PANDORA: Preserving Privacy in PRNU-Based Source Camera Attribution

Manoranjan Mohanty, Ming Zhang, Muhammad Rizwan Asghar, and Giovanni Russello

***Abstract*—Photo Response Non-Uniformity (PRNU) noise-based source camera attribution is a popular digital forensic method. In this method, a camera fingerprint computed from a set of known images of the camera is matched against the extracted noise of an anonymous questionable image to find out if the camera had taken the anonymous image. The possibility of privacy leak, however, is one of the main concerns of the PRNU-based method. Using the camera fingerprint (or the extracted noise), an adversary can identify the owner of the camera by matching the fingerprint with the noise of an image (or with the fingerprint computed from a set of images) crawled from a social media account. In this article, we address this privacy concern by encrypting both the fingerprint and the noise using the Boneh-Goh-Nissim (BGN) encryption scheme, and performing the matching in encrypted domain. To overcome leakage of privacy from the content of an image that is used in the fingerprint calculation, we compute the fingerprint within a trusted environment, such as ARM TrustZone. We present *PANDORA* that aims at minimizing privacy loss and allows authorized forensic experts to perform camera attribution.**

I. Introduction

Photo Response Non-Uniformity (PRNU) noise-based source camera attribution [1], [2] is an effective method to find out the source camera of an anonymous image. This method has been instrumental both to verify whether an anonymous questionable image, such as the one containing terrorist propaganda or child pornography, has been taken by a suspected camera, and in the case of no particular suspected camera, to identify the culprit camera from a database of suspected cameras.

The PRNU-based approach is based on the PRNU noise pattern that is present in a camera sensor due to the manufacturing impurities. Because of the impurities, each pixel in the sensor generates a different response to light intensity than the ideal noise-free case, resulting in the PRNU noise, which is basically the difference between the noise-free output and the actual output. The PRNU noise can act as a camera fingerprint as this noise is unique for each camera. Since obtaining the exact PRNU noise is not possible without the cooperation of the camera manufacturer, the PRNU noise is typically estimated from an image. Using the estimated PRNU noise of a set of known images of the camera, a camera fingerprint is computed. Then, this fingerprint is correlated with the estimated PRNU noise of an anonymous query image to determine if the camera has taken the anonymous image. In the rest of this article, for simplicity, we call the *estimated PRNU noise* as the *PRNU noise*.

Manoranjan Mohanty is with the Center for Cyber Security, New York University Abu Dhabi, UAE. Ming Zhang, Muhammad Rizwan Asghar, and Giovanni Russello are with the Department of Computer Science, The University of Auckland, New Zealand. They can be contacted by email: manoranjan.mohanty@nyu.edu, ming.zhang@auckland.ac.nz, r.asghar@auckland.ac.nz, and g.russello@auckland.ac.nz, respectively.

One of the concerns with the PRNU-based approach, however, is its potential to leak privacy [3], [4]. If someone's camera fingerprint is leaked, her identity can be known (even if the fingerprint is anonymized) by linking the fingerprint with the PRNU noise of images crawled from social media [5]. For example, in a court case, the identity of a suspect (who can be proved innocent later) or the identity of a witness can be known by unauthorized persons using the camera fingerprint of the suspect/witness. Below, we elaborate this possibility.

- A high-profile personality is under investigation as a suspect in a case of child pornography. Given the sensitivity of the case, the suspect's name is suppressed until proven guilty. Third-party experts extract a fingerprint from the suspect's phone to match it against a database of known child pornography images. However, the suspect's fingerprint is mishandled and is released to the public. In order to find out the suspect, journalists match PRNU noise of images from social media with the leaked fingerprint and find a matching image on a Facebook account belonging to a high-profile music personality. The suspect's name appears in the headlines of major national and international news describing him/her as a child abuser. However, later, the suspect is proved innocent. Nevertheless, the suspect's career is now ruined as he/she is now defamed.
- In a trial of a drug case, a witness has provided a picture showing the dealings of a syndicate. The identity of the witness is kept secret to ensure her safety. However, the image is (maliciously) leaked to the public. In order to identify who has taken the image, the drug syndicate matches the image with the fingerprint of a number of investigating journalists (the fingerprint can be previously known to the syndicate or can be freshly obtained using images from social media accounts). From the matchings, the syndicate identifies the journalist who has taken the image. Then, the journalist starts receiving threats from the syndicate.

Although the scenarios above could seem far-fetched, it is clear that camera fingerprint can be misused to identify users through their online presence. The problem is further complicated by the fact that law enforcement agencies are using cloud storage and external third-party experts for handling digital evidence. For example, the fingerprints could be stored on a third-party server (*e.g.,* cloud datacenter), or the fingerprints could be extracted and matched by external forensic experts.

