

Property-Preserving Hashing for ℓ_1 -Distance Predicates: Applications to Countering Adversarial Input Attacks

Hassan Jameel Asghar
hassan.asghar@mq.edu.au
Macquarie University
Sydney, Australia

Chenhan Zhang
chzhang@ieee.org
Macquarie University
Sydney, Australia

Dali Kaafar
dali.kaafar@mq.edu.au
Macquarie University
Sydney, Australia

ABSTRACT

Perceptual hashing is used to detect whether an input image is similar to a reference image with a variety of security applications. Recently, they have been shown to succumb to adversarial input attacks which make small imperceptible changes to the input image yet the hashing algorithm does not detect its similarity to the original image. Property-preserving hashing (PPH) is a recent construct in cryptography, which preserves some property (predicate) of its inputs in the hash domain. Researchers have so far shown constructions of PPH for Hamming distance predicates, which, for instance, outputs 1 if two inputs are within Hamming distance t . A key feature of PPH is its strong correctness guarantee, i.e., the probability that the predicate will not be correctly evaluated in the hash domain is negligible. Motivated by the use case of detecting similar images under adversarial setting, we propose the first PPH construction for an ℓ_1 -distance predicate. Roughly, this predicate checks if the two one-sided ℓ_1 -distances between two images are within a threshold t . Since many adversarial attacks use ℓ_2 -distance (related to ℓ_1 -distance) as the objective function to perturb the input image, by appropriately choosing the threshold t , we can force the attacker to add considerable noise to evade detection, and hence significantly deteriorate the image quality. Our proposed scheme is highly efficient, and runs in time $O(t^2)$. For grayscale images of size 28×28 , we can evaluate the predicate in 0.0784 seconds when pixel values are perturbed by up to 1%. For larger RGB images of size 224×224 , by dividing the image into 1,000 blocks, we achieve times of 0.0128 seconds per block for 1% change, and up to 0.2641 seconds per block for 14% change. Furthermore, the time to process the entire image can be considerably improved since the scheme is highly parallel.

KEYWORDS

Property-preserving hashing, adversarial attacks, error-correcting codes

1 INTRODUCTION

Consider a scenario in which an image needs to be checked against a database of images for any similarities. Privacy demands that both the image and the database not be revealed during this process. This scenario stems from several real-world use cases. For instance, this is required in face recognition for border control where the identity of a passenger is verified against a gallery of photos from other passengers on the same flight.¹ Likewise, a cloud service provider may wish to ensure that an image uploaded to its cloud is not one

of the flagged images in its database [1]. A possible solution to this is via *perceptual hashing* [1–3], variants of which have been used by Microsoft,² Meta,³ and Apple.⁴ Perceptual hashing is a type of *locality sensitive hashing* (LSH) [4] which produces similar hashes to perceptually similar images, unlike cryptographic hash functions. The result is that we can not only check images for their similarities, but also be able to do so more efficiently using their succinct hash digests.

Recently several attacks have been demonstrated on perceptual hashing. One type of attack, called an *evasion attack*, slightly perturbs an image to construct a perceptually similar image but with dissimilar hash digests under the perceptual hashing scheme [1, 3]. Perceptual hashing involves two main steps: extracting features from the image and creating a hash of the feature vector [3]. The guarantee that two perceptually similar images produce a similar hash digest is only in probability, which in practice is not negligible. Thus, there is significant space available to the attacker for image alterations to launch an evasion attack. For instance, the basic attack in [3] formulates the competing requirements of perceptual similarity and dissimilarity of the hash digests as an optimization problem, the solution to which is the required adversarial image.

As mentioned above, the reason for the success of evasion attacks is that perceptual hashing or locality sensitive hashing do not tend to have negligible *correctness error*. Informally, let \mathbf{x} and \mathbf{y} be two perceptually similar images, and let h be a perceptual hash function, then ideally the requirement is that $\Pr[h(\mathbf{x}) = h(\mathbf{y})] \geq 1 - \epsilon$ [3, 5, 6]. However, there are strong lower bounds suggesting that ϵ cannot be made negligibly small [6, 7]. A related notion of hashing is *property-preserving hashing* (PPH) [8]. Such hash functions have the property that the hash digests of two inputs preserve some predicate of the two inputs. For instance, a predicate that outputs 1 if the Hamming distance of its two inputs are within a certain threshold. Moreover, the definition postulates that the correctness error be negligible. Generally, these hash functions are sampled from a larger PPH family of functions. Recent works have constructed robust versions of PPH in which the adversary can choose inputs depending on the description of the hash function chosen from the family [6, 8–10]. These works have focused on the Hamming distance predicate.

Our interest in PPH for the aforementioned application of checking similarity of images stems from the fact that adversarial attacks to induce image misclassification introduce small perturbation in the images by using a distance metric such as the ℓ_2 -distance in the objective function [11, 12]. Furthermore, researchers in this space have also used the ℓ_2 -norm as a measure of perceptual similarity

²See PhotoDNA at <https://www.microsoft.com/en-us/photodna>.

³See <https://about.fb.com/news/2019/08/open-source-photo-video-matching/>.

⁴See “CSAM Detection – Technical Summary”, at https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf.

¹See “2024 Update on DHS’s Use of Face Recognition & Face Capture Technologies” at <https://www.dhs.gov/archive/news/2025/01/16/2024-update-dhss-use-face-recognition-face-capture-technologies>.

between images [3, 12]. Thus, if we have a PPH function family that preserves Euclidean distance predicates we can be assured that any evasion attacks possible in the hash space of the images are precisely those that are possible in the original space over the Euclidean distance. As a result, setting an appropriate threshold for similarity over the Euclidean distance ensures that the attacker can only succeed by substantially distorting the adversarial image, i.e., reduced perceptual similarity to the original image. Our contributions are as follows

- We propose the first PPH family for the asymmetric ℓ_1 -distance predicate for n -element vectors with each element from the set $\{0, 1, \dots, q-1\}$. Roughly, given two vectors \mathbf{x} and \mathbf{y} , the asymmetric ℓ_1 -distance predicate outputs 1 if both $\|\mathbf{x} \dot{-} \mathbf{y}\|_1$ and $\|\mathbf{y} \dot{-} \mathbf{x}\|_1$ are less than $t/2$ for some threshold t , where $\mathbf{x} \dot{-} \mathbf{y}$ is the vector whose i th element is defined as $\max\{x_i - y_i, 0\}$. Prior to this work, only known PPH constructions are for Hamming distance predicates [6, 8–10]. Our scheme is based on ℓ_1 -error correcting codes from Tallini and Rose [13]. The ℓ_1 -distance metric is related to ℓ_2 -distance (see Section 2), and can be used as a proxy for adversarial attacks or perceptual similarity.
- We show that our PPH construction is robust to one-sided errors: the predicate outputs 1 yet the PPH evaluates to 0. This property is important to prevent the aforementioned evasion attacks. We also give evidence that the other one-sided error can be made practically small in the non-robust setting. This error is important to rule out collisions, i.e., two dissimilar images that produce the same hash. We discuss collision attacks and inverting the hash function, i.e., finding the input vector given the hash digest, and show some evidence that they may be computationally expensive.
- We prove lower bounds on the possible compression, i.e., the length of the digest, to preserve any ℓ_1 -distance predicate. Our scheme produces hash digests of length $t \log_2 n$, which is considerably less than $n \log_2 q$ (size of images) if t is small. We show that for practical parameters this is close to the lower bounds for small t . For larger t the digest size is large, but on par with Hamming distance PPHs which achieve compression of around $t \log_2 n$ [6].
- We implement our scheme using the Python library `galois` [14] and show the computational time of our scheme against increasing values of t . We further implement two adversarial evasion attacks and two generic image transformations over the public Imagenette dataset [15], and show how the scheme can prevent such attacks with a tradeoff between adversarial attack prevention and computational time.

2 PRELIMINARIES

Images. Let q, n be positive integers. The image space is the set \mathbb{Z}_q^n , where $\mathbb{Z}_q = \{0, 1, \dots, q-1\}$, and n is the product of the number of elements in a pixel times the total number of pixels in the image. While we assume that $q \geq 2$, we are not interested in $q = 2$, as binary images can be handled via Hamming distance predicates.

Example 1. Consider a set S of 28×28 images with each pixel having a binary value, white or black. Each pixel is therefore from

\mathbb{Z}_2 . The 28×28 pixel matrix can be “flattened” in a vector of 784 pixels. Thus each image in S is in \mathbb{Z}_2^{784} . Consider instead that S contains only grayscale images. Each pixel now has a grayscale value in $\{0, 1, \dots, 255\}$. We can represent S as vectors from \mathbb{Z}_{256}^{784} (with $q = 256$). Consider now that the images in S are RGB colored images. Each pixel is now a vector containing R, G and B values, each in $\{0, 1, \dots, 255\}$, that is from the vector space \mathbb{Z}_{256}^3 . Thus the images in S are vectors from the product space $(\mathbb{Z}_{256}^3)^{784}$. This can be flattened further to obtain the product space \mathbb{Z}_{256}^{2352} . \square

Metrics. The Hamming, ℓ_1 and ℓ_2 distances between vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$ are denoted by $\|\mathbf{x} - \mathbf{y}\|_0$, $\|\mathbf{x} - \mathbf{y}\|_1$ and $\|\mathbf{x} - \mathbf{y}\|_2$, respectively. A refresher on these is given in Appendix A.

Dot Product. Let $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, then their dot product is defined as $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i$. The Cauchy-Schwartz inequality states that

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2.$$

See for example [16, §2.2].

Relation Between Norms. The following result relates the ℓ_1 -norm to the ℓ_2 -norm.

PROPOSITION 1. *Let \mathbf{x} and \mathbf{y} be images. Then,*

$$\|\mathbf{x} - \mathbf{y}\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_1 \leq \sqrt{n} \|\mathbf{x} - \mathbf{y}\|_2. \quad (1)$$

Furthermore, these bounds are tight.

PROOF. See Appendix B. \square

Thus, it suffices to focus on the ℓ_1 -norm. For instance, if we want to preserve $\|\mathbf{x} - \mathbf{y}\|_2 \leq t'$ for some threshold t' , we can set $t = \sqrt{nt'}$ as the threshold for the ℓ_1 -norm.

Min-Entropy. To prove lower bounds on the amount of compression achievable under Hamming distance, Holmgren et al [6] use the notion of average min-entropy as defined in [17]. The min-entropy $H_\infty(X)$ of the random variable X is defined as $-\log_2(\max_x \Pr(X = x))$. For a pair of random variables X and Y , the average min-entropy of X given Y is defined as $H_\infty(X | Y) = -\log_2(\sum_{y \in Y} \Pr(Y = y) \cdot (\max_x \Pr(X = x | Y = y)))$. The following result is from [6, 17].

PROPOSITION 2 (DODIS ET AL [17]). *Let X, Y, Z be random variables with Z having support over a binary string of length m . Then*

$$H_\infty(X | Y, Z) \geq H_\infty(X | Y) - m$$

2.1 Scenario and Threat Model

We assume a database \mathcal{D} of N images $\mathbf{x}_1, \dots, \mathbf{x}_N$ hosted by a server. A predicate $P : \mathbb{Z}_q^n \times \mathbb{Z}_q^n \rightarrow \{0, 1\}$ is a function applied to a pair of images. For instance, one such predicate for two images \mathbf{x}, \mathbf{y} is:

$$P(\mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if } \|\mathbf{x} - \mathbf{y}\|_1 \leq t, \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The predicate used in our scheme is different from the one above, but still based on the ℓ_1 -distance, as we shall see later. Given any input image \mathbf{y} from a client, the server checks if it satisfies the predicate P against any image $\mathbf{x}_i \in \mathcal{D}$.

We consider a hash function family $\mathcal{H} = \{h : \mathbb{Z}_q^n \rightarrow \{0, 1\}^m\}$. Given $\mathbf{x} \in \mathbb{Z}_q^n$, we call $h(\mathbf{x})$ for some $h \in \mathcal{H}$, the *hash digest* of the

image \mathbf{x} . Associated with the family, there is a deterministic polynomial time algorithm $\text{eval}_h : \{0, 1\}^m \times \{0, 1\}^m \rightarrow \{0, 1\}$. Informally, the hash function h should satisfy the following properties:

- (1) *Compression*: The output $h(\mathbf{x})$ should be compressed, i.e., the size of $h(\mathbf{x})$ should be less than the size of the image \mathbf{x} .
- (2) *Hiding*: It should be hard to find \mathbf{x} from $h(\mathbf{x})$.
- (3) *Property-preservation*: The function eval_h should be such that $\text{eval}_h(h(\mathbf{x}), h(\mathbf{y})) = P(\mathbf{x}, \mathbf{y})$ with high probability.

The first requirement is for utility, as one would want the hash function to at least compress the input space. The other two requirements are for privacy. We want the input image \mathbf{x} to be hidden from the server, with only the output of the predicate P revealed. The third requirement is related to an adversarial user who wishes to submit an image \mathbf{y} such that there is a mismatch between $P(\mathbf{x}, \mathbf{y})$ for some $\mathbf{x} \in \mathcal{D}$, and the eval_h function.

Adversarial Goals. We consider the following attacks.

- (1) *Inversion attack*: Given the hash digest $h(\mathbf{x})$ of an image \mathbf{x} chosen uniformly at random from \mathbb{Z}_q^n , find \mathbf{x} .
- (2) *Evasion attack*: Given $h \in \mathcal{H}$, find a pair of images \mathbf{x} and \mathbf{y} , such that $P(\mathbf{x}, \mathbf{y}) = 1$ yet $\text{eval}_h(h(\mathbf{x}), h(\mathbf{y})) = 0$.
- (3) *Collision attack*: Given $h \in \mathcal{H}$, find a pair of images \mathbf{x} and \mathbf{y} such that $P(\mathbf{x}, \mathbf{y}) = 0$ yet $\text{eval}_h(h(\mathbf{x}), h(\mathbf{y})) = 1$.

The first goal is in a similar flavour to inverting passwords given their hash digests. The defence against the second attack follows directly from the definition of PPH. This is the main attack that our scheme defends against. Prevention of the third attack is also desirable. We show that while this may still be possible, the probability of collisions can be made extremely low in the non-robust setting, i.e., two arbitrary images deemed similar under h .

2.2 Property-Preserving Hashing and Compression Bounds

We recall the definitions of PPH and robust PPH (RPPH) from [6, 8]. We assume n and m to be polynomials in a security parameter λ .

Definition 1 (Property Preserving Hash (PPH)). A (b, n, m) -property preserving hash family $\mathcal{H} = \{h : \{0, 1\}^n \rightarrow \{0, 1\}^m\}$ for a predicate $P : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ is a family of efficiently computable functions with the following algorithms

- $\text{samp}(1^\lambda)$ is a probabilistic polynomial time algorithm that outputs a random $h \in \mathcal{H}$.
- $\text{eval}_h(y_1, y_2)$ is a deterministic polynomial time algorithm that for an $h \in \mathcal{H}$ and $y_1, y_2 \in \{0, 1\}^m$ outputs a single bit.
- *b-Correctness*: For a bit $b \in \{0, 1\}$, for any $h \in \mathcal{H}$ and for all $x_1, x_2 \in \{0, 1\}^n$ we have

$$\Pr_{h \leftarrow \text{samp}(1^\lambda)} [P(x_1, x_2) \neq \text{eval}_h(h(x_1), h(x_2)) \mid P(x_1, x_2) = b] = \text{negl}(\lambda)$$

□

Note that we have slightly modified the definition of PPH from [6, 8] to include two-sided correctness. This is because, even though our scheme is 1-correct, 0-correctness is only guaranteed with a small but non-negligible probability. A PPH is considered to be an RPPH if it further satisfies the following definition.

