Robust Homomorphic Image Hashing

Priyanka Singh
Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar, Gujarat, India
Priyanka_Singh@daiict.ac.in

Hany Farid
University of California, Berkeley
Berkeley CA, USA
hfarid@berkeley.edu

Abstract

Most image forensic techniques are concerned with authenticating the contents of an image, linking an image to a device or class of devices, or extracting forensically useful information from an image. Another aspect of image forensics is the identification of previously identified content, particularly in the face of simple image modifications. Such so-called robust image hashing techniques can be highly effective at finding child sexual abuse material, revenge porn, terrorism-related material, and dangerous or hateful conspiracy material. The growing use of end-to-end encryption on commercial platforms makes identification of such material significantly more challenging. We describe a robust image hashing algorithm that is both robust to simple image manipulations and that can operate on an encrypted image, without the need or even ability to decipher the underlying encrypted image.

1. Introduction

In 2017 Facebook announced a trial program to combat the growing and troubling problem of revenge porn [7]. Facebook users wishing to remove and prevent future upload of personal and potentially embarrassing images of themselves were asked to send these images to Facebook. Facebook would in turn extract a distinct signature (a hash or robust hash) from this content, allowing the platform to eliminate a specific image from future upload. A similar technology – photoDNA – has been in use on Facebook and other platforms to prevent the upload of known child sexual abuse material [10].

Although well-intentioned, this program was met with some skepticism and ridicule. At a time when Facebook, and social media in general, is under intense scrutiny for their failure to protect user data and privacy, it seemed unreasonable to ask users to trust Facebook with some of their most personal and intimate images.

More recently, Facebook announced plans to move all messaging services within the Facebook ecosystem to use end-to-end encryption [6]. Without proper safeguards, it will be nearly impossible to contend with the trafficking of child sexual abuse, terrorism, and dangerous conspiracy material.

We propose that a balance can be found between privacy and security. Users should be able to communicate without fear of corporate or government interference, but at the same time, platforms should have safeguards in place to contend with the most abusive and dangerous content making its way through their platforms.

To this end, we describe a secure robust image hashing algorithm that extracts a distinct but stable (to some common image manipulations) signature that can be used to quickly and reliably identify a specific piece of content. The hash is designed to be distinct, stable, efficient, and amenable to being implemented within a partially homomorphic encryption system. In particular, by restricting the computations to addition and scalar multiplication of unencrypted pixel values (plaintext), we can extract the hash from an encrypted version (ciphertext) of the image, without requiring or even being able to decrypt the image.

We will show through large-scale experiments that the robust image hash is highly distinct and stable to a broad range of non-linear intensity adjustments (gamma correction), additive noise, compression quality, and re-scaling.

1.1. Related Work

There is a significant body of literature on so-called content-based image retrieval (CBIR) techniques [16, 8] Within this literature, there has also been some efforts on

developing secure or privacy-preserving CBIR techniques. We briefly review this literature and place our work in the context of these techniques.

Secure CBIR techniques consist of two basic steps consisting of the extraction of a distinct signature and comparison of that signature to a pre-computed database of image signatures.

Several specialized secure CBIR schemes have been proposed. For example, Erkin et al. [3] proposed a technique that allows for server-side facial recognition without having to transmit or store sensitive facial biometric information in an unencrypted format. Like our approach, this technique exploits the homomorphic properties of the Paillier encryption algorithm, but unlike our approach, only applies to facial recognition. Rahulamathavan et al. [14] proposed a similar secure facial recognition system. Both of these techniques require multiple client-server communications. In contrast, in our approach, we require only a single client-server communication.

Lu et al. [9] proposed a secure CBIR scheme to enable search over encrypted multimedia databases. In this scheme, a signature is extracted prior to encryption, the multimedia content is encrypted with a traditional algorithm (e.g., AES, DES), and this signature and encrypted data are sent to the server. Xia et al. [19] proposed a secure CBIR scheme that employs an encrypted stream cipher approach. While effective at preserving privacy, these approaches are not robust to simple image manipulations, whereas our approach is robust to luminance non-linearities, additive noise, re-compression, and re-scaling.

Several feature-based privacy-preserving transforms have previously been proposed [17, 5, 15, 1, 2]. These approaches generally do not reach the accuracies needed to operate on a large social-media platform like Facebook, that routinely sees billions of uploads per day. To this end, we seek detection accuracies in the mid to high 90-percentiles with false alarms on the order of 1 in ten million. Competing approaches to these feature-based approaches have also been proposed [12, 13, 18], but these techniques cannot compete with the computational efficiency of the feature-based approaches.

