Managing a Large Database of Camera Fingerprints

Miroslav Goljan, Jessica Fridrich, Tomáš Filler
Department of Electrical and Computer Engineering
State University of New York, Binghamton, New York, 13902-6000
{mgoljan,fridrich,tfiller}@binghamton.edu

ABSTRACT

Sensor fingerprint is a unique noise-like pattern caused by slightly varying pixel dimensions and inhomogeneity of the silicon wafer from which the sensor is made. The fingerprint can be used to prove that an image came from a specific digital camera. The presence of a camera fingerprint in an image is usually established using a detector that evaluates cross-correlation between the fingerprint and image noise. The complexity of the detector is thus proportional to the number of pixels in the image. Although computing the detector statistic for a few megapixel image takes several seconds on a single-processor PC, the processing time becomes impractically large if a sizeable database of camera fingerprints needs to be searched through. In this paper, we present a fast searching algorithm that utilizes special "fingerprint digests" and sparse data structures to address several tasks that forensic analysts will find useful when deploying camera identification from fingerprints in practice. In particular, we develop fast algorithms for finding if a given fingerprint already resides in the database and for determining whether a given image was taken by a camera whose fingerprint is in the database.

Keywords: Camera identification, image forensics, sensor fingerprint, fast search, image database.

1. INTRODUCTION

Systematic artifacts of imaging sensors can be used as their unique identifiers (fingerprints) for various forensic tasks, such as verifying digital media origin or their integrity. The basis of the recently proposed camera fingerprint is the Photo-Response Non-Uniformity (PRNU). The PRNU signal is caused by slightly varying pixel dimensions and inhomogeneity of the silicon wafer from which the sensor is made. Each image/video produced by the same sensor is overlaid with the same noise-like pattern, which is modulated by the scene light intensity. To provide evidence that a given image was taken by a certain camera, the image is shown to contain the camera fingerprint. References [1–5] describe the camera identification process in more detail.

Anticipating future use of camera identification from sensor fingerprint by law enforcement and government, we visualize a situation when a large database of N fingerprints is queried whether it already contains a given fingerprint or whether it contains a fingerprint of the camera that took a given query image. The search is a multiple-hypothesis testing problem with a correlation detector applied to each fingerprint from the database. The complexity of this direct brute-force search is $O(nN)$, where n is the number of pixels in each image/fingerprint. Considering the fact that most cameras today have $n > 10^6$ pixels and that there could be tens of thousands of fingerprints in the database ($N > 10^4$), the cost of the direct search could become prohibitively large. To provide forensic investigators with operability, flexibility, and ease of use, we develop fast algorithms for finding a given fingerprint in a database of fingerprints and for determining whether a given image was taken a by a camera whose fingerprint resides in the database. In the first case, we match a given fingerprint to other fingerprints, while the second task consists of matching a noise residual to fingerprints. Note that there is not really a fundamental difference between these two tasks as the noise residual from a single image can be considered as a poor-quality fingerprint.

Because the correlation detector has been established as an approximate version of the generalized likelihood ratio test for the problem of detecting the presence of the fingerprint,[1] reducing the computation time by further simplifying the detector will likely increase the probability of an erroneous decision. Thus, this direction is not pursued in this paper. Instead, we focus on finding a reasonable trade-off between the probability of identification (when the camera fingerprint is contained in the database) and the speed/memory requirements of the search. Interestingly, it is not

necessary to have a low false-alarm rate because the correlation detector can be run as a double check on all fingerprints identified as candidates by the search. Too many candidate fingerprints, however, would slow down the search.

In the next section, we introduce the notation, terminology, and some background material pertaining to camera identification using sensor noise. Section 3 contains a discussion of several possible strategies that one might use to decrease the computational complexity of the fingerprint search. By analyzing their weak points, we arrive at the fingerprint digest, a key concept based on which the fast search algorithm is constructed. The algorithm itself is detailed in Section 4. Experimental tests of the proposed search algorithm are included in Section 5, where we evaluate the time needed to find a fingerprint in the database and the time to determine that a fingerprint is not present in the database. This section also analyzes the effect of the search on identification errors when compared to the direct brute-force search. The paper is concluded in Section 6.

2. PRELIMINARIES

In this section, we introduce the basic notation, terminology, and some background material. Vectors and matrices will be always in bold upright font. Sometimes, we will index a matrix with a one-dimensional index, in which case it is assumed that the matrix has been converted to a vector, for example, by rows. Unless stated otherwise, all operations between vectors/matrices are understood as element-wise. Denoting the dot product of $\mathbf{X}$ and $\mathbf{Y}$ as $\mathbf{X} \odot \mathbf{Y} = \sum_{i=1}^{n} \mathbf{X}[i]\mathbf{Y}[i]$, $\| \mathbf{X} \| = \sqrt{\mathbf{X} \odot \mathbf{X}}$ is the norm of $\mathbf{X}$, and $\bar{\mathbf{X}}$ the sample mean. The normalized correlation is

$$corr(\mathbf{X}, \mathbf{Y}) = \frac{(\mathbf{X} - \bar{\mathbf{X}}) \odot (\mathbf{Y} - \bar{\mathbf{Y}})}{\| \mathbf{X} - \bar{\mathbf{X}} \| \cdot \| \mathbf{Y} - \bar{\mathbf{Y}} \|}.$$

In digital forensics using sensor noise, we usually work with image noise residuals obtained using a denoising filter F: $\mathbf{W} = \mathbf{I} - F(\mathbf{I}) = a\mathbf{I}\mathbf{K} + \boldsymbol{\Theta}$. Here, $\mathbf{W}$ is the noise residual of image $\mathbf{I}$, a is an attenuation factor, $\mathbf{K}$ is the multiplicative PRNU factor, which plays the role of the sensor fingerprint, and $\boldsymbol{\Theta}$ is the modeling noise. The fingerprint $\mathbf{K}$ can be estimated from noise residuals of several images taken by the same sensor (camera). The presence of the signal $\mathbf{I}\mathbf{K}$ in the noise residual of $\mathbf{I}$ can be interpreted as evidence that $\mathbf{I}$ was taken by a sensor with fingerprint $\mathbf{K}$. The presence is usually established using a correlation-based detector.[1–5]

Throughout this paper, we will assume that we are given a database $\mathcal{D}$, $\mathcal{D} = \{\mathbf{F}_i\}$, $i = 1, \ldots, N$, containing N camera fingerprints $\mathbf{F}_i$ of the same length n, which is also the length of the query fingerprint $\mathbf{X}$. Further, for simplicity, we assume perfect pixel synchronization between the query fingerprint and database fingerprints, i.e., no geometrical transformation of images and a search for its parameters is involved. We denote the fingerprints residing in the database with letter $\mathbf{F}$ instead of $\mathbf{K}$ because they are only estimates of the corresponding "true" PRNU factors. Without loss of generality, we also assume that the fingerprints are normalized:

$$\frac{1}{n} \sum_{i=1}^{n} (\mathbf{F}[i])^2 = 1, \ \forall \mathbf{F} \in \mathcal{D}.$$