Previous researches on addressing the privacy issue have mainly focused on distorting the PRNU noise such that a reliable PRNU noise cannot be extracted from an image [6]. As explained in Section III, these anti-forensic approaches however are not effective privacy-preserving methods as the PRNU noise is robust to a number of anti-forensic operations.

In this article, we present *PANDORA* (Preserving privAcy in prNu-baseD sOurce cameRa Attribution) that addresses the privacy issue in PRNU-based camera attribution by encrypting the fingerprint and the PRNU noise, and performing the matching operations in an encrypted domain. *PANDORA* allows the computation of the fingerprint within the device to be identified. The fingerprint is computed in a trusted environment where an adversary can neither access the content of an image (so that the privacy is not leaked from the image content) nor tamper with the fingerprint. In this work, we assume that mobile devices' cameras are the devices to be identified. As such, given its presence in most current mobile devices, we use ARM TrustZone as a secure environment to compute the fingerprint. However, without loss of generality, our work can be generalized to any computing platform with a similar secure environment (*e.g.,* Intel SGX) or where a secure element can be deployed in the device (*e.g.,* secure SIM cards).

In *PANDORA*, both the fingerprints and the PRNU noise of query images are encrypted using the Boneh-Goh-Nissim (BGN) encryption scheme [7] that is homomorphic to an arbitrary number of additions and one multiplication. The encrypted fingerprint can be matched against the PRNU noise of a query image on a third-party server without accessing neither the fingerprint nor the PRNU noise in cleartext. As such, *PANDORA* allows to outsource to third-party the most costly operations, such as storage and matching. Unlike previous PRNU noise distortion techniques, our scheme can ensure privacy while providing utility, as the privacy leak from the PRNU-based attribution method is minimized while ensuring that an authorized forensic expert can uninterruptedly perform her job.

The rest of this article is organized as follows. Section II provides an overview of PRNU-based source camera attribution method, and ARM TrustZone. Section III reviews related work on anti-forensic schemes to PRNU-based source attribution method. These methods can preserve privacy by denying extraction of a reliable PRNU noise from an image. Section IV describes our system model and threat model. Section V presents *PANDORA*. In Section VI, we describe our BGN-based encrypted domain camera attribution in detail. Construction details of *PANDORA* are given in Section VII, and security analysis is analysed in Section VIII. Section IX explains results and performance analysis of *PANDORA*. Finally, Section X concludes this article and provides directions for future work.

II. Background

In this section, first we provide an overview of PRNU-based source camera attribution method. Then, we describe some details of the ARM TrustZone.

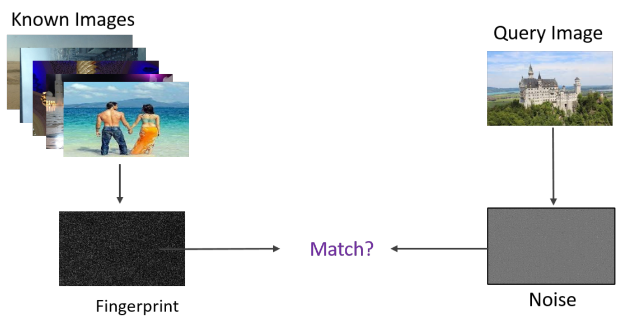

Fig. 1: PRNU-based source camera attribution: a fingerprint is computed from a set of known images of a camera while the noise is extracted from a query image. The fingerprint is matched with the extracted noise to determine if the query image has been taken from the same camera.

A. PRNU-Based Source Camera Attribution

PRNU-based source camera attribution is a well-studied method [1], [8]–[16]. This method is based on the fact that the sensor output I of a camera can be modeled as

$$I = I^{(0)} + I^{(0)}X + \phi,$$

where $I^{(0)}$ is the noise-free output, X is the PRNU noise representing the camera fingerprint, and ϕ is random noise. Using a de-noising filter $\mathscr{F}$ (such as a Weiner filter) and a set of images of a camera, we can estimate the camera fingerprint by first estimating the PRNU noise of the i^{th} image as $F_i = I_i - \mathscr{F}(I_i)$, and then combining the PRNU noise of all the images. For determining if a specific camera has taken a given query image I', we can first obtain the PRNU noise F' of the query image using $\mathscr{F}$, and then correlate F' with F to determine if the camera has taken I' (as illustrated in Figure 1). The correlation can be computed using normalized correlation (r) that is given as:

$$r(F,F') = \frac{\sum_{i=1}^{n}(F_i-\overline{F})(F_i'-\overline{F'})}{\sqrt{\sum_{i=1}^{n}(F_i-\overline{F})^2}.\sqrt{\sum_{i=1}^{n}(F_i'-\overline{F'})^2}}, \quad (1)$$

where $\overline{F} = \frac{\sum_i^n F_i}{n}$ and $\overline{F'} = \frac{\sum_i^n F_i'}{n}$. If the correlation value $r(F,F')$ is above a threshold, it is concluded that the query image belongs to the camera. The threshold is chosen per-camera.