In summary, unlike previous approaches, our technique is applicable to all images, is robust to common image manipulations, achieves a high degree of matching even with false matches on the order of 1 in 10 million, requires only a single client-to-server message, and is computationally efficient.

Our robust and secure CBIR leverages the properties of homomorphic encryption within the Paillier cryptosystem, but is not critically dependent on it. We begin with an overview of this encryption scheme and then describe the details of our technique.

2. Paillier Cryptosystem

The Paillier cryptosystem is a partially homomorphic, asymmetric encryption scheme [11]. We briefly describe this cryptosystem and then enumerate its homomorphic properties.

A public-private key pair is computed by first generating two large prime numbers p and q, from which the public keys n and g are computed as:

$$n = p \times q \tag{1}$$
$$g = n + 1. \tag{2}$$

The private keys λ and μ are computed as:

$$\lambda = \text{lcm}(p-1, q-1) \tag{3}$$
$$\mu = \text{mod}(\alpha, n), \tag{4}$$

where $\text{lcm}(\cdot)$ denotes the least common multiple operator, $\text{mod}(\cdot)$ is the modulo operator, and α is computed by the extended Euclidean algorithm. Specifically, γ is first computed as follows:

$$\gamma = \text{gcd}\left(\frac{\text{mod}(g^\lambda, n^2) - 1}{n}, n\right), \tag{5}$$

where $\text{gcd}(\cdot)$ is the greatest common divisor operator. The Bézout coefficients α and β satisfy:

$$\gamma = \alpha \times \frac{\text{mod}(g^\lambda, n^2) - 1}{n} + \beta \times n. \tag{6}$$

With the public (n, g) and private (λ, μ) keys generated, we next describe how to encrypt and decrypt a message.

A plaintext message m can be encrypted to ciphertext c as follows:

$$c = E(m, r; g, n) = \text{mod}(g^m \times r^n, n^2), \tag{7}$$

where r is a random integer satisfying $0 < r < n$. The incorporation of this random value ensures that the same plaintext is encoded as different ciphertexts under the same public key.

A ciphertext message c is decrypted as follows:

$$D(c; \lambda, \mu, n) = \text{mod}\left(\frac{\text{mod}(c^\lambda, n^2) - 1}{n} \times \mu, n\right). \tag{8}$$

The Paillier cryptosystem is partially homomorphic, satisfying multiplicative and exponentiation identities. Let ciphertexts c_1 and c_2 be the encrypted version of messages m_1 and m_2. The homomorphic properties are as follows:

- Under the appropriate modular arithmetic, the decrypted product of two ciphertexts is equal to the sum of their corresponding plaintexts:

$$D(\text{mod}(c_1 \times c_2, n^2); \lambda, \mu, n) = \text{mod}(m_1 + m_2, n). \tag{9}$$

- Under the appropriate modular arithmetic, the decrypted exponentiation of a ciphertext by a scalar value s is equal to the product of the scalar and corresponding plaintext:

$$D(\text{mod}(c_1^s, n^2); \lambda, \mu, n) = \text{mod}(s \times m_1, n). \quad (10)$$

Although these homomorphic properties are fairly limited, we will describe a robust image hashing scheme (extraction and comparison) that requires only addition and scalar multiplication in the plaintext domain, meaning that it can be fully implemented in the encrypted domain.

3. Robust Image Hashing

We describe a robust image hashing algorithm that extracts a compact, distinct, and robust (to common image manipulations) signature. In addition to these typical requirements, we seek a hashing algorithm that lends itself to computation on the corresponding ciphertext as described in the previous section. This limitation requires us to rely only on addition and scalar multiplication in the plaintext domain. We also seek to bundle all of the information needed to perform the hashing in the ciphertext domain into a single payload which, as we will see below, requires encoding additional information beyond the image pixels. We first describe this algorithm in the plaintext domain, and then in the ciphertext domain.

3.1. Plaintext domain

We begin by performing a series of simple pre-processing steps that collectively reduce the complexity of extracting a hash, reduces the length of the extracted hash, and increases its robustness to image modifications. These steps are not expected to be performed on the ciphertext and so need not conform to the homomorphic computation limitations.

The first pre-processing step converts a three-channel (RGB) color image to grayscale using a standard conversion of $I = 0.2989R + 0.5870G + 0.1140B$ (if the image is already grayscale, then this step is not performed). The grayscale image is then down-sized (or up-sized) to a fixed resolution of 302×302 pixels using bicubic (or similar) interpolation so as to minimize spatial aliasing (because we will eventually trim a one-pixel border, the final image size will be 300×300). These first two pre-processing steps serve to reduce the complexity of extracting a hash, reduce the length of the extracted hash, and make the hash robust to minor modifications that will be (typically) lost when down-sizing the image to a relatively low resolution. The rescaled grayscale image is then subjected to histogram equalization in which the grayscale image is adjusted to have a (nearly) uniform intensity distribution. This final pre-processing step improves the robustness of the hash to simple intensity/color modifications.