We further assume the following model for any two fingerprint estimates: $\mathbf{F}_1 = a\mathbf{K} + \boldsymbol{\Theta}_1$, $\mathbf{F}_2 = b\mathbf{K}' + \boldsymbol{\Theta}_2$, where $\mathbf{K}$ and $\mathbf{K}'$ are the PRNU factors and $\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2$ are i.i.d. Gaussians $N(0, \sigma_1^2)$, $N(0, \sigma_2^2)$, respectively. Having observed $\mathbf{F}_1$ and $\mathbf{F}_2$, matching two fingerprints leads to the following two-channel hypothesis testing problem:[1]

$$\begin{aligned} &H_0: \ \mathbf{K} \neq \mathbf{K}' && \text{(the two fingerprints are from } \textit{different} \text{ cameras)} \\ &H_1: \ \mathbf{K} = \mathbf{K}' && \text{(the two fingerprints are from the } \textit{same} \text{ camera).} \end{aligned} \tag{1}$$

A simplified version of the generalized likelihood ratio test for this problem decides between H_0 and H_1 by comparing the normalized correlation ρ,

$$\rho = corr\left(\mathbf{F}_1, \mathbf{F}_2\right), \tag{2}$$

to a threshold t. The detector decides H_1 when $\rho > t$. When the decision is H_1 while H_0 is true, we speak of a "false alarm." The other type of error is called "missed detection." The probability of the false alarm and missed detection will be denoted P_{FA} and P_{MD}, respectively. From the central limit theorem, ρ is approximately Gaussian $N(0, 1/n)$ under H_0. Note that the variance of the correlation decreases linearly with increasing length of the vectors. This implies that for a chosen probability of false alarm P_{FA},

$$t = \frac{1}{\sqrt{n}} Q^{-1}\left(P_{\mathrm{FA}}\right), \tag{3}$$

where $Q(x)$ is the probability that one realization of a standard normal variable exceeds x.

3. COMPLEXITY REDUCTION

To reduce the computational complexity of the fingerprint search, one needs to either decrease the number of correlations that need to be carried out or somehow decrease the length of the fingerprint. Note that both options will increase the error probabilities of the search. In this section, we explore both ideas, analyze their weaknesses, and then naturally arrive at the concept of the so-called fingerprint digest, a key ingredient in our fast search algorithm detailed in Section 4.

3.1 Composite fingerprints

The first and a quite natural idea is to combine multiple fingerprints into one composite fingerprint. By correlating a query fingerprint with the composite fingerprint, we will essentially perform a simultaneous correlation test against many fingerprints with the cost of computing a single correlation.

Without loss of generality, let the number of fingerprints in the database be $N = 2^m - 1$. Form m subsets $S_m \subset \{1, 2, \ldots, N\}$ of size $(N+1)/2$ such that $i \in S_m$ if and only if $b_m(i) = 1$, where $(b_1(i), b_2(i), \ldots, b_m(i))$ is the binary representation of i. For the database of N camera fingerprints, pre-compute m "composite fingerprints" $\mathbf{M}_r$, $r = 1, \ldots, m$:

$$\mathbf{M}_r = \frac{1}{|S_r|} \sum_{i \in S_r} \mathbf{F}_i. \tag{4}$$

Let $\mathbf{X}$ be the query fingerprint and let $\mathbf{F}_q$ be the fingerprint of the same camera from the database. In other words, we assume that the camera is in the database. The search algorithm computes m normalized correlations,

$$c^{(r)} = corr\left(\mathbf{X}, \mathbf{M}_r\right), \; r = 1, \ldots, m, \tag{5}$$

and then compares them to a pre-determined threshold t. The matching index from the database is given by its binary representation $(b_1, b_2, \ldots, b_m)$, $b_r = (c^{(r)} > t)$, $r = 1, \ldots, m$. A positive identification of the fingerprint will occur when the search algorithm finds q as the matching index. When $b_r = 0$, $\forall r$, the algorithm decides that $\mathbf{X}$ is not in $\mathcal{D}$. The complexity of this algorithm is $O(n \log_2 N)$, as the composite fingerprints can be pre-computed. The logarithmic complexity, however, comes with a significant performance drop. Due to lack of space, we only provide a brief qualitative argument, leaving out the details to the report [10], where the ROC for this detector is computed. Assuming $q \in S_1$, a necessary condition for positive identification is:

$$t < c^{(1)} = corr(\mathbf{X}, \mathbf{M}_1) = corr\left(\mathbf{X}, \sum_{i \in S_1} \mathbf{F}_i\right) = corr\left(\mathbf{X}, \mathbf{F}_q + \sum_{i \in S_1, i \neq q} \mathbf{F}_i\right) = \frac{\mathbf{X} \odot \mathbf{F}_q + \mathbf{X} \odot (\mathbf{M}_1 - \mathbf{F}_q)}{\| \mathbf{X} \| \cdot \| \mathbf{M}_1 \|} \approx \frac{c_q}{\sqrt{(N+1)/2}}. \tag{6}$$

This means that only sufficiently strong fingerprints satisfying $c_q > t_1 = t\sqrt{(N+1)/2}$ have a chance to be positively identified. The drop in correlation (6) indicates that for sufficiently large N this algorithm will fail with probability approaching 1. For example, for $n = 1{,}920{,}000$ (the number of pixels in iPhone cameras used in experiments in Section 5) and $c_q = 0.3$, the probability of detection $P_{\mathrm{D}} = 0.99$ for $N = 255$, but it decreases to $P_{\mathrm{D}} = 0.70$ for $N = 511$ and $P_{\mathrm{D}} = 0.045$ for $N = 4095$.

3.2 Sequential trimming

An alternative and quite simple way to speed up the search is to trim all fingerprints to a fixed length $k \leq n$, which would speed up the search by the factor of n/k. The expected correlation c_q does not change after this trimming, while the threshold t needs to be increased due to the shorter length of the correlated signals and the fact that N comparisons are carried out. A successful detection occurs when $\rho > t$ only for the matching qth fingerprint and $\rho \leq t$ for the

remaining fingerprints. In contrast to the approach based on composite fingerprints, the complexity of this algorithm is still linear in N, but no longer dependent on n. Another advantage is a reduced demand for memory space.

Most importantly, however, sequential trimming does not exhibit a performance floor with respect to increasing N. The detailed analysis, including a parametric expression for the ROC curve, appears again in [10]. Here, we limit the exposition to the same example as in the previous section. For $n = 1{,}920{,}000$ and the three values of N, the fingerprints were trimmed to such a length k to obtain the same speed up factor as in the approach using composite fingerprints. The probability of detection $P_{\mathrm{D}} \approx 1$ with very low P_{FA} for correlations as low as $c_q = 0.1$ for $N = 4095$ and $c_q = 0.03$ for $N \le 511$.

