Approximate Thumbnail Preserving Encryption

Byron Marohn[1], Charles V. Wright[2], Wu-chi Feng[2], Mike Rosulek[3], and Rakesh B. Bobba[3]

[1] Intel Corporation, Hillsboro, OR*
[2] Portland State University, Portland, OR†
[3] Oregon State University, Corvallis, OR‡

October 17, 2017

Abstract

Thumbnail preserving encryption (TPE) was suggested by Wright et al. [Information Hiding & Multimedia Security Workshop 2015] as a way to balance privacy and usability for online image sharing. The idea is to encrypt a plaintext image into a ciphertext image that has roughly the same thumbnail as well as retaining the original image format. At the same time, TPE allows users to take advantage of much of the functionality of online photo management tools, while still providing some level of privacy against the service provider.

In this work we present three new *approximate TPE* encryption schemes. In our schemes, ciphertexts and plaintexts have perceptually similar, but not identical, thumbnails. Our constructions are the first TPE schemes designed to work well with JPEG compression. In addition, we show that they also have provable security guarantees that characterize precisely what information about the plaintext is leaked by the ciphertext image.

We empirically evaluate our schemes according to the similarity of plaintext & ciphertext thumbnails, increase in file size under JPEG compression, preservation of perceptual image hashes, among other aspects. We also show how approximate TPE can be an effective tool to thwart inference attacks by machine-learning image classifiers, which have shown to be effective against other image obfuscation techniques.

1 Introduction

With the advent of cheap cell phone cameras and fast mobile networks, electronic devices are capturing more and more of our daily lives in photos and video. A vast number of digital images and videos are recorded every day, and many of these are also either transmitted or stored via some third party provider in the cloud. For example, Apple, Google, Dropbox, Microsoft, and others all offer services that automatically sync photos from mobile devices. Users of social networks like Facebook share more than one billion new photos each week [1].

Individually and as a society, we benefit from this ubiquitous, centralized data collection. We can access our files from anywhere, even if we lose or damage our mobile device. We keep in touch

*byron.marohn@intel.com
†cvwright@cs.pdx.edu, wuchi@cecs.pdx.edu
‡rosulekm@engr.orst.edu, rakesh.bobba@oregonstate.edu

with friends in distant cities, easily sharing large files asynchronously and without the need for peer-to-peer transfer. We get the benefit of security updates, virus scanning, and spam filtering, without having to become system administrators.

At the same time, volume of media being recorded and stored today presents a new and serious threat to privacy. Data breaches have become far too common; attackers have recently gained access to thousands of accounts on the most popular services [6, 16]. In other cases, the cloud services themselves have exploited user data for their own benefit [18] or to satisfy a secret request from a third party [3].

The obvious solution to these problems is to encrypt data from "end-to-end" on the user's device, so that the cloud service sees only ciphertext. Recent work in this space includes Cryptagram [19] and P3 [15]. Cryptagram encrypted photos are orders of magnitude larger than the original JPEG. P3 splits its encrypted images into two pieces, a public part and a private part; it requires a second cloud service to act as a semi-trusted third party to hold the private part of each image. P3 was also recently found to provide very limited protection against inference attacks. McPherson et al [11] trained a convolutional neural network to recognize faces in the public parts of P3 encrypted images. With P3's most conservative configuration, the neural network achieved 83% accuracy on the AT&T faces data set [17].

Moreover, in order to have success with users, any new protection mechanism must not sacrifice usability or lose the killer features that made today's large cloud-based services popular in the first place. Another recently proposed approach for this problem is thumbnail preserving encryption [21], or TPE. The idea is that an encrypted image should reveal only a reduced resolution version, i.e., a *thumbnail*, of the original image, and nothing more. Intuitively, the TPE scheme should preserve coarse features of the plaintext image that are larger than a thumbnail block, and hide all fine details smaller than one block. More concretely, in a thumbnail of block size B, each $B \times B$ of the original image produces a single pixel in the thumbnail. The goal of TPE is that each $B \times B$ block of the TPE ciphertext image reveals no more than the color/value of that corresponding thumbnail pixel. There are many algorithms for scaling an image down to create a thumbnail, but the simplest option for purposes of TPE is to let each pixel in the thumbnail be the average of the $B \times B$ pixels in the corresponding block of the original high-resolution image.

A key strength of the thumbnail preserving approach is that it maintains most of the convenience of working with plain, unencrypted media. Existing, unmodified cloud services retain the ability to perform a limited set of useful operations using only the encrypted media. For example, a photo management service like Google Photos can still generate accurate thumbnails using only encrypted photos, so a user can manage her collection online without needing to download the full-resolution version of every image (e.g., to decrypt on the client side). Misuse-detection systems based on perceptual hashing can detect known bad (e.g., illegal or copyrighted) images without learning too much about legitimate images.

The only existing approach for TPE [21] uses a pseudorandom shuffle of the pixels in each thumbnail block of the image. This technique preserves the original thumbnail exactly but unfortunately also has a number of drawbacks that limit its utility for protecting real photos. First, it reveals much more information than is necessary—not only the average value in each block, but also the full (un-ordered) list of plaintext values in the block. This leakage might be exploited in the future to reassemble some or all of the original image. Second, it does not work very well with JPEG compression. With larger thumbnail block sizes, it results in washed out colors in images after decryption. This occurs because JPEG applies its lossy compression on the shuffled image,

which tends to contain lots of high-frequency information that JPEG was specifically designed not to preserve.

Contributions In this paper, we present three new techniques for *approximate* thumbnail preserving encryption. Here we aim to preserve only an approximation of the average color in each block. We give up the ability to preserve the exact thumbnail, but in return we gain much stronger security, much better perceptual quality in the decrypted images, and much smaller encrypted images. The key to making our techniques work on lossy compressed JPEG images is that we apply our encryption mid-way through the JPEG compression process, after the lossy chroma subsampling and quantization steps, and before the lossless run-length and entropy coding. (For more background on JPEG compression, see "A Note on Image Formats" in Section 4.)

We give three constructions of approximate TPE with a different mix of features. Our schemes are *provably secure*, in the sense that we characterize succinctly and exactly what information about the plaintexts is leaked by ciphertext images.

An important open question for all TPE schemes is, *What thumbnail block size is necessary to achieve meaningful security?* We take a step towards answering this question by evaluating our constructions against the same deep neural network adversary that recently recognized faces in P3 encrypted images [11]. Our experimental results show that, for thumbnail blocks a little smaller than a face, we can reduce the neural network classifier's accuracy to not much better than random guessing. For even larger blocks, the neural network performs no better than random.

Outline The remainder of the paper is organized as follows. We review related work in Section 2. We define our notions of security formally in Section 3. In Section 4 we describe our new approximate thumbnail preserving constructions. In Section 5 we evaluate our schemes in terms of file size, image quality, preservation of perceptual hashes, and security against inference attacks with the deep neural network adversary from [11]. We conclude with some thoughts on directions for future work in Section 6.

2 Related Work

Multimedia encryption: While many encryption schemes for images (and other media) have been proposed over the years, format-preserving encryption schemes where the ciphertext is also an image are the most relevant to our work. Cryptagram [19] is one such scheme proposed for private image sharing over online social networks (OSNs). Cryptagram allows users to users to encrypt their images into ciphertexts that i) have the same format as the plaintext image, ii) do not reveal any information to the those without the decryption key, and iii) are able to tolerate transformations (e.g., compression etc) done by the OSN. However, Cryptagram significantly increases the image size during encryption, and its ciphertext images are opaque and cannot be distinguished without first decrypting them. While the latter may not be an issue for sharing images on OSNs, it is not suitable for applications such as cloud-based image storage where users may want to browse through their albums without having to download and decrypt every image. P3 [15] is another scheme proposed for privacy-preserving image sharing. P3 partitions an image into public and private parts and encrypts the much smaller private part and makes the public part of the image available on OSN as is. The public part has been shown to tolerate server side transformations such as compression, but it has been shown that P3 is vulnerable to face recognition [11] attacks. In

contrast, the proposed TPE techniques are format-preserving, allow users to browse and organize ciphertext images without the need for decryption, and resist face recognition attacks.