The hash is extracted from the pre-processed image $I(x, y)$ as follows. The image is first convolved with a pair of 1-D separable derivative filters to yield the following partial derivative images:

$$I_x(x, y) = I(x, y) \star d(x) \quad (11)$$
$$I_y(x, y) = I(x, y) \star d(y), \quad (12)$$

where $\star$ denotes convolution with the filter $d = [-1 \;\; 1]$ applied in the horizontal (x) and vertical (y) directions. Accurate discrete differentiation requires a pair of 1-D convolutions with a matched derivative- and pre-filters [4]. We jettison the pre-filter, sacrificing more accurate derivative measurements in favor of a significantly simplified cipher-domain convolution with no loss of distinctiveness or robustness of the underlying hash.

To avoid edge artifacts that arise due to the convolution, a one-pixel border around the entire image is removed prior to subsequent processing. This yields a final image size of 300×300.

Recall that the homomorphic properties in Equations (9) and (10) hold under the appropriate modular division with $n = p \times q$. Because p and q will be large values, their product will also be large and so it is not likely that the convolution of pixels values, on a scale of $[0, 255]$, will exceed n in size. We need not, therefore, concern ourselves with the modular division.

Each partial derivative image $I_x(\cdot)$ and $I_y(\cdot)$ is partitioned into 5×5 blocks each of size 60×60 pixels. A partial hash is extracted from each of these 25 blocks in the same way. The final hash is the concatenation of each of these partial hashes.

Consider a single 60×60 pixel block with corresponding original intensity values $I(x, y)$ and partial derivatives $I_x(x, y)$ and $I_y(x, y)$. A 2-D histogram is computed from these triple of values. The horizontal axis of this histogram corresponds to the original intensity values $I(\cdot)$ quantized into B bins. The vertical axis of this histogram corresponds to one of 4 bins corresponding to a positive horizontal derivative ($I_x(\cdot) \geq 0$), negative horizontal derivative ($I_x(\cdot) < 0$), positive vertical derivative ($I_y(\cdot) \geq 0$), and negative vertical derivative ($I_y(\cdot) < 0$).

Consider, for example, the 4×4 histogram in Figure 1. The columns in this histogram correspond to an intensity value in the range $[0, 64)$, $[64, 128)$, $[128, 192)$, or $[192, 255]$. The rows correspond to the positive/negative, horizontal/vertical derivatives. For example, a pixel (x_0, y_0) with intensity value $I(x_0, y_0) = 56$ and a corresponding negative horizontal derivative $I_x(x_0, y_0)$ and negative vertical derivative $I_y(x_0, y_0)$ results in the bins in column 1 and row 2 and 4 to each be incremented by one. That is, each pixel is counted in two bins corresponding to the sign of its

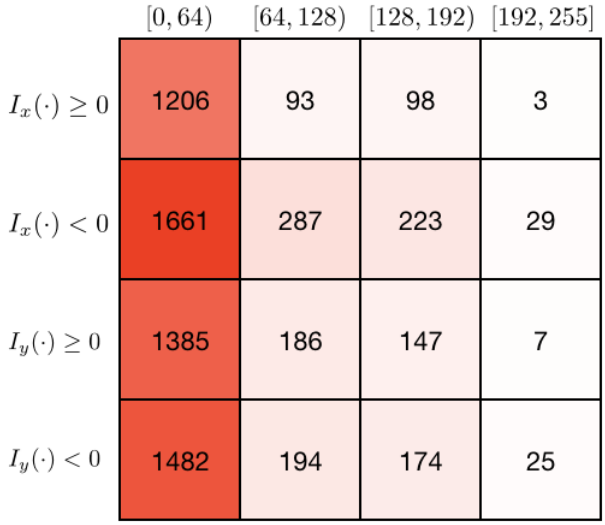

Figure 1. A 2-D histogram of intensities (columns) and derivatives (rows) used to construct an image hash. The numeric values in each cell correspond to the histogram count and each cell's color saturation is proportional to this count. Note that the values in this histogram, constructed for a single 60×60 pixel block, sum to $7,200$ – two values for each of $60 \times 60 = 3,600$ pixels.

horizontal and vertical derivative. This 2-D histogram embodies both the underlying appearance (intensity) and structure (derivative) of the image.