B. ARM TrustZone

The ARM TrustZone is a trusted execution environment technology offered by ARM for its ARMv6 processor architecture. Most of the recent ARM-powered mobile devices are shipped with the TrustZone. Non-ARM powered mobile devices are equipped with alternative solutions that still provide a trusted execution environment, *e.g.,* by Intel's SGX technology.

The TrustZone divides the processor core into two virtual isolated cores called *worlds*: the secure world (also referred to as the virtual core or simply TrustZone) and the normal world. Both worlds are separated by hardware extensions, and any access to the secure world is highly regulated. The TrustZone has its own secure operating system, secure set of applications, and secure data storage. Since unauthorized data access and processing request cannot be made to the secure world, highly sensitive applications, such as Digital Rights Management (DRM) applications that access protected multimedia content, are executed in the TrustZone. The general-purpose non-secure applications, on the other hand, are executed in the normal world.

III. Related Work

In this section, we review anti-forensic approaches that have been proposed to prevent privacy leaks in PRNU-based methods.

Various anti-forensic methods have been proposed to prevent PRNU-based source camera attribution [6]. These methods can either weaken the PRNU noise pattern or misalign the PRNU noise so that a reliable PRNU noise cannot be extracted from an image, and hence privacy of the image owner can be preserved.

PRNU noise can be weakened by strong signal processing operations. Gloe *et al.* [17] proposed two such PRNU noise weakening techniques by (i) applying an undetectable resampling operation to the image, and by (ii) forging image origin by removing PRNU of one camera and by adding PRNU of another camera. Karakucuk *et al.* proposed two adaptive PRNU denoising methods [18], [19] that iteratively remove PRNU noise from an image based on an estimated *gain factor* of the PRNU. Another approach to weakening the PRNU noise is to suppress the PRNU noise using flat-fielding [17], [20].

The PRNU noise can be misaligned either by applying geometric transformations, such as cropping and resizing, or by using more sophisticated irreversible transformation techniques, such as forced seam-carving [3], patch-based desynchronization [21], and image stitching [6].

Both the PRNU noise weakening method and the PRNU noise misaligning method, however, have been proved ineffective. The PRNU noise can withstand common signal processing operations such as scaling, cropping and compression [22], [23]. According to Rosenfeld *et al.* [24], even after eight rounds of de-noising, there is still a significant correlation between the noise pattern of a multiply-denoised image and the fingerprint of the camera. The flat-fielding approach is less practical, and it can be defeated for a specific use case [25]. Recently, Taspinar *et al.* [26] showed that even an irreversible transformation method, such as forced seam-carving, can also be ineffective. In addition, both the PRNU noise weakening method and the PRNU noise misaligning method do not guarantee utility as an authorized forensic expert cannot extract camera fingerprint for lawful use.

To the best of our knowledge, no previous effort has been made for encrypted domain camera attribution that guarantees both utility and privacy. Some previous works, however, have focused on encrypted domain image processing [27]–[30] using partial homomorphic encryption schemes, such as Shamir's secret sharing and Pailier encryption.

IV. System Model

In our work, we consider that the law enforcement agencies have outsourced most of the camera attribution tasks to a third-party entity, such as a forensic expert or a third-party organization. The third-party is responsible for storing fingerprints and performing matching operations between stored fingerprints and the noise of an anonymous query image. In our system, we have following entities:

- **Fingerprint Source:** This entity is an individual, an organization, or an application that computes camera fingerprint from a set of known images of the camera. This entity must ensure that the image content is not leaked to an adversary (to preserve privacy of the camera owner). In addition, this entity is also responsible to ensure that the fingerprint has not been tampered to provide false matchings. Since privacy can still be leaked from a computed fingerprint (even though it looks like a noise) by linking the fingerprint with images from social media, the Fingerprint Source encrypts the fingerprint.
  In this work, we assume that the application that generates the fingerprint is executed in a trusted environment. For instance, in case of a fingerprint generated from a mobile device, such as a smartphone, the application is executed in the ARM TrustZone of the phone. This prevents the phone owner to tamper with the fingerprint generation process. To prevent that the fingerprint is leaked in cleartext, the application encrypts the fingerprint still in the TrustZone before it sent off for further processing.
- **Third-Party Expert:** This entity stores encrypted fingerprints obtained from the Fingerprint Source, and matches a fingerprint with noise of a query image in the encrypted domain. This entity, however, does not know the information about the camera or the owner of the camera. Either an individual or an organization can act as a Third-Party Expert. In any case, this entity must have enough storage and processing power to store several thousand fingerprints and perform matchings in encrypted domain. To be cost effective, this entity can outsource storage and computation to public cloud service providers, such as Amazon, Google, and Microsoft.
- **Match Maker:** This entity represents a trusted organization (*e.g.,* law enforcement authority, a judge) who has the anonymous query image and who wants to find the camera that took the image. The fingerprint matching process, however, has been outsourced to the Third-Party Expert, who must not know the image content or the extracted noise of the image in plaintext. The Match Maker therefore extracts noise from the query image at her end, encrypts the noise, and sends the encrypted noise to the Third-Party Expert.
- **Match Maker Server:** This entity is under the control of the Match Maker and it is used for performing the final part of the matching on the values received from the Third-Party Expert. In comparison with the infrastructure managed by the Third-Party Expert, the Match Maker Server is smaller as it requires less computation power