Definition 2 (Robust Property Preserving Hash (RPPH)). A (b, n, m) -PPH family is a robust (b, n, m) -property preserving hash family if for all probabilistic polynomial time algorithms \mathcal{A}

$$\Pr_{\substack{h \leftarrow \text{samp}(1^\lambda) \\ x_1, x_2 \leftarrow \mathcal{A}(h)}} [P(x_1, x_2) \neq \text{eval}_h(h(x_1), h(x_2)) \mid P(x_1, x_2) = b] = \text{negl}(\lambda)$$

□

The main difference between an RPPH and a PPH is that the inputs x_1, x_2 that bring about a mismatch between the predicate and the evaluation function are adversarially chosen in the former who is also given the description of the sampled hash function h .

Lower Bound on Compression. Given a PPH for the Hamming distance predicate, Holmgren et al [6] derive a lower bound on m , i.e., the compression or digest length. In our case the inputs are from \mathbb{Z}_q^n instead of the generic set $\{0, 1\}^n$. We therefore, review their lower bound for inputs in \mathbb{Z}_q^n . Thus, we assume the PPH family to be $\mathcal{H} = \{h : \mathbb{Z}_q^n \rightarrow \{0, 1\}^m\}$, and the Hamming distance predicate to be 1 if $\|\mathbf{x} - \mathbf{y}\|_0 \leq t$ and 0 otherwise, for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$. The strategy used in [6] to get a bound on m is as follows. Given a random $h \in \mathcal{H}$, and a random variable X uniformly distributed over \mathbb{Z}_q^n , we first get the lower bound from Proposition 2:

$$H_\infty(X \mid h, h(X)) \geq H_\infty(X \mid h) - m \geq H_\infty(X) - m = n \log_2 q - m. \quad (3)$$

Here, the third inequality follows since X and h are independently distributed, and the last equality because min-entropy is maximum when X is uniformly distributed. Next, the task is to obtain an upper bound on $H_\infty(X \mid h, h(X))$, which would then give an upper bound on m after rearranging the inequalities. The strategy used in [6] to obtain the upper bound is to find a vector \mathbf{y} which is at a Hamming distance exactly t from \mathbf{x} , where \mathbf{x} is the vector hashed under the PPH, i.e., $h(\mathbf{x})$. They then exactly reconstruct \mathbf{x} by using the eval_h function of the PPH as an ‘‘oracle’’. More specifically, they first guess a vector \mathbf{y} which is exactly Hamming distance t from \mathbf{x} . The number of such vectors is exactly $\binom{n}{t}$. They then flip the bits of \mathbf{y} one at a time, and check whether the eval_h function outputs 0 or 1 on $h(\mathbf{x})$ and the hash of the version of \mathbf{y} with one bit flipped. This uses at most n applications of the eval_h function of the PPH. As long as the eval_h function has error less than $1/2n$, their algorithm can reconstruct \mathbf{x} with probability at least $\frac{\binom{n}{t}}{2^n} \cdot \frac{1}{2}$. Note that in their case $q = 2$. It is easy to change this for a general q to $\frac{\binom{n}{t}}{q^n} \cdot \frac{1}{2}$ by assuming that the eval_h function has error at most $1/2qn$. Now, let \mathcal{R} be their algorithm to reconstruct \mathbf{x} , then following [6]:

$$\begin{aligned} \Pr_{h, \mathbf{x}} (\mathcal{R}(h, h(\mathbf{x})) = \mathbf{x}) &\geq \Pr_{\mathbf{y}} (\|\mathbf{x} - \mathbf{y}\|_0 = t) \\ &\times \Pr_{\mathbf{x}, \mathbf{y}, h} (\mathcal{R}(h, h(\mathbf{x})) = \mathbf{x} \mid \|\mathbf{x} - \mathbf{y}\|_0 = t) \\ &> \frac{\binom{n}{t}}{q^n} \cdot \frac{1}{2} \end{aligned}$$

Now, abusing notation by letting \mathcal{H} also denote the random variable that takes on a random $h \in \mathcal{H}$, we get

$$\begin{aligned} H_\infty(X \mid h, h(X)) \\ = -\log_2 \left(\sum_h \Pr(\mathcal{H} = h) \cdot \left(\max_{\mathbf{x}} \Pr(X = \mathbf{x} \mid h, h(\mathbf{x})) \right) \right) \end{aligned}$$

$$\begin{aligned}
&\leq -\log_2 \left(\sum_h \Pr(\mathcal{H} = h) \cdot \Pr(\mathcal{R}(h, h(\mathbf{x})) = \mathbf{x} \mid h, h(\mathbf{x})) \right) \\
&= -\log_2 (\Pr(\mathcal{R}(h, h(\mathbf{x})) = \mathbf{x})) \\
&\leq 1 + n \log_2 q - \log_2 \binom{n}{t}
\end{aligned} \tag{4}$$

Combining the above with the inequality in Eq. (3), and noting that m is an integer, we obtain $m \geq \log_2 \binom{n}{t}$ [6]. This bound is for non-robust PPH families, and hence also applies to RPPH families.

3 COMPRESSION BOUNDS FOR ℓ_1 -DISTANCE PPH FAMILY

Assume we are given a PPH family $\mathcal{H} = \{h : \mathbb{Z}_q^n \rightarrow \{0, 1\}^m\}$ for the ℓ_1 -distance predicate of Eq. (2). We are interested in finding a lower bound on m similar to the one for the Hamming distance predicate (Section 2.2). Unfortunately, the strategy used in [6] does not work for the ℓ_1 distance. The main reason being that the predicate $\|\mathbf{x} - \mathbf{y}\|_1 \leq t$ does not reveal enough information about \mathbf{x} given some vector \mathbf{y} which is exactly an ℓ_1 distance of t from \mathbf{x} to be able to recover \mathbf{x} , apart from the fact that it can be used to sample a vector within distance t from \mathbf{x} . In the following we prove two bounds: one uses a large $t > 0.25qn$, and the other a much smaller $t \approx 0.1n$. We first prove a few results related to the ℓ_1 -norm of vectors in \mathbb{Z}_q^n .

3.1 The ℓ_1 -Ball of Radius t

For any $\mathbf{x} \in \mathbb{Z}_q^n$, let $B_1(\mathbf{x}, q, t)$ denote the set of vectors $\mathbf{y} \in \mathbb{Z}_q^n$, such that $\|\mathbf{x} - \mathbf{y}\|_1 \leq t$. Since $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, we can assume that t is an integer. Let $\mathbf{0}$ denote the all 0 vector.

PROPOSITION 3. For all $\mathbf{x} \in \mathbb{Z}_q^n$, we have $|B_1(\mathbf{x}, q, t)| \geq |B_1(\mathbf{0}, q, t)|$.

PROOF. See Appendix B. \square

PROPOSITION 4. Let $t \geq 0$ be an integer. Then

$$\binom{n+t}{t} - \binom{n-1+t-q}{t-q} \leq |B_1(\mathbf{0}, q, t)| \leq \binom{n+t}{t}.$$

PROOF. See Appendix B \square

PROPOSITION 5. Let $t \geq 0$ be an integer. Let $q \geq 2$ be even. Let \mathbf{y} be the n -element vector $(q/2, \dots, q/2)$. Then

$$|B_1(\mathbf{y}, q, t)| \leq \binom{n+t+1}{t+1}$$

PROOF. See Appendix B. \square

3.2 Bound on m for Large t

Our goal is to find a $\mathbf{y} \in \mathbb{Z}_q^n$ such that $\|\mathbf{x} - \mathbf{y}\|_1 \leq t$, without knowing \mathbf{x} . We assume $t = \gamma n$ for some $\gamma \geq 1$. Note that the maximum possible distance can be up to $(q-1)n \approx qn$. Thus, γ gives the relative distance to the maximum possible distance. The strategy for finding such a \mathbf{y} is depicted in Figure 1. Assume $n = 1$. Further assume that q is even. In our experiments $q = 256$, and so this is not a limitation. Suppose $t = \gamma n = \gamma \cdot q/2$. Then if we choose $y = q/2$, it includes all points in \mathbb{Z}_q , and hence x as well. Therefore, with probability 1, we find a y , namely $y = q/2$, which is a distance

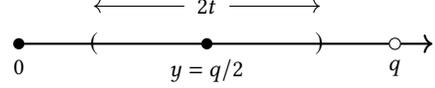


Figure 1: Choosing y as the mid-point.

$\leq t = q/2$ from x . Of course, if t is smaller than $q/2$, the probability decreases. We therefore find the probability over uniform random choices of \mathbf{x} that the vector $\mathbf{y} = (q/2, \dots, q/2)$ is within distance $t = \gamma n$ of \mathbf{x} .

PROPOSITION 6. Let \mathbf{x} be a vector sampled uniformly at random from \mathbb{Z}_q^n . Let $\mathbf{y} \in \mathbb{Z}_q^n$ be the vector each coordinate of which is $q/2$. Let D denote the random variable denoting the ℓ_1 distance between \mathbf{x} and \mathbf{y} . Then $\mathbb{E}(D) = qn/4$.

PROOF. See Appendix B. \square

PROPOSITION 7. Let D be the random variable as in Proposition 6. Let $t = \gamma n$ for some real $\gamma \geq 1$. Then if

$$\gamma \geq \left(\frac{1}{4} + \frac{1}{2\sqrt{2n}} \right) q$$

then $\Pr(D \leq t) \geq 1 - 1/e$.

PROOF. See Appendix B. \square

Thus, for large enough t , i.e., $t \approx qn/4$, the vector $\mathbf{y} = (q/2, \dots, q/2)$ is within ℓ_1 -distance t from a uniformly random vector in \mathbb{Z}_q^n with high probability. Assuming this to be the case, we then need to guess the vector \mathbf{x} whose hash digest has been provided to us. Our strategy is to simply sample a vector uniformly at random from $B_1(\mathbf{y}, q, t)$. By assumption, $\mathbf{x} \in B_1(\mathbf{y}, q, t)$, and therefore the probability of obtaining \mathbf{x} will be $1/|B_1(\mathbf{y}, q, t)|$. Sampling a vector uniformly at random from $B_1(\mathbf{y}, q, t)$ is not straightforward. However, there are techniques to sample a vector from an approximate uniform distribution. E.g., we can use the discrete hit-and-run sampler [18]. This produces a distribution arbitrarily close to uniform [19, §11.2]. This follows from the fact that $B_1(\mathbf{y}, q, t)$ is a diamond centered around \mathbf{y} . The resulting diamond can be enclosed within a cube which itself is within a cube and therefore the analysis in [18, §4.2] means that the algorithm will produce a distribution arbitrarily close to uniform distribution in polynomial time.

Now consider an algorithm \mathcal{R} which when given a random $h \in \mathcal{H}$ and an $h(\mathbf{x})$ where \mathbf{x} is sampled uniformly at random from \mathbb{Z}_q^n , does as follows. It computes $\text{eval}_h(h(\mathbf{x}), h(\mathbf{y}))$ with $\mathbf{y} = (q/2, \dots, q/2)$. If it outputs 1, it samples a vector uniformly at random from $B_1(\mathbf{y}, q, t)$. It outputs this vector as its guess for \mathbf{x} and stops. Assume that eval_h has correctness error $\delta < \frac{1}{2} \cdot \frac{e-2}{e-1}$. We get

$$\begin{aligned}
\Pr_{h, \mathbf{x}}(\mathcal{R}(h, h(\mathbf{x})) = \mathbf{x}) &\geq \Pr_{h, \mathbf{x}}(\mathcal{R}(h, h(\mathbf{x})) = \mathbf{x} \mid \|\mathbf{x} - \mathbf{y}\|_1 \leq t) \\
&\quad \times \Pr_{\mathbf{x}}(\|\mathbf{x} - \mathbf{y}\|_1 \leq t) \\
&> \left(1 - \frac{1}{2} \cdot \frac{e-2}{e-1} \right) \left(1 - \frac{1}{e} \right) \frac{1}{|B_1(\mathbf{y}, q, t)|} \\
&\geq \frac{2(e-1) - (e-2)(e-1)}{2(e-1)} \frac{1}{e} \frac{1}{\binom{n+t+1}{t+1}}
\end{aligned}$$

Nature	Regime	Color	t	n	Baseline	m	Compression
Bound	Large t	RGB	154918	$28 \times 28 \times 3$	18816	1194	6.346
			796466	$64 \times 64 \times 3$	98304	6495	6.607
			3165795	$128 \times 128 \times 3$	393216	26403	6.715
			9668908	$224 \times 224 \times 3$	1204224	81428	6.762
Bound	Large t	Gray-scale	52711	28×28	6272	378	6.027
			267937	64×64	32768	2115	6.454
			1060162	128×128	131072	8697	6.635
			3231539	224×224	401408	26957	6.716
Bound	Small t	RGB	235	$28 \times 28 \times 3$	18816	882	4.688
			1228	$64 \times 64 \times 3$	98304	5890	5.992
			4915	$128 \times 128 \times 3$	393216	23741	6.038
			15052	$224 \times 224 \times 3$	1204224	72754	6.042
Bound	Small t	Gray-scale	78	28×28	6272	883	14.078
			409	64×64	32768	1828	5.579
			1638	128×128	131072	7881	6.013
			5017	224×224	401408	24235	6.037
Our Scheme	Small t	RGB	235	$28 \times 28 \times 3$	18816	2631	13.983
			1228	$64 \times 64 \times 3$	98304	16682	16.970
			4915	$128 \times 128 \times 3$	393216	76600	19.480
			15052	$224 \times 224 \times 3$	1204224	258889	21.498
Our Scheme	Small t	Gray-scale	78	28×28	6272	749	11.942
			409	64×64	32768	4908	14.978
			1638	128×128	131072	22932	17.496
			5017	224×224	401408	78338	19.516

Table 1: Lower bounds on compression achievable through a PPH with small $t \approx 0.1n$ and large $t \approx 0.25qn$, and the actual compression through our scheme. The column labeled “Compression” is the percentage compression with respect to the baseline $n \log_2 q$. Here $q = 256$.

$$= \frac{1}{2} \frac{1}{\binom{n+t+1}{t+1}},$$

where we have used Proposition 5. Now using Eq (4), we get

$$H_\infty(X | h, h(X)) \leq -\log_2 \left(\Pr_{h,x}(\mathcal{R}(h, h(x)) = x) \right) < 1 + \log_2 \binom{n+t+1}{t+1}.$$

Combining the above with Eq. (3), we get

$$n \log_2 q - \log_2 \binom{n+t+1}{t+1} - 1 < m \Rightarrow m \geq n \log_2 q - \log_2 \binom{n+t+1}{t+1}, \quad (5)$$

where the last inequality follows from the fact that m is an integer. This implies that substantial compression is possible if t is large, i.e., $t > 0.25qn$. The compression rates for various image sizes n and t as computed through Proposition 7 are shown in Table 1. Even though high compression is possible, this value of t is too large for our application where it may produce a high false positive rate.

3.3 Bound on m for Small t

When t is much smaller, say $t \approx 0.1n$, there does not appear to be a better algorithm to find x than a random guess. Namely let \mathcal{R} be an algorithm which when given a random $h \in \mathcal{H}$ and $h(x)$ for some unknown x , samples a y uniformly at random from \mathbb{Z}_q^n as its guess for x . Assume that eval_h has correctness error $\delta < \frac{1}{2}$. Then,

$$\Pr_{h,x}(\mathcal{R}(h, h(x)) = x) > \frac{1}{2} \frac{|B_1(x, q, t)|}{q^n}$$

Now again using Eq (4), we get

$$H_\infty(X | h, h(X)) \leq -\log_2 \left(\Pr_{h,x}(\mathcal{R}(h, h(x)) = x) \right) < 1 - \log_2 |B_1(x, q, t)| + n \log_2 q.$$

Combining the above with Eq. (3), we get

$$m > n \log_2 q - n \log_2 q + \log_2 |B_1(x, q, t)| - 1 \Rightarrow m \geq \log_2 |B_1(x, q, t)|.$$

Now from Propositions 4 and 3, we have

$$|B_1(x, q, t)| \geq \binom{n+t}{t} - \binom{n-1+t-q}{t-q}$$

We next use the following result.