Denote the partial hash, of length $4B$, where B is the number of intensity bins, of each 60×60 pixel block as $\vec{h}_i$, $i = 1, \ldots, 25$, enumerated in column-order. The final hash, of length $25 \times 4B = 100B$, is the concatenation of these hashes:

$$\vec{H} \quad = \quad [\vec{h}_1 \;\; \vec{h}_2 \;\; \cdots \;\; \vec{h}_{25}]. \tag{13}$$

Note that since the hash is constructed from a histogram of intensity/derivative values, it can be encoded as positive integers.

The difference between two hashes $\vec{H}_1$ and $\vec{H}_2$ is measured using an L1-norm:

$$\Delta \quad = \quad \sum_{i=1}^{100B} |H_1(i) - H_2(i)| \tag{14}$$

where the index i corresponds to the i^{th} vector component. This distance Δ can then be compared against a specified threshold to determine if the two images are similar or not.

Because the convolution in Equations (11) and (12) require only the addition of scalar weighted pixel values, this computation can be performed on the corresponding ciphertext. The construction of the intensity/derivative histogram, of course, does not require any explicit computation. The histogram construction does, however, require comparing intensity/derivative values to place them in the appropriate histogram bin. As we will see in the next section, in order to perform this comparison on the corresponding ciphertext, additional information needs to be encoded alongside the encrypted image pixels. Because the final hash $\vec{H}$ will be computed in plaintext, the comparison of two hashes can be performed without relying on any homomorphic computations.

3.2. Ciphertext domain

In Section 3.1 we described a robust image hashing algorithm in the plaintext domain. Because all computations after the pre-processing stage consist entirely of addition and scalar multiplication, this hashing can be performed in the Paillier cryptosystem (Section 2). Specifically, we will show that it is possible to extract a hash from a pre-processed image encrypted under Paillier encryption that is equivalent to the plaintext hash.

To begin, each pixel of a pre-processed image $I(x, y)$ is independently encrypted using a random value $r(x, y)$ to yield an encrypted pixel $\tilde{I}(x, y)$, Equation (7). The 2-D intensity/derivative histogram is constructed from this pre-processed image. We will first describe the required intensity-based calculations followed by the derivative-based calculations.

In order to determine the corresponding intensity bin for each pixel, Figure 1, the encrypted intensity values are compared against the encrypted bin centers. Specifically, the intensity bin centers b_i $i = 1, \ldots, B$, are encrypted to yield $\tilde{b}_i$. Recall that each pixel value is encrypted with a different random value r, Equation (7). As such, the bin centers must be encrypted with the random values for each pixel in the 300×300 pre-processed image.

In plaintext, an intensity value $I(x, y)$ is placed in the closest bin b_k where:

$$k \quad = \quad \underset{i}{\text{argmin}} \, |(I(x, y) - b_i)| \, . \tag{15}$$

This difference can be computed in ciphertext by leveraging the homomorphic properties described in Section 2. Note, however, that only the difference $I(x, y) - b_i$ modulo n can be computed, Equation (9). By exploiting the properties of modulo division, the desired bin k under modulo division can be computed as follows:

$$k \quad = \quad \underset{i}{\text{argmin}} \left[\min(\Delta_i^l, \Delta_i^r) \right], \tag{16}$$

where:

$$\Delta_i^l \quad = \quad \text{mod}(I(x, y) - b_i, n) \tag{17}$$

$$\Delta_i^r \quad = \quad \text{mod}(b_i - I(x, y), n), \tag{18}$$

where we have replaced the absolute value operator with the minimum between the left and right distance between a pixel value and a histogram bin center.

In ciphertext, this desired bin is:

$$k = \underset{i}{\text{argmin}} \left[\min(\tilde{\Delta}_i^l, \tilde{\Delta}_i^r) \right], \quad (19)$$

where:

$$\tilde{\Delta}_i^l = \text{mod}(\tilde{I}(x,y) \times \text{mod}(\tilde{b}_i^{-1}, n^2), n^2) \quad (20)$$

$$\tilde{\Delta}_i^r = \text{mod}(\tilde{b}_i \times \text{mod}(\tilde{I}(x,y)^{-1}, n^2), n^2). \quad (21)$$

In the calculation of $\tilde{\Delta}_i^l$, the inner term $\text{mod}(\tilde{b}_i^{-1}, n^2)$ computes the negative of $\tilde{b}_i$ using the scalar multiplicative homomorphic property in Equation (10). The product of this value and $\tilde{I}(x,y)$ computes the difference between the intensity value and bin center using the additive homomorphic property in Equation (9). The second term $\tilde{\Delta}_i^r$ similarly computes the difference between the bin center and the intensity value.