3.3 Fingerprint digest

We now further improve on the sequential trimming and propose an approach that utilizes a selective dimensionality reduction. For each fingerprint $\mathbf{F}$ in the database, we keep its short *digest* consisting of an ordered list $\mathbf{V_F}$ of k largest (in magnitude) values from the fingerprint $\mathbf{F}$ and their locations $\mathbf{L_F}$ within the fingerprint captured using a one-dimensional index $l[d]$, $d = 1,\ldots, k$, $1 \le l[d] \le n$: $\mathbf{L_F} = \{l[d]\}_{d=1}^{k}$, $\mathbf{V_F} = \mathbf{F}[\mathbf{L_F}]$. For better readability, we will abbreviate $\mathbf{V}_{\mathbf{F}_i}$ as $\mathbf{V}_i$ and $\mathbf{L}_{\mathbf{F}_i}$ as $\mathbf{L}_i$. The likely elements of the digest are defective pixels, such as hot pixels represented with large positive values and dead pixels with large negative values. For a given query fingerprint $\mathbf{X}$, we compute the normalized correlation between each digest $\mathbf{V}_i = \mathbf{F}_i[\mathbf{L}_i]$ and $\mathbf{X}[\mathbf{L}_i]$:

$$\rho_i(\alpha) = corr\left(\mathbf{V}_i, \mathbf{X}[\mathbf{L}_i]\right), \tag{7}$$

where we denoted $\alpha = k/n$, the relative length of the digest. The hypothesis test (1) is resolved by comparing $\rho_i(\alpha)$ to a threshold t_k that returns the same probability of false alarm P_{FA} as the threshold for ρ in the original hypothesis test with detection statistic $\rho_i = corr(\mathbf{F}_i, \mathbf{X})$.

When $\mathbf{X}$ and $\mathbf{F}_q \in \mathcal{D}$ are correlated, e.g., when they are two fingerprints for the same camera corrupted by an additive white Gaussian noise, the expected normalized correlation (7) is positive. In contrast to the sequential trimming of Section 3.2, when $\rho_q(\alpha)$ was constant and equal to ρ for all α, ρ_q now becomes an increasing function of α (see the appendix for derivation):

$$\rho_q = \rho_q(\alpha, \rho) \cong \left(1 + \left(\frac{1}{\rho^2} - 1\right)\frac{1}{F(\alpha)}\right)^{-1/2}, \tag{8}$$

where

$$F(\alpha) = 1 + \frac{2}{\alpha\sqrt{2\pi}} Q^{-1}(\alpha/2) \exp\left\{-\left(Q^{-1}(\alpha/2)\right)^2/2\right\}. \tag{9}$$

Although (8) is only approximate, it is fairly accurate as shown in Figure 1 (middle and right) on simulated data. Note that although $F(\alpha) \to \infty$ and $\rho_q(\alpha, \rho) \to 1$, as $\alpha \to 0$ (see Figure 1 middle), the detector cannot improve with decreasing α. This is because the decreased length of the vectors increases the variance of the correlation, which, in turn, increases the decision threshold corresponding to a fixed P_{FA}: $t_k = t/\sqrt{\alpha} = Q^{-1}\left(P_{\mathrm{FA}}\right)/\sqrt{k}$, using formula (3). The p-value of ρ_q depends monotonically on the outlier measure $d^2 = \rho_q^2/Var[\rho|\mathrm{H}_0] = \rho_q^2 \alpha n$, which decreases with $\alpha \to 0$ slower than the same quantity in the sequential trimming (see Figure 1 right).

The digests allow us to keep k rather small (such as $k = 10{,}000$ or less, Figure 1 left) and thus speed up the search with a reasonable drop in reliability of testing even if we have to calculate ρ_i for all fingerprints $\mathbf{F}_i \in \mathcal{D}$. For example, for $\rho = 0.05$, when switching from the full-length fingerprints with $n = 3{,}000{,}000$ pixels to digests with $k = 10{,}000$ pixels, the outlier measure d^2 decreases only 29-times while it decreases $n/k = 300$ times for the non-selective trimming of Section 3.2. (See the curved line near zero in Figure 1 right.) The normalized correlations can thus be computed more than 10-times faster (for $\alpha = 0.0033$) with the same probability of miss compared to the sequential trimming and more than 3000-times faster when compared to the computation of full-length correlations.

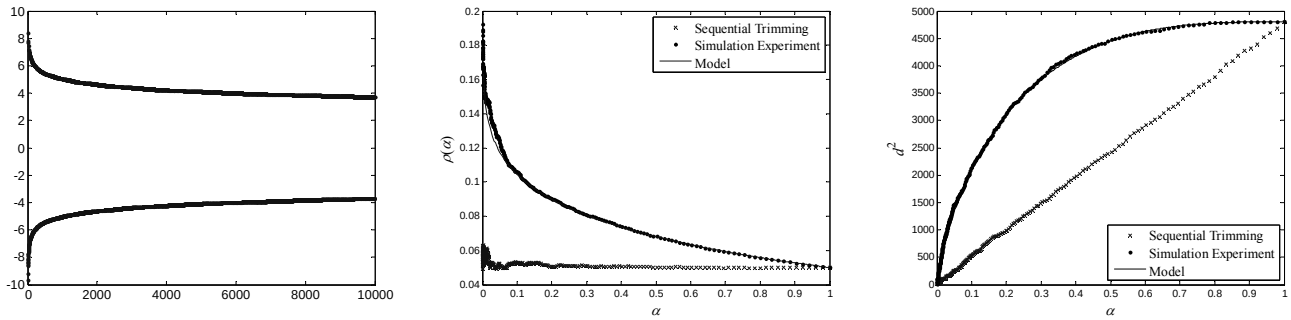

Figure 1. Scatter plot of a 10,000-pixel digest from a Nikon CoolPix-4300 4-Mpixel digital camera fingerprint estimated from 50 images (left), the normalized correlation (7) between digests simulated as i.i.d. Gaussian signals with $\rho = 0.05$ compared to the correlation computed from a cropped central part of the same length as the digest (middle), the outlier measure d^2 vs. relative digest length for the same simulated data (right).

4. FAST SEARCH ALGORITHM

In the previous section, we introduced the concept of a fingerprint digest and described a search algorithm that goes through all digests in the database and computes the correlation between each database digest $\mathbf{F}[\mathbf{L}]$ and the query fingerprint only at the digest locations $\mathbf{L}$. This already decreases the computational complexity by the factor of α, the relative digest length. We now describe a modification of this algorithm called the Approximate Rank Matching Search algorithm (ARMS) that can identify the correct fingerprint even faster provided the correlation $\rho = corr(\mathbf{X}, \mathbf{F})$ is large, e.g., $\rho > 0.1$. This would be the case, for example, when both $\mathbf{X}$ and $\mathbf{F}$ are good quality fingerprints.

