Proceedings of the 5th International Symposium on Communications, Control and Signal Processing, ISCCSP 2012, Rome, Italy, 2-4 May 2012

FAST CAMERA FINGERPRINT SEARCH ALGORITHM FOR SOURCE CAMERA IDENTIFICATION

Yongjian Hu[1,2], *Chang-Tsun Li*[1], *Zhimao Lai*[2], *and Shangfan Zhang*[2]

[1]Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK
[2]School of Electronic and Information Engineering,
South China University of Technology, Guangzhou 510641, P.R.China
{Yongjian.Hu, ctli}@dcs.warwick.ac.uk

ABSTRACT

To determine the source camera of a query image, the fingerprint from the query image needs to be compared with the fingerprints in the reference fingerprint database. Traditionally, the query fingerprint is compared with these reference fingerprints one by one in sequence. For a large database, however, such a brute-force search is inefficient and time-consuming. How to accurately locate the correct fingerprint in the reference fingerprint database is thus becoming a crucial problem for commercial applications of source camera identification. So far there have been few studies in literature addressing this problem. In this work, we propose a new solution to fast fingerprint search. We first store the information of the reference fingerprint digests in the separate-chaining hash table, and then introduce a new rule to select the candidate reference fingerprint digests before performing the correlation. The selection rule is incarnated with the search priority vector. Experimental results have shown that the proposed algorithm outperforms current algorithms.

Index Terms**—**Camera identification, correlation-based detection, fingerprint digest, separate-chaining hash table, priority vector

1. INTRODUCTION

In the last few years, digital image forensics, which is able to tell the origin and integrity of digital images, has attracted much attention. There are a number of forensic methods proposed for establishing the relationship between digital images and the imaging devices used to generate these images (e.g., [1]-[11]). One promising method is based on camera fingerprints. The principle is similar to gun identification based on the scratches left on the bullets fired by the gun. A digital image also has the "fingerprint/bullet scratch" of the imaging device hidden within its pixels [1]. For finding out the origin of a photo, what one needs to do is to find a match of the fingerprint extracted from the photo with the fingerprint in the reference fingerprint database.

All digital impressions or marks of a camera might be used as camera fingerprints. However, some of them are not stable and cannot ensure reliable detections. Lukas *et al*. [2] proposed to use the Photo-Response Non-Uniformity (PRNU) noise as device fingerprints for camera identification. The PRNU can remain stable under different ambient environments. Based on this pioneering work, different algorithms were proposed, some for improving the extraction process of fingerprints (e.g., [3]) while others for improving the quality of extracted fingerprints (e.g., [9]-[11]). In the meantime, topics related to the PRNU-based camera identification were also raised, for example, the security of detection was addressed in [12], the reliability of PRNU was discussed in [5], and the efficiency of locating a query camera in a large reference fingerprint database is revealed in [6].

Fast algorithm for camera identification is quite significant for real-world applications. In this paper, we propose a new fast algorithm for quickly determining whether a query image is taken by a specific camera in the known reference fingerprint database. The rest of the paper is organized as follows. We briefly review the related work in Section 2. In Section 3, we describe our fast search algorithm in detail. In Section 4, some experiments are given to demonstrate the effectiveness of our algorithm. We conclude our method in Section 5.

2. REVIEW OF RELATED WORK

The prototype proposed by Lukas *et al*. [1] for source camera identification is shown in Fig. 1. The extracted fingerprint/residual image/noise residual looks like a two-dimensional spread-spectrum watermark. So the framework for watermark detection is applicable to source camera identification. The fingerprint from a query/test image is compared with the camera reference fingerprint. This camera identification process can be formulated as a binary hypothesis testing problem. H_1 is true when the correlation is greater than the predefined threshold *t*, meaning the query image is taken by the reference camera; otherwise, H_0 is true.

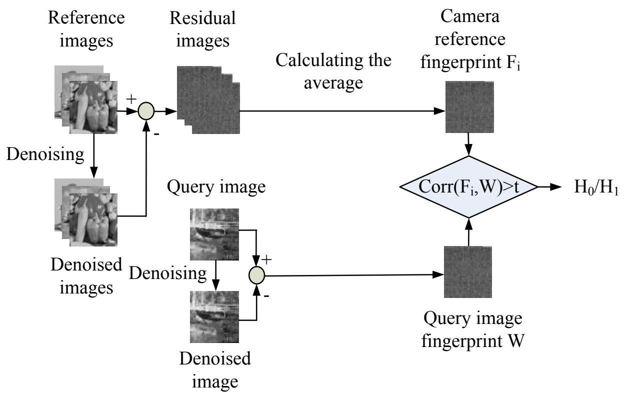

Fig. 1. Prototype of source camera identification.

