ROBUST IMAGE HASHING

R. Venkatesan[1], *S.-M. Koon*[2], *M. H. Jakubowski*[1], *and P. Moulin*[2]

[1]Cryptography Group, Microsoft Research, 1 Microsoft Way, Redmond, WA 98052-6399
[2] Univ. of Illinois, Beckman Inst. & ECE Dept., 405 N. Mathews Ave., Urbana, IL 61801
Email: {*venkie, mariuszj*}*@microsoft.com,* {*skoon, moulin*}*@ifp.uiuc.edu*

ABSTRACT

The proliferation of digital images creates problems for managing large image databases, indexing individual images, and protecting intellectual property. This paper introduces a novel image indexing technique that may be called an image hash function. The algorithm uses randomized signal processing strategies for a non-reversible compression of images into random binary strings, and is shown to be robust against image changes due to compression, geometric distortions, and other attacks.

This algorithm brings to images a direct analog of Message Authentication Codes (MACs) from cryptography, in which a main goal is to make hash values on a set of distinct inputs pairwise independent. This minimizes the probability that two hash values collide, even when inputs are generated by an adversary.

1. INTRODUCTION

Due to the popularity of digital technology, more and more digital images are being created and stored every day. This introduces a problem for managing image databases. One cannot determine if an image already exists in a database without exhaustively searching through all the entries. Further complication arises from the fact that two images appearing identical to the human eye may have distinct digital representations, making it difficult to compare a pair of images: e.g., an image and its watermarked version, a watermarked image and a copy attacked by software to remove watermarks, an image and its enhanced version using commercial software, an image stored using distinct transforms, and an image and its compressed version. Given a suitable algorithm to generate image identifiers, or an *image hash function*, one may use standard algorithms that search and sort n binary strings in time proportional to $\log n$ rather than to n [4].

Other applications of image hashing lie in the area of image authentication and watermarking [3, 8, 5, 1]. Given a suitable keyed image hash function $h(K, I)$ that authenticates an image I using a secret key K, one can proceed as in standard cryptographic methods [6] by appending a tag of the form $Encrypt(h(K, I))$. In watermarking, hash functions help to secure against some known attacks that use many images watermarked with the same random secret key: for an image I, use the watermarking key K_I derived from a master key K by computing $K_I = MD5(MK, hash(K, I))$.

Standard hash functions such as MD5 and SHA-1 cannot be used on images, since their output would be dramatically changed even if one bit of input changed [6, 8, 1]. However, digital images commonly undergo various manipulations, such as compression, enhancement, scaling, and cropping, just to name a few. Instead, the hash function should take into account the changes in the visual domain and produce hash values based on an image's visual appearance. In other words, the hash values obtained by applying the hash function to visually similar images should remain the same. For visually distinct images, the hash function should produce different hash values.

Here we introduce a novel algorithm that utilizes a wavelet representation of images and new randomized processing strategies for hashing. Constructions based on error-correcting codes serve to reduce the length of the hash value while keeping collision probability low. The statistical properties of the hash values on distinct images are similar to the properties used in cryptography to minimize collision probabilities [6]. An extension of [9, 10], this algorithm was extensively tested and found to be robust against common image processing and malicious attacks.

2. IMAGE HASHING ALGORITHM

A significant principle behind our design was to allow for the fact that image pixels are strongly correlated and may in fact be modified by an adversary. The hash function takes two inputs, an image I and a *secret* key K, to produce a hash value $h = H(I, K)$. The key K functions as a seed to the pseudorandom number generator used at various stages of the hashing algorithm. Given the same image, if a different key is used, the resulting hash will be statistically independent and thus completely different. The key is kept secret, and the hash value of a given image cannot be computed or verified by an unauthorized party.

We now outline the main steps of the algorithm, many of which are easily generalized. Our algorithm will use *coin flips* in an essential way, and may use a cryptographically

strong pseudorandom generator $G(s)$ with a secret random seed s.