For better clarity, from now on we strictly use i to index fingerprints/digests, j to index ordered pixels, d to index pixels in the digest, and u to index fingerprint candidates.

ARMS was inspired by the Spearman rank correlation.[6] The Spearman ranks $\mathbf{R_X}$ and $\mathbf{R_F}$ are the inverse permutations (on pixel indices) to the pixel location vectors $\mathbf{L_X}$ and $\mathbf{L_F}$. With $\mathbf{D} = \{\delta[j]\}_{j=1}^{n} = \mathbf{R_X} - \mathbf{R_F}$, the Spearman rank correlation is

$$\rho_S = corr\left(\mathbf{R_X}, \mathbf{R_F}\right) = 1 - \frac{6\sum_{j=1}^{n} \delta[j]^2}{n(n-1)(n+1)}. \tag{10}$$

The most influential indices j in (10) are those for which $\delta[j]$ is relatively small. Such indices are more likely located in the beginning of the digests when j is small. We will make use of the following sparse $n \times k$ matrix $\mathbf{S}$ whose elements are lists of fingerprint indices, $i \in \{1, \ldots, N\}$:

$$i \in \mathbf{S}[a,b] \text{ iff } \mathbf{F} = \mathbf{F}_i \in \mathcal{D} \text{ and } l_{\mathbf{F}}[b] = a. \tag{11}$$

The matrix $\mathbf{S}$ has as many rows as there are pixels in the fingerprints and as many columns as the digest length. The index i is in position $[a,b]$ when pixel a is the bth outlier in the ith digest. Because there are the total of Nk fingerprint indices, at most Nk lists $\mathbf{S}[a,b]$ are non-empty, and thus the density of $\mathbf{S}$ is $Nk/nk = N/n$. Consequently, providing the number of image pixels is much larger than the number of fingerprints in the database, $n \gg N$, the matrix $\mathbf{S}$ is sparse. This data structure will allow us to identify fingerprints that are more likely candidates for being correlated with the query fingerprint $\mathbf{X}$ *without having to evaluate the correlation*. For example, let the pixel $l_{\mathbf{X}}[1]$, which has the largest (positive or negative) value $\mathbf{V_X}[1]$ in $\mathbf{X}$ (it might be a hot or dead pixel), be ranked high in the camera fingerprint $\mathbf{F}_i$. This means that the camera index i is not far from the 1st column of $\mathbf{S}$ in row $l_{\mathbf{X}}[1]$. The algorithm retrieves all camera indices u (if there are any) from row $l_{\mathbf{X}}[1]$ between column 1 and $1+w$ and considers them as candidates whenever $\mathbf{V_X}[1]$ has the same sign as $\mathbf{V}_u[c]$, where u is the retrieved index and $c \in \{1, \ldots, 1+w\}$ is the column. The distance (window) w of how far we look is a parameter of ARMS. In the second iteration, the pixel (or row) $l_{\mathbf{X}}[2]$ is processed the same way and all camera indices from columns 1 to $2+w$ are retrieved. In the lth iteration, the columns $l-w$ to $l+w$ are processed in row $l_{\mathbf{X}}[l]$. Each candidate u is subjected to the hypothesis test in which the correlation $\rho_u(\alpha)$ given in (7) is compared with the threshold t_k. The iterations stop when $\rho_u(\alpha) > t_k$ (the matching fingerprint with index u is found) or when a time

limit is exceeded. The time limit is discussed further below. The larger the correlation between **X** and the matching fingerprint **F** is, the more likely will **F** appear early among the candidates. Notice that fingerprints **F** that appear early as candidates have $\delta[j] \le w$ in (10) for some small j, meaning that both the Spearman and the Pearson correlation receive large increments in their summations.

We improved this algorithm by further sifting through the candidates, which we now call potential candidates. A potential candidate becomes a valid candidate if its accumulated evidence **Z** exceeds a threshold t_{cand}. Clearly, we need to be able to compute **Z** very fast. In particular, its complexity must be much lower than $O(k)$. We propose to sum the squares of digest values as shown in the pseudo-code below. This is related to the estimation of an increment in the Pearson correlation (up to a scaling factor). The sparser the matrix **S** is, the fewer potential candidates ARMS will have to process and the faster the search will run.

The average computation time for finding the matching camera fingerprint when it is present in the database and identifiable by the correlation (7) depends (for fixed N, k, t) on ρ, w, t_{cand}, and on the particular implementation. Even though the correlation (7) is the costliest operation, the implementation of the data structure **S** is crucial for fast searching times as well. The number of considered candidates, w, may be minimized by choosing the threshold t_{cand} large enough, but not too large to miss the correct candidate.

So far, we focused on the case when the camera whose fingerprint is queried is in the database. When the database does not contain a matching fingerprint or when it is missed by the correlation test on digests of length k, the search time of this proposed algorithm can be much longer than if we used a simple brute-force search on digests described in Section 3.3. In order to limit the processing time, we introduce a *continue_criterion* (command line 4 in the pseudo-code below). As soon as this criterion becomes false, ARMS switches to a brute-force search on all fingerprints that were not considered yet. Optimization of this criterion with respect to the expected search time would require knowledge of the prior probability of finding a fingerprint in the database, which will not be available in practice. To keep the worst case time below twice the time for the brute force digest search, we simply set the criterion to 'false' once the current processing time exceeds that of the brute force search.

Psedo-code for ARMS

% ARMS finds a correlated (matching) item to **X** in database $\mathcal{D}$.

1. $k = 10{,}000$; $w = 1{,}000$; $t_{\text{cand}} = 0.2\sqrt{w}$; $t_k = \sqrt{60/k}$; % set up parameters (default values)
2. Compute digest $\mathbf{L_X}$ of length k from **X**.
3. $d = 0$; $\rho = 0$; $\mathbf{Z} = 0$;
4. `While` $\{\rho \le t_k$ `and` $d \le k - w$ `and` *continue_criterion*$\}$ `do`
5. $d = d + 1$; $c = \mathbf{L_X}[d]$; % move to the next element of **L**.
6. Retrieve fingerprint indices from all lists $\mathbf{S}[c, b]$, such that $|b-d| < w$, $U = \bigcup_{|b-d|<w} \mathbf{S}[c, b]$;
7. `While` $\{U$ is not empty `and` $\rho \le t_k\}$ `do`
8. choose $u \in U$; % potential candidate is $\mathbf{F}_u$
9. `If` $\{\text{sign}(\mathbf{V_X}[d]) = \text{sign}(\mathbf{V}_u[b])\}$
10. $\mathbf{Z}[u] = \mathbf{Z}[u] + (\mathbf{V}_u[b])^2$;
11. `end`
12. `If` $\{\mathbf{Z}[u] > t_{\text{cand}}$ `and` u has not been considered yet$\}$ % considered candidate
13. $\rho = corr(\mathbf{V}_u, \mathbf{X}[\mathbf{L}_u])$;
14. `end`
15. $U = U - \{u\}$;
16. `end`
17. `end`
18. `While` $\{\rho \le t_k$ `and` u has not been considered yet$\}$ `do` % brute force search on digests
19. $\rho = corr(\mathbf{V}_u, \mathbf{X}[\mathbf{L}_u])$;
20. `end`
21. `If` $\{\rho > t_k\}$ $q = u$; `end` % q is the matching fingerprint index in $\mathcal{D}$