Inference attacks: Approaches like blurring, pixelation or redaction have long been used for protecting privacy in images and text (*e.g.,* [24, 10]). However, while techniques like blurring and pixelation provide a more aesthetic approach to preserving privacy compared to redaction or replacement, it has been shown that such techniques are not very effective against both humans and automated recognition methods [24, 10, 13, 8, 11]. Specifically, it has been shown that it may be possible to deanonymize faces that are blurred or pixelated using machine learning techniques when pictures of the person are otherwise available. Similarly, it has been shown that pixelated or blurred text can be recovered using statistical tools like Hidden Markov Models (HMMs) [8]. Further, it has been shown that even if redaction techniques like replacing faces with white or black space are used, it may be possible to deanonymize when tagged pictures of the person are available using person recognition techniques [14]. Moreover, obfuscation techniques such as blurring, pixelation and redaction are not not typically reversible and thus are not suitable for applications such as image storage outsourcing where users want to store their albums in the cloud without having to worry about loss of privacy. In this work, we show that while the proposed schemes reveal some information about the image they are still resistant to machine learning based face recognition techniques from [11].

Property-preserving encryption: Property-preserving encryption [2, 4, 5] is an emerging cryptographic encryption paradigm where a certain property of the plaintext is preserved by the ciphertext. Format-preserving encryption, at a high-level, tends to constrain the ciphertexts to a specified format [2]. Similarly, order-preserving encryption preserves the numerical ordering of plaintexts among the ciphertexts [4, 5]. The TPE schemes proposed here can be considered a variant of property-preserving encryption schemes.

3 Security of Approximate TPE

The intuitive goal of approximate TPE has two parts. The first is a functionality goal: that TPE ciphertexts preserve the thumbnail of the plaintext images. That is, a plaintext image and its TPE-encrypted ciphertext have "similar" thumbnails. The second goal is a security goal, that TPE ciphertexts *leak no more* about the plaintext than its thumbnail (and original dimensions).

In this work, we discuss approximate TPE schemes that, in fact, leak slightly more information about the plaintext than just the thumbnail. However, our goal is to characterize exactly what information is leaked and provide schemes that *provably* leak no more than the stated information.

Threat model. Our security model considers a passive eavesdropper who is able to observe TPE-encrypted ciphertexts of chosen plaintext images. In this work, we do not consider active chosen-ciphertext attacks in which the adversary modifies ciphertexts or generates malicious ciphertexts for the honest parties to decrypt.

Our encryption schemes are nonce-based. That is, the encryption algorithm takes in a key, a plaintext image, and a *nonce* value. In practice, we imagine the nonce to be derived from public image metadata (e.g., filename, timestamp, geolocation, or a combination thereof). Importantly, our security definition requires that *nonces are globally unique.* No two images are to be encrypted with the same key+nonce combination. We believe this is a reasonable requirement for common situations, where users store their photos online without saving different versions of a photo using

identical metadata (e.g., in most mobile phones the default behavior when manipulating an image is to save the manipulated image as a copy with different filename and access date).

Formal security framework. We can formalize the above requirements by considering a chosen-plaintext attack game played against a distinguisher. The distinguisher can choose plaintext images and nonces but must never repeat a nonce. In one version of the game, the distinguisher receives TPE-encrypted images. In the other version of the game, the distinguisher receives simulated ciphertexts generated by some *simulator*. The simulator is not given the entire plaintext image (as the encryption algorithm is), but only some limited information like its thumbnail. If no distinguisher can distinguish between the two games, then the ciphertexts leak no more about the plaintexts than the information given to the simulator.

Definition 1. *Let* Enc *be an encryption algorithm taking a key, nonce, and plaintext as input. Let* $\mathcal{A}$ *be an adversary that makes calls to some oracle. We say that* $\mathcal{A}$ *is nonce-respecting if it never makes two oracle calls with the same first argument.*

Let f *be some function. We say that* Enc ***leaks no more than*** f *if there exists a simulator algorithm* $\mathcal{S}$ *such that for all nonce-respecting adversaries* $\mathcal{A}$*, the following is negligible in* λ*:*

$$\left| \Pr_{K \leftarrow \{0,1\}^\lambda} \left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot)}(\lambda) = 1 \right] - \Pr \left[\mathcal{A}^{\mathcal{S}(\cdot, f(\cdot))}(\lambda) = 1 \right] \right|$$

In the TPE schemes that follow, we show security by characterizing exactly what information about the plaintext images is leaked.

4 Approximate TPE Constructions

In this section we present two new constructions for approximate thumbnail preserving encryption of images that provide easy-to-understand provable security guarantees. A summary of their properties is given in Figure 14.

The first of these, described in Section 4.1, uses a simple dynamic range preserving encryption (DRPE) that provably reveals only the minimum and maximum pixel value in each thumbnail block. Despite not preserving the actual image thumbnail *per se*, ciphertext images and plaintext images still have perceptually similar thumbnails. The scheme does occasionally result in incorrect decryption of a thumbnail block (i.e., decrypted block does not equal the original plaintext block), but such an event happens with an exponentially small probability that is easy to characterize.

Our second construction (Sec 4.2) uses a novel application of least-significant bit embedding techniques from steganography. It provably reveals only (a bound on) the maximum pixel value in that block, along with the average of the plaintext in each block (i.e., the value of the thumbnail pixel corresponding to that block). The scheme provides a tunable trade off between thumbnail faithfulness and correctness of decryption: ciphertext images can be tuned to preserve the original plaintext thumbnail with arbitrary precision, at the expense of making the decrypted image only a noisy approximation of the original. Nevertheless, our experiments in Section 5.2 show that in practice the perceptual quality of both thumbnails and decrypted images is typically quite good.

Our third construction (Sec 4.3) combines elements of the first two to achieve a better balance between the accuracy of the thumbnail and the quality of the decrypted image. For each thumbnail block in the ciphertext image, this hybrid construction reveals both the dynamic range and the

Figure 1: Example image, original 461x364 pixels, ©Byron Marohn

average of each block in the plaintext. It is intended to produce a more accurate thumbnail than the DRPE scheme and a higher quality decrypted image than the LSB embedding scheme.

Examples Figure 1 shows an example photo. Figures 2–4 give flattened "mosaic" versions of the image, with 16x16, 32x32, and 64x64 pixel blocks, respectively. Figures 5–7, Figures 8–10, and Figures 11–13 show the same photo encrypted under the DRPE, LSB, and hybrid DRPE+LSB schemes. This image is particularly challenging for our scheme because of its wide range of colors and brightness levels. With the dynamic range preserving scheme, Figure 5 appears merely pixellated, and Figure 6 looks noisier when viewed at close to its native resolution. In Figure 7 with 64x64-pixel blocks, many of the thumbnail blocks have an extremely wide dynamic range in either the luminance channel, the chrominance, or both. Consequently the encrypted image appears very noisy, and it may be necessary for the reader to zoom out several times before the human eye registers that the thumbnail is correctly preserved. The LSB-encrypted versions of the same images are overall less perceptually chaotic, and their color palette is overall more muted, similar to the ideal thumbnail in the mosaic images. The hybrid images reduce the noise substantially in regions with small dynamic range, for example the area of dark black in the lower right corner.

A Note on Image Formats In this section, we describe our TPE constructions generically as operating on an image that consists of 2-dimensional blocks of *intensities*. For images in a lossless ("raw") format such as PNG, TIFF, or DNG, these techniques could be applied directly to the pixels in each color channel.