PROPOSITION 8. *If $q \geq 4$, $q-1 < t < 2.5n-2.5$, then*

$$\binom{n+t}{t} - \binom{n-1+t-q}{t-q} \geq \left(\frac{n-1+t}{t} \right)^{q-1} \binom{n-1+t-q}{t-q}$$

PROOF. See Appendix B. \square

Thus

$$m \geq \log_2 |B_1(x, q, t)| \geq \log_2 \left(\left(\frac{n-1+t}{t} \right)^{q-1} \binom{n-1+t-q}{t-q} \right) = (q-1) \log_2 \left(1 + \frac{n-1}{t} \right) + \log_2 \binom{n-1+t-q}{t-q} \quad (6)$$

Table 1 shows the lower bound of m through Eq. 6. Even with a smaller t considerable compression is possible in theory.

3.4 Feasibility of List Decoding

Some RPPH schemes for Hamming distance are based on error-correcting codes, in particular, syndrome decoding [6]. We are interested in knowing whether syndrome decoding is likely to be feasible for knowing whether syndrome decoding is likely to be feasible for ℓ_1 -distance. Consider the original image (assume binary images for now) x , and a candidate image y , the construction from [6] takes the syndrome of x and that of y , and then finds a list of errors of Hamming weight at most t . Due to the linearity of syndromes, if there is an error vector in the list which matches the difference of the syndromes, then y is within Hamming distance t of x . For syndrome list decoding to be efficient, the list of errors should be of polynomial size in n which itself is a polynomial in the security parameter λ . From Fact 2.9 in [6] syndrome list decoding is efficient if and only if list decoding is efficient.

Moving to our case, treating the target image x as a codeword, in light of the above, for a similar procedure to be efficient we need to find how many vectors are within the ℓ_1 -ball of x . If this size is big, then list decoding will not be an efficient solution. Thus, we estimate this size.

PROPOSITION 9. *Let $B_1(x, q, t)$ be the ℓ_1 -ball around $x \in \mathbb{Z}_q^n$, with $t > q-1$. Then*

$$|B_1(x, q, t)| > \left(1 + \frac{n-1}{t-q} \right)^{t-q} \quad (7)$$

PROOF. See Appendix B. \square

The lower bound from Eq. (7) is plotted in Figure 2 for grayscale images of various sizes with t ranging from 257 (i.e., from $q + 1$) to 266. Even for these extremely small values of t , we see that $|B_1(\mathbf{x}, q, t)|$ contains a large number of vectors. E.g., a grayscale image of size 28×28 contains at least 2^{60} possible vectors within its ℓ_1 -ball of radius $t = 266$. Thus, syndrome list decoding is not feasible in our case. We instead need an approach which directly checks if a given codeword is within distance t of the target codeword.

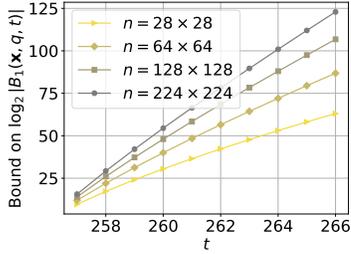


Figure 2: The lower bound in logarithmic scale of the number of images that lie within ℓ_1 -distance t of a given image. The list quickly becomes huge even for such small values of t .

4 CONNECTION TO ℓ_1 -DISTANCE ERROR CORRECTING CODES

Tallini and Rose [13] show a generic error correcting code for the ℓ_1 -distance, which we modify to use as a property-preserving hashing (PPH). To be precise, their scheme is based on the asymmetric ℓ_1 -distance. To understand the scheme, we introduce some notations. Let $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$. For $x, y \in \mathbb{Z}_q$, define:

$$x \dot{-} y = \max\{0, x - y\}$$

This definition is extended element-wise to $\mathbf{x} \dot{-} \mathbf{y}$. Now note that

PROPOSITION 10 ([13]). For any $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$

$$\mathbf{x} + (\mathbf{y} \dot{-} \mathbf{x}) = \mathbf{y} + (\mathbf{x} \dot{-} \mathbf{y}).$$

Furthermore,

$$\|\mathbf{x} - \mathbf{y}\|_1 = \|\mathbf{y} \dot{-} \mathbf{x}\|_1 + \|\mathbf{x} \dot{-} \mathbf{y}\|_1$$

PROOF. See Appendix B. □

Example 2. Let $\mathbf{x} = (2, 1, 0, 4)$ and $\mathbf{y} = (3, 0, 1, 4)$. Then $\mathbf{y} \dot{-} \mathbf{x} = (1, 0, 1, 0)$ and $\mathbf{x} \dot{-} \mathbf{y} = (0, 1, 0, 0)$. Thus,

$$\begin{aligned} \mathbf{x} + (\mathbf{y} \dot{-} \mathbf{x}) &= (2, 1, 0, 4) + (1, 0, 1, 0) \\ &= (3, 1, 1, 4) \\ &= (3, 0, 1, 4) + (0, 1, 0, 0) = \mathbf{y} + (\mathbf{x} \dot{-} \mathbf{y}) \end{aligned}$$

Moreover, $\|\mathbf{x} - \mathbf{y}\|_1 = 3$ and $\|\mathbf{y} \dot{-} \mathbf{x}\|_1 + \|\mathbf{x} \dot{-} \mathbf{y}\|_1 = 2 + 1 = 3$. □

This operation is nothing but set difference if we consider \mathbf{x} and \mathbf{y} as multisets. Next we define polynomials associated with a vector in \mathbb{Z}_q^n in a manner slightly different from [13]. Note that the goal in [13] is to construct error-correcting codes through which we could recover the original codeword \mathbf{x} given a received codeword \mathbf{y} . In our case, we only need to find whether the received codeword \mathbf{y} , i.e., the image, is within a certain ℓ_1 -distance away from \mathbf{x} , a database

image. Thus, we are not interested in recovering the original image \mathbf{x} , from which \mathbf{y} may have been adversarially created. Let \mathbb{F} be a finite field with $|\mathbb{F}| > n$. We shall assume $\mathbb{F} = \mathbb{Z}_p$, where $p > n$ is a prime. Thus $|\mathbb{Z}_p| \geq n + 1$. Let $\mathbf{a} = (a_1, \dots, a_n)$ be a vector composed of n distinct elements from $\mathbb{Z}_p - \{0\}$. The polynomial associated with $\mathbf{x} \in \mathbb{Z}_q^n$ is defined as

$$\sigma_{\mathbf{x}}(z) = \prod_{i=1}^n (1 - a_i z)^{x_i} \quad (8)$$

Note that this is a polynomial in $\mathbb{F}[Z]$, and the coefficient operations are in the field $\mathbb{F} = \mathbb{Z}_p$. The following is stated without proof in [13].

PROPOSITION 11. Let $\sigma_{\mathbf{x}} \in \mathbb{F}[Z]$ be as defined in Eq. (8). Then $\deg(\sigma_{\mathbf{x}}) = \|\mathbf{x}\|_1$.

PROOF. See Appendix B. □

The roots of $\sigma_{\mathbf{x}}$ are $a_i^{-1} \in \mathbb{F}$ with multiplicity x_i as can be easily seen. For any two polynomials in $\mathbb{F}[Z]$ define their greatest common divisor (gcd) as the zero or monic polynomial $d \in \mathbb{F}[Z]$ which divides both, and every other common divisor of the two polynomials divides d [20, §16.3]. The following key equation is proved in [13], with the proof reproduced here for completeness.

THEOREM 1. Let $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$. Then,

$$\sigma_{\mathbf{x}} \sigma_{\mathbf{y} \dot{-} \mathbf{x}} = \sigma_{\mathbf{y}} \sigma_{\mathbf{x} \dot{-} \mathbf{y}}$$

Furthermore,

$$\gcd(\sigma_{\mathbf{y} \dot{-} \mathbf{x}}, \sigma_{\mathbf{x} \dot{-} \mathbf{y}}) = 1.$$

PROOF. See Appendix B. □

Consider the polynomial $z^{t+1} \in \mathbb{F}[Z]$. We have:

PROPOSITION 12. Let $\mathbf{x} \in \mathbb{Z}_q^n$. Then $\gcd(\sigma_{\mathbf{x}}, z^{t+1}) = 1$.

PROOF. See Appendix B. □

We use the following results related to the extended Euclidean algorithm (EEA) whose proofs can be found in [21, §12.8].

THEOREM 2 ([21]). Let $r_0(z)$ and $r_{-1}(z)$ be polynomials with $\deg(r_0) \leq \deg(r_{-1})$ and $\gcd g(z)$. Then there exist polynomials u and v such that

$$u(z)r_{-1}(z) + v(z)r_0(z) = g(z), \quad (9)$$

with $\deg(u)$ and $\deg(v)$ less than $\deg(r_{-1})$. Furthermore, in the i th round, if r_i and r_{i-1} are the polynomials used in the division in EEA with $i \geq 0$, then

$$\begin{bmatrix} r_{i-1}(z) \\ r_i(z) \end{bmatrix} = (-1)^i \begin{bmatrix} v_{i-1}(z) & -u_{i-1}(z) \\ -v_i(z) & u_i(z) \end{bmatrix} \begin{bmatrix} r_{-1}(z) \\ r_0(z) \end{bmatrix}, \quad (10)$$

where

$$\begin{aligned} u_{-1}(z) &= 0, & u_0(z) &= 1, \\ v_{-1}(z) &= 1, & v_0(z) &= 0, \end{aligned}$$

Moreover, we have

- u_i and v_i are relatively prime for all i ,
 - $\deg(u_i) = \deg(r_{-1}) - \deg(r_{i-1})$,
 - $\deg(u_i) = \sum_{j=1}^i \deg(q_j)$,
 - $\deg(r_{i-1}) = \deg(r_{-1}) - \sum_{j=1}^i \deg(q_j)$.
-

We have not specified the polynomials u_i and v_i apart from the initial values of i , as their expressions are not necessary for our results. Now take the key equation in Theorem 1 modulo z^{t+1} :

$$\begin{aligned}\sigma_x(z)\sigma_{y-x}(z) &\equiv \sigma_y(z)\sigma_{x-y}(z) \pmod{z^{t+1}} \\ \sigma_{y-x}(z) &\equiv \sigma_x^{-1}(z)\sigma_y(z)\sigma_{x-y} \pmod{z^{t+1}} \\ \sigma_{y-x}(z) &\equiv \tilde{\sigma}_{x,y}(z)\sigma_{x-y}(z) \pmod{z^{t+1}},\end{aligned}\quad (11)$$

where

$$\tilde{\sigma}_{x,y}(z) = \sigma_x^{-1}(z)\sigma_y(z) \pmod{z^{t+1}}.$$

The inverse $\sigma_x^{-1}(z)$ exists since $\gcd(\sigma_x, z^{t+1}) = 1$ from Proposition 12, and can be obtained via the EEA (Eq. (9)).

4.1 When the Asymmetric ℓ_1 -Distances are Less Than the Thresholds

We first consider the case when $\|y - x\|_1 \leq t_+$ and $\|x - y\|_1 \leq t_-$, where t_+ and t_- are non-negative integers satisfying $t_+ + t_- = t$. From Proposition 10 this means that $\|y - x\|_1 \leq t$. Moreover, from Proposition 11 this implies that $\deg(\sigma_{y-x}) \leq t_+$ and $\deg(\sigma_{x-y}) \leq t_-$. The following theorem shows that if these conditions are met then we can find the solution to Eq. (11) given $\tilde{\sigma}_{x,y}$ and z^{t+1} , and from the solution we can exactly recover $\|y - x\|_1$ and $\|x - y\|_1$. From this, we can establish that $\|y - x\|_1 \leq t$. This result is stated in [13] with the proof deferred to the full version of that paper. However, could not find the full version of the paper. The authors did in fact state that this is based on the proof of Theorem 16 in [21, §12.8]. We therefore, provide a full proof here based on this theorem.

THEOREM 3. *Let t_+ and t_- be nonnegative integers satisfying $t_+ + t_- = t$, for a nonnegative integer t . Assume $\deg(\sigma_{y-x}) \leq t_+$ and $\deg(\sigma_{x-y}) \leq t_-$. Set $r_{-1}(z) = z^{t+1}$ and $r_0(z) = \tilde{\sigma}_{x,y}(z)$ in the EEA, and run the algorithm until reaching an $r_k(z)$ such that*

$$\deg(r_k) \leq t_+ \quad \text{and} \quad \deg(r_{k-1}) > t_+.$$

Set

$$\alpha(z) = (-1)^k r_k(z), \quad \beta(z) = u_k(z)$$

Then $\deg(\alpha) = \deg(\sigma_{y-x})$ and $\deg(\beta) = \deg(\sigma_{x-y})$.

PROOF. Set $r_{-1}(z) = z^{t+1}$ and $r_0(z) = \tilde{\sigma}_{x,y}(z)$. Run the EEA until reaching an $r_k(z)$ such that

$$\deg(r_k) \leq t_+ \quad \text{and} \quad \deg(r_{k-1}) > t_+.$$

Note that this is guaranteed since we start with $\deg(r_{-1}) = t+1$, and $\deg(r_i) < \deg(r_{i-1})$ at the i th iteration. Furthermore, $\gcd(\tilde{\sigma}_{x,y}, z^{t+1}) = 1$. So, the degrees of r_i 's are decreasing and go down to 0. Similarly we start with $\deg(u_0) = 1$, and at the i th iteration, we have $\deg(r_{i-1}) > \deg(r_i)$, which means (from Theorem 2)

$$\deg(u_i) = \deg(r_{-1}) - \deg(r_{i-1}) > \deg(r_{-1}) - \deg(r_{i-2}) = \deg(u_{i-1}),$$

and hence the degrees of the u_i 's are increasing. From the same theorem:

$$\deg(u_k) = \deg(r_{-1}) - \deg(r_{k-1}) < t + 1 - t_+ = t_- + 1 \leq t_- \quad (12)$$

Now set

$$\begin{aligned}\alpha(z) &= (-1)^k r_k(z), \\ \beta(z) &= u_k(z).\end{aligned}\quad (13)$$

Thus $\deg(\alpha) = \deg(r_k) \leq t_+$, and $\deg(\beta) = \deg(u_k) \leq t_-$. From Eq. (10), we have

$$\begin{aligned}r_k(z) &= (-1)^k (-v_k(z)r_{-1}(z) + u_k(z)r_0(z)) \\ \Rightarrow (-1)^k r_k(z) &= -v_k(z)r_{-1}(z) + u_k(z)r_0(z) \\ \Rightarrow \alpha(z) &= \beta(z)\tilde{\sigma}_{x,y}(z) - v_k(z)z^{t+1} \\ \Rightarrow \alpha(z) &\equiv \beta(z)\tilde{\sigma}_{x,y}(z) \pmod{z^{t+1}}.\end{aligned}$$