4.1 Analysis of ARMS

We now investigate the conditions under which ARMS correctly finds the matching fingerprint without computing too many correlations (lines 13 and 19). Figure 2 depicts the distribution of $\mathbf{Z}[u]$ for randomly generated correlated Gaussian fingerprints with correlations (2) $\rho = 0$, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3. The plotted values correspond to $w = 1{,}000$ and $k = 10{,}000$. For better clarity, in Figure 2 the sample distributions are displayed through their Gaussian fits. These distributions would be obtained in experiments if we always let ARMS go through all k iterations. It is the positive correlation ρ that determines how much the distributions overlap with the one for $\rho = 0$ and thus ρ determines the probability of considering a non-matching fingerprint as a candidate (line 13 of the pseudo-code). The separation between hypotheses H_0 and H_1 (the distribution for $\rho = 0$ and for a fixed $\rho > 0$) improves with increasing w. This means that w should be larger than 1,000 if we expect ρ to be below 0.2 (see Figure 2). On the other hand, w should not be too large as it increases the search time. Because the separation improves with ρ, which is typically unknown to us, one possible remedy can be introducing a varying w. One simple choice is to set $w = l + w_1$ in the lth iteration, i.e., the fingerprint indices u are taken from the first $2l + w_1$ columns of $\mathbf{S}$. Repeating the simulation experiment for such a varying window w with $w_1 = 1000$, gives the plot in Figure 3.

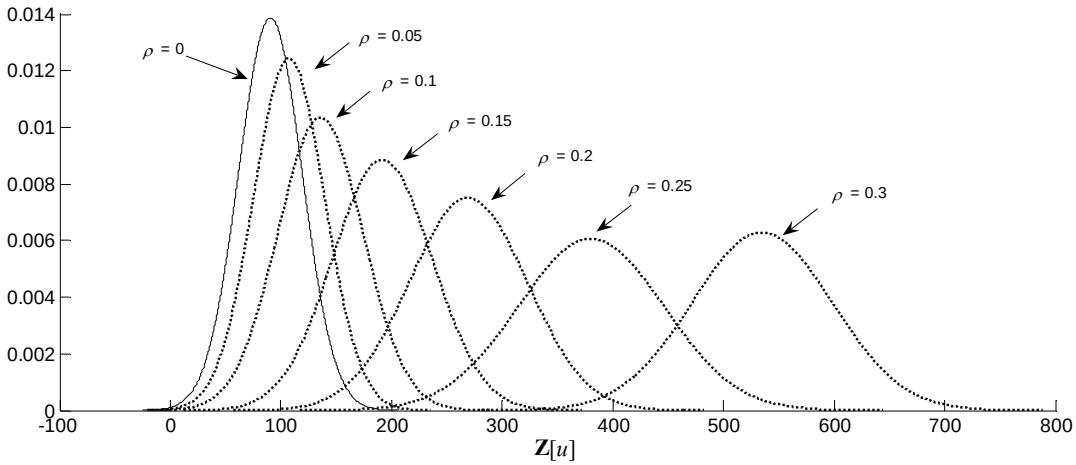

Figure 2. Gaussian fits of histograms of $\mathbf{Z}[u]$ (after k iterations of ARMS) for randomly generated fingerprints $\mathbf{F}_u$ that have correlation ρ with $\mathbf{X}$ as denoted in the plot (all simulated data, $w = 1{,}000$, $k = 10{,}000$, $n = 1200 \times 1600$).

The threshold t_{cand} also influences the number of iterations of ARMS (before it switches to the brute-force loop) and should be chosen rather small to reduce overhead computations. If ARMS stops after l ($l < k$) iterations, $\mathbf{Z}[u]$ will be k/l times smaller than what plotted in Figure 2 andFigure 3. If we want ARMS to find a matching fingerprint $\mathbf{F}_u$ with correlation $\rho > \rho_0$ (formula (1)) within l iterations with probability at least p without resolving to the brute-force search (lines 18–20), we need to choose t_{cand} as zl/k, where z is such that $\mathbf{Z}[u] > z$ with probability p, where $\mathbf{Z}$ is distributed according to the pdf from Figure 2 labeled with ρ_0.

We note that the sparse matrix $\mathbf{S}$ needs to be updated with every new fingerprint included in $\mathcal{D}$. With every next query, all data structures are reused. Alternatively, for one query $\mathbf{X}$, we can build only the needed portion of $\mathbf{S}$ "on the fly." This is because, in each iteration of ARMS, we need to access only k rows corresponding to the pixels represented by $\mathbf{L}_\mathbf{X}$ (for a single query $\mathbf{X}$) and $2w+1$ columns. We will speak of a "truncated matrix $\mathbf{S}$" when it has size $k \times (2w+1)$.

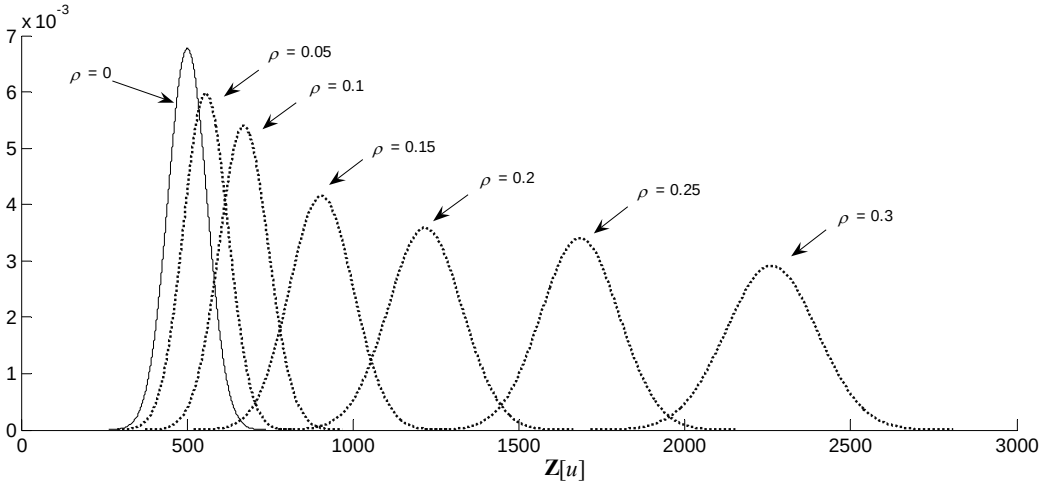

Figure 3. Gaussian fits of histograms of $\mathbf{Z}[u]$ (after k iterations of ARMS) for a varying window parameter w. The randomly generated fingerprints $\mathbf{F}_u$ have correlation ρ with $\mathbf{X}$ as denoted in the plot (all simulated data, $w_1 = 1{,}000$, $k = 10{,}000$, $n = 1200\times1600$).

