate P.

$$P_{x_1}(x_2) = \begin{cases} 1 & \text{if } x_1 > x_2 \text{ and } |x_1 - x_2| > k \\ 0 & \text{if } x_1 \leq x_2 \text{ and } |x_1 - x_2| > k \\ * & \text{otherwise} \end{cases}$$

Theorem 46. *For $k \geq 1$, the OWC complexity of gap-k* GREATERTHAN *is*

$$R^{A \to B}_{\text{negl}(n)}(\mathcal{P}) = n - \log(k) - 1$$

and in fact there exists a protocol compressing by $\log(k) + 1$ bits that has no error.

Proof. Assume k is a power of 2. All other k follow: we will be unable to compress by more than $\lceil \log(k) \rceil + 1$ bits. Let P be the Gap-k GREATERTHAN predicate.

Designing a PPH. The proof that we can compress by $\log(k) + 1$ is simply that our hash function h just removes the last $\log(k) + 1$ bits. However, P' must do a little work:

- Let $L = h^{-1}(h(x)) = \{x_0, x_1, \ldots, x_{2k}\}$ be the list, in order, of all elements mapping to $h(x)$, where $x_0 + 2k = x_1 + 2k - 1 = \cdots = x_{2k}$.
- If $a \leq x_k$, $P'_a(h(x)) = 0$, and if $a > x_k$, $P'_a(h(x)) = 1$.

First we show this algorithm is correct. For every $x, a \in [2^n]$, if $P_a(h(x)) = 0$, then $a \leq \min h^{-1}(h(x)) + k \leq x + k$. If $a < x$, then $P_a(h(x))$ answers correctly, but if $a > x$, we get that $|x - a| \leq k$, and so our output is still alright since a is within the gap around x. Similarly, if $P_a(h(x)) = 1$, then $a > \max h^{-1}(h(x)) - k \geq x - k$. For the same reasons the output is either correct or within the gap.

Lower bound. Now we want to show that if h compresses by more than $\log(k) + 1$ bits, we can non-adaptively find two inputs such that $P'_a(h(x)) \neq P_a(x)$ with non-negligible probability. In fact we will show that we can guess a, x with probability at least $\frac{1}{400n}$. Our method will be first to guess an x that is "bad" (collides with an x' more than $\frac{5}{2}k$ from it), guess if it is smaller or bigger than x' (b), and then guess by how much x' is smaller or bigger than x ($2^{s-1} \leq |x - x'| \leq 2^s$):

1. $x \xleftarrow{\$} [N]$
2. $b \xleftarrow{\$} \{0, 1\}$ and $s \xleftarrow{\$} \{\log k + 1, \ldots, n\}$.
3. If $b = 0$: $a \xleftarrow{\$} \{x - 2^s, \ldots, x - (k + 1)\}$ and output x and a.
   If $b = 1$: $a \xleftarrow{\$} \{x + (k + 1), \ldots, x + 2^s\}$ and output x and a.

Let h be any function hashing n bits to $n - (\log(k) + 2)$ bits (again assume k is a power of 2). Let $K = 3.5k$, we want to bound the probability we choose a random input x and end up with $h(x)$ having a pre-image size at least size K:

$$\begin{aligned} \Pr_{x \xleftarrow{\$} [N]}\left[|h^{-1}(h(x))| \geq K\right] &= 1 - \Pr_{x \xleftarrow{\$} [N]}\left[|h^{-1}(h(x))| < K\right] \\ &\geq 1 - (K - 1)(2^{n - \log(k) - 1} - 1)2^{-n} \\ &\geq 1 - (K - 1)(\frac{1}{4k}) \\ &\geq 1 - \frac{3.5k}{4k} = \frac{1}{8} \end{aligned}$$

Consider any preimage $h^{-1}(y)$ of size at least $3.5k$: if we sort the set $h^{-1}(y)$, then pair off the first x in the sorted list with the $\frac{5}{2} \cdot k$'th, the second with the $\frac{5}{2}k + 1$'th and so on, then choosing $x \xleftarrow{\$} h^{-1}(y)$, with probability $\frac{4}{7}$, x will have an x' it is paired with. For all a in between x and x', and at least distance k from *both* of them, $P'_a(h(x))$ is wrong more often than $P'_a(h(x'))$ or vice-versa. For each x, x' pair, consider all a in between x and x' and at least distance k from both of them: $P'_a(h(x)) = P'_a(h(x'))$. So, for one of x or x', this will evaluate incorrectly. Therefore, one of x or x' will have that, for at least half of the a's in between and distance k from both, P'_a evaluates incorrectly. We also have that since x and x' are at least $\frac{5}{2}k$ apart, there are at least $\frac{k}{2}$ elements a that are distance k from both of them. If we choose at random from elements at least distance k from x, we get the probability of choosing an a at distance k from both x and x' is $\frac{1}{3}$.