In the real world, most photos are captured and stored using JPEG compression. JPEG is a DCT-based compression format for images. Each RGB image is converted into the YUV color space and then divided into 16x16 pixel squares called *macroblocks*. The Y values are further split into 4 8x8 pixel *blocks*; the U and V values are then sub-sampled into 2 8x8 blocks, one for U and the other for V. Each block is then discrete cosine transformed. The upper left corner of

Figure 2: Mosaic, 16x16 blocks

Figure 3: Mosaic, 32x32 blocks

Figure 4: Mosaic, 64x64 blocks

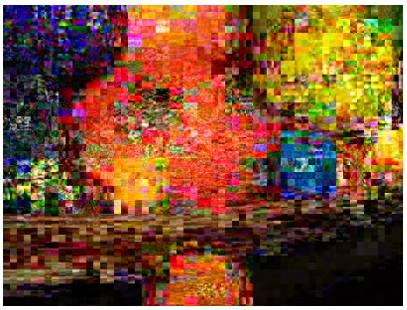

Figure 5: Encrypted with DRPE, 16x16 blocks

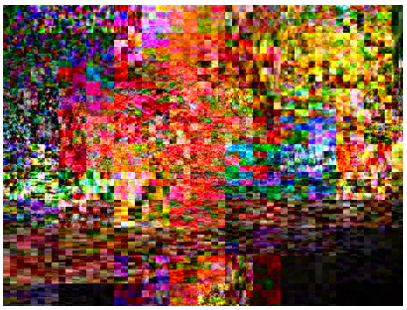

Figure 6: Encrypted with DRPE, 32x32 blocks

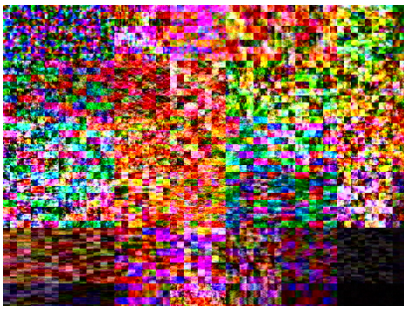

Figure 7: Encrypted with DRPE, 64x64 blocks

Figure 8: Encrypted with TPE-LSB, 3 bits, 16x16 blocks

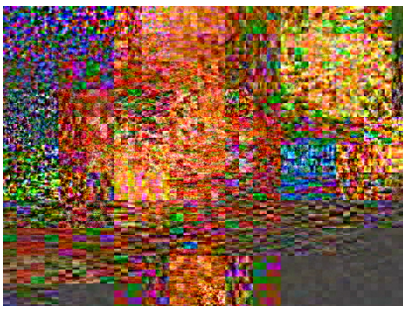

Figure 9: Encrypted with TPE-LSB, 3 bits, 32x32 blocks

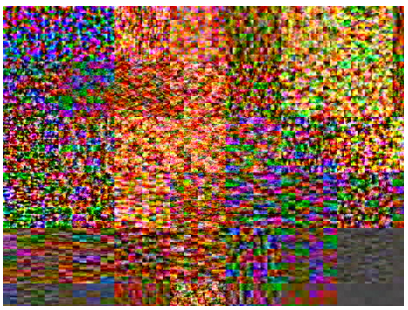

Figure 10: Encrypted with TPE-LSB, 3 bits, 64x64 blocks

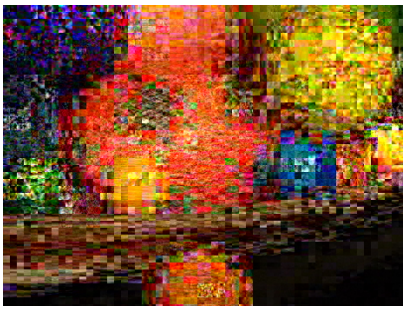

Figure 11: Encrypted with hybrid TPE, 3 bits, 16x16 blocks

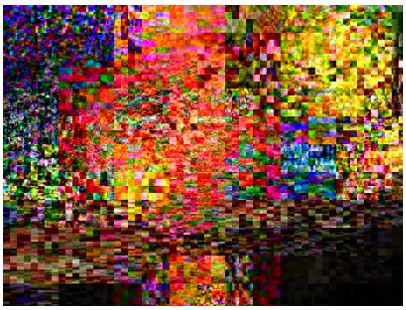

Figure 12: Encrypted with hybrid TPE, 3 bits, 32x32 blocks

Figure 13: Encrypted with hybrid TPE, 3 bits, 64x64 blocks

each block is called the *DC* value (i.e., the average value of the entire block) while the remaining cells hold *AC* coefficients. Once transform coded, each of the DC and AC values are quantized; coarser quantization leads to lower file sizes and lower quality, while finer quantization leads to higher file sizes and higher fidelity. Generally speaking, it is difficult to do any kind of encryption before quantization as the decoder is unable to guarantee that the bits needed for decryption will be available.

To encrypt a JPEG image, if the thumbnail block size is a multiple of 8x8, we can simply apply our thumbnail-preserving encryption to the DC coefficients as if they were pixels, and this is sufficient to preserve the thumbnail. Then we apply a less leaky construction to encrypt the AC coefficients, because we want to reveal as little as possible about the fine details in each block. We discuss the challenge of efficiently encrypting the AC coefficients in more detail in Section 4.4.

Assumptions Throughout this section, we assume that the user has a secret passphrase of sufficient complexity for use in deriving a cryptographic symmetric key, and that the user has already provided the passphrase out-of-band to any other users with whom they want to share encrypted photos. We also assume that each image carries with it some set of unique, public metadata, such as a filename, timestamp, or EXIF tags, that can be used as a unique nonce.

Common Elements Both of our constructions share some common framework elements. To encrypt an image with dimensions $H \times W$ with thumbnail blocks of $B \times B$ pixels, we proceed as follows.

- Let K be a high-entropy, long-term secret key (in practice, derived from a passphrase using a key derivation function).
- Let N be a nonce, unique to this image, derived from the image's public metadata.
- For each thumbnail block, at coordinates (x, y) in color channel c, derive $K^N_{x,y,c} := F(K, N\|x\|y\|c)$, where F is a pseudorandom function (PRF). The security of a PRF is that each such key is indistinguishable from an independent, uniformly chosen key. $K^N_{x,y,c}$ is a subkey that can be used to encrypt this specific thumbnail block of this specific image.

We therefore proceed to describe the operation of each encryption scheme on a single block. The use of a PRF to derive subkeys for each block ensures that each block receives an independently pseudorandom subkey. Since our security model assumes that nonces are never repeated, our subkey derivation ensures that each subkey is used only once globally. Hence we describe the encryption schemes at a block-level and need only show security under *one-time use* of a block-specific subkey.

Scheme	leakage per block	decryption
DRPE	approx. dynamic range	perfect w/h/p
LSB	approx. max value & average	noisy
Hybrid	approx. dynamic range & average	noisy

Figure 14: Summary of properties of our two schemes. "Leakage per block" means that the ciphertext leaks *no more than* the given information about each $B \times B$ thumbnail block. Approximate dynamic range (resp. max value) means that only a rounded version (to the nearest power of two) of the dynamic range (resp. max value) is leaked, not the exact value.

4.1 Dynamic Range Preserving TPE

In this section, we describe our first approximate thumbnail preserving encryption technique. It is not guaranteed to preserve an accurate thumbnail, but it does (approximately) preserve the minimum and maximum intensity in each thumbnail block of the image. Here, intensities may be raw pixel values as in uncompressed images, or DC coefficients in a JPEG image.

At the core of this construction is a very simple dynamic range preserving encryption (DRPE) technique, which we describe in greater detail below. To encrypt n plaintext elements, the DRPE requires n pseudorandom elements of key material like a one-time pad.

4.1.1 Dynamic Range Preserving Encryption (DRPE)

We define the dynamic range (DR) of a list of values as the closed interval bounded above and below by the maximum and the minimum values in the list, respectively. More concretely, if X is a vector of integers $X = (x_1, x_2, \ldots)$, then $DR(X) = [\min(X), \max(X)]$. Here we give a simple approach for constructing a dynamic range preserving encryption that preserves the dynamic range of the plaintext data within a factor of two.

Dynamic Range Expansion (DRX) A necessary preprocessing step of our encryption and decryption routines is to expand the original dynamic range of the input values so that its size is a power of two. Given a vector of n integers $A = (a_1, a_2, \ldots, a_n)$, we define the expanded dynamic range $DRX(A) = [l, u]$ as the smallest closed interval such that, for some non-negative integer b, we have:
$l \equiv 0 \mod 2^b$; $u - l \equiv (2^b - 1) \mod 2^b$; and $a_i \in [l, u], \forall i \in [1, n]$.