The next step in the computation of the robust hash is the calculation and binning of the partial image derivatives. Because the computation of the derivatives, Equations (11)-(12), requires only 1-D convolutions, we will formulate their computation as 1-D convolutions:

$$I_x(x) = I(x) \star d(x). \quad (22)$$

Consider now the derivative in the plaintext domain at pixel location (x_0):

$$I_x(x_0) = I(x_0+1) - I(x_0). \quad (23)$$

Because the homomorphic properties in Equations (9) and (10) hold under the appropriate modular division with n, we must consider this derivative calculation under modular division. But, because n, the product of two large prime numbers, will be significantly larger than the convolution of pixels values, on a scale of $[0, 255]$, we need not concern ourselves with the modular division.

Based on the homomorphic properties in Equations (9)-(10), this derivative in the ciphertext domain is:

$$\tilde{I}_x(x_0) = \text{mod}(\tilde{I}(x_0+1) \times (\tilde{I}(x_0))^{-1}, n^2). \quad (24)$$

Once computed, the horizontal and vertical derivative at each pixel is binned based on whether it is greater than or equal to, or less than 0, Figure 1. This comparison is performed in the same way as with the intensity values where the value of b_i in Equations (15)-(21) is simply 0.

Recall that the intensity bin centers were encrypted with the random values used to encrypt each pixel in the 300×300. In the case of the derivative calculations, however, the reference value 0, must be encrypted with the random value $r_1 \times r_0^{-1}$, where r_0^{-1} represents modular inverse of r_0 under modulo n^2 and where r_1 and r_0 are the random numbers used to encrypt pixel values $I(x_0+1)$ and $I(x_0)$, respectively.

To see why the reference value of 0 must be encrypted with this combination of random variables, consider first the representation of the individual terms $\tilde{I}(x_0)$ and $\tilde{I}(x_0+1)$ in ciphertext, as specified by Equation (7):

$$\tilde{I}(x_0) = \text{mod}(g^{I(x_0)} \times r_0^n, n^2) \quad (25)$$

$$\tilde{I}(x_0+1) = \text{mod}(g^{I(x_0+1)} \times r_1^n . n^2). \quad (26)$$

Substituting these two ciphertext-based derivatives into Equation (24) yields:

$$\tilde{I}_x(x_0) = \text{mod}(\text{mod}(g^{I(x_0+1)} \times r_1^n, n^2) \times \text{mod}(g^{I(x_0)} \times r_0^n, n^2)^{-1} . n^2) \quad (27)$$

Using the modular exponent property[1], this expression can be rewritten as:

$$\tilde{I}_x(x_0) = \text{mod}(\text{mod}(g^{I(x_0+1)} \times r_1^n, n^2) \times \text{mod}((g^{I(x_0)} \times r_0^n)^{-1}, n^2) . n^2) \quad (28)$$

The inverse in the second term can be distributed as follows:

$$\tilde{I}_x(x_0) = \text{mod}(\text{mod}(g^{I(x_0+1)} \times r_1^n, n^2) \times \text{mod}((g^{-I(x_0)} \times r_0^{-n}), n^2) n^2). \quad (29)$$

Using the modular multiplicative property[2], this expression can be rewritten as:

$$\tilde{I}_x(x_0) = \text{mod}((g^{I(x_0+1)} \times r_1^n) \times (g^{-I(x_0)} \times r_0^{-n}), n^2). \quad (30)$$

The $g-$ and $r-$terms can now be collected as follows:

$$\begin{aligned} \tilde{I}_x(x_0) &= \text{mod}((g^{I(x_0+1)} \times g^{-I(x_0)} \times r_1^n \times r_0^{-n}), n^2) \\ &= \text{mod}((g^{I(x_0+1)-I(x_0)} \times (r_1 \times r_0^{-1})^n), n^2) \\ &= \text{mod}((g^{I_x(x_0)} \times (r_1 \times r_0^{-1})^n), n^2), \end{aligned} \quad (31)$$

from which we see that the random number that effectively encodes the spatial derivative $I_x(x_0)$ is $r_1 \times r_0^{-1}$, which is therefore the same value used to encrypt the reference value 0 to which spatial derivatives will be compared.

To summarize, the desired intensity/derivative histogram is computed by first placing each encrypted intensity pixel value into one of B intensity bins, Equation (19). The horizontal and vertical derivatives are then computed from the encrypted pixel values, Equation (24). These encrypted derivative values are then each placed into two of four bins depending on the sign of the horizontal and vertical derivative. The 2-D histogram is computed by simply counting the encrypted pixels that occupy each bin. This resulting plaintext histogram (hash) can then be directly compared against a database of hashes using the L1-norm in Equation (14).