We will call an x *bad* if it has an x' such that $h(x) = h(x')$ and $|x - x'| \geq 3.5k$ (which we call 'paired'), and for all the a in between x and x' and distance k from both, more than half of them evaluate incorrectly on x.

$$\begin{aligned}
\Pr_x[x \text{ is bad}] &\geq \Pr[x \text{ is bad} || h^{-1}(h(x))| \geq K] \cdot \Pr[|h^{-1}(h(x))| \geq K] \\
&\geq \Pr[x \text{ is bad} || h^{-1}(h(x))| \geq K \wedge \exists \text{ paired } x'] \cdot \Pr[\exists \text{ paired } x' || h^{-1}(h(x))| \geq K] \cdot \frac{1}{8} \\
&\geq \Pr[x \text{ is bad} || h^{-1}(h(x))| \geq K \wedge \exists \text{ paired } x' \wedge x \text{ has more incorrect } a] \\
&\quad \cdot \frac{1}{2} \cdot \frac{4}{7} \cdot \frac{1}{8} \\
&\geq \frac{1}{28}
\end{aligned}$$

Now, assume that we have chosen a bad x. We will compute the probability we choose an a that $P_a(x) \neq P'_a(h(x))$. Consider x and its pair x': x is wrong on at least half of the a between x and x', so our goal is to sample between x and x' without knowing x'. First, we guess whether $x < x'$ or vice-versa (our choice of the bit b). Then, we guess how far apart they are to the nearest power of 2 (our choice of $s \in [n]$). Finally, if we have guessed both of these correctly, we sample in the range $x \pm s$, and with probability at least $1/2$ we are sampling in the range (x, x'), and again with probability at least $1/2$, we sample an a that evaluates incorrectly for x. Formally, we have:

- Assume x is bad, and so is paired with an x'. $\Pr_{x,b,s,a}[P'_a(h(x)) \neq P_a(x)] \geq \Pr_{x,b,s,a}[P'_a(h(x)) \neq P_a(x) | b \text{ is correct}] \cdot \frac{1}{2}$.
- Now assume that both x is bad and b is chosen correctly (that is, we know $x < x'$ or $x' < x$). $\Pr_{x,b,s,a}[P'_a(h(x)) \neq P_a(x)] \geq \Pr_{x,b,s,a}[P'_a(h(x)) \neq P_a(x) | 2^{s-1} < |x - x'| \leq 2^s] \cdot \frac{1}{n}$.
- Assume x is bad, b is chosen correctly, and s is also guessing the range between x and x' correctly. We have $\Pr_{x,b,s,a}[P'_a(h(x)) \neq P_a(x)] \geq \Pr_{x,b,s,a}[P'_a(h(x)) \neq P_a(x) | a \in (x, x')] \cdot \frac{1}{2}$ since $\Pr_{x,b,s,a}[a \in (x, x')] \geq \frac{1}{2}$ given $a \in (x, x + 2^s)$ or $a \in (x - 2^s, x)$ when $b = 0$ or $b = 1$ respectively.
- Assume we have chosen an $a \in (x, x')$ or (x', x) (whichever is correct). The probability that $|x - a|$ and $|x' - a| > k$ is at least $\frac{1}{3}$ (since we guarantee that a is at least distance k from x).
- Finally, assume all of the previous points. We get $\Pr_{x,b,s,a}[P'_a(h(x)) \neq P_a(x)] \geq \frac{1}{2}$ because x is bad and we are choosing $a \in (x, x')$ or (x', x) (whether $b = 0$ or 1) where a is distance more than k from both x and x'. So, $P'_a(x)$ will evaluate incorrectly on at least half of all such a's.

- Putting all of these conditionals together (multiplying them), we have

$$\Pr_{x,b,s,a}[P'_a(h(x)) \neq P_a(x)] \geq \frac{1}{28} \cdot \frac{1}{2} \cdot \frac{1}{n} \cdot \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{336n}.$$

This completes the proof: if we try to compress more than $\lceil \log(k) \rceil + 1$ bits, we end up being able to use the above attack to find a bad input on the hash function with probability at least $\frac{1}{400n}$. □

A Different Lower Bound for Two-input Greater-Than. Here we will intuitively describe why the lower bound for Two-Input Greater-Than is $n - \log(k)$, one bit less than the lower bound on the single-input version. Consider the construction for a single-input gap-k Greater-Than PPH, as described in the proof above. $h(x)$ removes the last $\log(k) + 1$ bits from x, and we are able to compare a value a to $h^{-1}(x)$, checking if a is in the lower half or the higher half. If we instead are only given $h(x)$ and $h(a)$, we can only check that a is in the $h^{-1}(x)$, and have no sense of where.
So, in the two-input case, we have a simple upper bound: our hash $h'(x)$ removes exactly the $\log(k)$ lowest bits, and $P'(h(x), h(a))$ simply compare $h(x) > h(a)$. The proof that this is optimal follows the same structure as above, except we let $K = 2k$. Now, if our adversary finds a random collision, $h(x) = h(a)$, there is a large enough chance that $|x - a| > k$, which would violate the correctness of the PPH.