This dynamic range expansion has an intuitive interpretation. When all input values are positive (or all negative), then all of the inputs share a common prefix when written in binary. Let the longest common prefix consist of all but the b least-significant bits. When the input contains a mix of positive and negative values, then b gives a bound on their magnitude, namely $|a_i| < 2^{b-1}$ for all $i \in [1, n]$.

The DRX step is necessary for two reasons. First, it allows us to more reliably recover the dynamic range of the plaintexts given only the ciphertexts. Second, it allows us to encrypt values with a simple one-time pad.

Encryption Given a block-specific subkey $\tilde{K}$ and a vector of n plaintext integers $M = (m_1, m_2, \ldots, m_n)$, we encrypt as follows:

1. Compute the expanded dynamic range of the plaintexts $[l, u] = DRX(M)$, and let $b = \lceil \log_2(u - l) \rceil$. Intuitively, all integers in M have the same prefix (described by l) and differ only in their b least significant bits.

2. Use a pseudorandom generator to expand $\tilde{K}$ into nb pseudorandom bits $(k_1, \ldots, k_n)$, with each k_i exactly b bits long.

3. Encrypt the least significant b bits of each plaintext integer using a one-time pad. Hence, set

$$c_i = l + ((m_i - l) \oplus k_i)$$

The resulting ciphertext is $C = (c_1, \ldots, c_n)$. This basic approach for encrypting numbers in a fixed range was first described by Gutmann in 1997 [7].

Decryption Given the reliance on one-time pad, decryption is exactly the same as encryption. Given the vector of ciphertext integers and the subkey $\tilde{K}$, we compute the expanded dynamic range, compute the pseudorandom bits $(k_1, \ldots, k_n)$, and perform one-time pad on the b least significant bits of each plaintext integer.

Correctness It is possible that a ciphertext encrypted with this scheme may fail to decrypt correctly. Specifically, this happens when the encryption and decryption algorithm compute different expanded dynamic ranges. For example, suppose the plaintext integers have a common 5-bit prefix. By construction, the ciphertext integers also have the same common 5-bit prefix, but there it is also possible that all ciphertext integers actually have a common 6-bit prefix.

Fortunately, the probability of such an error drops off exponentially as the number of elements n increases. Note that the least significant b bits of ciphertext integers are distributed (pseudo)randomly. The ciphertext integers fall into a smaller expanded dynamic range than the plaintext integers if and only if all ciphertext integers agree in their bth least significant bit. Hence, the probability of decryption error is bounded by the probability that n specific random bits all agree, thus:

$$\Pr[\text{decryption error}] = \Pr[n \text{ coin tosses all agree}] = \left(\frac{1}{2}\right)^{n-1}$$

Hence, for thumbnail blocks of size $B \times B$, the probability of decryption error for a *specific* block is $(1/2)^{B^2-1}$. For uncompressed image formats, this is astronomically small; however, JPEG compressed images share a single DC value for all pixels in an 8×8 block, which increases this probability to $(1/2)^{(B/8)^2-1}$. In this work, the smallest block size we consider is 16×16, leading to error probability $(1/2)^3 = 1/8$, which is substantial. For small TPE block sizes, these errors can cause observable damage to the decrypted plaintext as shown in Section 5.2.

Leakage and Security Analysis The scheme leaks no more than the expanded dynamic range $[l, u]$ of the plaintext. Recall that it is sufficient to demonstrate security for a one-time use of the block-specific subkey.

In the scheme, the ciphertext integer c_i consists of all but the last b bits of l, followed by b bits $(m_i - l) \oplus k_i$.

The k_i values are derived from a pseudorandom generator (PRF) applied to the (assumed random) subkey $\tilde{K}$. The security of a PRG is that its output bits are indistinguishable from uniformly random bits. But when k_i is uniformly (independently) random, then $(m_i - l) \oplus k_i$ is uniformly distributed. Hence, c_i is indistinguishable from a string agreeing with l in all but the last b bits, followed by b uniformly random bits.

Such a probability distribution can be sampled knowing only l and b (equivalently, l and u). Since the ciphertext distribution is indistinguishable from a distribution that depends only on the expanded dynamic range, we can say that ciphertexts leak no more than the expanded dynamic range of the plaintexts.

Since the scheme is based on a one-time pad, this construction provides no integrity protection against ciphertext tampering attacks. If an encrypted bit is flipped in the ciphertext, then the error will silently be propagated in the decryption, so that the same bit will also be flipped in the plaintext. Indeed, providing such ciphertext authenticity would require ciphertext expansion, which we avoid in this work.

4.2 Approximate TPE with LSB Embedding

Our second approximate TPE scheme reveals no more than a bound on the maximum pixel intensity in each block and the average intensity in each block (i.e., the thumbnail value for that block). The resulting ciphertext can match the thumbnail of the plaintext image arbitrarily closely, but this comes at the cost of introducing increased noise in the image upon decryption.

Encryption Given a vector of n plaintext integers $M = (m_1, m_2, \ldots, m_n)$ and a block-specific subkey $\tilde{K}$, we encrypt as follows:

1. We begin by applying a simple form of dynamic range expansion. Here we find the smallest integer b such that $m_i \in [-2^b, 2^b - 1]$ for all $i \in [1, n]$.

2. Use a pseudorandom generator to expand $\tilde{K}$ into nb pseudorandom bits $(k_1, \ldots, k_n)$, with each k_i exactly b bits long.

3. We apply a standard one-time-pad-like encryption.

$$c_i = m_i \oplus k_i$$

   Note that so far this hides everything about the plaintexts other than their largest magnitude b. As such, the ciphertext currently has no relationship to the plaintext thumbnail.

4. Next, we reverse the order of the b bits in each element c_i of the block; i.e., $c'_i = \mathsf{reverse}(c_i)$ for each i.

5. We compute the sum of plaintext integers $\sum_i m_i$ and of ciphertext integers $\sum_i c'_i$. Our goal is to modify the c'_i values to push their sum closer to the "target" sum $\sum_i m_i$.

6. Borrowing a trick from steganographic LSB embedding, we flip bits in the c'_i values. We start with the *most significant* bits of the c'_i values. Flipping these bits makes the greatest change to the sum $\sum_i c'_i$. But because the c'_i values represent *reversed* plaintext values, flipping their most-significant bits flips the least-significant bits of the corresponding plaintext, doing the least damage to the image payload.

   The encryption process is randomized. A c'_i "victim" is chosen uniformly from the available pixels in the block; if the victim has a `1` at the current bit position and the ciphertext's magnitude is higher than the plaintext, then it is flipped to a `0` and vice-versa. When the difference between $\sum_i c'_i$ and the target $\sum_i m_i$ is smaller than 2^{b-2}, then flipping the bth bit actually makes $\sum_i c'_i$ *farther* from the target (it changes the sum by 2^{b-1}. At that point, the process proceeds to the next-most-significant bit. The process ends after some limit (e.g., at most 3 bits) set by the user.

The resulting $(c'_1, \ldots, c'_n)$ values are the ciphertext.

Observe that this is only an approximate method for preserving the thumbnail. When plaintext pixels are b bits long, they are encrypted to ciphertext pixels whose expected average is roughly 2^{b-1}. If the plaintext pixels have a target average that is far from this average (e.g., close to 0 or close to 2^b), then substantially more bits must be flipped in step (6) to bring the ciphertext average for close to that of the plaintext. To avoid damaging too many bits of the plaintext, this embedding process must be stopped early (for example, after only damaging up to two or three bits).

This difficulty for very bright and very dark blocks is clearly shown in Figures **??-??**, especially in the lower right corner that is supposed to be entirely black. In this example, embedding was stopped early (at 3 bits) to prevent excessive damage to the decrypted result while still revealing an approximate thumbnail.

Decryption Decryption simply reverses the first 3 steps of encryption, and does not attempt to correct for the LSB embedding steps of encryption. The decryption process recomputes the same pseudorandom k_i values, reverses the bits in each integer in the block, and then computes the plaintext as $m_i = c_i \oplus k_i$.