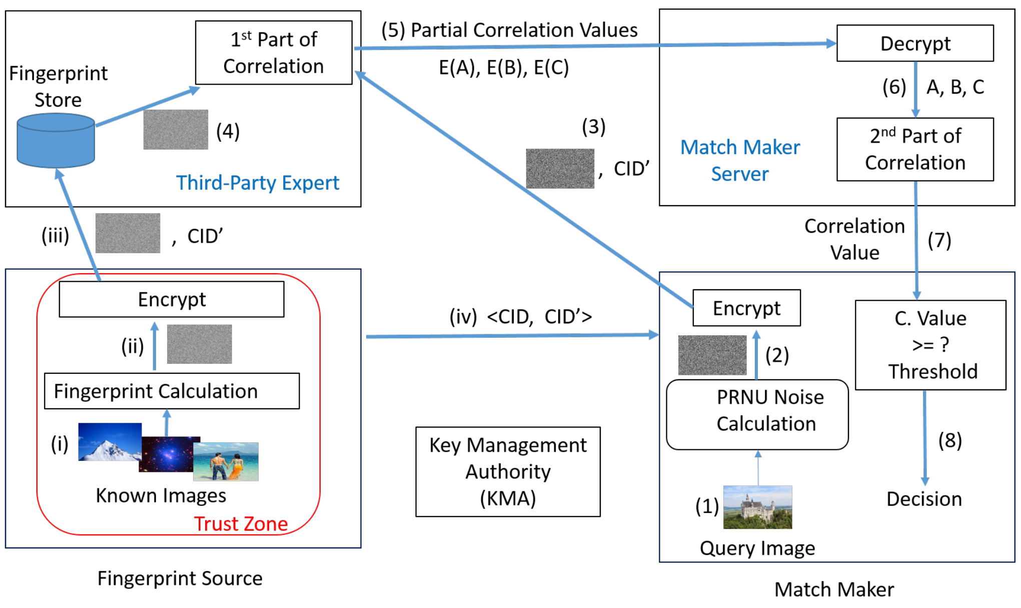

Fig. 2: The overview of *PANDORA*: The TrustZone takes as input some a set of known images (Step i) and calculates a fingerprint (Step ii). The fingerprint is encrypted and sent to the Fingerprint Store (Step iii). Upon request, a Match Maker takes as input a query image (Step 1) and calculates the PRNU noise (Step 2). The noise is encrypted and sent to the Third-Party Expert (Step 3). The Third-Party Expert fetches the stored fingerprints from the Fingerprint Store and, after performing encrypted matching, sends partial correlation values to the Match Maker Server (Step 5). The Match Maker Server decrypts partial correlation values (Step 6), computes the final correlation value and sends it to the Match Maker (Step 7). The Match Maker makes the final decision based on whether the final correlation value is above a certain threshold (Step 8).

and storage for performing its tasks.

- **Key Management Authority (KMA):** This entity is responsible for generating public and private keys of BGN encryption.

Threat Model: We assume the KMA as a fully trusted entity. The KMA can be under the direct control of the law enforcement authority. We also assume that the KMA securely transfers the keys to the Fingerprint Source, the Match Maker, and the Match Maker Server. The one-time key generation process can be pre-processed. Thus, the KMA can be assumed to be kept offline in most of the time. The Fingerprint Source, the Match Maker, and the Match Maker Server are also assumed to be fully trusted entities.

The Third-Party Expert is assumed to be *honest but curious* entity. That is, the Third-Party Expert is trusted to honestly perform its duty. However, it is not trusted to guarantee data confidentiality. The adversary can be either an outsider or even an insider, such as an unfaithful employee working for the Third-Party Expert. Furthermore, we assume that the Third-Party Expert has mechanisms to deal with the data integrity and availability.

V. Proposed Approach

In this section, we provide an overview of our approach for encrypted source camera attribution. Figure 2 shows how the different entities in our architecture are interconnected. In our approach, the KMA generates encryption keys, and sends the public key to the Fingerprint Source and the Match Maker, and the private key to the Match Maker Server.