D Proofs for Section 4

Here we will include the omitted proofs for our Gap-Hamming PPH constructions. These proofs tend to involve basic Chernoff bound calculations.

D.1 Proofs for Subsection 4.1: CRHFs for a Gap-Hamming PPH

Here is the proof of the lemma left out in Section 4.1.

Lemma 47. *For n sufficiently large, $k = O(n)$, constant $a \in (0, 1)$, $\gamma = o(\frac{k}{n \log(n/k)})$, and $D = \Theta(\log(1/\gamma))$. For any constant $\delta \in (0, 1)$, δ-biased (n, k, D, γ, aD)-expanders exist and are constructible in time $\mathrm{poly}(n)$.*

Proof. First, we will show that with the right parameter settings for D and γ, we can show δ-biased (n, k, D, γ, aD)-expanders exist via random sampling. In fact, we will show that with constant probability p, we will sample such an expander. Then, we will show that with probability at least $1 - \mathrm{negl}(n)$, the graph we sample via this method is δ-balanced. So, the probability we will sample a graph that is both an (n, k, D, γ, aD)-expander and δ-biased is a union bound: $p - \mathrm{negl}(n)$. This means that we will sample a δ-biased (n, k, D, γ, aD)-expander with constant probability.

Sampling an expander with constant probability. First we will show that we can sample these expanders with constant probability. We will bound the following sampling procedure: for each node in L, uniformly sample D distinct neighbors in R and add those edges. We have that for any set $S \subseteq [n]$ and $K \subseteq [k]$,

$$\Pr[N(S) \subseteq K] \leq \left(\frac{|K|}{k}\right)^{D \cdot |S|}$$

since for this upper bound, we can just model this process as every node in L choosing D edges to R independently.

G is *not* a (n, k, D, γ, aD)-expander if there exists a subset S of R such that $|S| \leq \gamma n$, and a subset $K \subseteq [k]$, $|K| < aD|S|$ such that $N(S) \subseteq K$. We will upper bound this probability with a constant. We start by turning this probability into an infinite series:

$$\begin{aligned}
\Pr_G[G \text{ is not a } (\gamma, k, D, \gamma, aD)\text{-expander}] &\leq \sum_{S,|S|\leq\gamma n} \sum_{K,|K|=aD|S|} \Pr_G[N(S) \subseteq K] \\
&\leq \sum_{s=1}^{\gamma n} \sum_{S,|S|=s} \sum_{K,|K|=aDs} \Pr_G[N(S) \subseteq K] \\
&\leq \sum_{s=1}^{\gamma n} \binom{n}{s} \cdot \binom{k}{aDs} \cdot \left(\frac{aDs}{k}\right)^{Ds} \\
&\leq \sum_{s=1}^{\gamma n} \left(\frac{ne}{s}\right)^{s} \cdot \left(\frac{ke}{aDs}^{aDs}\right) \cdot \left(\frac{\alpha s}{k}\right)^{Ds} \\
&\leq \sum_{s=1}^{\gamma n} x'^{s} \text{ for } x' = \left(\frac{ne}{s}\right)\left(\frac{ke}{aDs}\right)^{aD}\left(\frac{aDs}{k}\right)^{D}
\end{aligned}$$

Notice that for s varying from 1 to γn, this quantity is maximized when $s = \gamma n$. So, let $x = \left(\frac{e}{\gamma}\right)\left(\frac{ke}{aD\gamma n}\right)^{aD}\left(\frac{aD\gamma n}{k}\right)^{D}$, where $x' \leq x$. We will interpret this as a geometric series and get

$$\sum_{s=1}^{\gamma n} x'^{s} \leq \sum_{s=1}^{\gamma n} x^{s} \leq \sum_{s=1}^{\infty} x^{s} = \frac{x}{1-x} \text{ if } |x| < 1.$$

We need this upper bound to be some constant less than 1, which means we want x to be a constant strictly less than $\frac{1}{3}$. This is where our restrictions on k, γ, and D will come into play.

$$\begin{aligned}
\left(\frac{e}{\gamma}\right)\left(\frac{ke}{aD\gamma n}\right)^{aD}\left(\frac{aD\gamma n}{k}\right)^{D} &< \frac{1}{3} \\
\left(\frac{e}{\gamma}\right)e^{D}\left(\frac{aD\gamma n}{k}\right)^{D(1-a)} &< \frac{1}{3} \\
e^{D}\left(\frac{aD\gamma n}{k}\right)^{D(1-a)} &< \frac{\gamma}{3e}
\end{aligned}$$

Now, we have $k = o(n)$. Assume that we have $\gamma = o(k/(n(\log(n/k))))$ and $D = O(\log(1/\gamma)))$. With these parameters, we have $D\gamma = o(k/n)$. This is fairly easy to show: if $\gamma = o(k/n(\log(n/k)))$, then $\gamma = k/(nf(\log(n/k)))$ for some $f(a) = \omega(a)$, and we have two cases to analyze:

- $\log(n/k) \geq \log(f(\log(n/k)))$. Now, because $f(a) = \omega(a)$, this means $f(\log(n/k)) = \omega(\log(n/k))$. And given the inequality, $f(\log(n/k)) = \omega(\log(n/k) + \log(f(\log(n/k)))$.
- $\log(n/k) < \log(f(\log(n/k)))$. So, $f(\log(n/k)) = \omega(\log(f(\log(n/k))))$ (and is exponentially larger). Given the inequality, this implies $f(\log(n/k)) = \omega(\log(n/k) + \log(f(\log(n/k)))$.