Because the intensity and derivative histogram comparisons require encryption of the reference values with the appropriate pixel-level random numbers, Equation (7), the

[1] $(a \bmod b)^p = a^p \bmod b$

[2] $((a \bmod c) \times (b \bmod c)) \bmod c = (a \times b) \bmod c$

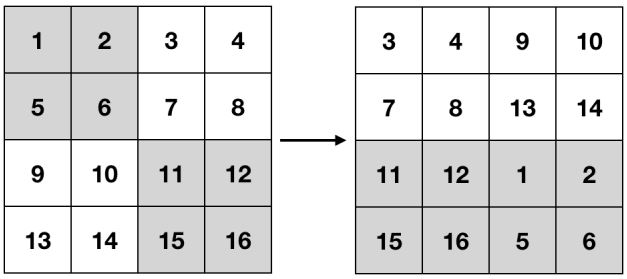

Figure 2. A 4×4 pixel image is partitioned into non-overlapping regions of size 2×2. These regions are scrambled while preserving the pixel-ordering within each region.

following information must be provided alongside the encrypted values: (1) each of B intensity bin centers encrypted with the random value r for each of 300×300 pixels; and (2) the value 0 encrypted with the "convolved" random value, for each of 300×300 pixels. With $B = 4$ as in Figure 1, the final payload of encrypted pixel and reference values is $540,000$ values or the equivalent of a 900×600 raw grayscale image.

3.3. Obfuscation

We encode and transmit the reference intensity values so that each pixel value can be placed in the appropriate intensity histogram bin. An obvious undesirable side-effect of this is that the server can reconstruct a $\log(B)$-bit grayscale version of the image, where B is the number of intensity bins used to create the image hash. To contend with this, we propose that within each 5×5 block (each of size 60×60 pixels), the client partitions each block into non-overlapping regions of size 2×2 pixels. These regions are then randomly scrambled within each block prior to transmission, Figure 2.

When the server receives this scrambled image, the intensity histogram is unaffected as is the partial derivative histograms computed at only the top-left pixel in each 2×2 region. At the same time, the server will be unable to reconstruct the original intensity image. The only cost of this scrambling is that the 2-D intensity/derivative hash, Figure 1, is computed from $1/4$ of the pixels. We have found, however, that this has no impact on the distinctiveness or robustness of the hash.

One could argue that this obfuscation should be enough to conceal the contents of the image. Given the sensitivity and even illegality (e.g., child sexual abuse imagery) of some material that may be hashed, it is critical to ensure that the image cannot be reconstructed. The added level of security offered by encrypting the scrambled pixels virtually guarantees that this would be impossible.

3.4. Validation and Robustness

We constructed a dataset of approximately $15,000$ images of randomly downloaded flickr images. Using the plaintext image hashing described above, we hashed all of these images and computed a pairwise comparison of all images to remove any duplicates. Any image pairs with an L1-norm below a specified threshold were manually reviewed to determine if they were duplicates. One of the pair of any confirmed duplicates was removed from our dataset yielding a final image count of $14,685$.

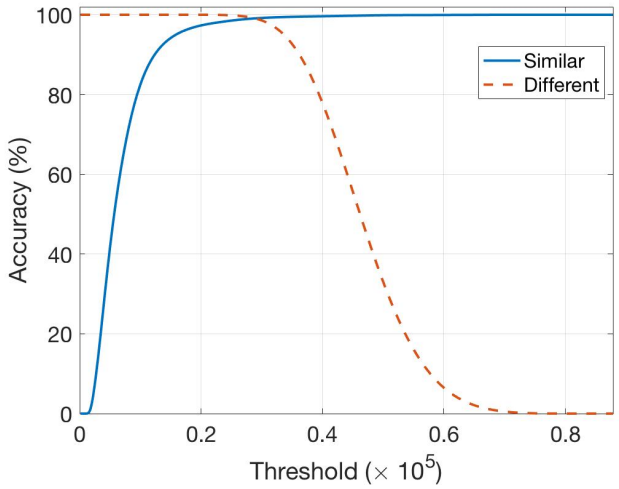

Figure 3. Accuracy of classifying similar (solid-blue) and different (dashed-red) images as a function of the threshold on L1-norm between two hashes. The cross-over point is 99.1%. See also Figure 4.

We validated the robust hashing by confirming that the plaintext and ciphertext version as extracted from all $14,685 \times 25 = 367,125$, 30×30 pixel blocks were identical.