Thus (α, β) is a solution of Eq. (11). We next show that if (α', β') is any other solution of Eq. (11) satisfying $\deg(\alpha') \leq t_+$, and $\deg(\beta') \leq t_-$, and $\gcd(\alpha', \beta') = 1$ then necessarily $\deg(\alpha') = \deg(\alpha)$ and $\deg(\beta') = \deg(\beta)$. Since we know that $\deg(\sigma_{y-x}) \leq t_+$ and $\deg(\sigma_{x-y}) \leq t_-$, they satisfy Eq. (11) and $\gcd(\sigma_{y-x}, \sigma_{x-y}) = 1$ from Theorem 1, this shows that the solution through EEA, i.e., Eq. (13), will satisfy $\deg(\alpha) = \deg(\sigma_{y-x})$ and $\deg(\beta) = \deg(\sigma_{x-y})$, and we are done. So assume (α', β') is another solution. Then

$$\begin{aligned}\alpha'(z) &\equiv \beta'(z)\tilde{\sigma}_{x,y}(z) \pmod{z^{t+1}} \\ \alpha'(z)\beta(z) &\equiv \beta'(z)\tilde{\sigma}_{x,y}(z)\beta(z) \pmod{z^{t+1}} \\ \alpha'(z)\beta(z) &\equiv \alpha(z)\beta'(z) \pmod{z^{t+1}}\end{aligned}\quad (14)$$

The degree of each side of this congruence is $\leq t_+ + t_- = t$, and hence we have

$$\alpha'(z)\beta(z) = \alpha(z)\beta'(z),$$

i.e., without the modulus. Since α' divides both sides, we have

$$\beta(z) = \frac{\alpha(z)\beta'(z)}{\alpha'(z)}$$

Since α' and β' are relatively prime, α' must divide α . Define:

$$\mu(z) = \frac{\alpha(z)}{\alpha'(z)},$$

which implies $\beta(z) = \mu(z)\beta'(z)$. From Eq. (13) and (10) we have

$$\begin{aligned}\alpha(z) &= (-1)^k r_k(z) \\ &= -v_k(z)r_{-1}(z) + u_k(z)r_0(z) \\ &= -v_k(z)z^{t+1} + \beta(z)\tilde{\sigma}_{x,y}(z) \\ \Rightarrow \mu(z)\alpha'(z) &= -v_k(z)z^{t+1} + \mu(z)\beta'(z)\tilde{\sigma}_{x,y}(z).\end{aligned}$$

Since (α', β') is a solution we have for some polynomial ψ ,

$$\alpha'(z) = \beta'(z)\tilde{\sigma}_{x,y}(z) + \psi(z)z^{t+1}$$

Putting this in the previous equation, we get

$$\begin{aligned}\mu(z)\beta'(z)\tilde{\sigma}_{x,y}(z) + \mu(z)\psi(z)z^{t+1} &= -v_k(z)z^{t+1} + \mu(z)\beta'(z)\tilde{\sigma}_{x,y}(z) \\ -\mu(z)\psi(z) &= v_k(z).\end{aligned}$$

Thus, μ divides v_k . On the other hand $\mu(z)\beta'(z) = \beta(z) = u_k(z)$. Thus, μ also divides u_k . But u_k and v_k are relatively prime from Theorem 2. Thus, μ is a constant, and hence $\deg(\alpha') = \deg(\alpha)$ and $\deg(\beta') = \deg(\beta)$. \square

4.2 When the Asymmetric ℓ_1 -Distances are More Than the Thresholds

We now consider the other case, i.e., when $\|y \dot{-} x\|_1 > t_+$ or $\|x \dot{-} y\|_1 > t_-$, where t_+ and t_- are non-negative integers satisfying $t_+ + t_- = t$. From Proposition 11, equivalently, this means that $\deg(\sigma_{y \dot{-} x}) > t_+$ or $\deg(\sigma_{x \dot{-} y}) > t_-$. The proof of Theorem 3 tells us that even if the key equation (Eq. (11)) is not satisfied, there will still be a solution if we run the EEA till $\deg(r_k) \leq t_+$ and $\deg(r_{k-1}) > t_+$, since we have $\deg(u_k) \leq t_-$ (from Eq. (12)) which has the same degree as $\sigma_{x \dot{-} y}$. However, if we change the condition to running the algorithm until $\deg(r_k) < t_+$ and $\deg(r_{k-1}) \geq t_+$, then we see that from Theorem 2

$$\deg(u_k) = \deg(r_{-1}) - \deg(r_{k-1}) \leq t + 1 - t_+ = t_- + 1.$$

Furthermore from Theorem 2,

$$\begin{aligned} \deg(r_k) &= \deg(r_{-1}) - \sum_{i=1}^{k+1} \deg(q_i) \\ \Rightarrow \deg(u_k) + \deg(q_{k+1}) &= \deg(r_{-1}) - \deg(r_k) \\ &> t + 1 - t_+ = t_- + 1, \end{aligned}$$

where we have used the fact that $\deg(u_k) = \sum_{i=1}^k \deg(q_i)$ from Theorem 2. Together, we get the condition:

$$t_- + 1 \geq \deg(u_k) > t_- + 1 - \deg(q_{k+1})$$

where q_{k+1} is defined as

$$r_{k-1} = q_{k+1}r_k + r_{k+1}.$$

Since $\deg(r_{k+1}) < \deg(r_k)$, we have

$$\begin{aligned} \deg(r_{k-1}) &= \deg(q_{k+1}) + \deg(r_k) \\ \Rightarrow \deg(q_{k+1}) &= \deg(r_{k-1}) - \deg(r_k) \geq 1. \end{aligned}$$

Thus if $\deg(q_{k+1}) = 1$ then

$$t_- + 1 \geq \deg(u_k) > t_- \Rightarrow \deg(u_k) = t_- + 1.$$

However, $\deg(q_{k+1})$ could be greater than 1. In general if $\deg(q_{k+1}) = \delta + 1$, where $\delta \geq 0$ is an integer, we get that

$$t_- + 1 \geq \deg(u_k) > t_- - \delta.$$

Thus, once the EEA stops at the condition $\deg(r_k) < t_+$ and $\deg(r_{k-1}) \geq t_+$, we could flag it as a non-solution if $\deg(u_k) > t_- - \delta$ for some fixed integer $\delta \geq 0$. $\delta = 0$ is guaranteed to happen. But $\delta = 1$ is less probable, $\delta = 2$ even less so and so on. But how probable is this? Let τ denote the degree of r_{k-1} . Then from the equation:

$$r_{k-2} = q_k r_{k-1} + r_k,$$

we see that the polynomial on the left hand side and the one on the right hand side is of the form:

$$A_\tau z^\tau + A_{\tau-1} z^{\tau-1} + \dots + A_1 z + A_0 = B_\tau z^\tau + B_{\tau-1} z^{\tau-1} + \dots + B_1 z + B_0 + r_k(z).$$

Since $\deg(r_{k-2}) > \deg(r_{k-1})$, we have $A_\tau = B_\tau$. Now, $A_{\tau-1} = B_{\tau-1}$ implies that

$$\deg(r_{k-1}) - \deg(r_k) = \deg(q_{k+1}) \geq 2,$$

and in general $A_{\tau-1} = B_{\tau-1}, \dots, A_{\tau-\delta} = B_{\tau-\delta}$ implies that

$$\deg(r_{k-1}) - \deg(r_k) = \deg(q_{k+1}) \geq \delta + 1.$$

The EEA starts by dividing z^{t+1} by $\tilde{\sigma}_{x,y}(z)$. The coefficients of $\tilde{\sigma}_{x,y}$ are sums of products of random elements (without replacement)

of \mathbb{F} (due to the vector \mathbf{a}). Thus, we can model them as random coefficients in \mathbb{F} . Thus the probability can be approximated as

$$\Pr[\deg(q_{k+1}) \geq \delta + 1] \approx \frac{1}{|\mathbb{F}|^\delta} = \frac{1}{|\mathbb{Z}_p|^\delta} = \frac{1}{p^\delta}. \quad (15)$$

If \mathbb{F} is large enough, say around 1000, then setting $\delta = 3$ suffices, as the chance of $\delta \geq 3$ is approximately one in a billion. Unfortunately, we do not have an analytical proof of this, which we leave as an open problem. However, we can show through simulations that this is a very good estimate of the probability, as shown in Figure 3. We have chosen very small values of n and t , since the probabilities are very small. The probabilities are obtained by implementing the scheme in the Python library galois [14]. For each value of n we choose p is a prime larger than n . We then sample uniformly random images $\mathbf{x} \in \mathbb{Z}_p^n$, and make changes such that $\|y \dot{-} x\|_1 \geq t_+$ and $\|x \dot{-} y\|_1 > t_-$. We can see that across all cases the empirical probability of the left hand side of Eq. 15 is less than $p^{-\delta}$. Furthermore, the probability decreases significantly as we increase p , as is likely to be the case with the values of n used in actual images.

5 PPH CONSTRUCTION AND SECURITY ANALYSIS

We first precisely define the asymmetric ℓ_1 -distance predicate on images from \mathbb{Z}_q^n .

Definition 3. Let n be a positive integer. Let $q \geq 2$ and $\delta \geq 0$ be integers. Let t be a positive integer and let t_+ and t_- be non-negative integers with $t = t_+ + t_-$. The two-input asymmetric ℓ_1 -distance predicate P_{as} is defined as

$$P_{\text{as}}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if } \|y \dot{-} x\|_1 < t_+ \text{ and } \|x \dot{-} y\|_1 \leq t_- - \delta \\ 0, & \text{otherwise} \end{cases}$$

□

The (m, n) -PPH construction for this predicate is shown in Construction 1. We use the fact that a degree- t polynomial with coefficients in \mathbb{Z}_p can be represented as a vector in \mathbb{Z}_p^{t+1} , where the i th coefficient of the polynomial is the i th element in the vector representation. Thus, $m = (t + 1) \log_2 p \approx t \log_2 n$.

5.1 Correctness and Efficiency

PROPOSITION 13. Construction 1 is a $(1, m, n)$ -PPH for the asymmetric ℓ_1 -distance predicate P_{as} , with $m = (t + 1) \log_2 p$ and $p > n$ being a prime.

PROOF. The 1-correctness of the PPH follows almost directly from Theorem 3. More elaborately, if \mathbf{x} and \mathbf{y} are such that $\|y \dot{-} x\|_1 < t_+$ and $\|x \dot{-} y\|_1 \leq t_- - \delta$, then $P_{\text{as}}(\mathbf{x}, \mathbf{y}) = 1$. Let σ_x and σ_y be the corresponding σ -polynomials. From Proposition 12, we have $\gcd(\sigma_x, z^{t+1}) = 1$, and therefore the inverse $\sigma_x^{-1}(z)$ exists. Thus, we can obtain $\tilde{\sigma}_{x,y}(z) = \sigma_x^{-1}(z)\sigma_y(z) \pmod{z^{t+1}}$. We can then run the EEA with inputs $\tilde{\sigma}_{x,y}$ and z^{t+1} . Changing the stopping condition to $\deg(r_k) < t_+$ and $\deg(r_{k-1}) \geq t_+$ does not change the result of Theorem 3 as can be easily verified. Therefore, since $\deg(\sigma_{y \dot{-} x}) < t_+$ and $\deg(\sigma_{x \dot{-} y}) \leq t_- - \delta \leq t_-$, the theorem guarantees that the degrees of the polynomials α and β , obtained through the EEA,

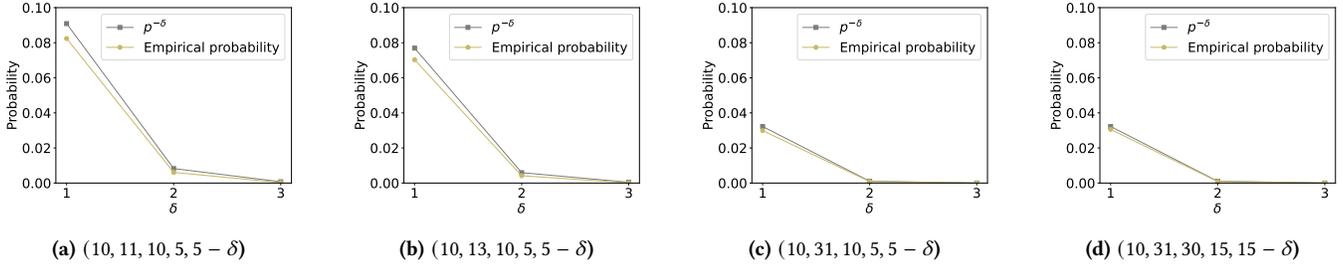


Figure 3: The probability $p^{-\delta}$ from Eq. (15) versus the empirical probability obtained after 10^6 runs with varying n and $p > n$. We use $q = 5$ in all plots. In all cases, the empirical probability is lower than $p^{-\delta}$. Tuples are the values (n, p, t, t_+, t_-) .

Construction 1: An (m, n) -PPH for the Asymmetric ℓ_1 -Distance Predicate

Parameters: Security parameter λ , positive integers $n = n(\lambda)$ and $q \geq 2$ for \mathbb{Z}_q^n , positive integers $t = t(\lambda)$, t_+ and t_- , with $t = t_+ + t_-$, integer $\delta \geq 0$, input image $\mathbf{x} \in \mathbb{Z}_q^n$.

- $\text{samp}(1^\lambda)$:
 1. Set p as the first prime after n
 2. Generate $\mathbf{a} = (a_1, \dots, a_n)$ as a vector with n distinct elements from $\mathbb{Z}_p - \{0\}$.
 3. Output the following hash function h :

$$h(\mathbf{x}) = \sigma_{\mathbf{x}}(z) \pmod{z^{t+1}} \in \mathbb{Z}_p^{t+1}$$

where

$$\sigma_{\mathbf{x}}(z) = \prod_{i=1}^n (1 - a_i z)^{x_i} \text{ with coefficients in } \mathbb{Z}_p$$

- $\text{eval}_h(X, Y)$: Let $X, Y \in \mathbb{Z}_p^{t+1}$.

1. Compute:

$$\tilde{\sigma}_{\mathbf{x}, \mathbf{y}}(z) = \sigma_{\mathbf{x}}^{-1}(z) \sigma_{\mathbf{y}}(z) \pmod{z^{t+1}}.$$

2. Set $r_{-1}(z) = z^{t+1}$ and $r_0(z) = \tilde{\sigma}_{\mathbf{x}, \mathbf{y}}(z)$ in the EEA, and run the algorithm until reaching an $r_k(z)$ such that

$$\deg(r_k) < t_+ \text{ and } \deg(r_{k-1}) \geq t_+.$$

3. Output 1 if $\deg(r_k) \leq t_- - \delta$, else output 0.
-

are equal to the degrees of $\deg(\sigma_{\mathbf{y}-\mathbf{x}})$ and $\deg(\sigma_{\mathbf{x}-\mathbf{y}})$, respectively. Thus eval_h will output 1 in this case. Thus,

$$\Pr_{h \leftarrow \text{samp}(1^\lambda)} [P(x_1, x_2) \neq \text{eval}_h(h(x_1), h(x_2)) \mid P(x_1, x_2) = 1] = 0$$

□

For 0-correctness we only have the following conjecture due to Eq. (15):

$$\Pr_{h \leftarrow \text{samp}(1^\lambda)} [P(x_1, x_2) \neq \text{eval}_h(h(x_1), h(x_2)) \mid P(x_1, x_2) = 0] \approx p^{-\delta}$$

Asymptotically, $p^{-\delta}$ is not a negligible function of λ , as p is the next prime to n , which itself is polynomial in λ . However, in practice,

this can be made extremely small. For instance, for $n = 224 \times 224 \times 3$ and $\delta = 3$, we have $p^{-\delta} < 2^{-51}$. Since the above theorem holds for all vectors \mathbf{a} sampled through $\text{samp}()$, we also have:

PROPOSITION 14. *Construction 1 is a robust $(1, m, n)$ -PPH for the asymmetric ℓ_1 -distance predicate P_{as} , with $m = (t + 1) \log_2 p$ and $p > n$ being a prime.* □

We do not have an equivalent conjecture to claim that our scheme is also $(0, m, n)$ -RPPH, as it may be possible to find a pair of images \mathbf{x} and \mathbf{y} , given an instant of the hash function h from the family, i.e., the prime p and vector \mathbf{a} . Given the σ -polynomial $\sigma_{\mathbf{x}}$ of an image \mathbf{x} , even though we can trivially find a collision in the polynomial space, e.g., $z^{t+1} + \sigma_{\mathbf{x}}$, the resulting polynomial needs to be a valid σ -polynomial of some image \mathbf{y} for it to be a collision in the image space. We discuss reverse-engineering the original image in Section 5.3, which also discusses finding collisions in the image space.