To test the effectiveness of PRNU on a large collection of images, Goljan *et al*. [5] constructed a database of over a million images taken by 6896 individual cameras which cover 150 models. Their work verifies the reliability of the PRNU as the role of camera fingerprint; on the other hand, it also shows that such forensic investigation often needs a large amount of computation.

The same authors proposed a fast fingerprint search algorithm, ARMS (the Approximate Rank Matching Search algorithm), to reduce the computational complexity and speed up the detection process [6]. ARMS is based on *fingerprint digests*. Instead of directly computing the correlation between the full-length query fingerprint and the full-length reference fingerprint, a subset of pixels of the query and reference fingerprints, i.e., the two digests, were employed. In principle, ARMS was inspired by the Spearman rank correlation. Based on the most influential indices/elements between the test digest and the reference digests, it assigns the priority to the reference digests for correlation comparison. The accumulated evidence was introduced for further improving the selection rule of ARMS. In essence, ARMS only uses the information of partial elements of the query digest and the reference digests. The reader is referred to [6] for more details.

3. OUR FAST SEARCH ALGORITHM

In this work, we propose a fast search algorithm that can use more comprehensive information from both the query fingerprint and the reference fingerprints in the database (or called the database fingerprints). Let $\mathbf{F}_i$ and $\mathbf{W}$ denote the *i*th database fingerprint and the query fingerprint, respectively. The manner of calculating $\mathbf{F}_i$ and $\mathbf{W}$ has been shown in Fig. 1. $\widetilde{\mathbf{F}}_i$ and $\widetilde{\mathbf{W}}$ denote their corresponding digests, which consist of *m* largest magnitude components of $\mathbf{F}_i$ and $\mathbf{W}$, respectively. Note that we do not distinguish matrices, vectors and sequences since they can be easily converted from one form to another. Unless stated clearly, all operations between them are element-wise. $\mathbf{F}_i$ and $\mathbf{W}$ have the same size and both of them have *n* components/elements/pixels. Assume that there are N reference cameras. D denotes the reference fingerprint database. $D = \{\widetilde{\mathbf{F}}_i\}$, $i = 1,2,...,N$. For simplicity, we only use the fingerprint of one channel (e.g., green channel) image to explain our algorithm. It can be easily extended to the synthetic fingerprint constructed using information of all the three channel images.

Our fast search algorithm aims at finding the source camera of a query image in the reference fingerprint database. It includes four major parts: (i) obtaining the camera fingerprints from the reference images and the query image, and using the fingerprint digests, $\widetilde{\mathbf{F}}_i$, $i = 1,2,...,N$, to construct the reference fingerprint database, D ; (ii) generating a separate-chaining hash table, H , and storing the information of the reference fingerprint digests in it; (iii) building the search priority vector $\mathbf{R}_p$ after comparing $\widetilde{\mathbf{W}}$ with $\widetilde{\mathbf{F}}_i, i = 1,2,...,N$, in H ; (iv) computing the correlation between $\widetilde{\mathbf{W}}$ and the database fingerprint digests selected by $\mathbf{R}_p$, and making the final decision.

3.1. Building the Reference Fingerprint Database Using Camera Reference Fingerprint Digests

After extracting $\mathbf{F}_i$ and $\mathbf{W}$ in a way shown in Fig. 1, we perform the ZM (zero-meaning) operation [3] on them to suppress the non-unique artifacts (i.e., the noise components coming from cameras of the same make or model) [5]. The fingerprint digest, $\widetilde{\mathbf{F}}_i$, is simply constructed by using the first m largest magnitude elements of $\mathbf{F}_i$. Using all the fingerprint digests, we obtain $D = \{\widetilde{\mathbf{F}}_i\}, i = 1,2,...,N$.