Correctness Whenever we flip a MSB in the encryption step in order to better match the thumbnail, we cause an LSB to be flipped in the decrypted image. Similarly, whenever we flip one of the second-most significant bits in the ciphertext, we cause an error in a second-least significant bit in the plaintext, and so forth. The expected number of errors in a block is determined by the average plaintext intensity in the block and the number of bits that we are willing to patch up in order to match that average in the ciphertext. A block whose average intensity is very close to the midpoint of its allowable range will typically require only a few bits of correction in the ciphertext, because the ciphertext pixels (before any flipping) are expected to have an initial sum very close to the target plaintext sum. Blocks that are either very bright or very dark will require greater correction. If we patch up only one or two bit positions, the errors will appear in the decrypted image as high-frequency noise. As we flip more and more bits, the errors will become more significant. Our experimental results in Section 5.2 show that flipping a small number of bits provides a reasonable trade-off on real JPEG photos.

Leakage and Security Analysis The analysis is very similar to that of the previous scheme. The first three steps of encryption leak no more than than the value b (magnitude of largest plaintext value).

By construction, the final 3 steps of encryption use no information about the plaintext pixels except for their sum $\sum_i m_i$ (and the user-specified limit on the number of bits to flip). Hence, the overall ciphertext distribution is indistinguishable from a distribution that can be sampled knowing only b and $\sum_i m_i$. The ciphertexts therefore leak no more than these properties of the plaintext.

4.3 A Hybrid DRPE+LSB Construction

Our hybrid construction combines elements of the DRPE and LSB-embedding TPE mechanisms. In a nutshell, it first applies the DRPE encryption from Section 4.1 to encrypt each block. Then, it applies the LSB embedding scheme from Section 4.2, which ensures that the average of the ciphertext block is not too far from the average of its corresponding plaintext block.

Encryption Given a block-specific subkey $\tilde{K}$ and a vector of n plaintext integers $M = (m_1, m_2, \ldots, m_n)$, we encrypt as follows:

1. Compute the expanded dynamic range of the plaintexts $[l, u] = DRX(M)$, and let $b = \lceil \log_2(u - l) \rceil$. Intuitively, all integers in M have the same prefix (described by l) and differ only in their b least significant bits.

2. Use a pseudorandom generator to expand $\tilde{K}$ into nb pseudorandom bits $(k_1, \ldots, k_n)$, with each k_i exactly b bits long.

3. Encrypt the least significant b bits of each plaintext integer using a one-time pad. Hence, set

$$c_i = l + ((m_i - l) \oplus k_i)$$

4. Reverse the order of the b encrypted bits in each element c_i of the block; i.e., $c'_i = \mathsf{reverse}(c_i, b)$ for each i.

5. We compute the sum of plaintext integers $\sum_i m_i$ and of ciphertext integers $\sum_i c'_i$.

6. If the sum of the ciphertexts is too far from the sum of the plaintexts, we flip bits in the c'_i values, starting with the most significant of the encrypted bits. For each bit position B, when the difference between $\sum_i c'_i$ and the target $\sum_i m_i$ is smaller than 2^{B-2}, we proceed to the next bit position. The process ends after some limi set by the user, e.g. modifying at most the three most-significant bits.

   Also, starting with B as the most-significant encrypted bit position, we must be careful that we do not set bit B to be the same value in all ciphertexts in the block. If we do, then in the decryption step, we will not recover the correct dynamic range, and bit B will not be decrypted.

The resulting $(c'_1, \ldots, c'_n)$ values are the ciphertext.

Decryption To decrypt, we reverse the encryption process, without attempting to correct any bits flipped in the LSB embedding step.

Correctness Like the LSB embedding scheme, the hybrid scheme may introduce errors in the plaintext. However, because the hybrid scheme already preserves the dynamic range of the plaintexts before it flips any bits, it can handle many blocks with no additional errors. For example, suppose the user allows modification of up to three bits; then if all the plaintexts in a block share a 4 or 5-bit prefix, the LSB embedding step does not need to flip any bits.

Leakage and Security Analysis Like the previous two schemes, the hybrid construction reveals no more than (i) the expanded dynamic range of each block and (ii) the approximate average value in each block.

4.4 Handling AC Coefficients in JPEG

In JPEG, the DC coefficients describe the average pixel values within each 8x8 block (i.e., information that TPE aims to preserve) while the AC coefficients describe the "texture" (information that TPE aims to hide). Therefore, for TPE with thumbnail blocks larger than 8x8, we should reveal as little information as possible about the AC coefficients.

Fully encrypt AC coefficients. The simplest, and safest, approach then is simply to fully encrypt all the AC coefficients. Unfortunately this has at least two important drawbacks. First, while fully-encrypted AC coefficients do not impact the thumbnail, they make the ciphertext image extremely noisy when viewed at any resolution much higher than that of the thumbnail. In a small test with photos from the Holidays data set [9], we were unable to tell anything meaningful from such encrypted images despite them ostensibly having the same thumbnail as the plaintext. Second, the ciphertext images are extremely large. JPEG compression of AC coefficients relies on two assumptions: that many quantized AC values will be zero; and that nonzero values will be relatively low-entropy. Encrypting the AC coefficients breaks both of these assumptions, causing both JPEG's run-length coding and its entropy coding to be ineffective at reducing the file size.

Compromise: Revealing dynamic range of AC coefficients. As a compromise between secrecy and file size, we propose to use the dynamic range preserving encryption from Section 4.1 to protect the AC coefficients in the same way as the DC coefficients.

Assume we have a plaintext image and wish to perform TPE with block size $B \times B$, where B is a multiple of 8. Hence each thumbnail block consists of several atomic JPEG 8x8 blocks. Given a thumbnail block from the plaintext image, we extract the DC coefficient from each 8x8 JPEG block in the thumbnail block, and we encrypt them all as a single block using either of our TPE constructions above.

We propose to handle the AC coefficients in the same way. Given a thumbnail block, we extract all the AC coefficients of frequency 1 from the JPEG blocks and encrypt them together as a block using the DRPE. This leaks the approximate dynamic range of all the frequency-1 coefficients. Then we do the same for the frequency 2 coefficients, the frequency 3 coefficients, and so on, up to frequency 63.

We have found that this approach substantially reduces the size of our ciphertext images; see Section 5.1. Overall, images encrypted in this way will leak no more than the following information about each thumbnail block:

- Whatever is leaked in the DC domain by the usual approximate TPE scheme, depending on which scheme is used (e.g., the approximate dynamic range of the block, or the max value and sum of pixels in the block).
- The approximate dynamic range of frequency-1 AC coefficients in the block, of frequency-2 coefficients, etc.

Compromise: preserving zeroes. As a further compromise in favor of space efficiency, we can weaken the DRPE scheme a little bit more by preserving zeroes in the AC coefficients.

In more detail, when we encounter a zero in the plaintext, we do not encrypt it but instead leave it as-is. To avoid encrypting non-zero AC coefficients to zero ciphertext (which would cause decryption errors), we can do the following: Given that a nonzero coefficient is in some range $[1, v]$ (note that the dynamic-range-preserving scheme for AC coefficients already leaks an upper bound v on the magnitude of coefficients), we (a) subtract one to get a value in the range $[0, v-1]$; (b) encrypt by adding a one-time pad mod v, rather than XOR as before; (c) add one to get a ciphertext value in the range $[1, v]$ again.

In addition to what is leaked above, this modification also leaks whether or not individual AC coefficients are zero in the plaintext image. We suspect that preserving zeroes in the AC's must

substantially reduce the security of the TPE scheme. Unfortunately the impact of this weakness is difficult to predict or quantify. As a first step toward understanding the real-world security impact, we performed an experiment pitting both versions of our DRPE construction against a state-of-the-art face recognition technique. We found no significant difference in the two sets of results; see Section 5.4 for more information on the experiment. We evaluate the storage space trade off in Section 5.1.

5 Empirical Evaluation

Data We use two well known, publicly available image data sets in our evaluation. First, to measure the image quality and storage space overhead that our constructions achieve in the real world, we use the INRIA *Holidays* corpus [9]. It consists of 1491 JPEG images from consumer-level digital point-and-shoot cameras, taken by amateur photographers on trips to computer vision conferences at various locations around the world.