From the theorem, the compression achieved by our scheme is $\approx t \log_2 n$. For small t the compression rate is close to the lower bound as shown in Table 1. Unfortunately, for large t , not much compression is possible. It is unclear whether further compression is possible for robust PPH families, since the compression bounds from Section 3 are for non-robust PPH families. For efficiency of the construction, we have the following theorem.

THEOREM 4. *Let h be a hash function from Construction 1. Then h and eval_h can be computed in $O(nt^2)$ and $O(t^2)$ time, respectively.*

PROOF. See Appendix B. □

5.2 Application to Adversarial Image Detection

For the adversarial image search scenario, we first need a setup algorithm to store the σ -polynomials, or rather their inverses, of all images $\mathbf{x}_i \in \mathcal{D}$. At the time of submitting an input image, the user needs to prepare the σ -polynomial for his/her image to send to the server. Finally, the server runs the detection algorithm which evaluates to 1 if for any image in the dataset we have a match according to the asymmetric ℓ_1 -distance predicate. These algorithms are detailed in Algorithms 2, 3 and 4, respectively.

5.3 Information Leakage and Inverting the Hash Function

What Does the eval_h Function Reveal? From Theorem 2, the EEA returns two polynomials $\alpha(z)$ and $\beta(z)$. Although not explicitly

Algorithm 2: Setup

Input: All inputs to Construction 1, database \mathcal{D} of N images $\mathbf{x}_1, \dots, \mathbf{x}_N$.

- 1 Run $\text{samp}(1^\lambda)$ to obtain the hash function h
 - 2 **for** $i = 1$ **to** N **do**
 - 3 Obtain $h(\mathbf{x}_i) = \sigma_{\mathbf{x}_i}(z) \pmod{z^{t+1}}$
 - 4 Compute $\sigma_{\mathbf{x}_i}^{-1}(z)$ modulo z^{t+1}
 - 5 Replace \mathbf{x}_i with $h(\mathbf{x}_i)^{-1} = \sigma_{\mathbf{x}_i}^{-1}(z)$ in \mathcal{D}
 - 6 **return** \mathcal{D}
-

Algorithm 3: Prepare

Input: Hash function h from Construction 1, image y .

- 1 **return** $h(y) = \sigma_y(z) \pmod{z^{t+1}}$
-

Algorithm 4: Detect

Input: Hash function h from Construction 1, hash digest $h(y)$, database \mathcal{D} of inverse σ -polynomials $\sigma_{\mathbf{x}_1}^{-1} = h(\mathbf{x}_1)^{-1}, \dots, \sigma_{\mathbf{x}_N}^{-1} = h(\mathbf{x}_N)^{-1}$.

- 1 **for** $i = 1$ **to** N **do**
 - 2 **if** $\text{eval}_h(h(\mathbf{x}_i)^{-1}, h(y)) = 1$ **then**
 - 3 **return** 1
 - 4 **return** 0
-

stated these polynomials could be exactly the polynomials σ_{y-z} and σ_{x-y} . We can then factor these polynomials using a variety of efficient polynomial factorization algorithms [20]. From these factorized polynomials and knowing \mathbf{x} or y one can recover the other, given that the vector \mathbf{a} is public information. This is obviously not surprising as this method was initially proposed to correct errors [13]. However, not knowing the initial images \mathbf{x} and y , one only learns the absolute difference in pixel values between \mathbf{x} and y for all pixels. This is certainly more information than a simple 1 or 0 answer to the fact that \mathbf{x} and y satisfy P_{as} . Since the server computes the eval_h function, we can reduce this information leakage by never storing the vector \mathbf{a} at the server. Otherwise, our scheme assumes an honest server.

What Do the σ -Polynomials Reveal? The question then arises if someone adversarial gets hold of the dataset \mathcal{D} of inverse σ -polynomials, how difficult it is to find an original image? In Section 2.1 we mentioned that one of the adversarial goals is to invert the hash digest $h(\mathbf{x})$. To be precise, for each image $\mathbf{x} \in \mathcal{D}$, we store $\sigma_{\mathbf{x}}^{-1}(z) \pmod{z^{t+1}}$, which is a degree t polynomial. We can easily compute its inverse to obtain $\sigma_{\mathbf{x}}(z) \pmod{z^{t+1}}$, which is again a degree t polynomial. The general form of this polynomial is:

$$\sigma_{\mathbf{x}}(z) = A_t z^t + \dots + A_1 z + 1$$

So, the question reduces to what do the coefficients A_i 's reveal about \mathbf{x} ? To answer this, we have the following theorem:

THEOREM 5. *Let \mathbf{x} be an image with the σ -polynomial:*

$$\sigma_{\mathbf{x}}(z) = \prod_{i=1}^n (1 - a_i z)^{x_i}.$$

as given by Eq. (8). Let $m = \sum_{i=1}^n x_i$ be the degree of this polynomial as given by Proposition 11. Let A_j be the j th coefficient of this polynomial, with $0 \leq j \leq m$. Then,

$$A_j = (-1)^j S(j, n),$$

where

$$S(j, n-k) = \sum_{i=0}^j \binom{x_{k+1}}{i} a_{k+1}^i S(i, n-k-1),$$

for $0 \leq k \leq n$.

PROOF. See Appendix B. □

So, for example $S(0, n) = S(0, n-1) = \dots = S(0, 0) = 1$, and hence $A_0 = 1$. Likewise, $S(1, n) = x_1 a_1 S(0, n-1) + S(1, n-1) = x_1 a_1 + S(1, n-1)$. By the recursive nature of the definition, we get $S(1, n-1) = x_2 a_2 + S(1, n-2)$. Continuing on, we get $S(1, 1) = x_n a_n + S(1, 0) = x_n a_n$. Thus, $S(1, n) = \sum_{i=1}^n x_i a_i$, and so $A_1 = -\sum_{i=1}^n x_i a_i$. While these values can be easily computed if we know the vectors \mathbf{a} and \mathbf{x} , not knowing the later means that we need to try q^n possibilities, i.e., all possible values of x_1 , times all possible values of x_2 , and so on, to see which ones match A_j . Thus, the complexity of finding \mathbf{x} from the σ -polynomial of \mathbf{x} through this way is proportional to $\mathcal{O}(q^n)$.

6 EXPERIMENTAL RESULTS

We use the Imagenette dataset [15] (Imagenette-320 to be specific) for our adversarial image detection application, which is a smaller subset of the well-known Imagenet dataset [22]. This is a dataset of 9,459 RGB images of size 224×224 . Thus $n = 224 \times 224 \times 3$.

Similarity Metrics. To measure the difference between original and adversarial images we use three similarity metrics: (a) the Learned Perceptual Image Patch Similarity (LPIPS) metric [23], which is widely used as a proxy for human perceptual similarity. Generally, a perturbed image is similar to the original one when its LPIPS is lower than 0.2, and the difference can be perceived significantly when it is more than 0.3 [23], (b) Pixel Change Ratio, which shows the percentage change in absolute pixel values compared to the original image, and (c) normalized asymmetric ℓ_1 -distance (NAD) for two images \mathbf{x} and \mathbf{x}^* :

$$\text{NAD}(\mathbf{x}, \mathbf{x}^*) = \frac{\max\{\|\mathbf{x} - \mathbf{x}^*\|_1, \|\mathbf{x}^* - \mathbf{x}\|_1\}}{qn} \times 100 \quad (16)$$

Recall that the maximum possible distance between two images is $(q-1)n \approx qn$. In this section we assume that $t_+ = t_- = \frac{t}{2}$. Note that for $P_{\text{as}}(\mathbf{x}, \mathbf{x}^*)$ to evaluate to 1, we must have:

$$t_+ = t_- = \frac{t}{2} \leq \frac{qn \times \text{NAD}(\mathbf{x}, \mathbf{x}^*)}{100} \Rightarrow t \leq \frac{qn \times \text{NAD}(\mathbf{x}, \mathbf{x}^*)}{50} \quad (17)$$

6.1 Possible Values of t

Ideally, the threshold t for the asymmetric ℓ_1 -distance predicate P_{as} should be such that for any two images in the database \mathcal{D} the predicate evaluates to 0, thus ensuring zero false positives. To do so, we define the *empirical error* on the database \mathcal{D} as:

$$\text{err}_{P_{\text{as}}}(\mathcal{D}) = \sum_{1 \leq i < j \leq N} \frac{P_{\text{as}}(\mathbf{x}_i, \mathbf{x}_j)}{\binom{N}{2}}, \quad (18)$$

and the parameters of P_{as} are $t_+ = t_- = \frac{t}{2}$ and $\delta = 3$. Although in general $P_{as}(x_i, x_j) \neq P_{as}(x_j, x_i)$, with $t_+ = t_-$ and large t the difference is not profound enough to matter. We plot the error $\text{err}_{P_{as}}(\mathcal{D})$ in Figure 4 against increasing values of t . For all values of $t \leq 325,000$ we get $\text{err}_{P_{as}}(\mathcal{D}) = 0$. This is $t \approx 0.008qn$, or $\text{NAD} \approx 0.4$ from Eq. (17). Thus, any value of t less than this should produce no false positives for images in this dataset. Unfortunately, as we shall see, this value of t is not high enough to defend against some attacks. We note that until $t = 2,000,000$ which is $t \approx 0.05qn$ or $\text{NAD} \approx 2.5$, the error rate is less than 2%. Thus, we can discard the few images that cause ‘‘collisions’’ which is most likely because these images are similar, to use a higher value of t .

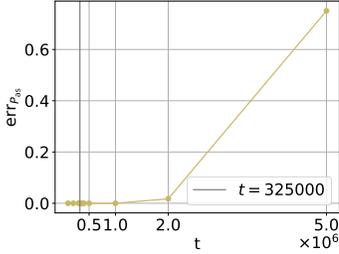


Figure 4: The empirical error on the Imagenette dataset \mathcal{D} . We get non-zero error with $t \geq 350,000$.

6.2 Impact of Adversarial Attacks

The Fast Gradient Sign Method (FGSM) [24] and Projected Gradient Descent (PGD) [11] are two well-known adversarial input attacks. In both attacks the noise parameter ϵ can be adjusted to add more noise to the input image, with the cross-entropy loss as the objective function. Figure 5 shows the impact on the quality of an image through LPIPS, pixel change ratio, and NAD as we increase ϵ in FGSM. At $\epsilon = 0.4$, the NAD is 0.4512, which approximately corresponds to $t = 325,000$ from Eq (17). At this t , the LPIPS is 0.1786. At $\epsilon = 0.1$, with $\text{NAD} = 1.1277$, which corresponds to $t \approx 869,122$, we have LPIPS 0.5714, and hence the image quality has significantly degraded. Due to lack of space, a similar graph for PGD is relegated to Figure 7 in Appendix C.

To show the impact on a larger number of images, we sample 1,000 images from the imagenette dataset and apply both the FGSM and PGD attacks on them by varying ϵ . The results are shown in Table 2. For FGSM, the average NAD of 1.1287, and for PGD an average NDA of 0.8163 produces LPIPS of 0.5596 and 0.4485, respectively, which is significant perceptual loss on the images.

6.3 Impact of Image Transformations

Apart from adversarial input attacks, an image can also be changed to evade detection via simple image transformations [1]. These include filters such as increasing brightness, adjusting contrast, rotating, and cropping the image. We select two such techniques: changing brightness and contrast. By adjusting an *enhancement factor* we can vary the brightness and contrast, with a value of 1 giving the original image.⁵ We use the same symbol, i.e., ϵ , to denote

⁵See <https://pillow.readthedocs.io/en/stable/reference/ImageEnhance.html>

ϵ	Attack	LPIPS	NAD	Attack	LPIPS	NAD
0.000	FGSM	0.0000	0.0000	PGD	0.0000	0.0000
0.005		0.0026	0.0580		0.0016	0.0379
0.020		0.0524	0.2266		0.0145	0.1005
0.050		0.2454	0.5644		0.0470	0.1768
0.100		0.5596	1.1287		0.1187	0.2973
0.200		0.9913	2.2576		0.2871	0.5523
0.300		1.2105	3.3857		0.4485	0.8163

Table 2: Statistics for different ϵ values on FGSM and PGD.

the enhancement factor. We show the impact of ϵ on the image quality using the same sample image in Figure 7 in Appendix C. One thing to note from the figure is that LPIPS is lower for these two techniques, even though the pixel change ratio is very high compared to the FGSM and PGD attacks. Thus, for these two attacks a lower value of LPIPS will suffice to deteriorate the adversarial image. In Table 3 we show the average LPIPS and NAD on the 1,000 images chosen in our experiment. For both techniques, $\epsilon = 0.250$ results in LPIPS of more than 0.3.

ϵ	Attack	LPIPS	NAD	Attack	LPIPS	NAD
0.250	Brightness	0.3183	6.8516	Contrast	0.3204	1.7338
0.500		0.1140	4.5666		0.1109	1.1559
0.750		0.0244	2.3057		0.0236	0.5695
1.000		0.0000	0.0000		0.0000	0.0000
1.250		0.0258	1.9095		0.0179	0.5303
1.500		0.0800	3.5170		0.0526	0.9300
2.000		0.1955	5.8659		0.1208	1.4895

Table 3: Statistics for Brightness and Contrast attacks.

6.4 Computational Time

We implement the PPH scheme using the Python library `galois` [14]. We set $t = 0.01qn$ which is above $0.008qn$ established in Section 6.1. We choose $t_+ = t_- = t/2$, and $\delta = 3$. For small grayscale images, such as the 28×28 size images used in the MNIST dataset [25], our scheme can compute the σ -polynomial of an image x in about 0.26 seconds. This is a one-off cost for the client, and hence not prohibitive. For database creation, we need to convert each image $x \in \mathcal{D}$ to its σ -polynomial and then invert it. This can be done in time 0.66 seconds per image. Finally the eval_h function can be computed in 0.08 seconds given the σ -polynomials of the input image and the inverse σ -polynomial of the target image.

For larger image sizes, the time can grow large as our algorithms are of the order $O(t^2)$. Our idea is to divide the image into blocks, and then compute the predicate P_{as} per block with $n_B = n/B$ being the size of each block and $t_B = t/B$ being the threshold per block, for a block size of B . Note that dividing the image into blocks is not unprecedented. It is done by the Blockhash perceptual hashing algorithm [26], and this strategy is known to be more robust against local changes to images [3, 27]. With this, for an RGB 224×224 image, the eval_h can be computed in about 0.013 seconds per block or 13 second per image. Detailed times are shown in Table 4 which is the result of running the algorithms a total of 1,000 times. To

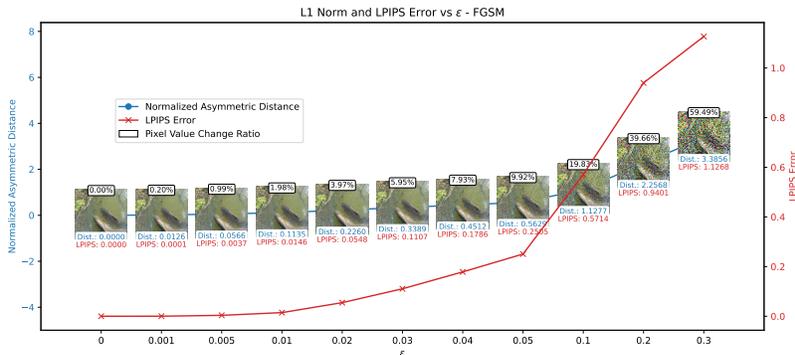


Figure 5: The impact of adding noise to the image using the FGSM attack on the metrics LPIPS, pixel change ratio and NAD.