Given a set of non-tampered known images (Step i) of a camera, the Fingerprint Source first extracts the noise from the images, and then generates a fingerprint by combining the noise (as discussed in Section II). The Fingerprint Source ensures the integrity of the images either by taking and storing the images in a trusted environment (such as in TrustZone), or detect tampered images (and hence discard them) using various image forensic techniques [31]. The generated fingerprint is encrypted using the public key received from the KMA (Step ii). The real Camera ID (CID) associated with the encrypted fingerprint is anonymized by introducing a dummy CID$'$ such that a camera cannot be linked to the Fingerprint Source. The encrypted fingerprint together with the CID$'$ is sent to the Third-Party Expert for storage and further processing (Step iii). The $\langle$CID, CID$'\rangle$ tuple is sent to the Match Maker (Step iv)[1].

The Match Maker will either verify if a particular suspected camera has taken an anonymous query image (*i.e.,* if a suspected CID is attached to the query image), or in the case of no suspected camera, identify the culprit camera (*i.e.,* CID of the query image). Let us discuss the verification process first. Identification is similar to verification.

Given a non-tampered query image (Step 1), the Match Maker first extracts PRNU noise. The extracted noise (Step 2) is then encrypted. The encrypted noise and its CID$'$ (Step 3) are sent to the Third-Party Expert. Using the CID$'$, the Third-Party Expert retrieves the corresponding encrypted

[1]We assume that the communication between the Fingerprint Source and the Match Maker is protected.

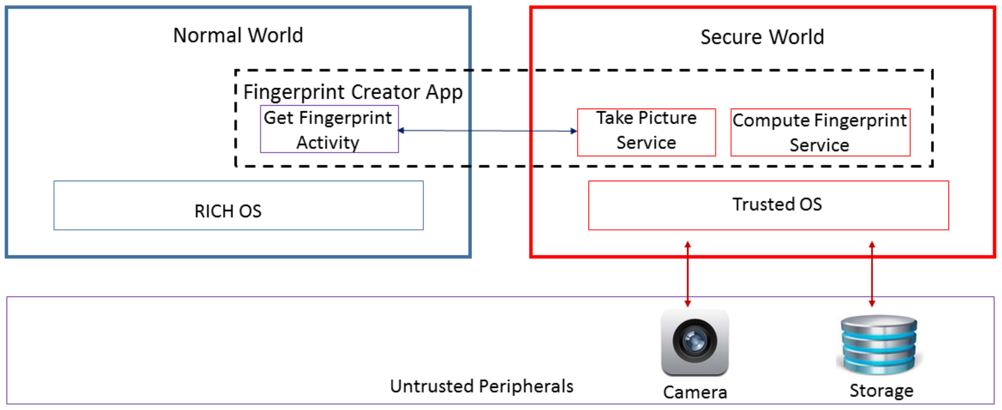

Fig. 3: Fingerprint computation using TrustZone.

fingerprint (Step 4), and correlates the fingerprint with the encrypted PRNU noise (*i.e.,* in the encrypted domain). Since the correlation process involves more complex operations than what homomorphically possible under the BGN encryption, the Third-Party Expert performs part of the correlation process on its end. The partial correlation result is sent to the Match Maker Server (Step 5). The Match Maker Server decrypts the encrypted result, and using the decrypted values (Step 6), completes the correlation operation. The correlation value is sent to the Match Maker (Step 7) that compares the correlation value with a threshold. If the correlation value is greater than or equal to the threshold, it is concluded that the image was taken by CID$'$, and hence by CID (Step 8).

The identification is similar to verification as the identification can be performed using a series of verifications. In case of identification, the Match Maker can perform one-by-one matching for the $\langle$CID, CID$'\rangle$ tuples in its database till the correlation value of a matching is more than or equal to the threshold.

VI. Solution Details

The main idea behind *PANDORA* is to compute the camera fingerprint and the PRNU noise of a query image in a trusted environment (such as the ARM TrustZone), and match the fingerprint with the PRNU noise in encrypted domain. In the following sections, we explain in details some of the steps executed in *PANDORA*.

A. Computing Fingerprint

In this article, we assume that the ARM TrustZone is available to securely compute the fingerprint of a device. Figure 3 outlines our approach. The images used in computing the fingerprint must be non-tampered. Thus, we take and store the images in the TrusZone.

In our approach, the suspect installs the *Fingerprint Creator App*, an application that has the main activity *Get Fingerprint* in the Rich Operating System (OS) running in the normal world. When the user launches the app, *Get Fingerprint* activity requests the support of the Trusted OS to start the *Take Picture* service in the TrustZone. The *Take Picture* service takes several pictures with the phone camera. Because the camera is not under the TrustZone secure environment, an adversary that does not want that the correct fingerprint is generated might try to modify the pictures before they reach the *Take Picture* service. To detect tampered images, the service uses various image forensic techniques [31]. If it detects that the pictures have been tampered with, it will discard them. Otherwise, the pictures are encrypted (before leaving the TrustZone) and stored in local storage. Once n number of pictures have been taken, the *Compute Fingerprint* service (still in the TrustZone) is invoked. The *Compute Fingerprint* service first fetches the images from the local storage, and then computes fingerprint using Winer filter-based method (as discussed in Section II-A). Finally, the *Compute Fingerprint* service encrypts the fingerprint using our encryption mechanism described in details in the following section.