5. EXPERIMENTS

The ARMS functionality and performance was verified on images downloaded from the Flickr image-sharing website. While this source provides a convenient access to a large number of diverse cameras for our tests, we have no control over the image content and their source. For example, some users post completely black images, which only taint our tests and need to be eliminated from experiments. Moreover, some users share their cameras or exchange images. We believe that shared images constitute only a small fraction of all images and that we will actually be able to pinpoint the users that use the same camera.

We test ARMS on a large number of cameras of exactly the same model, the Apple iPhone, with a built-in 2 Mpixel camera. Its resolution is 1200×1600 pixels, which is rather small compared to typical cameras in use today. We chose this homogenous database of one camera model for two reasons. First, in practice the search algorithm will likely be deployed only on cameras of the same model, as one can extract this information from the EXIF header. Second, fingerprints of sensors from cameras of the same model contain residuals of weak non-unique artifacts (NUAs) that may slightly increase the false alarm rate. In short, we intentionally chose the source of database fingerprints and their resolution far from the most favorable case for camera identification.

For experiments, we collected more than 200,000 color images in full resolution from 2425 Flickr users. Images in landscape orientation were arranged in groups of 60 and the camera PRNU was estimated from each group using the maximum likelihood estimator.[1] For some users, more than 120 images were downloaded to allow us to obtain two different fingerprint estimates for their camera. Thus, the total number of fingerprints was 3092. The fingerprints were estimated for each color channel separately and then processed by zero-meaning as described in Ref. [9]. This process separates the three color channels and also the green, blue, and red pixels (assuming the sensor has the Bayer color filter array), subtracts the averages from each column and row and in each color and then re-arranges the colors back to form a colored noise fingerprint. After converting it to a grayscale real-valued matrix, the fingerprint is filtered with a Wiener filter in the DFT domain (magnitude only). The resulting 2-D signal is converted to a vector – the final fingerprint – by column-wise scanning.

5.1 Model validation

Our first test involves two fingerprints of the same iPhone camera computed from two different sets of 60 images. The goal here is to compare our theoretical model for which relation (8) was derived. Figure 4 shows a noticeable mismatch between the model and the measured data. For digests of length $k = 10{,}000$, $n = 1200\times1600$, $\alpha_0 = k/n = 0.0052$, we computed $\rho(\alpha_0) = 0.6728$, while formula (8) gives $\rho(\alpha_0) = 0.7074$. A closer inspection revealed that iPhone fingerprints

exhibit some residual periodic NUAs that might be responsible for this discrepancy. The difference between the shapes of the plots in Figure 4 and Figure 1 (right) is due to a large initial correlation $\rho(1) = 0.307$ between the two fingerprints of the same camera.

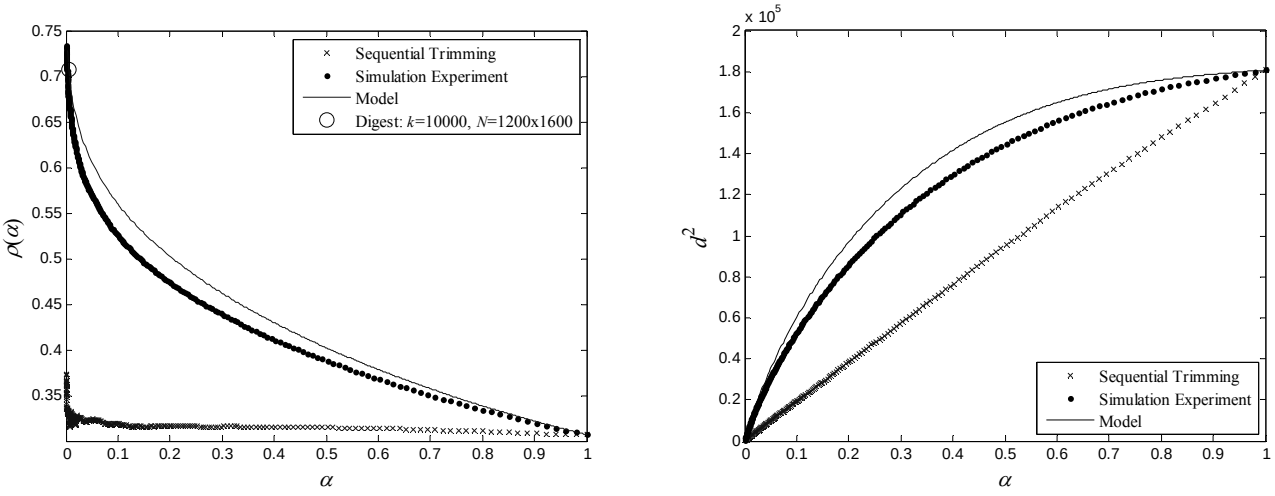

Figure 4. Function $\rho(\alpha)$ (left), outlier measure d^2 (the curved line on the right).

5.2 Finding a matching fingerprint in the database

This is the main experiment that demonstrates the performance of ARMS on the above database of 3092 fingerprints from (at most) 2425 iPhone cameras. The experiment consisted of 667 searches for which the camera of the query fingerprint was present in the database. We repeat that the query and the database fingerprints were estimated from a different set of images. The fingerprint chosen as the query was removed from the database, leaving $N = 3091$ others to search among. The digest length was fixed to $k = 10{,}000$. We recorded the following quantities: $\rho_u(\alpha_0)$, the number of ρ evaluations for the considered candidates, and the search time after the data was loaded in the memory. To determine the appropriate thresholds, we considered the results of the large scale experiments with the correlation detector reported in Ref. [7], even though the tests there were run on cameras other than iPhones. In particular, this work showed that in order to obtain the probability of false alarm $P_{\mathrm{FA}} \approx 10^{-6}$, one needs to use a larger threshold than $Q^{-1}(P_{\mathrm{FA}})/\sqrt{k}$ given by (3). Based on the results of Ref. [7], we set $t_k = \sqrt{60/k} = 0.07746$. The expected correlation ρ between two full-length i.i.d. Gaussian fingerprints with $\rho(\alpha, \rho) = 0.07746$ is $\rho = 0.025$. This value of ρ is the digest's sensitivity limit that will determine, along with the fingerprints' quality, the probability of a missed detection. When $\rho < 0.025$ for the matching fingerprint, a missed identification will likely occur.

Because the typical value of the correlation ρ between two matching fingerprints (from the same camera) in our test was around $\rho = 0.25$ (ten times our digests' sensitivity limit), we expect no missed detections. The average correlation $\rho_u(\alpha_0)$ for pairs of digests from the same camera was 0.5988.