Size	Color	Blocks	n_B	t_B	Time σ	Time σ^{-1}	Time $eval_h$
224×224	RGB	1000	150	384	0.0235	0.0963	0.0128
128×128	RGB	100	491	1256	0.1238	0.3712	0.0455
64×64	RGB	100	122	312	0.0185	0.0777	0.0101
28×28	RGB	10	235	601	0.0436	0.1584	0.0204
28×28	Gray	1	784	2007	0.2596	0.6667	0.0784

Table 4: Total time in seconds taken by our scheme to produce the σ -polynomial of an image, the σ -polynomial and its inverse of an image, and the solution using the $eval_h$ function, where the images are divided into blocks with $t = 0.01qn$.

Attack	ϵ	LPIPS	NAD	n_B	t_B	Time σ	Time σ^{-1}	Time $eval_h$
FGSM	0.100	0.5596	1.1287	150	869	0.0303	0.1993	0.0309
	0.300	0.4485	0.8163	150	629	0.0280	0.1506	0.0220
Brightness	0.250	0.3183	6.8516	150	5280	0.0847	1.3360	0.2651
	2.000	0.1955	5.8659	150	4520	0.0763	1.0865	0.2143
Contrast	0.250	0.3204	1.7338	150	1336	0.0386	0.2999	0.0489
	2.000	0.1208	1.4895	150	1147	0.0358	0.2592	0.0422

Table 5: Total time in seconds taken by our scheme to produce the σ -polynomial of an image, the σ -polynomial and its inverse of an image, and the solution using the $eval_h$ function, where t is chosen such that LPIPS is high for each attack.

calculate the time of $eval_h$ we alter the image such that the predicate is not satisfied, which is the worst-case time. In Table 5, we further show the time taken when we choose t such that LPIPS is high for each of the four attacks. Here t is chosen from NAD according to Eq. 17. The corresponding values of ϵ and LPIPS are taken from Tables 2 and 3.

Our implementation was done on a standard Apple M3 ARM processor with 8 (performance) cores, and 16 GB RAM. We note that with more cores, and involving GPUs, these times can be substantially improved, as the algorithms are parallelizable: each block and each database image can be evaluated separately.

7 RELATED WORK

The notion of robust property-preserving hash (RPPH) functions was formally introduced by Boyle et al. [8], where they also give a construction of an RPPH for gap-Hamming distance predicate: the predicate outputs 1 if the Hamming distance is lower than one

threshold, 0 if it is higher than the other, and a special symbol if it lies within the gap. The authors also show a construction of a non-robust PPH for gap-Hamming distance predicate using a locality sensitive hash (LSH) from [28]. Following their work, new constructions for gap-Hamming distance as well as exact Hamming distance predicates have been proposed [6, 9, 10]. In particular, the construction from [6] is based on the idea of efficient list decoding of linear codes. This inspired us to search for list decoding of errors measured in the Euclidean distance (ℓ_2 -distance). Mook and Peikert [29] show a construction of list decoding of error-correcting codes based on Reed-Solomon codes for the ℓ_2 -distance. Unfortunately, as we show in Section 3.4, this procedure is unlikely to be efficient even for small values of the distance threshold t , as the size of the list (possible vectors) blow up. While our result is for the ℓ_1 -metric, from Proposition 1, it also applies to the ℓ_2 -metric. Our construction is based on ℓ_1 -error correcting codes from [13], which themselves are derived from their earlier construction of error-correcting codes for the Hamming distance [30]. Several adversarial attacks have been shown against perceptual hashing algorithms [1, 3]. In particular, evasion attacks add small adversarial noise to cause a mismatch in the hashes of two perceptually similar images. This attack does not apply to property-preserving hashing since the probability of a mismatch between the predicate on the original domain and the hash domain is required to be negligible.

8 CONCLUSION

We have proposed the first property-preserving hash (PPH) function family for (asymmetric) ℓ_1 -distance predicates, with applications to countering adversarial input attacks. Our construction is efficient, as shown through our implementation. Our work leaves a number of avenues for future research. While the proposed hash function shows high compression for smaller distance thresholds t , our theoretical results show that further compression may be possible, especially when t is large. Furthermore, our scheme is only robust against one-sided errors and only handles asymmetric ℓ_1 -distance predicates. It remains an open problem to find a robust PPH for the exact ℓ_1 -distance predicate. There may also be interest in finding a robust PPH scheme for the Euclidean distance predicate. Finally, the implementation of our scheme has the potential to be further sped up due to its highly parallel nature.

ACKNOWLEDGMENTS

We are grateful to Quang Cao, Ron Steinfeld and Josef Pieprzyk for helpful discussions on earlier ideas on this topic.

REFERENCES

- [1] L. Struppek, D. Hintersdorf, D. Neider, and K. Kersting, "Learning to break deep perceptual hashing: The use case neuralhash," in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 58–69.
- [2] E. Klingren and D. Starkweather, "phash: The open source perceptual hash library," 2013.
- [3] Q. Hao, L. Luo, S. T. Jan, and G. Wang, "It's not what it looks like: Manipulating perceptual hashing based applications," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 69–85.
- [4] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 604–613.
- [5] C. Zauner, "Implementation and benchmarking of perceptual image hash functions," 2010.
- [6] J. Holmgren, M. Liu, L. Tyner, and D. Wichs, "Nearly optimal property preserving hashing," in *Annual International Cryptology Conference*. Springer, 2022, pp. 473–502.
- [7] R. O'Donnell, Y. Wu, and Y. Zhou, "Optimal lower bounds for locality-sensitive hashing (except when q is tiny)," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 1, pp. 1–13, 2014.
- [8] E. Boyle, R. LaVigne, and V. Vaikuntanathan, "Adversarially robust property preserving hash functions," *Cryptology ePrint Archive*, 2018.
- [9] N. Fleischhacker and M. Simkin, "Robust property-preserving hash functions for hamming distance and more," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021.
- [10] N. Fleischhacker, K. G. Larsen, and M. Simkin, "Property-preserving hash functions for hamming distance from standard assumptions," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2022.
- [11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=rjZiBfZAb>
- [12] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. Ieee, 2017, pp. 39–57.
- [13] L. G. Tallini and B. Bose, "On 1-1-distance error control codes," in *2011 IEEE International Symposium on Information Theory Proceedings*. IEEE, 2011, pp. 1061–1065.
- [14] M. Hostetter, "Galois: A performant NumPy extension for Galois fields," 11 2020. [Online]. Available: <https://github.com/mhostetter/galois>
- [15] J. Howard and S. Guggen, "Imagenette: A smaller subset of 10 easily classified classes from imagenet," 2019. [Online]. Available: <https://github.com/fastai/imagenette>
- [16] J. Shurman *et al.*, *Calculus and analysis in euclidean space*. Springer, 2016.
- [17] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.
- [18] S. Baumert, A. Ghatge, S. Kiatsupaibul, Y. Shen, R. L. Smith, and Z. B. Zabinsky, "Discrete hit-and-run for sampling points from arbitrary distributions over subsets of integer hyperrectangles," *Operations Research*, vol. 57, no. 3, pp. 727–739, 2009.
- [19] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.
- [20] V. Shoup, *A computational introduction to number theory and algebra*, 2nd ed. Cambridge university press, 2009.
- [21] F. MacWilliams, "The theory of error-correcting codes," *Elsevier Science Publishers BV google schola*, vol. 2, pp. 39–47, 1977.
- [22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [23] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.
- [24] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
- [25] Y. LeCun, C. Cortes, and C. Burges, "The mnist database of handwritten digits," 1998. [Online]. Available: <http://yann.lecun.com/exdb/mnist/>
- [26] B. Yang, F. Gu, and X. Niu, "Block mean value based image perceptual hashing," in *2006 International Conference on Intelligent Information Hiding and Multimedia*. IEEE, 2006, pp. 167–172.
- [27] M. Schneider and S.-F. Chang, "A robust content based digital signature for image authentication," in *Proceedings of 3rd IEEE international conference on image*

- processing*, vol. 3. IEEE, 1996, pp. 227–230.
- [28] E. Kushilevitz, R. Ostrovsky, and Y. Rabani, "Efficient search for approximate nearest neighbor in high dimensional spaces," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 614–623.
- [29] E. Mook and C. Peikert, "Lattice (list) decoding near minkowski's inequality," *IEEE Transactions on Information Theory*, vol. 68, no. 2, pp. 863–870, 2021.
- [30] L. G. Tallini and B. Bose, "On a new class of error control codes and symmetric functions," in *2008 IEEE International Symposium on Information Theory*. IEEE, 2008, pp. 980–984.
- [31] M. O'Searcoid, *Metric spaces*. Springer Science & Business Media, 2006.
- [32] D. S. Mitrinović, J. E. Pečarić, and A. M. Fink, *Bernoulli's Inequality*. Dordrecht: Springer Netherlands, 1993, pp. 65–81. [Online]. Available: https://doi.org/10.1007/978-94-017-1043-5_3
- [33] H. J. Asghar, R. Steinfeld, S. Li, M. A. Kaafar, and J. Pieprzyk, "Algebraic attacks on human identification protocols," *Cryptology ePrint Archive*, 2014.
- [34] R. Brualdi, *Introductory Combinatorics*, ser. Pearson Education. Pearson Prentice Hall, 2012.

A SOME USEFUL RESULTS

Metrics. Let S be a set. A function $d : S \times S \rightarrow \mathbb{R}$ is called a metric on S if for all $x, y, z \in S$, (1) $d(x, y) \geq 0$ with equality if and only if $x = y$, (2) $d(x, y) = d(y, x)$, and (3) $d(x, y) \leq d(x, z) + d(z, y)$ [31]. In this case S is called a metric space. For $x, y \in \mathbb{Z}_q$, i.e., pixels, define the function $d : \mathbb{Z}_q \times \mathbb{Z}_q \rightarrow \mathbb{R}$ as:

$$d(x, y) = |x - y| \quad (19)$$

It follows that d is a metric on \mathbb{Z}_q , as can be easily verified. Similarly, define the Hamming distance $d_H : \mathbb{Z}_q \times \mathbb{Z}_q \rightarrow \mathbb{R}$ as:

$$d_H(x, y) = \begin{cases} 1, & \text{if } x \neq y, \\ 0, & \text{otherwise} \end{cases} \quad (20)$$

Then d_H is also a metric. Thus, \mathbb{Z}_q together with d or with d_H is a metric space. It follows that the following are metrics spaces on the set \mathbb{Z}_q^n [31, §1.6]:

- (1) The set \mathbb{Z}_q^n together with the metric $d_0(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n d_H(x_i, y_i)$.
- (2) The set \mathbb{Z}_q^n together with the metric $d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n d(x_i, y_i)$.
- (3) The set \mathbb{Z}_q^n together with the metric $d_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n (d(x_i, y_i))^2}$.
- (4) The set \mathbb{Z}_q^n together with the metric $d_\infty(\mathbf{x}, \mathbf{y}) = \max_i \{d(x_i, y_i)\}$,

for all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, with x_i and y_i being their i th elements. With these metrics we define the following norms for all $\mathbf{x} \in \mathbb{Z}_q^n$:

- (1) ℓ_0 -norm (Hamming weight): $\|\mathbf{x}\|_0 = d_0(\mathbf{x}, \mathbf{0})$.
- (2) ℓ_1 -norm: $\|\mathbf{x}\|_1 = d_1(\mathbf{x}, \mathbf{0})$.
- (3) ℓ_2 -norm (Euclidean): $\|\mathbf{x}\|_2 = d_2(\mathbf{x}, \mathbf{0})$.
- (4) ℓ_∞ -norm: $\|\mathbf{x}\|_\infty = d_\infty(\mathbf{x}, \mathbf{0})$.

Technically these are not norms as we have not even defined the vector space of images. However, we will continue to use the term by assuming the operations to be over the vector space \mathbb{R}^n .

Miscellaneous Results.

PROPOSITION 15. *Let $a > 1.4$ be a real number, and let $q \geq 4$ be a positive integer. Then $a^q - 1 \geq a^{q-1}$.*

PROOF. We prove this via induction. For $q = 4$, the left hand side is $a^4 - 1$ and the right hand side is a^3 . For the inequality to hold we must have $a^3(a - 1) \geq 0$. Since this is an increasing function of a , and direct substitution shows that this inequality is satisfied by $a = 1.4$, it follows that it is satisfied by all $a > 1.4$. Now suppose the

result holds for $q = k$, i.e., $a^k - 1 \geq a^{k-1}$. Multiplying both sides by a , we get

$$\begin{aligned}(a^k - 1)a &\geq a^{k-1}a \\ a^{k+1} - a &\geq a^k \\ a^{k+1} - 1 &\geq a^k + a - 1 \geq a^k,\end{aligned}$$

where the last step follows from the fact that $a > 1$. \square

Bernoulli's Inequality. Let $x > -1$ be a real number and let q be a positive integer then

$$(1+x)^q \geq 1+qx \quad (21)$$

See for example [32].

PROPOSITION 16. Let $q \geq 2$ be an integer. Then the function

$$f(q) = \left(1 + \frac{0.9}{q}\right)^q$$

is an increasing function of q .

PROOF. Consider the ratio:⁶

$$\begin{aligned}\frac{f(q+1)}{f(q)} &= \frac{\left(1 + \frac{0.9}{q+1}\right)^{q+1}}{\left(1 + \frac{0.9}{q}\right)^q} = \frac{\left(1 + \frac{0.9}{q+1}\right)^{q+1}}{\left(1 + \frac{0.9}{q}\right)^q} \left(1 + \frac{0.9}{q}\right) \\ &= \left(\frac{q+1+0.9}{q+1} \frac{q}{q+0.9}\right)^{q+1} \left(1 + \frac{0.9}{q}\right) \\ &= \left(\frac{(q+1+0.9)q+0.9-0.9}{(q+1)(q+0.9)}\right)^{q+1} \left(1 + \frac{0.9}{q}\right) \\ &= \left(1 + \frac{-0.9}{(q+1)(q+0.9)}\right)^{q+1} \left(1 + \frac{0.9}{q}\right)\end{aligned}$$

Now since $q \geq 2$, we have

$$\frac{-0.9}{(q+1)(q+0.9)} \geq -\frac{0.9}{3 \times 2.9} > -1.$$

Hence we can apply Bernoulli's inequality (Eq. 21) to obtain:

$$\frac{f(q+1)}{f(q)} \geq \left(1 - \frac{0.9}{q+0.9}\right) \left(1 + \frac{0.9}{q}\right) = \left(\frac{q+0.9-0.9}{q+0.9}\right) \left(\frac{q+0.9}{q}\right) = 1.$$

Thus $f(q+1) \geq f(q)$. \square

B PROOFS

B.1 Proof of Proposition 1

PROOF. Let $\mathbf{z} = \mathbf{x} - \mathbf{y}$. Then, equivalently, we need to show that

$$\|\mathbf{z}\|_2 \leq \|\mathbf{z}\|_1 \leq \sqrt{n}\|\mathbf{z}\|_2$$

Note that \mathbf{z} may not be an image, i.e., \mathbf{z} may not be a member of \mathbb{Z}_q^n .