Although the application presented is based on TrustZone, our work can be generalized to any computing platform where a secure environment is present, such as the Intel SGX. Another alternative, is to use a secure element such as a secure SIM card that can be deployed in the suspect's device.

B. Encrypting Fingerprint

We consider that the fingerprint vector is represented using a vector $F = \{F_1, F_2, \ldots, F_n\}$ of length n. Our goal is to encrypt the value of an element of the vector (*i.e.,* each F_i, where $1 \leq i \leq n$) using the BGN encryption. The positions of the vector are kept in plaintext to facilitate future correlations.

Each F_i, however, is a floating point number that is incompatible with the modular prime operation of the BGN scheme. Before encryption, we convert F_i to an integer F_i^{int} by first rounding of F_i by d decimal places and then multiplying 10^d to the round-off value.

The BGN scheme is computationally expensive. According to our experiments, encryption of a single floating point number requires 1.2 millisecond (ms). The encryption of a fingerprint computed from a standard 720×720 image requires 11 minutes (as the dimension of the fingerprint is equal to the dimension of the image). Such high computation overhead is not desirable for a practical solution. To overcome this issue, we used the concept of fingerprint digest [32].

A fingerprint digest is a trimmed down version of the fingerprint. The correlation of the fingerprint digest with the PRNU digest (produced by trimming down the PRNU noise) can produce similar error rates (*i.e.,* False Acceptance Rate – FAR – and False Rejection Rate – FRR) to that of the correlation of the fingerprint with the PRNU noise. The fingerprint digest therefore can be used instead of the fingerprint where low overhead is desired (at the cost of slightly higher error rate). The fingerprint digest of m (where $m < n$) elements contains those m pixels which are more significant to the correlation. For example, the defective hot pixels (represented with large positive values) and dead pixels (represented with large negative values) can be part of the fingerprint digest. In practice, the fingerprint digest is found by first sorting the fingerprint and then selecting m highest absolute value elements.

In our work, we replace the full fingerprint F with fingerprint digest $F_D \subset F$, and encrypt the integral representation of each element of F_D. The encrypted fingerprint digest $E(F_D)$ is then sent to the Third-Party Expert.

One of the key requirements of our approach is to determine the number of elements in the digest, *i.e.,* to find value of m. The value must be chosen in such a way that both error rate and overheads are reasonable. In this article, we experimentally set the value of m to $10,000$. For this number, FAR is 0.0521 and FRR is 0.12. In comparison to the non-digest method, the computational cost is decreased by more than 50 times.

TABLE I: Error rates for different k when $m = 10000$.

k	2000	5000	6000	7000	8000	9000
FAR	0.0222	0.0556	0.0444	0.0444	0.0667	0.0778
FRR	0.05	0.05	0.1	0.25	0.3	0.35

One optimization to further decrease the overhead can be encrypting a subset of k (where $k < m$) elements from the fingerprint digest and keep the remaining $m-k$ elements unencrypted. We explored this possibility, and found that this approach leaks information about the fingerprint. As a matter of fact, the correlation of $m-k$ unencrypted elements (even when k is large) can still attribute the correct camera with low error rate. For $m = 10000$, Table I shows the error rates for different k that we experimentally obtained. Therefore, to avoid leaking information that could lead to privacy breaches, we decided to encrypt all elements of the fingerprint digest F_D. The encrypted fingerprint digest $E(F_D)$ is then sent to the Third-Party Expert to perform the encrypted matching with the encrypted PRNU noise $E(F')$ of a query image. The details of this step are discussed in the next section.

C. Matching Fingerprint with the PRNU Noise

The Third-Party Expert obtains the encrypted fingerprint digest $E(F_D)$ from the suspect's device and the encrypted PRNU noise $E(F')$ of a query image from the court. To perform the match, the Third-Party Expert first obtains an encrypted PRNU digest $E(F'_D)$ from the encrypted PRNU using the location information of the elements of the fingerprint digest. The encrypted fingerprint digest and the encrypted PRNU digest are then input to the Pearson correlation coefficient formula given in Equation 1.