In both cases, we have $f(\log(n/k)) = \omega(\log(n/k) + \log(f(\log(n/k)))$. This yields

$$D\gamma = \frac{\log(n/k) + \log(f(\log(n/k)))}{f(\log(n/k))} \cdot \frac{k}{n} = o(1) \cdot \frac{k}{n} = o(k/n).$$

$D\gamma = o(k/n)$ gives us

$$\frac{aD\gamma n}{k} = \frac{a \cdot c_1 \log(1/\gamma) \cdot c_2 \gamma n}{k} = \frac{o(k)}{k} = o(1).$$

Therefore, for any constant $0 < b < 1$ and sufficiently large n, we have $\frac{aD\gamma n}{k} < b$. Substituting that term with b, we have

$$\begin{aligned} e^D \left(\frac{aD\gamma n}{k}\right)^{D(1-a)} < e^D \cdot b^{D(1-a)} &< \frac{\gamma}{3e} \\ D(\log(e) + (1-a)\log(b)) &< \log(\gamma) - \log(3e) \\ D((1-a)\log(1/b) - \log(e)) &> \log(1/\gamma) + \log(3e) \\ D &> \frac{\log(1/\gamma) + \log(3e)}{(1-a)\log(1/b) - \log(e)} \end{aligned}$$

Now, since we can use any constant b, we will choose b to make the denominator a positive constant: let $b > e^{(a-1)}$, and we have that

$$D = \Omega(\log(1/\gamma)).$$

Given these matching upper and lower bounds for D, we have $D = \Theta(\log(1/\gamma))$.

Sampling a Balanced Expander. This will be a simple application of a Chernoff bound. Consider the expected value of the degree of nodes on the right: $\mathbb{E}_G[|N(v)|] = nD/k$. Notice that $|N(v)| = \sum_{u \in L} \mathbb{1}(v \in N(u))$, and for every $u \in L$ and $v \in R$, we have $\Pr_G[v \in N(u)] = D/k$, which is independent for every $u \in L$. A Chernoff bound tells us for every $v \in R$ and any constant $0 \leq d \leq 1$, $\Pr_G[|N(v)| \leq (1-\delta)nD/k] \leq \exp[\frac{-\delta^2 nD}{2k}]$. Because $k = o(n)$ and δ is constant, $\exp[\frac{-\delta^2 nD}{2k}] = 2^{-n^{O(1)}} = \mathrm{negl}(n)$. Now, via a simple union bound over all $v \in R$, we have

$$\begin{aligned} \Pr_G[\exists v \in R, |N(v)| < (1-\delta)nD/k] &\leq \sum_{v \in R} \Pr_G[|N(v)| < (1-\delta)nD/k] \\ &\leq k \cdot \exp\left[\frac{-\delta^2 nD/k}{2}\right] \\ &= k/2^{n^{O(1)}} = \mathrm{negl}(n). \end{aligned}$$

Therefore, the probability that our random graph is δ-balanced is at least $1 - \mathrm{negl}(n)$.

This completes the proof: we can simply sample a random D-left-regular bipartite graph, check if it is a balanced expander with the desired parameters, and with constant probability it will be. □

E Proof of Theorem 28: A Double-Oracle Equality PPH implies OWFs

We will show that without OWFs, given *any* compressing family of functions $\mathcal{H}$, we can find a collision given only an oracle to $h \in \mathcal{H}$ using algorithm 2, and finding such a collision is equivalent to finding a pair of inputs breaking the guarantees of the PPH.

Before our proof, we will need to define bins.

Definition 48. *A bin $B_y \subseteq \{0,1\}^n$ is defined by an element in $y \in Im(g)$ for some function g: $B_y = \{x : g(x) = y\}$.*

We will typically fix a bin and analyze the properties of that bin, so we will drop the subscript.

We will also assume that OWFs do not exist, which allows us to have non-uniform inverters.

Theorem 49 ([GIL+90]). *If OWFs do not exist, then weak OWFs do not exist. This means that for every function $f : \{0,1\}^n \to \{0,1\}^m$, there exists a non-uniform inverter $\mathcal{A}_f$ and a polynomial Q,*

$$\Pr_x \left[y \leftarrow f(x), x' \leftarrow \mathcal{A}_f(y) : f(x') = y \right] \geq 1 - \frac{1}{Q(n)}.$$

Here is an overview of our proof: we will use our oracle access to h and our ability to invert function to produce an approximation h' of h. We then need to have a noticeable probability of choosing an element x and getting an inverse x' from an inverter $\mathcal{A}_h$ such that neither x nor x' disagree between h and h'. The way to ensure this is to think of the output bin defined by x and bound the fraction of elements that are bad (that disagree between h and h'). Since we may have some bad elements in our bin, even if they are a small fraction, an arbitrary inverter of h' might always choose to give us inverses that disagree between h and h'. So, we will want to split the bin into sub-bins using a pairwise independent hash function $f \xleftarrow{\$} \mathcal{F}_k$ for some k – lemma 53 tells us that there exists a $k \in [n]$ so that there are very few bad elements in our sub-bin in expectation.