3.2. Generating the Separate-Chaining Hash Table

The building of separate-chaining hash table is the important part of our fast algorithm. We need the hash table to store the information of D . Let $\mathbf{V}_{\mathbf{F}_i}$ be the set of *m* largest magnitude elements of $\mathbf{F}_i$ and let $\mathbf{L}_{\mathbf{F}_i}$ be the set of the locations of those elements in $\mathbf{F}_i$. Using a one-dimensional index $l[d]$, $d = 1,...,m$, $1 \le l[d] \le n$, we have $\mathbf{L}_{\mathbf{F}_i} = \{l[d]\}_{d=1}^{m}$, $\mathbf{V}_{\mathbf{F}_i} = \mathbf{F}_i[\mathbf{L}_{\mathbf{F}_i}]$. The steps of creating the separate-chaining hash table are described as follows:

Step 1: Create a separate-chaining hash table H with n list heads to save the camera fingerprint indices, $i, i \in \{1,\ldots,N\}$.

Step 2: For each fingerprint digest, insert the camera fingerprint index i to H according to both the location information and the value of each element. Specifically, if $\mathbf{V}_{\mathbf{F}_i}(d) \geq 0$, i is inserted into the $\mathbf{L}_{\mathbf{F}_i}[d]th$ list head of the H ; otherwise, $-i$ is inserted into the $\mathbf{L}_{\mathbf{F}_i}[d]th$ list head of the H .

Since $\widetilde{\mathbf{F}}_i$ has m elements, all the information of elements can be stored in H after m iterations. The reader is referred to [13] for more information of separate-chaining hash table.

3.3. Building the Search Priority Vector

The most important part of our fast search algorithm is to build the search priority vector $\mathbf{R}_p$. Before introducing $\mathbf{R}_p$, we give an observation in the following figure.

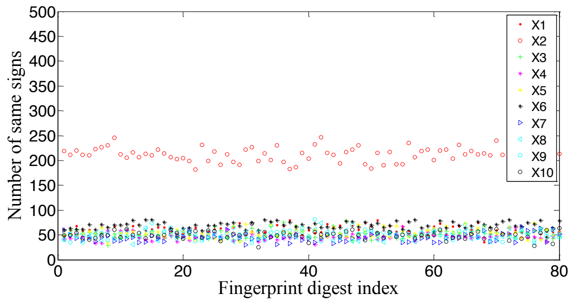

Fig. 2. The number of elements with the same sign between the query fingerprint digest and the reference digests from 10 cameras. The query fingerprint digests come from Nikon D100 (i.e., X2). The length of each fingerprint digest is 10000.

In Fig. 2, we compare the query fingerprint digest with the database fingerprint digests and count the number of the elements located in the same spatial locations that possesses the same sign. The 10 cameras involved, from X1 to X10, are Nikon CoolPix L3, Nikon D100, Canon A40, Canon A620, Canon A720, Canon EOX Kiss X2, Panasonic DMC-LX2, Sony DSC-T10, Fujifilm FinePix F200EXR, Nikon D90, respectively. The fingerprint digests come from all the 10 cameras. Each camera contributes 80 images. When computing the fingerprints, we only take a 1024×1024 image block from the center of each photo so that the fingerprints from the photos of different cameras have the same size. For the construction of fingerprint digests, we let *m*=10000. It can be seen that the number of elements with the same sign for Nikon D100 is around 200 while that number is around 50 for the other 9 cameras. We repeat the same experiment on the photos taken by other cameras and can observe the similar phenomena. This result inspires us to use the number of elements with the same sign as the measurement for fast fingerprint search. We build our search priority vector in the following way. Let $\mathbf{R} = [r_1, r_2, \ldots, r_N]$ to be the comparison results, where r_i refers to the number of elements with the same signs between $\widetilde{\mathbf{W}}$ and $\widetilde{\mathbf{F}}_i$. The steps of obtaining $\mathbf{R}$ are described as follows:

Step 1: Select the first m largest magnitude elements from $\mathbf{W}$ to construct $\widetilde{\mathbf{W}}$. Let $\mathbf{V}_{\mathbf{W}}$ denote the set of the values of the elements of $\widetilde{\mathbf{W}}$, and let $\mathbf{L}_{\mathbf{W}}$ denote the locations of the elements of $\widetilde{\mathbf{W}}$ in $\mathbf{W}$.

Step 2: Choose d *th* element of $\mathbf{V}_{\mathbf{W}}$ and traverse the list of the ($\mathbf{L}_{\mathbf{W}}[d]$ *th*) list head of H . If there is i and $\mathbf{V}_{\mathbf{W}}[d] \geq 0$, r_i is increased by 1; If there is $-i$ and $\mathbf{V}_{\mathbf{W}}[d] < 0$, r_i is also increased by 1; otherwise, keep r_i unchanged.