To evaluate the protection provided by our constructions against inference attacks, we use the AT&T Faces data set [17]. It consists of a total of 400 tightly cropped face images, showing 10 different angles for each of 40 human subjects. Each image contains 92x112 grayscale pixels.

5.1 Impact on JPEG Compression Efficiency

When we encrypt the DC and AC coefficients in a JPEG image, we increase their entropy. As a consequence, we also decrease the effectiveness of the lossless run-length coding and entropy coding steps that JPEG uses to compress the coefficients. To measure the real-world cost, we compared the storage space consumed by the encrypted Holidays corpus to that of the plaintext.

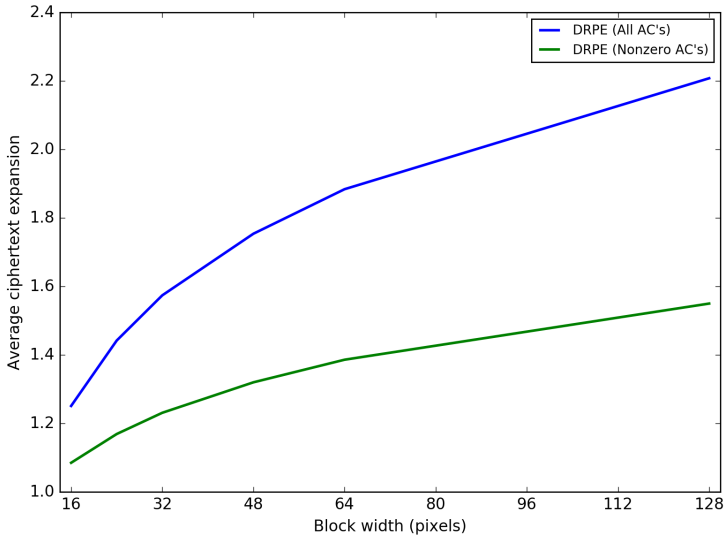

Figure 15: Increase in encrypted file size for dynamic range preserving TPE

Figure 15 shows the result of an experiment using our DRPE constructions to investigate the effect of the AC coefficients on the ciphertext file size. We used our DRPE thumbnail preserving

encryption to encrypt two copies of the Holidays data set. In one, we encrypted only the non-zero AC coefficients; in the other, we encrypted all of them. In both cases, increasing the size of the thumbnail block decreases the effectiveness of the compression. This happens because, with a larger block size, more JPEG blocks fall into thumbnail blocks that span wider ranges of values. Even if the plaintext coefficients have low entropy (i.e. they take on only a few values within their range), their ciphertexts will be uniformly distributed across the whole allowable range, and therefore they will be much more difficult to compress.

When we encrypt only the nonzero AC coefficients, the expansion is quite small. With small 16x16-pixel blocks, the encrypted image is only about 5% larger than the plaintext, and even with large 128x128-pixel blocks, the overhead is still below 50%. When we encrypt all AC coefficients, the increase in storage space is 4-5 times larger. With 16x16 blocks, the increase in file size is about 25%, and with 128x128 blocks, the ciphertexts are 120% larger than the plaintexts. Even when we encrypt all the AC coefficients, our construction is still significantly more efficient than the original thumbnail-preserving scheme from [21], which increased JPEG file sizes up to a factor of 6.

Based on the results in Figure 15, we elected to preserve the zeroes in the AC coefficients in the following experiments with our LSB embedding TPE. Figure 16 shows the increase in the size of the encrypted images produced by the LSB scheme when we correct up to 1, 2, 3, and 6 bits of the encrypted DC coefficients to match the original thumbnail, with the results for the DRPE scheme included for comparison. In general, the LSB embedding scheme produces larger encrypted images, but as we correct more bits we also reduce the overhead. This result is intuitive, since correcting bits reduces the entropy of the encrypted JPEG coefficients, making them more easily compressible.

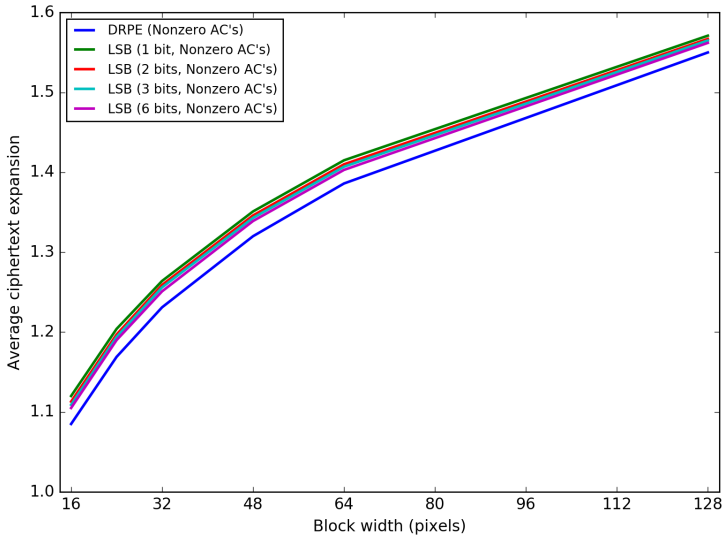

Figure 16: Increase in encrypted file size for LSB embedding TPE

5.2 Perceptual Quality of Ciphertext Thumbnails and Decrypted Images

Figure 17 shows how the perceptual quality of the decrypted image, as compared to the original plaintext, changes with the TPE block size for each of our constructions. Here, we use structural

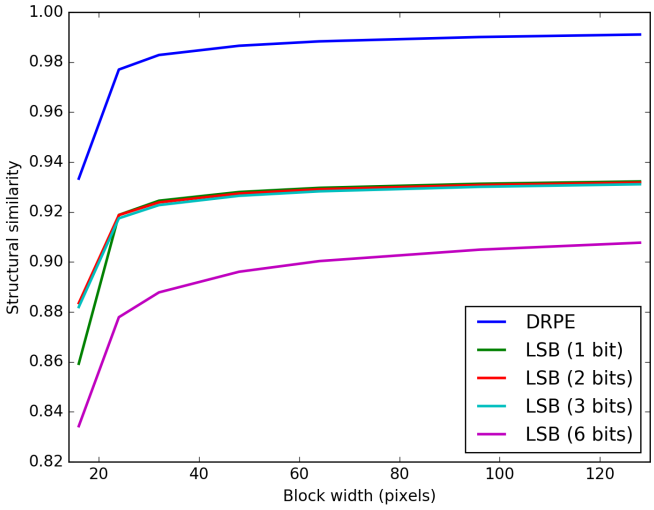

Figure 17: Average structural similarity between original and decrypted plaintext images

similarity (SSIM) [20] as an objective measure of perceptual image quality. SSIM uses knowledge of the human visual system's sensitivity to various kinds of errors in order to improve on simpler image quality metrics such as the mean squared error (MSE) or peak signal-to-noise ratio (PSNR). We used the open source tool `pyssim` to compute structural similarity between the original plaintext Holidays images and the decrypted versions, and between the thumbnails of the plaintexts and the thumbnails of the ciphertexts. In this experiment we used thumbnails of size 100x100 pixels, independent of the resolution of the original image.

With each of our constructions, the quality of decrypted image is substantially reduced when the thumbnail block size is small. This happens because, with small blocks, there is a non-negligible probability of an error in recovering the expanded dynamic range of a block. For blocks larger than 32x32 or 64x64 pixels, the quality of decrypted images from the dynamic range preserving scheme is near perfect. With the LSB scheme, the quality degrades as we increase the number of least-significant bits flipped to match the thumbnail in the encrypted image. Nevertheless, even when we flip up to 6 bits, the structural similarity is still around 0.9, which represents a moderate level of perceptual quality.

Compared to the pixel-shuffling TPE scheme from [21], our constructions are better able to handle large block sizes with little degradation in quality, while the approach in [21] tends to produce decrypted images with washed out colors. Our DRPE scheme in particular provides near-perfect quality for blocks larger than 24x24 pixels.