Algorithm 2: FindCollisions$(\mathcal{O}_h, \{\mathcal{F}_k\}_{k\in[n]}, \mathcal{A})$

Input: $\mathcal{O}_h$ is an oracle for h. We have $n-3$ families of pairwise independent hashes: $\mathcal{F}_k = \{f : \{0,1\}^n \to \{0,1\}^k\}$. We also have a set of $n-3$ non-uniform PPT inverters $\mathcal{A} = \{\mathcal{A}_{g_k}\}$ for each $g_k \in \{0, \cdots, n-3\}$ for $g_k(h, f_k, \mathbf{x}, x) = f_k, \mathbf{x}, h(\mathbf{x}), f_k(x)||h(x)$ where $f_k \leftarrow \mathcal{F}_k$.

Output: (x, x') so that $x \neq x'$ and $h(x) = h(x')$ or $\perp$ on failure.

Choose $\mathbf{x} = x_1, \cdots, x_t \xleftarrow{\$} \{0,1\}^n$ and $x \xleftarrow{\$} \{0,1\}^n$;
Compute $\mathbf{y} = (y_1 = \mathcal{O}_h(x_1), \cdots, y_k = \mathcal{O}_h(x_k))$;
Choose $k \xleftarrow{\$} \{0, \cdots, n-3\}$;
Choose $f \xleftarrow{\$} \mathcal{F}_k$;
Compute $f(x)||\mathcal{O}_h(x) = f(x)||h(x)$;
Compute $h', f, x_1, \cdots, x_t, x' \leftarrow \mathcal{A}_{g_k}(f, \mathbf{x}, \mathbf{y}, f(x)||h(x))$ or output $\perp$ if $\mathcal{A}_{g_k}$ fails ;
if $x \neq x'$ *and* $\mathcal{O}_h(x) = \mathcal{O}_h(x')$ **then**
  return (x, x')
end
else
  return $\perp$
end

Theorem 50. *Double-Oracle robust PPHs for equality imply OWFs.*

Proof. Throughout this proof, we will reference a few lemmas proved later. We will present this proof first to motivate the lemmas. We will prove that algorithm 2 finds collisions in any compressing function with noticeable probability.

To put this all together, let's label some events:

- InverseSuccess(IS) is the event that $\mathcal{A}_{g_k}$ outputs a correct inverse.

- Collision is the event we get a collision with algorithm 2.
- GoodApprox is the event that $\mathcal{A}_{g_k}$ produces an h' that disagrees with h on at most $1/(100n^c)$-fraction of inputs. A *bad* element is an element x such that $h(x) \neq h'(x)$.
- GoodBin is the event that the x we chose landed in a bin with at most $1/n^c$ bad elements and has more than 1 element.
- CleanSubBin is the event that after choosing f from $\mathcal{F}_k$ for $k = \max(0, i-3)$, we end up in a sub-bin with no bad elements and more than one element. Otherwise we refer to the sub-bin as dirty.

We will first assume that $\mathcal{A}_{g_k}$ succeeds and that $k = \max(0, i-3)$. We will compute the probability of failure using a union bound later, and since k is chosen independently at random of all other events, we will include that with probability $1/n < 1/(n-3)$.

We dissect the probability, over our choice of $\mathbf{x} \xleftarrow{\$} \{0,1\}^{n \times t}$, $x \xleftarrow{\$} \{0,1\}^n$, and $f \xleftarrow{\$} \mathcal{F}_k$ where $k = \max(0, i-3)$, of getting a collision *given* InverseSuccess; e.g. in the next lines $\Pr[\mathsf{Collision}]$ really means $\Pr[\mathsf{Collision}|\mathsf{IS}]$:

$$\begin{aligned}
\Pr_{\mathbf{x},x,f}[\mathsf{Collision}] &\geq \Pr[\mathsf{Collision}|\mathsf{CleanSubBin}] \cdot \Pr[\mathsf{CleanSubBin}] \\
&\geq \Pr[\mathsf{Collision}|\mathsf{CleanSubBin}] \cdot \Pr[\mathsf{CleanSubBin}|\mathsf{GoodBin}] \cdot \\
&\quad \Pr[\mathsf{GoodBin}] \\
&\geq \Pr[\mathsf{Collision}|\mathsf{CleanSubBin}] \cdot \Pr[\mathsf{CleanSubBin}|\mathsf{GoodBin}] \cdot \\
&\quad \Pr[\mathsf{GoodBin}|\mathsf{GoodApprox}] \cdot \Pr[\mathsf{GoodApprox}]
\end{aligned}$$

Now, let's analyze each of these probabilities one-by-one, starting with $\Pr[\mathsf{GoodApprox}|\mathsf{IS}]$.