Now we see that

$$\begin{aligned}\|\mathbf{z}\|_1^2 &= \left(\sum_{i=1}^n |z_i|\right)^2 \\ &= \left(\sum_{i=1}^n |z_i|\right) \left(\sum_{i=1}^n |z_i|\right)\end{aligned}$$

⁶Part of this proof is derived from: <https://math.stackexchange.com/questions/1589429/how-to-prove-that-logxx-when-x1/>.

$$\begin{aligned}&= \sum_{i=1}^n |z_i|^2 + \sum_{i,j:i \neq j} |z_i||z_j| \\ &\geq \sum_{i=1}^n |z_i|^2 = \|\mathbf{z}\|_2^2,\end{aligned}$$

from which it follows that $\|\mathbf{z}\|_2 \leq \|\mathbf{z}\|_1$. For the second inequality, let $\mathbf{1}$ be the vector of all 1's, and let \mathbf{b} be such that $b_i = |z_i|$ for all i . Then,

$$\|\mathbf{z}\|_1 = \sum_{i=1}^n |z_i| = \langle \mathbf{1}, \mathbf{b} \rangle \leq \|\mathbf{1}\|_2 \|\mathbf{b}\|_2 = \sqrt{n}\|\mathbf{z}\|_2,$$

where we have used the Cauchy-Schwartz inequality. To show that the bounds are tight, let us first consider the left hand side inequality. Assume that for some positive constant c we have $c\|\mathbf{x} - \mathbf{y}\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_1$. Consider the two images $\mathbf{x} = (1, 0, \dots, 0)$ and $\mathbf{y} = \mathbf{0}$, where $\mathbf{0}$ is the zero vector. Then,

$$c\|\mathbf{x} - \mathbf{y}\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_1 \Rightarrow c\sqrt{1} \leq 1 \Rightarrow c \leq 1.$$

Thus, $c = 1$ is tight. For the RHS, assume that for some positive constant c , we have $\|\mathbf{x} - \mathbf{y}\|_1 \leq c\|\mathbf{x} - \mathbf{y}\|_2$. Consider the two images $\mathbf{x} = (q-1, q-1, \dots, q-1)$ and $\mathbf{y} = \mathbf{0}$. Then,

$$\|\mathbf{x} - \mathbf{y}\|_1 \leq c\|\mathbf{x} - \mathbf{y}\|_2 \Rightarrow n(q-1) \leq c\sqrt{n}(q-1) \Rightarrow c \geq \sqrt{n}.$$

\square

B.2 Proof of Proposition 3

PROOF. Let $\mathbf{y}' \in B_1(\mathbf{0}, q, t)$. Construct the vector \mathbf{y} , whose i th coordinate is:

$$y_i = \begin{cases} x_i - y'_i, & \text{if } x_i \geq y'_i, \\ y'_i, & \text{otherwise} \end{cases}$$

Then clearly $\mathbf{y} \in \mathbb{Z}_q^n$. Now $\|\mathbf{x} - \mathbf{y}\|_1 = \sum_{i=1}^n |x_i - y_i|$. Consider the i th term in the sum. If $x_i \geq y'_i$, then

$$|x_i - y_i| = |x_i - (x_i - y'_i)| = |y'_i|,$$

otherwise if $x_i < y'_i$, then

$$|x_i - y_i| = |x_i - y'_i| = y'_i - x_i \leq y'_i = |y'_i|,$$

where the inequality follows from the fact that $x_i \geq 0$. Thus, in both cases $|x_i - y_i| \leq |y'_i|$. Therefore, $\|\mathbf{x} - \mathbf{y}\|_1 \leq \|\mathbf{y}'\|_1 \leq t$. Thus, $\mathbf{y} \in B_1(\mathbf{x}, q, t)$.

Next, we show that this mapping is injective. Consider two different vectors $\mathbf{y}', \mathbf{y}'' \in B_1(\mathbf{0}, q, t)$. Assume one of the coordinates they differ in is the i th coordinate. Assume to the contrary that the map defined above yields a vector with the same i th coordinate y_i for both. If $x_i \geq y'_i$ and $x_i \geq y''_i$, or if $x_i < y'_i$ and $x_i < y''_i$, then we get $y'_i = y''_i$ through the map above, which is a contradiction. Thus, either $x_i \geq y'_i$ and $x_i < y''_i$, or $x_i < y'_i$ and $x_i \geq y''_i$. Assume $x_i \geq y'_i$ and $x_i < y''_i$. Then, we get $y_i = x_i - y'_i = y''_i \Rightarrow x_i = y'_i + y''_i$. Since $y'_i \geq 0$, this means that $x_i \geq y''_i$. But this contradicts the fact that $x_i < y''_i$. The case when $x_i < y'_i$ and $x_i \geq y''_i$ is analogous. \square

B.3 Proof of Proposition 4

PROOF. To simplify notation let $C(t, n)$ denote $|B_1(\mathbf{0}, q, t)|$. Then,

$$C(t, n) = \sum_{i=0}^{q-1} C(t-i, n-1).$$

That is, we fix one element of the vector to i , and count all vectors of $(n-1)$ elements whose sum is $t-i$. The count is then complete by summing over all possible values the fixed element in the original vector can take. Also, note that

$$\begin{aligned}
C(t-1, n) &= \sum_{i=0}^{q-1} C(t-1-i, n-1) \\
&= C(t-1, n-1) + C(t-2, n-1) + \dots \\
&\quad + C(t-q+1, n-1) + C(t-q, n-1) \\
&= C(t, n-1) + C(t-1, n-1) + C(t-2, n-1) + \dots \\
&\quad + C(t-q+1, n-1) \\
&\quad + C(t-q, n-1) - C(t, n-1) \\
&= C(t, n) + C(t-q, n-1) - C(t, n-1) \\
&\Rightarrow C(t, n) = C(t-1, n) + C(t, n-1) - C(t-q, n-1).
\end{aligned}$$

The recurrence relation $C(t-1, n) + C(t, n-1)$ has the solution $\binom{n+t}{t}$ (see for example [33, §C]). Therefore,

$$C(t, n) = \binom{n+t}{t} - C(t-q, n-1) \leq \binom{n+t}{t},$$

since $C(t-q, n-1) \geq 0$. From the above, we see that

$$\begin{aligned}
C(t-q, n-1) &= \binom{n-1+t-q}{t-q} - C(t-2q, n-2) \\
&\leq \binom{n-1+t-q}{t-q}.
\end{aligned}$$

Therefore

$$C(t, n) \geq \binom{n+t}{t} - \binom{n-1+t-q}{t-q}$$

□

B.4 Proof of Proposition 5

PROOF. Again, to simplify notation let $C(t, n)$ denote $|B_1(\mathbf{y}, q, t)|$. Let $C'(t, n)$ denote $|B_1(\mathbf{y}, q+1, t)|$. Then it is easy to see that $C(t, n) \leq C'(t, n)$. We will work with $C'(t, n)$. First note that

$$C'(t, n) = 1 + 2 \sum_{i=0}^{q/2-1} C'(t-i, n-1).$$

This is because, if we fix one element of a vector $\mathbf{x} \in \mathbb{Z}_{q+1}^n$ within the ℓ_1 ball of \mathbf{y} to i , we are left with counting all vectors of $n-1$ elements whose sum is $t-i$. Without loss of generality, let us assume that this is the first element of \mathbf{x} . Since the corresponding element in \mathbf{y} is fixed at $q/2$, $|i - q/2|$ ranges from 1 to $q/2 - 1$. This counts all vectors whose first element is fixed at $i \in \{0, 1, \dots, q/2 - 1\}$. The first element of \mathbf{x} can also take on the values $q/2 + 1$ to q . For each such value, there is a corresponding value between 0 and $q/2 - 1$. Thus, we need to count such vectors is twice of the sum of $C'(t-i, n-1)$ as i ranges from 0 to $q/2 - 1$. The remaining case is when \mathbf{x} is identical to \mathbf{y} . This is a single vector. Now,

$$\begin{aligned}
&C'(t-1, n) \\
&= 1 + 2 \sum_{i=0}^{q/2-1} C'(t-1-i, n-1) \\
&= 1 + 2 (C'(t-1, n-1) + C'(t-2, n-1) + \dots
\end{aligned}$$

$$\begin{aligned}
&\quad + C'(t-q/2+1, n-1) + C'(t-q/2, n-1)) \\
&= 1 + 2 (C'(t, n-1) + C'(t-1, n-1) + C'(t-2, n-1) + \dots \\
&\quad + C'(t-q/2+1, n-1) + C'(t-q/2, n-1) - C'(t, n-1)) \\
&= \left(1 + 2 \sum_{i=0}^{q/2-1} C'(t-i, n-1) \right) \\
&\quad + 2C'(t-q/2, n-1) - 2C'(t, n-1) \\
&= C'(t, n) + 2C'(t-q/2, n-1) - 2C'(t, n-1) \\
&\Rightarrow C'(t, n) \\
&= C'(t-1, n) + 2C'(t, n-1) - 2C'(t-q/2, n-1) \\
&= \binom{n+t}{t} + C'(t, n-1) - 2C'(t-q/2, n-1) \\
&\leq \binom{n+t}{t} + C'(t, n-1),
\end{aligned}$$

where we have used the same identity for the recurrence relation as in Proposition 4, and the inequality follows since $C'(t-q/2, n-1) \geq 0$. The recurrence relation

$$C'(t, n) \leq \binom{n+t}{t} + C'(t, n-1),$$

implies that

$$C(t, n) \leq C'(t, n) \leq \binom{n+t}{t} + \binom{n-1+t}{t} + \dots + \binom{t}{t} = \binom{n+t+1}{t+1},$$

where the last equality is the so-called *hockey-stick identity*. See for example [34, §5]. □

B.5 Proof of Proposition 6

PROOF. For $i \in [n]$, let D_i be the random variable denoting the distance $|x_i - y_i|$. Then through linearity of expectation, $\mathbb{E}(D) = \sum_{i=1}^n \mathbb{E}(D_i)$. Since $0 \leq x_i < q$, we have $0 \leq D_i \leq q/2$. D_i is 0 when $x_i = q/2$, and $D_i = q/2$, when $x_i = 0$. Any other value of D_i has two possible choices of x_i . For instance $D_i = 1$, if $x_i = q/2 + 1$ or $q/2 - 1$. Therefore, since x_i is uniformly distributed over \mathbb{Z}_q :

$$\begin{aligned}
\mathbb{E}(D_i) &= 0 \cdot \frac{1}{q} + \frac{q}{2} \cdot \frac{1}{q} + 1 \cdot \frac{2}{q} + 2 \cdot \frac{2}{q} + \dots + \left(\frac{q}{2} - 1\right) \cdot \frac{2}{q} \\
&= \frac{1}{2} + \frac{2}{q} \left(1 + 2 + \dots + \frac{q}{2} - 1\right) \\
&= \frac{1}{2} + \frac{2}{q} \frac{q}{2} \left(\frac{q}{2} - 1\right) \frac{1}{2} = \frac{q}{4}
\end{aligned}$$

Therefore, $\mathbb{E}(D) = qn/4$. □

B.6 Proof of Proposition 7

PROOF.

$$\begin{aligned}
\Pr(D \leq t) &= \Pr(D - qn/4 \leq t - qn/4) \\
&= 1 - \Pr(D - qn/4 > t - qn/4) \\
&\geq 1 - \Pr(D - qn/4 \geq t - qn/4) \\
&\geq 1 - \exp\left(-\frac{2(t - qn/4)^2}{\sum_{i=1}^n (q/2 - 0)^2}\right),
\end{aligned}$$

where the last inequality follows from Hoeffding's inequality for $t > qn/4$, and the expected value and range of D from Proposition 6.

By plugging in $t = \gamma n > qn/4$, we see that $\gamma > q/4$. Assuming γ to be such, we have

$$\begin{aligned} \Pr(D \leq t) &\geq 1 - \exp\left(-\frac{2(\gamma n - qn/4)^2}{\sum_{i=1}^n (q/2)^2}\right) \\ &= 1 - \exp\left(-\frac{2(4\gamma - q)^2 n^2}{q^2 n} \frac{4}{16}\right) \\ &= 1 - \exp\left(-\left(\frac{4\gamma}{q} - 1\right)^2 \frac{n}{2}\right). \end{aligned}$$

Now if

$$\left(\frac{4\gamma}{q} - 1\right)^2 \geq \frac{2}{n}, \quad (22)$$

Then

$$\Pr(D \leq t) \geq 1 - \exp(-1) = 1 - 1/e,$$

and we are done. The solution to Eq. (22) gives a bound on γ as stated in the statement of the proposition. \square

B.7 Proof of Proposition 8

PROOF. First note that

$$\binom{n+t}{t} = \frac{(n+t)!}{t!n!} = \frac{n+t}{n} \frac{(n-1+t)!}{t!(n-1)!} \geq \frac{(n-1+t)!}{t!(n-1)!} = \binom{n-1+t}{t}$$

Thus,

$$\begin{aligned} |B_1(\mathbf{x}, q, t)| &\geq \binom{n-1+t}{t} - \binom{n-1+t-q}{t-q} \\ &= \binom{n-1+t}{t} \binom{n-1+t-1}{t-1} \dots \\ &\quad \left(\frac{n-1+t-q+1}{t-q+1}\right) \binom{n-1+t-q}{t-q} - \binom{n-1+t-q}{t-q} \\ &\geq \left(\frac{n-1+t}{t}\right)^q \binom{n-1+t-q}{t-q} - \binom{n-1+t-q}{t-q} \quad (23) \\ &= \left(\left(\frac{n-1+t}{t}\right)^q - 1\right) \binom{n-1+t-q}{t-q} \end{aligned}$$

where we have used the fact that for $t > q-1$, and all $0 \leq i \leq q-1$ we have

$$\left(\frac{n-1+t-i}{t-i}\right) \geq \left(\frac{n-1+t}{t}\right).$$

Note that $t > q-1$ is true in our case. Now, assume $q \geq 4$ and that $(n-1+t)/t > 1.4$ which implies that $t < 2.5(n-1)$. Then we can apply Proposition 15 from Appendix A in the previous result, and get

$$|B_1(\mathbf{x}, q, t)| \geq \left(\frac{n-1+t}{t}\right)^{q-1} \binom{n-1+t-q}{t-q}$$

\square

B.8 Proof of Proposition 9

PROOF. From Eq. 23 in the proof of Proposition 8 we have

$$|B_1(\mathbf{x}, q, t)| \geq \left(\frac{n-1+t}{t}\right)^q \binom{n-1+t-q}{t-q} - \binom{n-1+t-q}{t-q}$$

which holds for $t > q-1$, which is true by assumption. Now,

$$\left(\frac{n-1+t}{t}\right)^q = \left(1 + \frac{n-1}{t}\right)^q$$

$$\begin{aligned} &> \left(1 + \frac{n-1}{qn}\right)^q, \\ &\geq \left(1 + \frac{0.9}{q}\right)^q, \end{aligned}$$

where we have used the fact that $t \leq (q-1)n < qn$ and $n \geq 10$, which is most likely to be the case. From Proposition 16 in Appendix A, $(1 + 0.9/q)^q$ is an increasing function of q , with $q \geq 2$. Therefore, since $q \geq 2$, we have for all $q \geq 2$:

$$\left(1 + \frac{0.9}{q}\right)^q \geq \left(1 + \frac{0.9}{2}\right)^2 = 2.1025 > 2.$$

Thus,

$$|B_1(\mathbf{x}, q, t)| > 2 \binom{n-1+t-q}{t-q} - \binom{n-1+t-q}{t-q} = \binom{n-1+t-q}{t-q}$$

Now using the fact that $\binom{a}{b} \geq \left(\frac{a}{b}\right)^b$, we get:

$$\begin{aligned} |B_1(\mathbf{x}, q, t)| &> \binom{n-1+t-q}{t-q} \geq \left(\frac{n-1+t-q}{t-q}\right)^{t-q} \\ &= \left(1 + \frac{n-1}{t-q}\right)^{t-q} \end{aligned}$$

\square

B.9 Proof of Proposition 10

PROOF. Take the i th element. If $x_i > y_i$, then the left hand side is $x_i + 0 = x_i$. And the right hand side is $y_i + x_i - y_i = x_i$. On the other hand if $x_i \leq y_i$, then the left hand side is $x_i + y_i - x_i = y_i$. And the right hand side is $y_i + 0 = y_i$.