Let $d = 1, 2, \ldots, m$ and repeat Step 2 m times, and then $\mathbf{R}$ is obtained. We sort the elements of $\mathbf{R}$ in a descending order to obtain $\mathbf{R}_p$.

3.4. Performing the Correlation Detection

The first element of $\mathbf{R}_p$ has the largest value. Its subscript corresponds to the index of the best candidate database fingerprint digest that is thought to be correlated with the query digest. Likewise, the subscript of the second element of $\mathbf{R}_p$ corresponds to the second best candidate digest, and so on. In this way, all the database fingerprint digests in the database are sorted.

The correlation coefficient between a query fingerprint digest and the database fingerprint digest, ρ , can be calculated as follows:

$$\rho = corr(\mathbf{V}_{\mathbf{F}_i}, \mathbf{V}_{\mathbf{W}}[\mathbf{L}_{\mathbf{F}_i}]) = \frac{(\mathbf{V}_{\mathbf{F}_i} - \overline{\mathbf{V}}_{\mathbf{F}_i}) \bullet (\mathbf{V}_{\mathbf{W}}[\mathbf{L}_{\mathbf{F}_i}] - \overline{\mathbf{V}}_{\mathbf{W}}[\mathbf{L}_{\mathbf{F}_i}])}{\left\| \mathbf{V}_{\mathbf{F}_i} - \overline{\mathbf{V}}_{\mathbf{F}_i} \right\| \times \left\| \mathbf{V}_{\mathbf{W}}[\mathbf{L}_{\mathbf{F}_i}] - \overline{\mathbf{V}}_{\mathbf{W}}[\mathbf{L}_{\mathbf{F}_i}] \right\|} \quad (1)$$

where $\mathbf{X} \bullet \mathbf{Y} = \sum_{i=1}^{m} \mathbf{X}[i]\mathbf{Y}[j]$, $\|\mathbf{X}\| = \sqrt{\mathbf{X} \bullet \mathbf{X}}$ is the norm of $\mathbf{X}$, $\overline{\mathbf{X}}$ is the mean of $\mathbf{X}$.

The resulting correlation coefficient is compared with a predefined threshold t . Similar to watermark detection, this detection threshold can be determined using the Neymann-Pearson criterion. This work does not discuss how to choose

t in detail. The reader is referred to [7] for more information about the calculation of t.

We summarize the description of our algorithm in Fig. 3.

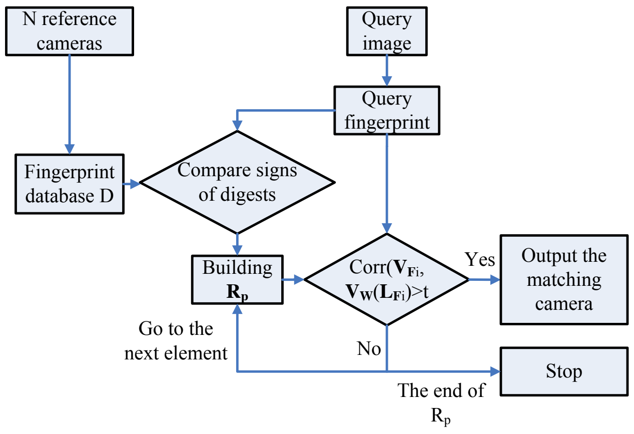

Fig. 3. The framework of our algorithm.

4. EXPERIMENT AND DISCUSSION

To evaluate the proposed algorithm, we conduct experiments on natural photos taken by 50 digital cameras of different models and brands. We use 90-100 images of each camera as the reference images to calculate the camera fingerprint and 60-80 smooth images (e.g., blue sky images) of each camera to constitute the query image subsets. All the 1024×1024 images are taken from the photos in a way described in subsection 3.3. Therefore, $D=\{\widetilde{\mathbf{F}}_i\}$, $i=1,...,50$. Since m=10000, each fingerprint digest only takes about 1% pixels of $\mathbf{F}$. Apparently, the computational load for calculating the correlation is lighter than that of the full-length fingerprints correlation. The detection threshold is set to 0.03.