- $\Pr[\mathsf{GoodApprox}|\mathsf{IS}] \geq 99/100$.

  This is a straight application of lemma 51. We guarantee that with probability $99/100$, h' agrees with h on all but $1/p(n)$-fraction of inputs for any polynomial $p(n)$ depending on our number of queries t. Now, we have chosen $100n^c$ for our polynomial.

- $\Pr[\mathsf{GoodBin}|\mathsf{GoodApprox} \wedge \mathsf{IS}] \geq 49/100$.

  From lemma 52, we know that $99/100$-fraction of our inputs end up in a bin where at most $100/(100n^c) = 1/n^c$-fraction of the inputs disagree between h and h'. Now, we could end up in a bin with only one element, but at most $1/2$ of the elements could map to bins of size 1. This is because if we compress even just by one bit and map as many of our 2^n elements to bins with a single element in them, we can map the first $2^{n-1}-1$ to single-element bins, but the last bin must contain the rest, and $\frac{2^{n-1}-1}{2^n} < \frac{1}{2}$. Compressing by more bits only decreases this fraction.

  Therefore, $\Pr[\mathsf{GoodBin}|\mathsf{GoodApprox} \wedge \mathsf{IS}] = 1 - \Pr[\text{only one element in bin or in a bad bin}]$. By a union bound, this gives us $1 - (1/2 + 1/100) = 49/100$.

- $\Pr[\mathsf{CleanSubBin}|\mathsf{GoodBin} \wedge \mathsf{IS}] \geq 79/100$.

  From lemma 53, we know that our sub-bin contains bad elements with probability at most $16/n^c$. Assume we have chosen c large enough so that $16/n^c \leq 1/100$. Then, from lemma 55, we know that with probability at least $8/10$, we have more than 1 element. So, $\Pr[\mathsf{CleanSubBin}|\mathsf{GoodBin} \wedge \mathsf{IS}] = 1 - \Pr[\text{only element in sub-bin or in a dirty sub-bin}]$. By a union bound this is at least $1 - (2/10 + 1/100) = 79/100$.

- $\Pr[\mathsf{Collision}|\mathsf{CleanSubBin} \wedge \mathsf{IS}] \geq \frac{1}{2}$.

  We are guaranteed to have at least two elements in our sub-bin. So, when $\mathcal{A}_{g_k}$ produces a pre-image for $h'(x)||f(x)$, then $\mathcal{A}$ has probability at most $1/2$ of giving us $x' = x$, but we are guaranteed that $h'(x') = h(x')$ since we are in a clean sub-bin.

This means,

$$\Pr[\mathsf{Collision}|\mathsf{IS}] \geq \frac{1}{2} \cdot \left(\frac{79}{100} \cdot \frac{49}{100} \cdot \frac{99}{100}\right) \geq \frac{19}{100}$$

We're almost done. We need to finish this analysis by consider the case that $\mathcal{A}_g$ does not find an inverse and that we correctly chose $k = \min(0, i-3)$. First, note that $\Pr[\neg\mathsf{IS}] \leq 1/100$ from theorem 49 because the input to $\mathcal{A}_g$ looks like a random output from g. Finally,

$$\begin{aligned}\Pr[\mathsf{Collision}] &\geq \Pr[\mathsf{Collision}|\mathsf{IS}] \cdot \Pr[\mathsf{IS}] \cdot \Pr[\ k = \min(0, i-3)\ ] \\ &\geq \frac{19}{100} \cdot \frac{99}{100} \cdot \frac{1}{n} \geq \frac{18}{100n}\end{aligned}$$

□

Helpful lemmas Here we will go through the lemmas that make our analysis possible. We will start by showing that with polynomially many queries t, we have a 99/100 chance of getting an approximate h' that agrees on almost all queries with h.

Lemma 51. *Let $p(n)$ be any polynomial and assume OWFs do not exist. Let $t \geq (r+7)p(n)$, and we define a function $g(h, x_1, \ldots, x_t) = x_1, \ldots, x_t, h(x_1), \ldots, h(x_t)$. Let $\mathbf{x} = (x_1, \cdots, x_t) \xleftarrow{\$} \{0,1\}^{t \cdot n}$. For any non-uniform inverter $\mathcal{A}_g$ that succeeds in producing an inverse h',*

$$\Pr_{h,\mathbf{x}}\left[h' \leftarrow \mathcal{A}_g(\mathbf{x}, \mathbf{y}) : \Pr_x\left[h(x) = h'(x)\right] \geq 1 - 1/p(n)\right] \geq 98/100.$$