First, a wavelet decomposition of the image is computed, and each subband is randomly tiled into small rectangles. Each rectangle's statistics (either averages or variances, depending on the subband) are calculated and quantized using a *randomized rounding* (i.e., probabilistic quantization) method that is crucial to ensure the the resulting hash values as binary strings are random. These quantized statistics can be thought of as elementary image features, many of which are relatively robust against attacks. The quantized statistics are then input to the decoding stage of a suitably chosen error-correcting code to generate the final hash value. This step helps ensure that hash value remains the same if the decoder input remains within a suitably defined decoding region.

More specifically, the steps of the algorithm are as follows:

Step 1:*Random tiling transform and statistics calculation*: A random tiling of each subband of the image is generated as illustrated in Fig. 1. This can be thought of as a randomized analog of methods for computing integrals through approximation. The number of samples is large enough to ensure that there is enough information about the original image while purposely introducing irreversibility and some unpredictability. Averages of coefficients in the rectangles are calculated in the coarse subband, and variances are calculated in the other subbands. This results in a length–l image–statistics vector $m = \sigma(I, K)$, where I is the image, K is the key that determines the tiling, and σ is the feature (statistics) mapping operation.

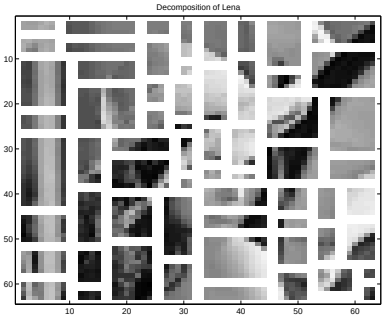

Fig. 1. Random tiling of Lena's coarse subband (in a three-level wavelet decomposition using Haar wavelet).

Step 2: *Randomized rounding*: This technique is used in theoretical computer science in design of algorithms and analysis of their behaviour on inputs that are chosen by an *adversary* [7]. We are not aware of any other use of this technique where it is necessary for both theoretical *and* practical reasons. The statistics vector m calculated from Step 1 is quantized using a randomized quantizer Q. This gives the rounded statistic increased robustness against attacks. We recall that to minimize the probability that any two distinct inputs collide, each output bit of a hash function must be random, and hash values must be pairwise independent; we treat inputs as produced by unknown sources, and randomized rounding is the crucial source of randomness (or entropy) in the hash function's output. The function Q uses the pseudorandom seed K from Step 1 to generate the probabilistic distribution used for quantization. Step 2 produces a length–l vector

$$x = Q(m, K) \in \{0, 1, \cdots, 7\}^l$$

whose entries are 3–bit data.

Step 3: The quantized statistics vector x is decoded by a first-order Reed-Muller error-correcting decoder D to produce the length-n binary hash value $h = D(x) \in \{0, 1\}^n$ [2]. The metric used for decoding is the exponential pseudo-norm, which approximates the sup norm and confers additional robustness to the decoding scheme. The decoding stage of the algorithm serves several important purposes:

- It cancels small perturbations (that remain in decoding spheres) in quantized statistics of visually similar images, yielding the same hash value.
- On distinct images, it results in statistically uncorrelated hash values.
- It condenses the randomness introduced by randomized rounding into fewer bits to make the outputs random.

Step 4 Another decoding stage of a linear code with random parameters maps the current intermediate hash value into an even shorter string.

The above steps may be repeated with different random keys and different domains (e.g., spatial and DCT) to yield many independent random strings, and the algorithm can check if a majority of the hash values agree. This increases the strength of the hash function significantly. For example, even given the hash value, finding an image that yields the same hash value on all (or most) of the invocations or repetitions involves a significant computational load in the final step. A formal analysis of the steps involved in the hash computation will appear elsewhere.