Figure 18 shows how the quality of the ciphertext image's thumbnail, as compared to the original plaintext's thumbnail, changes with the TPE block size. We resized each image to a small 100x100 pixel thumbnail, then computed the structural similarity between the original thumbnail and the thumbnail of the encrypted image. Overall, the perceptual quality of the thumbnail decreases as the size of our encrypted thumbnail block approaches and then surpasses the actual number of original pixels that contribute to each pixel in the thumbnail image. However, even with larger blocks, thumbnail quality is still reasonable. As one might have expected, the thumbnail quality

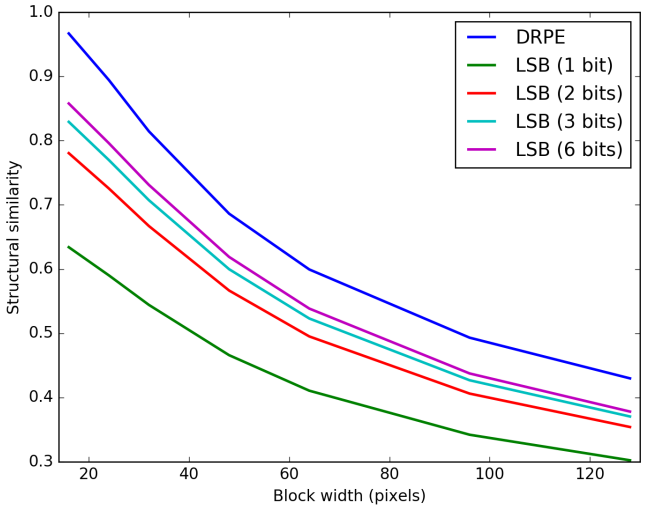

Figure 18: Average structural similarity between plaintext thumbnails and ciphertext thumbnails

is worst with the LSB embedding scheme when we only modify 1 bit per DC coefficient, but the quality improves quickly as we flip more bits.

5.3 Preservation of Perceptual Hashes

One unique advantage of (approximate) thumbnail preserving encryption is that it preserves features of the original image that are often used in automated image recognition techniques for content-based image retrieval or misuse detection. For example, Microsoft's PhotoDNA [12] system, which is used by Facebook, Dropbox, and others for detecting child exploitation images, uses a perceptual hash algorithm as a sort of robust fingerprint for known illegal content. Other perceptual hash functions, pHash [23] and BlockHash [22], have been used to detect pornography and copyright violations, respectively. Typically, a perceptual hash algorithm, including those in PhotoDNA, pHash, and BlockHash, begins by scaling down the input image to a very low resolution (e.g. 9x9 grayscale pixels).

By design, a thumbnail preserving encryption should preserve most of the bits in the perceptual hash. To test this hypothesis, we used the `pyssim` tool to compute the BlockHash of each plaintext and encrypted image from the Holidays data set. The Holidays corpus was originally designed for evaluation of content-based image retrieval ("query by example") techniques, so its 1491 images consist of 500 query images and 991 other images. Each non-query image is intended as a match for exactly one of the query images. For each of our TPE constructions, we compared the BlockHash of each plaintext query image to the hashes of (*i*) the encryption of the same image, (*ii*) the encryptions of its designated matching images, and (*iii*) the encryptions of all other, non-matching images.

Figure 19 shows the average number of bits that differed in the BlockHash for each group of images for each TPE scheme. The BlockHash is 256 bits long, so we should expect that two unrelated images would differ in about 128 of these bits. In our experiment, we see that the actual Hamming distance is slightly lower; for each configuration of our TPE schemes, unrelated images

differ on average by about 125 bits. In contrast, when we compare the same image to its own ciphertext, we see that most of the bits of the hash are preserved. As we increase our thumbnail block size, we corrupt more bits of the hash. The DRPE scheme is most sensitive to changes in the block size, since it does not preserve the correct average intensity in each block. With the LSB scheme, by patching up more bits to preserve the correct average, we also preserve more of the perceptual hash. Based on these results, it should be feasible to detect known images when they appear in encrypted form, by setting a threshold of about 70 or 80 bits.

On the other hand, our results are not as promising for content-based image retrieval based on BlockHash. Across our TPE schemes, encrypted matching images have an average Hamming distance of about 100–105 bits from their plaintext query images. While the average distance for matching images is clearly lower than the average distance for unrelated images, the standard deviation of each population is about 20 bits, so there is substantial overlap. More work will be required to determine whether a perceptual hash algorithm can be used to achieve better separation between similar and unrelated images with TPE.

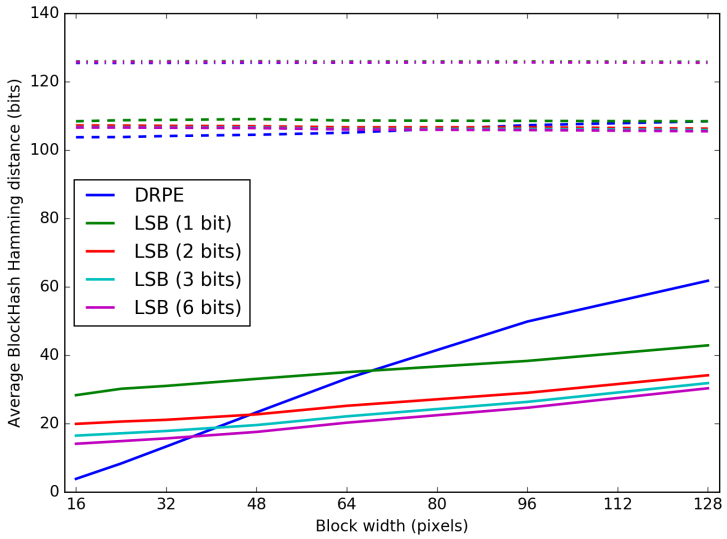

Figure 19: BlockHash distance for plaintext versus encrypted images: same image (solid line), similar images (dashed line), and unrelated images (dotted line).

5.4 Security Against Inference Attacks

A recent study by McPherson et al [11] showed that simply pixellating, or *mosaicing*, faces in pictures is not sufficient to thwart face recognition using modern convolutional neural networks. Since thumbnail preserving encryption is essentially a reversible, randomized mosaicing, we replicate some of their experiments here in order to better understand the protection that TPE can provide against face recognition in encrypted images.

We use the same AT&T face data set [17] that was used in the McPherson experiments. Although there are many newer and larger face data sets, we use the AT&T data because it gives almost every reasonable advantage to our adversary. In the real world, before an adversary can

attempt to recognize a face in an encrypted image, it must first detect the face and draw a bounding box around it. In the AT&T data set, the face detection step is not necessary, as each image contains nothing but the face itself. The adversary gets an additional advantage in that every face bounding box in the AT&T data is also aligned to the start of a thumbnail block. In the real world, faces could occur anywhere within a block, and such perfect alignment would be rare. Also in the real world, many applications for face recognition require the recognizer to identify people from a very large population. With only 40 subjects, AT&T Faces represents a much simpler scenario, such as identifying people in pictures on a social network where the user's set of friends is small and known in advance.

A final advantage of the AT&T data is that its small sizes allows us to run experiments quickly on even a modest GPU. We ran our experiments with the Torch7 machine learning framework on CentOS Linux 7, running on a quad-core Intel i7-3770 CPU and an nVidia GTX 1050 Ti GPU with 768 CUDA cores and 4 GB of onboard RAM.

We encrypted the AT&T data set with our DRPE and LSB thumbnail preserving encryption with block sizes of 16x16, 32x32, 48x48, 64x64, 80x80, 96x96, 112x112, and 128x128 pixels. For a direct comparison with [11] we also generated a non-reversible mosaiced version of the data set for each of the same block sizes. We pseudorandomly generated 10 testing/training splits of the corpus, where in each split we assign two images of each subject to the testing set and use the remaining eight images for training. Following the experiment in [11], we train the neural network from McPherson et al on each training set for 150 iterations, and we record its accuracy on the testing set at the 150th iteration. We report the average 150th-iteration accuracy over the 10 testing/training splits.