Proof. Let $r = n^{O(1)}$ be the number of bits required to describe a hash function in $\mathcal{H}$. Now, fix h, the hash function we have oracle-access to. We consider the following function, which we use in line 2 of algorithm 2:

$$\begin{aligned}g &: \{0,1\}^r \times \{0,1\}^{tn} \to \{0,1\}^{tn} \times \{0,1\}^{tm} \\ g &: (h, x_1, \cdots, x_t) \mapsto (x_1, \cdots, x_t, h(x_1), \cdots, h(x_t))\end{aligned}$$

Since OWFs do not exist, we have an inverter $\mathcal{A}_g$ which can invert g to get an h' on at least $99/100$-fraction of possible outputs of g. First, assume that $\mathcal{A}_g$ produces a correct inverse. Now, we will bound the probability that h' differs on *more than* $1/p(n)$-fraction of inputs to h:

$$\Pr_{\mathbf{x}}\left[\forall x_i, h(x_i) = h'(x_i)\right] \leq (1 - \frac{1}{p(n)})^t.$$

Since h' has only r bits to describe itself, we can bound the probability that there even exists such an h' with a union bound:

$$\Pr_{\mathbf{x}}\left[\exists h' : \forall x_i, h(x_i) = h'(x_i)\right] \leq 2^r \left(1 - \frac{1}{p(n)}\right)^t.$$

We want this probability to be less than $\frac{1}{100}$, so we can bound the number of queries t as follows:

$$2^r \left(1 - \frac{1}{p(n)}\right)^t \leq \frac{1}{100} \iff t \geq (r + \log(100)) \cdot \frac{1}{\log\left(\frac{p(n)}{p(n)-1}\right)}$$

We notice that this ugly term $\frac{1}{\log \frac{p(n)}{p(n)-1}} \leq p(n)$ for all n. It turns out that $p(n)$ is a very good upper bound of this term: assuming $p(n)$ is increasing, for all exponents $0 \leq c < 1$, there exists n' so that for all $n > n'$, $\frac{1}{\log \frac{p(n)}{p(n)-1}} \geq p(n)^c$.

Given that $7 > \log_2(100)$, we can choose t such that

$$t \geq (r+7)p(n).$$

□

Now, we will assume that h and h' agree on all but $1/p(n)$ inputs and prove, using a simple counting argument, that $99/100$ of our inputs land in bins with at most $100/p(n)$-fraction of bad bins.

Lemma 52. *If h and h' disagree on at most $q(n) = 2^n/p(n)$, then there exists a set of bins containing $99/100$-fraction of all inputs such that each bin contains at most a $1/p'(n)$-fraction of bad inputs where $p'(n) = p(n)/100$.*

Proof. For sake of contradiction, assume that the lemma is not true: for *every* set of $99/100 \cdot 2^n$ inputs $x \in \{0,1\}^n$, at least one of the x falls into a bin B, where strictly more than $1/p'(n)$-fraction of the inputs $y \in B$ are "bad," mapping $h(y) \neq h'(y)$.

So, let us consider the set where we map as many inputs as we can to good bins, bins with $\leq 1/p'(n)$-fraction of the inputs are bad. This means that the rest of the inputs map to bins where $> 1/p'(n)$-fraction of the inputs are bad. In the best case for this, we can find good bins for $99/100 \cdot 2^n - 1$ of the inputs, but not for the last one.

Let $q' \geq 2^n/100 + 1$ be the number of inputs that are left, and therefore all map to bad bins. In fact, these q' elements fill the remaining bins exactly, so if we try counting the number of bad elements:

$$\#bad > \sum_{B \text{ is a bad bin}} \left(|B| \cdot \frac{1}{p'(n)} \right) = \frac{q'}{p'(n)}.$$

Since $q' > 2^n/100$, this implies $\#bad > 2^n/(100p'(n)) = 2^n/p(n)$. This is a contradiction since we assumed $\#bad \leq 2^n/p(n)$.

Therefore, there exists a subset $S \subset \{0,1\}^n$ where $|S| \geq 2^n/100$, and all $x \in S$ map to bins with fraction at most $100/p(n)$ of the elements in those bins are bad. □

Now, let us assume that we are in a good bin where h' agrees with h on all but $1/n^c$ elements in this bin. We will prove that with noticeable probability, we can split the bin into sub-bins, where most of them are completely clean (and in fact that we will land in a clean bin).

Lemma 53. *Let B be a bin of size between 2^i and 2^{i+1}, and at most $1/n^c$-fraction of elements $x \in B$ are bad.*

Let $k = \max(0, i-3)$ and $f : \{0,1\}^n \to \{0,1\}^k \xleftarrow{\$} \mathcal{F}$ a pairwise independent hash-function family. For an arbitrary fixed $x \in B$,

$$\Pr_{f \xleftarrow{\$} \mathcal{F}} \left[\nexists x' \in B \text{ so that } h(x') \neq h'(x') \wedge f(x') = f(x) \right] \geq \frac{16}{n^c}$$

Proof. Let $|B| = s$. By assumption, $2^i \leq s \leq 2^{i+1}$. We will be focusing on the number of bad elements in B. Let q be the number of bad elements in B. Again, by assumption, $q \leq s/n^c$.