3. EXPERIMENTAL RESULTS

Extensive tests on over a hundred images were performed, and no failure cases were noted. We present some details

next. The normalized Hamming distance between intermediate hash values before and after image distortions is used to measure the algorithm's performance. We refer to this distance as the *equality percentage (EP)*, or the percentage of equal vector components in the intermediate hash vectors used in the above algorithm. This measure satisfies two key properties: first, EP is high (near 100 %) for visually similar images, because the hash values are similar; second, EP is low for distinct images, because the hash values are quite different. In fact, the expected value of of EP is 50% for two vectors independently drawn from a uniform random distribution on $\{0,1\}^n$.

TEST 1: RESISTANCE TO ATTACKS: A variety of attacks, many from the well known *StirMark* software package, were performed on about 100 images, including the traditional Lena and Baboon. The hash function was robust against a range of attacks:

- Rotation by up to 2 degrees;
- Cropping up to 10% of image area;
- Scaling by up to 10%;
- Random deletion of up to 5 lines;
- Shifting by up to 5%;
- JPEG compression using quality factor of 10% or higher;
- 4×4 median filtering.

The EP's remained close to 100%, and the final hash values were invariant without exception.

TEST 2: PROBABILITY OF COLLISION FOR UNRELATED IMAGES: We used Baboon as our original image and compared its hash value with that of 150 other images. The resulting EPs were all in the range $[45\%, 65\%]$. The final hash values were distinct.

TEST 3: KEY (PSEUDORANDOM SEED) DEPENDENCY OF THE ALGORITHM: Many steps of the algorithm, such as the geometric decomposition of the image and the rounding of the coefficients, are controlled by a random seed. Distinct seeds for the same image yielded uncorrelated random hash values as though hashing were performed on distinct images, while the EP was in the range $[50\%, 65\%]$. In terms of security, such a strong dependency on the key is significant: once a key is broken, the user can simply change it, like a password. No systemic changes are necessary after attacks or improvements to the algorithm, aside from recomputing hash values of images in a database.

4. CONCLUSION

We presented an image-hashing algorithm that converts images into short, robust bit strings. Using this algorithm, we can compare two images by checking two bit strings for exact equality, rather than attempting the far more involved problem of comparing malleable, "fuzzy" image data. In our experiments, image hashes were robust to various attacks, including both common image processing and malicious distortions. The hashing algorithm combines various ideas from the fields of signal processing, error-correcting codes, and cryptography.

Hashing can be used in place of watermarking, with the benefit that nothing is added to images. From a game-theoretic point of view, hashing may be stronger than watermarking, since hashing algorithms can be adapted to attacks after these occur and without the need to modify and re-release deployed images; with watermarking, on the other hand, the first move belongs to the embedding algorithm, which commits to a particular watermark that the attacker can subsequently try to remove. A formal analysis of image hashing, as well as hashing of other media, will be covered in future work.

5. REFERENCES

[1] S. Bhattacharjee and M. Kutter, "Compression tolerant image authentication," *Proc. ICIP-'98*, vol. 1, pp. 435-439, Sep. 1998.

[2] R. E. Blahut, *Theory and Practice of Error-Correcting Codes*, Addison Wesley, 1994.

[3] G. L. Friedman, "The trustworthy digital camera: restoring credibility to the photographic image," *IEEE Transactions on Consumer Electronics*, vol. 39 no.4, pp. 905-910, Nov. 1993.

[4] D. E. Knuth, *The Art of Computer Programming, Vol I, II*, Addison Wesley, 1998.

[5] C.-Y. Lin and S.-F. Chang, "Generating robust digital signature for image/video authentication," *Proc. Multimedia and Security Workshop at ACM Multimedia '98*, Bristol, U.K., Sep. 1998.

[6] A. Menezes, V. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1998.

[7] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1996.

[8] M. Schneider and S.-F. Chang, "A robust content based digital signature for image authentication," *Proc. ICIP-96*, vol. 3, pp. 227-230, Sep. 1996.

[9] R. Venkatesan and M. H. Jakubowski, "Image hashing," *DIMACS Conf. on Intellectual Property Protection*, Piscataway, NJ (USA), Apr. 2000.

[10] R. Venkatesan and S.-M. Koon, "Robust image hashing into binary strings," manuscript, 1999.