In order to determine the robustness of our hash to common image manipulations, we generated $7,500$ variations of each image with varying and random amounts of gamma correction $I^{\gamma}(x, y)$ with $\gamma \in [0.5, 2]$, additive Gaussian noise with SNR in the range $[15, 60]$ (db), JPEG compression with quality in the range $[70, 100]$ $(\%)$, and scaling by an amount in the range $[0.25, 1]$. These manipulations were applied to the image prior to the pre-processing stage of converting to grayscale, down-sizing, and histogram equalizing. We computed the L1-norm between the original image and each of its variants, yielding a total of over 110 million ("similar") comparisons. We also computed the L1-norm between the hash of each of the original images and the remaining $14685-1$ images, yielding a total of over 107 million ("different") comparisons.

A robust hash should have relatively small "similar" distances and large "different" distances. The receiver operating characteristic (ROC) curve in Figure 3 reports the accuracy of correctly classifying "similar" (solid-blue) and "different" (dashed-red) images as a function of the L1-norm

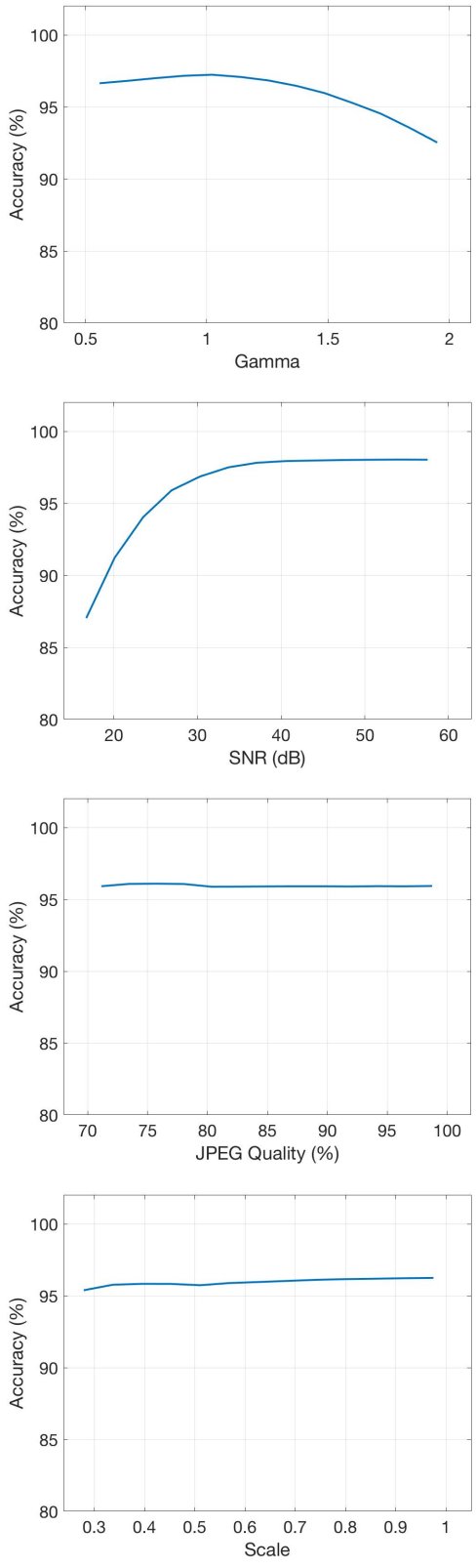

Figure 4. Accuracy of classifying similar images as a function of gamma correction, additive noise, JPEG quality, and scaling, for a fixed false alarm rate of 0.001%. See also Figure 3.

threshold. The cross-over point of these curves is at 99.1% accuracy. With a false positive rate (incorrectly identifying different images as the same) of 1%, 0.1%, 0.01%, and 0.001% the true positive rate (correctly identifying similar images as similar) is 99.1%, 98.4%, 97.4%, and 95.9%. At a false positive rate of 1 in 10 million, the true positive rate remains relatively high at 93.5%.

Shown in Figure 4, from top to bottom, is the true positive rate (with a fixed false positive rate of 0.001%) as a function of the amount of gamma correction, additive noise, JPEG quality, and scaling. Each data point in each panel corresponds to all images with the specified distortion and integrated across all other distortions. For example, the data point with an SNR of 30db corresponds to all images with this much noise and a random and varying amount of gamma correction, JPEG compression, and scaling. From top to bottom, we see that: (1) the sensitivity to gamma correction is asymmetric with slightly lower robustness to gamma values greater than 1.0; (2) the hash is robust to SNR larger than 25 db; and (3) the hash is robust to a range of compression qualities and scalings.