The Pearson correlation coefficient contains additions, scalar multiplications, one multiplication, a division, and a square root operations. The additions, scalar multiplications, and one multiplication operation can be performed in the encrypted domain as the BGN encryption is homomorphic to these operations. Thus, using the encrypted fingerprint digest $E(F_D)$ and the encrypted PRNU noise digest $E(F')$, the Third-Party Expert computes in the encrypted domain the following components of the Pearson correlation coefficient:

$$E(A) = \sum_{i=1}^{m} \left(E(F_i^{int}) - \overline{E(F^{int})}\right)\left(E(F_i^{int,\prime}) - \overline{E(F^{int,\prime})}\right),$$

$$E(B) = \sum_{i=1}^{|m} \left(E(F_i^{int}) - \overline{E(F^{int})}\right)^2,$$

and

$$E(C) = \sum_{i=1}^{m} \left(E(F_i^{int,\prime}) - \overline{E(F^{int,\prime})}\right)^2,$$

where $\overline{E(F^{int})}$ and $\overline{E(F^{int,\prime})}$ represent mean of $E(F^{int})$ and $E(F^{int,\prime})$ respectively. Then, the Third-Party Expert sends the encrypted components $E(A)$, $E(B)$, and $E(C)$ to the Match Maker Server for further processing.

The Match Maker Server performs the remaining operations of the Pearson correlation coefficient in plaintext form. To this end, (1) the Match Maker Server obtains A, B, and C by decrypting $E(A)$, $E(B)$, and $E(C)$; (2) converts back A, B, and C to their float values (by dividing 10^d); and (3) computes the correlation coefficient $r'(F_D, F'_D)$ as

$$r'(F_D, F'_D) = \frac{A}{\sqrt{BC}}.$$

As last step of the process, the Match Maker obtains the correlation coefficient from the Match Maker Server, and compares the coefficient with a threshold to determine if the query image was taken by the suspect's camera.

Note that since our scheme operates in integer domain and we consider fingerprint digest, it could be the case that our scheme might introduce some error when compared to conventional camera attribution method that considers full fingerprint and operates in floating point domain. However, the effect of fingerprint digest in camera attribution has been well studied in earlier work [32]. In the following section, we will therefore only study the effect of rounding error introduced by *PANDORA*.

D. Rounding Error Analysis

Without loss of generality, let us assume that both an element of the fingerprint digest, *i.e.,* F_i, and an element of the PRNU digest, *i.e.,* F_i', are rounded off by d decimal places. Suppose the error in rounding the element of the fingerprint and rounding the element of the PRNU are denoted as ε_{F_i} and $\varepsilon_{F_i'}$ respectively. Then, we can write $F_i^r = F_i + \varepsilon_{F_i}$ and $F_i^{r,\prime} = F_i' + \varepsilon_{F_i'}$, where F_i^r and $F_i^{r,\prime}$ are round-off values of F_i and F_i', respectively. The fingerprint and PRNU are assumed to be Gaussian with zero mean and unit variance. Thus, the rounding errors can also be assumed to be Gaussian with zero mean and unit variance.

Using Equation 1, we can get the error in correlation coefficient ε_r as

$$\varepsilon_r = \frac{\sum_{i=1}^n(\varepsilon_{F_i}F_i^{r,\prime} + \varepsilon_{F_i'}F_i^r + \varepsilon_{F_i}\varepsilon_{F_i'})}{\sqrt{\sum_{i=1}^n(F_i^r)^2}\sqrt{\sum_{i=1}^n(F_i^{r,\prime})^2}}. \quad (2)$$

Due to the property of Pearson correlation coefficient, we know that

$$r = \frac{\sum_{i=1}^n F_i^r F_i^{r,\prime}}{\sqrt{\sum_{i=1}^n(F_i^r)^2}\sqrt{\sum_{i=1}^n(F_i^{r,\prime})^2}}$$

is bounded by ± 1. We also know that both ε_{F_i} and $\varepsilon_{F_i'}$ satisfy $-0.5\times 10^{-d} \le \varepsilon_{F_i} \le 0.5\times 10^{-d}$ and $-0.5\times 10^{-d} \le \varepsilon_{F_i'} \le 0.5\times 10^{-d}$, respectively. In average case, $|\varepsilon_{F_i}| < F_i^r \times 10^{3-d}$ and $|\varepsilon_{F_i^{r,\prime}}| < F_i' \times 10^{3-d}$, as very few elements in the fingerprint and the PRNU are less than 0.01 (based on experimental observation). By putting the average case values in Equation 2, we obtain

$$|\varepsilon_r| < \frac{3r}{10^{3-d}}.$$

TABLE II: Change in error rates for different d when $p = 0.0157$.

d	1	2	3	4
FAR	0.0313	0.0500	0.0563	0.0563
FRR	0.5000	0.1000	0.0500	0.0500

In other words, ε_r can change the value of r maximum by $|\frac{3r}{10^{3-d}}|$, *i.e.,* $r' \le r + \frac{3r}{10^{3-d}}$. This change can affect the FAR and FRR rates of the correlation since r' can be greater than (or less than) a threshold T while r is less than (or greater than) T. To decrease this error in the correlation, a higher d must be chosen such that the probability of wrongful attribution decreases (as ε_r will be small enough to make an impact). A way to choose d can be $d > 3 + p$, where p represents the number of decimal places in T. The higher the value of d, the lower the error (as shown in Table II for a single camera). For minimal error in a practical scenario, the value of d can be fixed to the machine precision. Note that fixing the value of d to machine precision of a normal PC (where a float is typically represented by a 4-byte or 8-byte number) will not increase the computation cost and data overhead since the number has to be represented as a big number (*e.g.,* a 32-byte number) after the encryption.