4.1. Experiments on 50 Cameras

The query image set consists of 3414 images from 50 cameras. We run our fast search algorithm in D. After 8084 rounds of search, the source cameras for the entire query image set are identified; specifically, 2358 images are identified in the first round, 311 images in the second round and 209 images in the third rounds. That is, 84.3% (=(2358+311+209)/3414) of the images are matched to their source cameras in the first three rounds. In the worse case, there is an image that takes 42 rounds of search. The average search round for a query image is 8084/3414≈3. The complete search results are shown in Fig. 4.

For comparison, we also test ARMS (with parameters w=10000, $t_{cand}=0.2\sqrt{w}$) [6] under the same situation. It needs 28763 rounds in total. In particular, 1401 images are matched in the first round, 328 images in the second round and 206 images in the third rounds. So in the first three rounds, 56.7% (=(1401+328+206)/3414) of the images are matched to their source cameras. Apparently, its search accuracy is inferior to our algorithm.

When the brute-force search scheme is used, the average search round for each query image is 50/2=25. So it needs much more search rounds than our algorithm. In fact, it is the lowest efficient algorithm among the three.

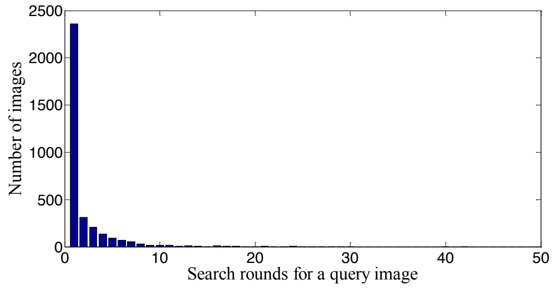

Fig. 4. Search results for 3414 query images in the database containing 50 camera fingerprint digests.

4.2. Experiments in a Larger Reference Fingerprint Database

To collect thousands of different cameras for experiments is not easy due to the price of cameras. Although it is possible to get some images from the internet, copyright is an issue that needs to be taken seriously. In order to simulate a large database that contains as many as possible different physical cameras, we propose a way for the simulation of camera fingerprints. Let us first investigate the distribution of PRNU fingerprints. We find that they approximately follow the generalized Gaussian distribution (GGD). The function of GGD is described in [14] as

$$f(x;\alpha,\beta,\mu)=\frac{1}{2\alpha\Gamma(1/\beta)}e^{-(\frac{|x-\mu|}{\alpha})^{\beta}} \qquad (2)$$

where α is the scaling factor, β the shape parameter and μ the mean. We can use the method in [14] to estimate the GGD parameters. In Fig. 5, we give an example to show the distribution of camera fingerprints and the GGD fitting situation. In general, we find that over 99% of fingerprint values fall into the interval [-2, 2]. So we propose to create computer-generated (CG) random camera fingerprints in this interval using GGD functions by varying α , β , and μ . Those GGD parameters can be easily obtained using a random number generator.

We create 4000 CG camera fingerprints and combine them with 50 real camera fingerprints to build a larger database. Afterwards, we repeat the same experiment as described in Subsection 4.1. For 3414 query images, the total rounds of search for our algorithm are 25504. So the average search round for a query image is 25504/3414≈8. The results are shown in Fig. 6.

In contrast, ARMS needs 1024171 rounds, meaning (1024171/3414)≈300 rounds for the search of every query image. As for the brute-force search algorithm, the average search round for each query image is 2025. Still, our algorithm is superior to both ARMS and the brute force algorithm in terms of search accuracy.

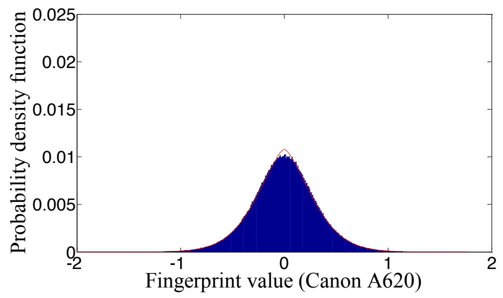

Fig. 5. Sample distribution of camera fingerprints and its GGD fitting.