Our Matlab implementation of ARMS was run without any attempt to optimize the speed of the implementation (the sparse matrix **S** was created with the `spalloc` command), with two different thresholds t_{cand}. The first one (the first column in Table 1) leads to a small average number of considered candidates and promises a faster search time after code optimization and using a pre-computed matrix **S**. The second test with a smaller value of t_{cand} leads to a short average search time of 0.227 seconds (including the time for updating the truncated matrix **S**). This processing time should be compared to 4.8 sec needed for the brute-force search on fingerprint digests (the last column in Table 1) and to 35 minutes needed when searching on full-length fingerprints that requires reading the large fingerprints from external memory. All tests were run on a Dual Core AMD Opteron™ Processor, 2.2 GHz.

When the query fingerprint belongs to a camera that is not contained in $\mathcal{D}$, i.e., when H_0 (1) is true, the running time of ARMS is upper bounded by twice the processing time for the brute force search on digests as stated in Section 4. The

time for the brute-force search will be about twice the average time for the matching fingerprints because the algorithm has to go through all fingerprints rather than its half.

Table 1. The search for a matching fingerprint in the database of 3091 fingerprints of 2425 different iPhone cameras.

	ARMS	ARMS	Brute-force search on fingerprint digests
N	3091	3091	3091
K	10,000	10,000	10,000
t_{cand}	$\sqrt{w} = 31.623$	$0.2\sqrt{w} = 6.3246$	–
t_k	0.07746	0.07746	0.07746
mean($\rho_u(\alpha_0)$) (min, max)	0.5988 (0.2377, 0.9405)	0.5988 (0.2377, 0.9405)	0.5988 (0.2377, 0.9405)
mean($\rho_u(1)$) (min, max)	0.2522 (0.0829, 0.8327)	0.2522 (0.0829, 0.8327)	0.2522 (0.0829, 0.8327)
Missed/false identification	0/0	0/0	0/0
Average number of evaluations of correlation ρ (best 95% cases)	2.030 (1.106)	28.95 (12.615)	1545 (77)
Time for testing 1 candidate	0.0028 sec	0.0028 sec	0.0028 sec
Average search time	13.1 sec	0.227 sec	4.80 sec

Surprisingly, the test found two matching fingerprints with high correlations $\rho_u(\alpha_0) \approx 0.63$ belonging to different users. Further inspection of the images confirmed that the camera users were closely related if not the same, suggesting that the images were, indeed, all taken with identical cameras. Consequently, in the caption of Table 1 we decreased the total number of different cameras in this test to 2423.

5.3 Find the source camera in the database for a given image

The last experiment deals with the task of finding the source camera for a single query image, which corresponds to the case when the query fingerprint is estimated from a single image and is thus of a very low quality. We used the database of $N = 2423$ fingerprints of 2423 different iPhone cameras and tested randomly chosen 691 images from 691 different cameras. We stress that none of the 691 images was used to estimate the database fingerprints. Because the correlation between the matching fingerprint and the query fingerprint is likely to be quite low, we did not apply ARMS in this case and instead used the brute-force search on digests described in Section 3.3. The average search time was 3.4 sec ($= N/2 \times 0.0028$ sec.), the empirical missed detection rate was $P_{\text{MD}} = 0.19$ and $P_{\text{FA}} = 0$ (while for $k = 30{,}000$, $P_{\text{MD}} = 0.067$ and $P_{\text{FA}} = 0$). The typical value of the correlation ρ between the image noise residual (PRNU estimate from one image) and the matching camera fingerprint was only around 0.04, which is quite close to the digest sensitivity limit (0.025) for $k = 10{,}000$. A brief inspection of Figure 2 andFigure 3 tells us that we would have to increase k substantially to provide ARMS an advantage over the brute-force search on digests. The proper value of k could be obtained by extending the error analysis of sequential trimming [10] to digests. Due to time and space constraints, we postpone this analysis to our future work.

5.4 Memory requirements

A straightforward implementation requires keeping in the memory the digests $\mathbf{V}_i$ and the indices $\mathbf{L}_i$, $i = 1, \ldots, N$, of the fingerprints in the database, the sparse matrix $\mathbf{S}$, the query fingerprint $\mathbf{X}$, and its digest. All digests take $(4+4)(N+1)k$ bytes when a single precision is opted for, or $(8+4)(N+1)k$ bytes for double precision. The sparse matrix takes at most double the space required for the digests. The required space is thus proportional to kN and, as long as kN is not too large, the requested memory space is confined. When N is large (or comparable to the number of pixels n) and the above requirements exceed the available memory, the matrix $\mathbf{S}$ can be prepared in advance and its (partial) rows read whenever necessary. This approach pays off when the number of queries n_{q} to the same database is such that $n_{\text{q}} \sim n/k$. Otherwise, the matrix $\mathbf{S}$ does not have to be even fully available. Instead, during the lth iteration only the $2w+1$ elements in columns $l-w, \ldots, l+w$ and row $l_{\mathbf{X}}[l]$ of $\mathbf{S}$ need to be obtained one at a time until the search stops. This is the implementation (in Matlab) we used in the experiments.

6. CONCLUSION

Camera identification based on detecting sensor noise is nowadays an established technology that has already been used in court[8] to provide evidence in prosecution whenever a crime has been committed by taking a digital picture or video. It is anticipated by the authors of this article that this technology will be used increasingly more often not only in criminal cases but in digital forensic analysis in general. In particular, analysts may need to decide whether a given sensor fingerprint already resides in a large database of camera fingerprints or whether a given image was taken by a camera whose fingerprint is in the database. The straightforward approach in which the query fingerprint is correlated with each fingerprint from the database has complexity proportional to the product of the database size and the number of pixels in the fingerprint. With increasing resolution of digital sensors, the search time may become prohibitively large even for moderately sized fingerprint database, when the search may require hours to finish. The complexity becomes an even more serious issue when an analyst needs to cluster images according to their common source. Here, the processing time is quadratic in the number of clusters.

In this paper, we describe a fast search algorithm whose worst-case complexity is still proportional to the database size but does not depend on the sensor resolution. As the constant of proportionality is very small, the search is very fast in practice. For example, when the fingerprint resides in a database of 3000 two-megapixel fingerprints, the average search time is about 0.2 seconds. The algorithm works by extracting a digest of the query fingerprint formed by the most extreme 10,000 fingerprint values and then approximately matches their positions with the positions of pixels in the digests of all database fingerprints. The algorithm requires a sparse data structure that needs to be updated with every new fingerprint included in the database. The algorithm is designed to make sure that the probability of a match and false alarm for the fast search is identical to the corresponding error probabilities of the direct brute-force search. Experiments on a database of 2000 iPhones prove the feasibility of the proposed approach.

The fast search algorithm introduced here does not rely on any structure or special properties of the fingerprints in the database. It can be utilized in any application where a database contains n-dimensional elements, and n is a fixed large number. The only requirement is that the elements consist of real numbers or integers from a large range. Integers from a small range would lead to ill-defined ranks. An extreme case when the rank correlation and, consequently, the fast search algorithm cannot be used, are binary vectors (one bit for each vector element).