3.5. Efficiency

Because of the pre-processing step, the run-time efficiency will depend on the original image resolution. Nevertheless, averaged over the $15K$ images in our dataset, the average run-time of extracting our robust hash in plaintext is 0.03 seconds. In ciphertext, the average run-time is 10.2 seconds. The plaintext robust hash is implemented in MatLab and the ciphertext version is implemented in Java. All run-times are reported for a laptop running macOS with an Intel Core i5, 2.3 GHz.

4. Discussion

We believe that, although not without its challenges, a balance can be found between privacy and security. We believe that some material is so horrific or dangerous that all effort should be made to eliminate it from our online communities.

The type of privacy-preserving robust image matching described here finds just this balance. This approach allows users to share encrypted images without fear of corporate or government oversight while at the same time allowing online platforms the ability to eliminate the transmission of previously identified illegal or dangerous content.

Although we have only described an image matching technique, this basic approach extends to video, albeit with higher bandwidth and computational requirements. We are currently refining this approach.

Acknowledgment

This research was developed with funding from Google, Microsoft Corporation, and the Defense Advanced Research Projects Agency (DARPA FA8750-16-C-0166). The views, opinions, and findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] Yu Bai, Li Zhuo, Bo Cheng, and Yuan Fan Peng. Surf feature extraction in encrypted domain. In *IEEE International Conference on Multimedia and Expo*, pages 1–6, 2014. 2

[2] Reda Bellafqira, Gouenou Coatrieux, Dalel Bouslimi, and Gwénolé Quellec. Content-based image retrieval in homomorphic encryption domain. In *Engineering in Medicine and Biology Society, 2015 37th Annual International Conference of the IEEE*, pages 2944–2947, 2015. 2

[3] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 235–253, 2009. 2

[4] Hany Farid and Eero Simoncelli. Differentiation of discrete multi-dimensional signals. *IEEE Transactions on Image Processing*, 13(4):496–508, 2004. 3

[5] Chao-Yung Hsu, Chun-Shien Lu, and Soo-Chang Pei. Image feature extraction in encrypted domain with privacy-preserving SIFT. *IEEE Transactions on Image Processing*, 21(11):4593–4607, 2012. 2

[6] Mike Isaac. Facebook's Mark Zuckerberg says he'll shift focus to users' privacy. March 6, 2019. 1

[7] Katie Kim and Lisa Capitanini. Facebook's new program fighting 'revenge porn' stirs controversy. June 27, 2018. 1

[8] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern recognition*, 40(1):262–282, 2007. 1

[9] Wenjun Lu, Ashwin Swaminathan, Avinash L Varna, and Min Wu. Enabling search over encrypted multimedia databases. In *Media Forensics and Security*, volume 7254, page 725418, 2009. 2

[10] Jeff Meisner. Facebook to use Microsoft's PhotoDNA technology to combat child exploitation. May 19, 2011. 1

[11] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999. 2

[12] Zhan Qin, Jingbo Yan, Kui Ren, Chang Wen Chen, and Cong Wang. Towards efficient privacy-preserving image feature extraction in cloud computing. In *22nd ACM International Conference on Multimedia*, pages 497–506, 2014. 2

[13] Qin, Zhan and Yan, Jingbo and Ren, Kui and Chen, Chang Wen and Wang, Cong. Secsift: Secure image SIFT feature extraction in cloud computing. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 12(4s), 2016. 2

[14] Yogachandran Rahulamathavan, Raphael C-W Phan, Jonathon A Chambers, and David J Parish. Facial expression recognition in the encrypted domain based on local fisher discriminant analysis. *IEEE Transactions on Affective Computing*, 4(1):83–92, 2013. 2

[15] Matthias Schneider and Thomas Schneider. Notes on non-interactive secure comparison in image feature extraction in the encrypted domain with privacy-preserving SIFT. In *2nd ACM workshop on Information hiding and multimedia security*, pages 135–140, 2014. 2

[16] Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):1349–1380, 2000. 1

[17] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):355–368, 2010. 2

[18] Q. Wang, S. Hu, K. Ren, J. Wang, Z. Wang, and M. Du. Catch me in the dark: Effective privacy-preserving outsourcing of feature extractions over image data. In *35th Annual IEEE International Conference on Computer Communications*, pages 1–9, 2016. 2

[19] Zhihua Xia, Xinhui Wang, Liangao Zhang, Zhan Qin, Xingming Sun, and Kui Ren. A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing. *IEEE Transactions on Information Forensics and Security*, 11(11):2594–2608, 2016. 2