We have two cases: $i < 4$ and $i \geq 4$. For the case that $i < 4$, we can choose c to be large enough so that $2^i/n^c < 1$. When this is the case, there are no bad elements in B, and therefore

for $k = 0$, the sub-bin defined by $h'(x)||f(x) = h'(x)$ is entirely clean. In fact, for $n > 2$, we choose $c \geq 4$ and we are guaranteed this fact. When $i \geq 4$, we need to do a bit more analysis.

Fix a sub-bin B' for an arbitrary element in the image of $f \in \mathcal{F}$. Let $\hat{X} = \sum_{j=1}^{q} \hat{X}_j$ be the sum of indicator values $\hat{X}_j$ where $\hat{X}_j$ is 1 if the j^{th} *bad* element in our starting bin B is mapped to bin B'. So, $X = |B'|$ is a non-negative random element. We can bound the mean of $\hat{X}$ as $\hat{\mu} = \mathbb{E}[\hat{X}] \leq \frac{s}{n^c} \cdot \frac{1}{2^k}$. Since s is between 2^i and 2^{i+1}, $\hat{\mu} \leq \frac{16}{n^c}$. With a Markov bound, $\Pr[\hat{X} \geq 1] \leq \frac{\hat{\mu}}{1} \leq \frac{16}{n^c}$. □

Finally, we need to make sure that our sub-bin is large enough. So, we will assume that we are in a bin of size s between 2^i and 2^{i+1} and let $k = i - 3$, as in the previous theorem. We will show using the assymetric Chebyshev theorem, theorem 54, that with probability $8/10$, we have a sub-bin of at least 2 elements.

Theorem 54 (Asymetric Chebyshev)**.** *For a random variable X of unknown or assymetric distribution with mean μ and variance σ^2 and for two integers, $k_1 + k_2 = 2\mu$,*

$$\Pr\left[k_1 < X < k_2\right] \geq \frac{4\left((\mu - k_1)(\mu - k_2) - \sigma^2\right)}{(k_2 - k_1)^2}.$$

Lemma 55. *Let B be a bin of size between 2^i and 2^{i+1} for $i \geq 1$, and at most $1/n^c$-fraction of elements $x \in B$ are bad.*

Let $k = \max(0, i - 3)$ and $f : \{0,1\}^n \to \{0,1\}^k \xleftarrow{\$} \mathcal{F}$ a pairwise independent hash-function family. With probability at least $8/10$, there are at least 2 elements in a sub-bin defined by $h'(x)||f(x)$.

Proof. In this proof we consider the bin B as a whole. Let $X = \sum_{j=1}^{s} X_j$ where X_j indicates if the j^{th} element in B maps to our sub-bin B' defined by $h'(x)||f(x)$. Note that since f is pairwise independent, X is the sum of pairwise independent variables.

Again, the first case, where $i < 4$, is simple to analyze. Here $k = 0$, and we are looking at B as a whole, which by assumption $i \geq 1$ has at least 2 elements. The second case, where $i \geq 4$, requires more analysis.

Let μ be the mean of X. By linearity of expectation $\mu = \sum \mathbb{E}[X_j] = \sum \frac{1}{2^k} = \frac{s}{2^k}$. Since $k = i - 3$, $8 \leq \mu \leq 16$.

Let σ^2 be the variance of X. We compute σ^2 as follows, keeping in mind that the X_j are pairwise independent so covariance between 2 variables is 0:

$$\begin{aligned}
\sigma^2 = \text{Var}\left(\sum_{j=1}^{s} X_j\right) &= \sum_{j=1}^{s} \text{Var}(X_j) + \sum_{j \neq \ell} \text{Cov}(X_j, X_\ell) \\
&= \sum_{j=1}^{s} \text{Var}(X_j) = \sum_{j=1}^{s} \frac{1}{2^k}\left(1 - \frac{1}{2^k}\right) \\
&= \frac{s}{2^k}(1 - \frac{1}{2^k}) < \mu.
\end{aligned}$$

Since we have variance and mean, we can use theorem 54, the Chebyshev inequality. If we let $k_1 = 1$ and $k_2 = 2\mu - 1$, we satisfy $k_1 + k_2 = 2\mu$, and can compute

$$\begin{aligned}
\Pr[X > 1] \geq \Pr[1 < X < 2\mu - 1] &\geq \frac{4(\mu - 1)(2\mu - 1 - \mu) - \sigma^2}{(2\mu - 1 - 1)} \\
&= \frac{4}{4(\mu - 1)^2} \cdot \left((\mu - 1)^2 - \sigma^2\right) \geq \frac{\left((\mu - 1)^2 - \mu\right)}{(\mu - 1)^2}.
\end{aligned}$$

Recall that $8 \leq \mu \leq 16$ and this function increases with μ. So, plugging in $\mu = 8$ gives us a minimum:

$$\Pr[X > 1] \geq \frac{(8-1)^2 - 8}{(8-1)^2} = \frac{41}{49} > \frac{8}{10}.$$

□