For the second part, consider the i th summand in computing the ℓ_1 -distance. If $x_i > y_i$ then the i th summand of $\|\mathbf{x} - \mathbf{y}\|_1$ is $|x_i - y_i| = x_i - y_i$. The i th summands in $\|\mathbf{y} \dot{-} \mathbf{x}\|_1$ and $\|\mathbf{x} \dot{-} \mathbf{y}\|_1$ are 0 and $x_i - y_i$, respectively. Next assume $x_i \leq y_i$. Then the i th summand of $\|\mathbf{x} - \mathbf{y}\|_1$ is $|x_i - y_i| = y_i - x_i$. The i th summands in $\|\mathbf{y} \dot{-} \mathbf{x}\|_1$ and $\|\mathbf{x} \dot{-} \mathbf{y}\|_1$ in this case are $y_i - x_i$ and 0, respectively. \square

B.10 Proof of Proposition 11

PROOF. First note that $\deg(\sigma_{\mathbf{x}}) \leq \sum_{i=1}^n x_i = \|\mathbf{x}\|_1$. Furthermore, the coefficient of $z^{\|\mathbf{x}\|_1}$ is given by [21, §8.6, p. 244]:

$$(-1)^{\|\mathbf{x}\|_1} \prod_{i=1}^n a_i^{x_i}.$$

Since a_i are non-zero elements of \mathbb{F} , it follows that the coefficient of $z^{\|\mathbf{x}\|_1}$ is non-zero. \square

B.11 Proof of Theorem 1

PROOF. From the definition of the σ -polynomials in Eq. (8), together with Proposition 10, we have:

$$\begin{aligned} \sigma_{\mathbf{x}}(z) \sigma_{\mathbf{y} \dot{-} \mathbf{x}}(z) &= \left(\prod_{i=1}^n (1 - a_i z)^{x_i}\right) \left(\prod_{i=1}^n (1 - a_i z)^{y_i \dot{-} x_i}\right) \\ &= \prod_{i=1}^n (1 - a_i z)^{x_i + (y_i \dot{-} x_i)} \\ &= \prod_{i=1}^n (1 - a_i z)^{y_i + (x_i \dot{-} y_i)} \end{aligned}$$

$$\begin{aligned}
&= \left(\prod_{i=1}^n (1 - a_i z)^{y_i} \right) \left(\prod_{i=1}^n (1 - a_i z)^{x_i - y_i} \right) \\
&= \sigma_y(z) \sigma_{x-y}(z)
\end{aligned}$$

For the second part of the theorem, first note that $1 - a_i z$ does not divide $1 - a_j z$ for $i \neq j$. To see this, divide $1 - a_j z$ by $1 - a_i z$. We get the remainder $1 - a_i^{-1} a_j$. For this to be 0, we should have $a_i = a_j$, which is not possible as all the a_i 's are distinct. Let γ be any common factor of σ_{y-x} . Since Eq. (8) is the factorization of σ_{y-x} into irreducible monic polynomials, γ must have the factor $1 - a_i z$ for some $i \in [n]$. But for this to be in σ_{y-x} we must have that $y_i - x_i > 0$, which implies that $y_i > x_i$. Therefore, $x_i - y_i = 0$, and hence the term $1 - a_i z$ is absent in the product form of σ_{x-y} . Since $1 - a_i z$ does not divide any other term in σ_{x-y} as established above, we have that γ cannot be a common factor of σ_{y-x} and σ_{x-y} . \square

B.12 Proof of Proposition 12

PROOF. First note that $\sigma_x(0) = 1$, i.e., the constant term of σ_x is 1. Let $m = \deg(\sigma_x)$. This polynomial can be written as:

$$\sigma_x(z) = A_m z^m + \dots + A_{t+1} z^{t+1} + \dots + A_1 z + 1,$$

where $A_i \in \mathbb{F}$. Now, all the divisors of z^{t+1} are z^i for $0 \leq i \leq t + 1$. Pick any z^i with $i > 0$. Then dividing σ_x by z^i leaves the remainder $A_{i-1} z^{i-1} + \dots + A_1 z + 1$, where $A_0 = 1$. This is non-zero regardless of the A_i 's. Therefore, the gcd is 1. \square

B.13 Proof of Theorem 4

PROOF. Since the σ -polynomial only needs to be stored modulo z^{t+1} , we can use the following algorithm:

```

1 Set  $s \leftarrow 1$ 
2 for  $i = 1$  to  $n$  do
3    $r \leftarrow (1 - a_i z)^{x_i} \pmod{z^{t+1}}$ 
4    $s \leftarrow sr \pmod{z^{t+1}}$ 
5 return  $s$ 

```

Step 3 multiplies $1 - a_i z$ with itself up to q times. Thus, this can be done in up to q steps. Reduction modulo z^{t+1} can be done via the division algorithm. Since this involves a polynomial of degree up to q and another with degree $t + 1$, this can be done in time $\mathcal{O}(qt)$ [20, §17.1]. Step 4 involves multiplying two polynomials of degrees less than or equal to t . This can be done in $\mathcal{O}(t^2)$ time [20, §17.1]. Finally, reduction modulo z^{t+1} , as above, can then be done in $\mathcal{O}(t^2)$, as sr is of degree at most $2t$. Thus, h can be computed in time $\mathcal{O}(n(qt + t^2 + t^2)) = \mathcal{O}(nt^2)$.

For the eval_h function, the first step is to find the inverse of σ_x modulo z^{t+1} . This can be done using the EEA. The EEA takes time $\mathcal{O}(t^2)$ [20, §17.3] as the polynomials are of degrees t and $t + 1$, respectively. This is followed by multiplication of degree t polynomials σ_x^{-1} and σ_y , and then by reduction modulo z^{t+1} , both taking $\mathcal{O}(t^2)$ time as discussed above. Finally, running the EEA algorithm on $\tilde{\sigma}_{x,y}$ and z^{t+1} for at most $t_+ < t$ steps again takes time $\mathcal{O}(t^2)$. Thus, eval_h can be computed in $\mathcal{O}(t^2)$ overall time. \square

B.14 Proof of Theorem 5

PROOF. consider the polynomial in Eq. (8) for the vector $\mathbf{x} = (x_1, \dots, x_n)$:

$$\prod_{i=1}^n (1 - a_i z)^{x_i}.$$

Let us relabel the a_i 's so that multiple occurrences of a_i 's are labelled by different labels, as shown in Figure 6.

Here we have overloaded notation to also use \mathbf{a} to denote the tuple containing multiple occurrences of elements of the vector \mathbf{a} . Then the polynomial can be rewritten in terms of \mathbf{a}' as:

$$\prod_{i=1}^m (1 - a'_i z),$$

where $m = \sum_{i=1}^n x_i$. Then the j th coefficient of this polynomial is given by the elementary symmetric polynomial [21, §8]:

$$\begin{aligned}
e_j(a_1, a_2, \dots, a_n) &= e_j(a'_1, a'_2, \dots, a'_n) \\
&= (-1)^j \sum_{1 \leq i_1 < i_2 < \dots < i_j \leq n} a'_{i_1} a'_{i_2} \dots a'_{i_j}
\end{aligned}$$

Let us call the sum on the right $S(j, n)$. We would like to get an expression of $S(j, n)$ in terms of the original vector \mathbf{a} . Now $S(j, n)$ is the sum in the j th elementary symmetric polynomial in terms of the elements of the tuple \mathbf{a} , $S(j, n - 1)$ is the sum in the j th elementary symmetric polynomial in terms of the elements of the tuple \mathbf{a} without a_1 , and so on. Under this notation $S(0, i) = 1$ for all integers $i \geq 0$, and $S(i, i - 1) = 0$ for all integers $i \geq 1$.

Each summand in $S(j, n)$ is a product of j elements of the tuple \mathbf{a} . To calculate $S(j, n)$, first consider a_1 . There are a total of x_1 occurrences of a_1 in the tuple \mathbf{a} . Taking j at a time, we therefore have a total of $\binom{x_1}{j}$ occurrences of a_1^j in $S(j, n)$. Next we consider a_1^{j-1} with the last coefficient being any of the other coefficients. We can have $\binom{x_1}{j-1}$ possible arrangements that yield a_1^{j-1} . For each of these arrangements we need to determine the last coefficient in the j -term product. We are left with $n - 1$ coefficients: a_2, \dots, a_n and we are taking them one at a time. Thus, we are computing the quantity $S(1, n - 1)$. Likewise for a_1^{j-2} we need to consider the number of possible arrangements that yield a_1^{j-2} which are $\binom{x_1}{j-2}$ and the number of possible ways in which the last two spots can be filled by the remaining $n - 1$ elements, which is $S(2, n - 1)$. Continuing on this way, once we reach a_1^0 , we see that all j spots in the product are taken by the rest of the elements in \mathbf{a} . Thus, we are computing $S(j, n - 1)$. Collecting these counts, we get

$$\begin{aligned}
S(j, n) &= \binom{x_1}{j} a_1^j + \binom{x_1}{j-1} a_1^{j-1} S(1, n-1) + \binom{x_1}{j-2} a_1^{j-2} S(2, n-1) \\
&\quad + \dots + \binom{x_1}{1} a_1 S(j-1, n-1) + \binom{x_1}{0} a_1^0 S(j, n-1) \\
&= \sum_{i=0}^j \binom{x_1}{i} a_1^i S(i, n-1).
\end{aligned}$$

From the above equation we see that:

$$S(j, n - 1) = \sum_{i=0}^j \binom{x_2}{i} a_2^i S(i, n - 2),$$

$$\begin{array}{rcl}
\mathbf{a} & = & \overbrace{(a_1, \dots, a_1)}^{x_1}, \overbrace{(a_2, \dots, a_2)}^{x_2}, \dots, \overbrace{(a_n, \dots, a_n)}^{x_n} \\
& & \downarrow \qquad \qquad \downarrow \qquad \downarrow \qquad \qquad \downarrow \qquad \qquad \downarrow \qquad \qquad \downarrow \\
\mathbf{a}' & = & (a'_1, \dots, a'_{x_1}, a'_{x_1+1}, \dots, a'_{x_1+x_2}, \dots, a'_{m-x_n+1}, \dots, a'_m)
\end{array}$$

Figure 6: Replacing the labels in the tuples \mathbf{a} with unique labels.

and so on. Thus,

$$A_j = e_j(a_1, \dots, a_n) = (-1)^j S(j, n).$$

□

C FURTHER EXPERIMENTAL RESULTS

Figure 7 shows the change in image quality as we continue to alter the original image using the PGD attack, brightness and contrast.

We note that through brightness and contrast adjustments, at certain ϵ values even though the quality of the image is both visibly and through the pixel change ratio metric, quite bad, the LPIPS is rather small. For instance, at $\epsilon = 0.5$, the LPIPS is only 0.1271 for the brightness attack, even though the pixel change ration is 50.18%. Thus a smaller value of LPIPS may suffice for the brightness and contrast attacks.

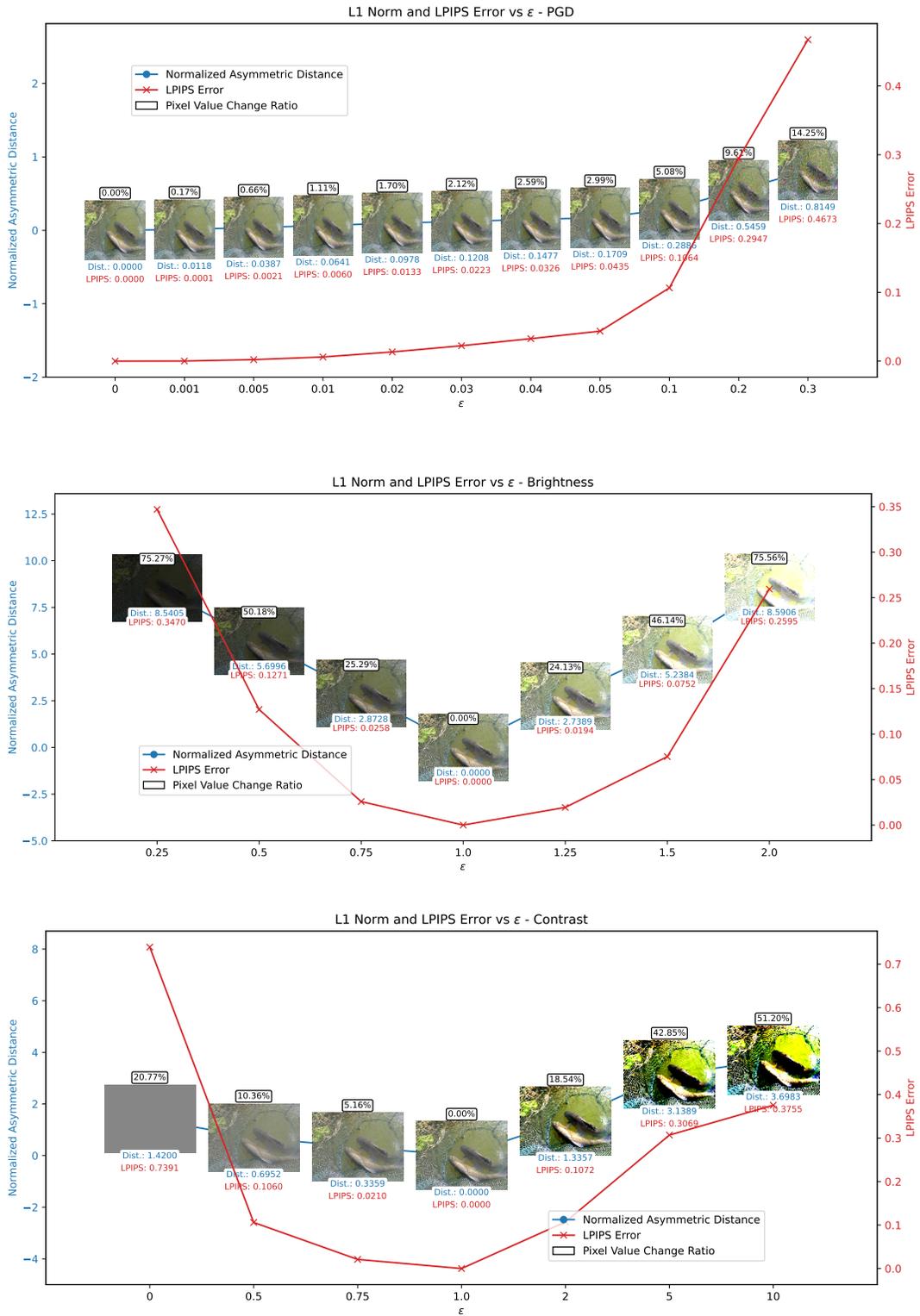


Figure 7: The impact of increasing alterations to the image through the PGD attack, brightness, and contrast, on the metrics LPIPS, pixel change ratio and NAD.