7. ACKNOWLEDGEMENTS

This research was supported by an NSF award no. CNF-0830528.

REFERENCES

[1] Fridrich, J.: "Digital Image Forensic Using Sensor Noise," *IEEE Signal Processing Magazine*, vol. 26(2), pp. 26–37, 2009.

[2] Lukáš J., Fridrich J., and Goljan M.: "Digital Camera Identification from Sensor Pattern Noise," *IEEE Transactions on Information Forensics and Security*, vol. 1(2), pp. 205–214, June 2006.

[3] Chen, M., Fridrich, J., Goljan, M., and Lukáš, J.: "Determining Image Origin and Integrity Using Sensor Noise," *IEEE Transactions on Information Security and Forensics*, vol. 1(1), pp. 74–90, March 2008.

[4] Chen, M., Fridrich, J., and Goljan, M.: "Source Digital Camcorder Identification Using Sensor Photo-Response Non-Uniformity," *Proc. SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505, San Jose, CA, January 28 – February 1, pp. 1G–1H, 2007.

[5] Goljan, M. and Fridrich, J.: "Camera Identification from Cropped and Scaled Images," *Proc. SPIE Electronic Imaging*, *Forensics, Security, Steganography, and Watermarking of Multimedia Contents X*, vol. 6819, San Jose, CA, January 28–30, pp. 0E–0F, 2008.

[6] Spearman's rank correlation coefficient. http://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient.

[7] Goljan, M., Fridrich, J., and Filler, T.: "Large Scale Test of Sensor Fingerprint Camera Identification," *Proc. SPIE Electronic Imaging, Media Forensics and Security XI*, vol. 7254, San Jose, CA, January 17–21, pp. 0I–0J, 2009.

[8] Spy Blog, May 9, 2009. http://p10.hostingprod.com/@spyblog.org.uk/blog/2009/05/

[9] Goljan, M.: "Digital Camera Identification from Images – Estimating False Acceptance Probability," Digital Watermarking, 7th International Workshop, IWDW 2008, Busan, Korea, *LNCS*, vol. 5450, Springer Berlin / Heidelberg, pp. 454–468, 2009.

[10] Fridrich, J. and Goljan, M.: "Derivation of ROCs for Composite Fingerprints and Sequential Trimming," Technical Report, Binghamton University, January 2010. *http://www.ws.binghamton.edu/fridrich/Research/rocs.pdf* .

8. APPENDIX

In this appendix we compute the expected value of the normalized correlation between the database fingerprint digest and its (query) noisy version (formulas (8) and (9)). The fingerprint $x_1, \ldots, x_n$ is modeled as n i.i.d. samples of a Gaussian random variable $\xi \sim N(0,\sigma^2)$. Without loss of generality, let us further assume that the pixels are already ordered so that $|x_j| \geq |x_{j+1}|$ for each $j \in \{1, \ldots, n-1\}$. We remind that the fingerprint digest is formed by the $k = \alpha n$ most extreme values in the estimated fingerprint: $x_1, \ldots, x_k$. To avoid working with rather complicated order statistics, we adopt the following simplification. We will assume that the k outlier values $x_1, \ldots, x_k$, all come from a random variable ϕ_T distributed on $U = (-\infty, -T] \cup [T, \infty)$, where $T > 0$ is such that the probability $\Pr(\xi \in U) = \alpha$. The pdf $f_T(x)$ of ϕ_T is just a scaled Gaussian density: $f_T(x) = 1/(\sqrt{2\pi}\sigma\alpha)e^{-x^2/(2\sigma^2)}$, for $x \in U$, and $f_T(x) = 0$ otherwise. Using the Q-function, $2Q(T/\sigma) = \alpha$ or $T = \sigma Q^{-1}(\alpha/2)$.

We now compute the expected value of the correlation between the database fingerprint digest $\mathbf{x} = (x_1, \ldots, x_k)$ and the query digest, which is a noisy version of $\mathbf{x}$, $\mathbf{x} + \boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is a zero-mean white Gaussian noise uncorrelated with $\mathbf{x}$:

$$\rho(\alpha) = corr(\mathbf{x}, \mathbf{x}+\boldsymbol{\eta}) = \frac{\sum_{j=1}^{k} x_j(x_j + \eta_j)}{\sqrt{\sum_{j=1}^{k} x_j^{\,2}}\sqrt{\sum_{j=1}^{k}(x_j+\eta_j)^2}} \simeq \frac{1}{\sqrt{1+\frac{1}{SNR}}}, \tag{A.1}$$

where

$$SNR = \frac{\frac{1}{k}\sum_{j=1}^{k} x_j^{\,2}}{\frac{1}{k}\sum_{j=1}^{k} \eta_j^{\,2}} \simeq \frac{Var[\phi_T]}{Var[\eta]}. \tag{A.2}$$

Thus, all we need to establish is the variance of ϕ_T. By integrating by parts and substituting for $T = \sigma Q^{-1}(\alpha/2)$:

$$\begin{aligned} Var[\phi_T] &= 2\int_T^\infty \frac{x^2 e^{-x^2/(2\sigma^2)}}{\sqrt{2\pi}\sigma\alpha}dx = \{x/\sigma = t\} = \frac{2\sigma^2}{\alpha}\int_{T/\sigma}^\infty \frac{t^2 e^{-t^2/2}}{\sqrt{2\pi}}dt = \{u = t, v' = te^{-t^2/2}\} \\ &= \frac{2\sigma^2}{\alpha}\left[\frac{T}{\sigma\sqrt{2\pi}}e^{-T^2/(2\sigma^2)} + Q(T/\sigma)\right] = \frac{2\sigma^2}{\alpha\sqrt{2\pi}}Q^{-1}(\alpha/2)e^{-\frac{1}{2}(Q^{-1}(\alpha/2))^2} + \sigma^2 \\ &= \sigma^2\left[1 + \frac{2}{\alpha\sqrt{2\pi}}Q^{-1}(\alpha/2)e^{-\frac{1}{2}(Q^{-1}(\alpha/2))^2}\right] \triangleq \sigma^2 F(\alpha). \end{aligned} \tag{A.3}$$

Using (A.3), we write the SNR (A.2) as $SNR = \sigma^2 F(\alpha)/Var[\eta]$. Because the correlation between the full-length fingerprints is $\rho = (1 + \sigma^2/Var[\eta])^{-1/2}$, we have $\sigma^2/Var[\eta] = \rho^{-2} - 1$ and thus (A.1) can be finally written as

$$\rho(\alpha) = \left(1 + \frac{1}{SNR}\right)^{-1/2} = \left(1 + \left(\frac{1}{\rho^2} - 1\right)\frac{1}{F(\alpha)}\right)^{-1/2}. \tag{A.4}$$