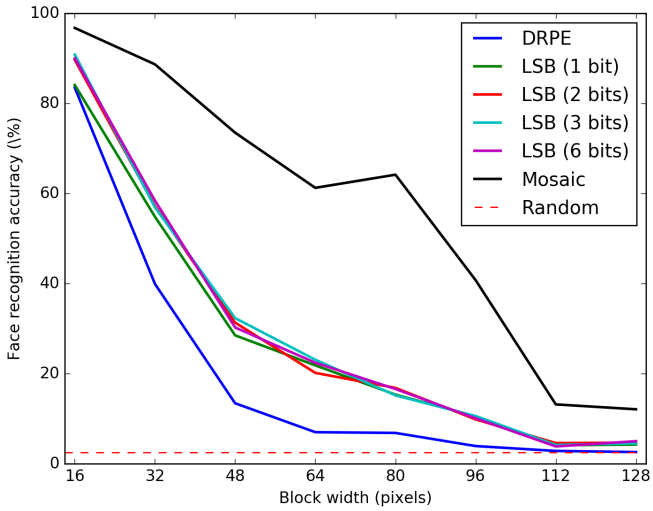

Figure 20: Face recognition accuracy for McPherson et al [11] classifier

Figure 20 shows how the neural network's accuracy degrades as we increase the thumbnail block size. Our results for the mosaiced images are consistent with those from the previous work, although they only tested with very small blocks up to 16x16 pixels. For 16x16, they reported 96.25% accuracy, while our experiment yielded 96.75%. For larger mosaic blocks, the adversary's

accuracy drops off substantially, although he still performs over 20 times better than random guessing with 64x64 or 80x80 blocks.

The adversary's accuracy is lower for TPE images. We suspect that much of the drop in accuracy is due to the randomness in the TPE ciphertexts. Two blocks with the same average intensity will appear identical under a mosaicing operation, but with TPE they will be encrypted differently as long as they are in different files or at different locations in the image. With the dynamic range preserving encryption and 64x64 blocks, the adversary's accuracy is only 7%–slightly less than three times better than random guessing. Because the LSB embedding scheme preserves a more accurate thumbnail than DRPE, it also enables the adversary to recognize faces more accurately. Flipping only 1 or 2 bits appears sufficient to give the adversary this boost. He receives surprisingly little additional benefit with 3 or 6 bits. With very large 112x112 blocks, each image contains only a single block, so it is not surprising that the adversary performs no better than random guessing against TPE. With large mosaic blocks, it appears that knowing only the average intensity in the image is sufficient to reveal the subject's identity 13.2% of the time.

To evaluate the impact of preserving zeroes in the JPEG AC coefficients, we repeated the experiment using the DRPE scheme and encrypting all AC coefficients. We compared the results to those for DRPE in Figure 20, and we could find no statistically significant differences.

Discussion We hesitate to generalize too much from this small initial experiment. But based on our results, it appears that TPE can provide substantial protection against state-of-the-art face recognition, as long as the subject's face is smaller than about 2x3 thumbnail blocks in the encrypted image. This is probably not realistic for portraits or head shots, such as a profile picture on a social networking site. Our TPE schemes could still be a good fit for most vacation photos, which contain a few people with larger objects in the background, or for large group photos, where each person's face is relatively small.

6 Conclusion

We proposed the idea of *approximate* thumbnail preserving encryption and described three new constructions that satisfy this notion. Our first scheme preserves the dynamic range of the values in each thumbnail block of the plaintext image. Our second scheme preserves both the rough order of magnitude in each block and also an approximation of the average value in the block. Our third scheme combines elements of the first two in order to achieve a better balance of accuracy in the thumbnail and perceptual quality in the decrypted image. Compared to the only previous work on thumbnail preserving encryption, our constructions offer improved, provable security; smaller encrypted files; and better perceptual quality of the decrypted images, especially with larger thumbnail block sizes. Our experiments show that our techniques also preserve thumbnails with acceptable perceptual quality, and they preserve a perceptual hash well enough to allow for identification of known bad content in encrypted form. In experiments against a state-of-the-art inference attack using deep neural networks, our techniques outperformed simple mosaicing and another recent scheme for sharing encrypted images on online services.

In future work, we will continue to refine the inference attacks presented here in order to better understand the impact of various trade-offs between security and compression efficiency. We will also work on extending our techniques for encrypting JPEG images to also handle recent compressed video formats, e.g. the MPEG family of standards.

Acknowledgements

The authors would like to thank Seth Terashima for many interesting discussions and ideas in the early stages of this project. The second author is partially supported by NSF award 1623400. The fourth author is partially supported by NSF awards 1149647 & 1617197.

References

[1] D. Beaver, S. Kumar, H. C. Li, J. Sobel, P. Vajgel, et al. Finding a needle in haystack: Facebook's photo storage. In *OSDI*, volume 10, pages 1–8, 2010.

[2] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. In M. J. J. Jr., V. Rijmen, and R. Safavi-Naini, editors, *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2009.

[3] T. Bergin. Yahoo email scanning prompts European ire. *Reuters*, October 2016.

[4] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill. Order-preserving symmetric encryption. In *Advances in Cryptology-EUROCRYPT 2009*, pages 224–241. Springer, 2009.

[5] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Advances in Cryptology–CRYPTO 2011*, pages 578–595. Springer, 2011.

[6] A. Greenberg. The police tool that pervs use to steal nude pics from Apple's iCloud. *Wired*, September 2014.

[7] P. Gutmann. Encrypting data with a restricted range of values, January 1997. Posted to sci.crypt news group. See `https://groups.google.com/d/msg/sci.crypt/bK8mSWeC41k/hpVUcQBicLYJ`.

[8] S. Hill, Z. Zhou, L. Saul, and H. Shacham. On the (in)effectiveness of mosaicing and blurring as tools for document redaction. *Proc. Privacy Enhancing Technologies*, 2016(4):403–17, Oct. 2016.

[9] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. *Computer Vision–ECCV 2008*, pages 304–317, 2008.

[10] K. Lander, V. Bruce, and H. Hill. Evaluating the effectiveness of pixelation and blurring on masking the identity of familiar faces. *Applied Cognitive Psychology*, 15(1):101–116, 2001.

[11] R. McPherson, R. Shokri, and V. Shmatikov. Defeating Image Obfuscation with Deep Learning. *ArXiv e-prints*, Sept. 2016.

[12] Microsoft. PhotoDNA newsroom. See `http://news.microsoft.com/presskits/photodna/`.

[13] E. M. Newton, L. Sweeney, and B. Malin. Preserving privacy by de-identifying face images. *IEEE Trans. on Knowl. and Data Eng.*, 17(2):232–243, Feb. 2005.

[14] S. J. Oh, R. Benenson, M. Fritz, and B. Schiele. *Faceless Person Recognition: Privacy Implications in Social Media*, pages 19–35. Springer International Publishing, Cham, 2016.

[15] M.-R. Ra, R. Govindan, and A. Ortega. P3: Toward privacy-preserving photo sharing. In *NSDI*, pages 515–528, 2013.

[16] J. Robertson. Dropbox confirms 2012 breach bigger than previously known. *Bloomberg*, August 2016.

[17] F. S. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 138–142. IEEE, 1994.

[18] S. Sengupta and K. J. O'Brien. Facebook can id faces, but using them grows tricky. *The New York Times*, September 2012.

[19] M. Tierney, I. Spiro, C. Bregler, and L. Subramanian. Cryptagram: Photo privacy for online social media. In *Proceedings of the First ACM Conference on Online Social Networks*, COSN '13, pages 75–88, New York, NY, USA, 2013. ACM.

[20] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[21] C. V. Wright, W.-c. Feng, and F. Liu. Thumbnail-preserving encryption for jpeg. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '15, pages 141–146, New York, NY, USA, 2015. ACM.

[22] B. Yang, F. Gu, and X. Niu. Block mean value based image perceptual hashing. In *Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP'06. International Conference on*, pages 167–172. IEEE, 2006.

[23] C. Zauner. Implementation and benchmarking of perceptual image hash functions. Master's thesis, Upper Austria University of Applied Sciences, Hagenberg Campus, 2010. See also `http://www.phash.org/`.

[24] Q. A. Zhao and J. T. Stasko. The awareness-privacy tradeoff in video supported informal awareness: A study of image-filtering based techniques. Technical report, Goergia Tech, http://hdl.handle.net/1853/3452, 1998.