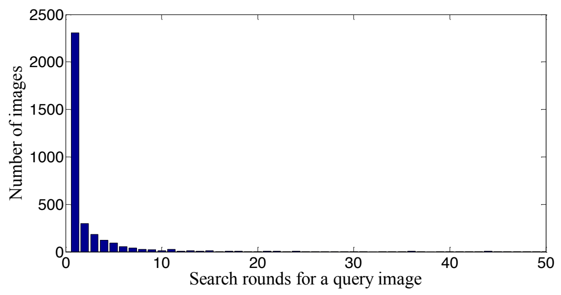

Fig. 6. Search results for 3414 query images in the database containing 4050 camera fingerprint digests. Note that only the first 50 of 4050 bins are shown.

5. CONCLUSION

We have proposed a fast camera identification algorithm. Our main contributions include the introduction of the separate-chaining hash table for the storage of the information of the reference fingerprint database and the invention of the search priority vector for the selection of candidate database fingerprints. Essentially, the building of our search priority vector depends on the information from every element of both the query and the database fingerprint digests. Therefore, our search priority vector can more accurately reflect the relationship between the query and the database fingerprint digests and thus leads to higher accuracy of finding the matching database digest. The unique data structure of the separate-chaining hash table facilitates the building of our search priority vector. Experimental results have proved that our algorithm has better performance than ARMS and the brute-force algorithms in terms of search accuracy.

Acknowledgement. This work was partially supported by the EU FP7 Digital Image and Video Forensics project (Grant Agreement No.251677) and NSF of China 60772115.

6. REFERENCES

[1] J. Lukas, J. Fridrich, and M. Goljan, "Digital "bullet scratches" for images," in *Proc. of IEEE Int. Conf. Image Processing*, Genova, Sep. 2005, vol. III, pp. 65–68.

[2] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Trans. Inf. Forensics Security*, vol. 1, no. 2, pp. 205–214, June 2006.

[3] M. Chen, J. Fridrich, M. Goljan, and J. Lukas, "Determining image origin and integrity using sensor noise," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 1, pp. 74–90, Mar. 2008.

[4] M. Goljan and J. Fridrich, "Camera identification from cropped and scaled images," in P*roc. SPIE Electronic Imaging, Forensics, Security, Steganography, and Watermarking of Multimedia Contents X*, San Jose, CA, Jan. 28–30, 2008,vol. 6819, pp. 0E-1–0E-13.

[5] M. Goljan, J. Fridrich and T. Filler, "Large scale test of sensor fingerprint camera identification," in *Proc. SPIE, vol.7254, Electronic Imaging, Media Forensics and Security XI*, San Jose, CA, pp. 0I 1-0I 12, January 18-22, 2009.

[6] M. Goljan, J. Fridrich and T. Filler, "Managing a large database of camera fingerprints," in *Proc. SPIE, 7541, Electronic Imaging, Media Forensics and Security XII*, San Jose, CA, pp. 08-01 - 08-12, January 17, 2010.

[7] Y. Hu, B. Yu, and C. Jian, "Source camera identification using large components of sensor pattern noise," *Proc. 2nd International Conference on Computer Science and its Applications*, pp. 1–5, South Korea, Dec. 2009.

[8] C.-T. Li and Y. Li, "Color-Decoupled Photo Response Non-Uniformity for Digital Image Forensics," *IEEE Trans. on Circuits and Systems for Video Technology* (Accepted).

[9] C.-T. Li, "Source camera identification using enhanced sensor pattern noise," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 2, Jun. 2010.

[10] X. Kang, Y. Li, Z. Qu and J. Huang, "Enhancing Source Camera Identification Performance with a Camera Reference Phase Sensor Pattern Noise," *IEEE Trans. Inf. Forensics Security*. (accepted)

[11] R. Caldelli, I. Amerini, and F. Picchioni, "Distinguishing between camera and scanned images by means of frequency analysis," *International Journal of Digital Crime Forensics*, vol. 2, no. 1, Jan./Mar. 2010.

[12] T. Gloe, M. Kirchner, A. Winkler, and R. Bohme, "Can we trust digital image forensics?," in *15th International Conference on Multimedia*, 2007, pp. 78–86.

[13] F. Pfenning, "Lecture notes on hash tables," [Online]. Available: www.cs.cmu.edu/~wlovas/15122-r11/lectures/13-hashtables.pdf .

[14] J. A. Dominguez-Molina, G. Gonzalez-Farias, and R. M. Rodriguez-Dagnino, "A practical procedure to estimate the shape parameter in the generalized Gaussian distribution," [Online]. Available: http://www.cimat.mx/reportes/enlinea/I-01-18_eng.pdf .