Note that rounding errors are not affected by encryption and decryption as both encryption and decryption are lossless operations.

VII. Construction Details

PANDORA leverages the BGN scheme proposed by Boneh, Goh and Nissim in [7]. The BGN scheme is somewhat homomorphic in a sense that it allows an arbitrary number of additions and a single multiplication operation. The proposed scheme consists of the following algorithms.

- **KeyGen(1^k).** The KMA runs the key generation algorithm in order to generate the public key PK and the secret key SK. It takes as input a security parameter k and generates two prime numbers q_1 and q_2. It computes $n = q_1q_2$. It picks two random generators $g, u \xleftarrow{R} \mathbb{G}$ of order n. It computes $h = u^{q_2}$, which is a random generator of the subgroup of $\mathbb{G}$ of order q_1. It outputs $\mathbb{G}_{\mathbb{T}}$. It defines a bilinear map: $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_{\mathbb{T}}$, which has the properties of *bilinearity*, *computability* and *non-degeneracy* [33]. The public key is $PK = (n, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, e, g, h)$. The secret key is $SK = q_1$.
- **Enc(PK, m).** To encrypt a message m (where $m < q_2$) using PK, the user runs the encryption algorithm. It picks a random $r \xleftarrow{R} \mathbb{Z}_n$. It computes $C = g^m h^r \in \mathbb{G}$. It outputs C.
- **Dec(SK, C).** To decrypt a ciphertext C using SK, the user runs the decryption algorithm. It computes $C^{q_1} = (g^{q_1})^m$. Let $\hat{g} = g^{q_1}$. Let $\hat{C} = C^{q_1}$. To recover m, it takes the discrete log of $\hat{C}$ base $\hat{g}$, *i.e.,* $m = \log_{\hat{g}}\hat{C}$.
- **Add(PK, C_1, C_2).** To add two ciphertexts $C_1 = g^{m_1}h^{r_1}$ and $C_2 = g^{m_2}h^{r_2}$, the server runs the addition algorithm. It picks a random $r \xleftarrow{R} \mathbb{Z}_n$. It calculates h^r. It computes C as follows.

$$\begin{aligned} C &= C_1 \cdot C_2 \cdot h^r \\ &= g^{m_1}h^{r_1} \cdot g^{m_2}h^{r_2} \cdot h^r \\ &= g^{m_1+m_2}h^{r_1+r_2+r} \end{aligned}$$

  Note that h^r is optional. More specifically, h^r is used for re-randomization. Avoiding blinding with h^r will make the homomorphic computation deterministic.
- **Mul(PK, C_1, C_2).** To multiply two ciphertexts $C_1 = g^{m_1}h^{r_1}$ and $C_2 = g^{m_2}h^{r_2}$, the server runs the multiplication algorithm. It computes $g_1 = e(g, g)$, which is of order n. Next, it computes $h_1 = e(g, h)$, which is of order q_1. It picks a random $r \xleftarrow{R} \mathbb{Z}_n$. It calculates ${h_1}^r$. Recall that $h = u^{q_2}$, which can also be re-written as $h = g^{\alpha\cdot q_2}$ for

some (unknown) $\alpha \in \mathbb{Z}$. It computes C as follows.

$$\begin{aligned}
C &= e(C_1,C_2)\cdot {h_1}^r \\
&= e(g^{m_1}h^{r_1},g^{m_2}h^{r_2})\cdot {h_1}^r \\
&= e(g^{m_1},g^{m_2})\cdot e(g^{m_1},h^{r_2})\cdot e(h^{r_1},g^{m_2})\cdot e(h^{r_1},h^{r_2})\cdot {h_1}^r \\
&= e(g,g)^{m_1m_2}\cdot e(g,h)^{m_1r_2}\cdot e(h,g)^{r_1m_2}\cdot e(h,h)^{r_1r_2}\cdot {h_1}^r \\
&= {g_1}^{m_1m_2}\cdot {h_1}^{m_1r_2}\cdot e(g^{\alpha q_2},g)^{r_1m_2}\cdot e(g^{\alpha\cdot q_2},h)^{r_1r_2}\cdot {h_1}^r \\
&= {g_1}^{m_1m_2}\cdot {h_1}^{m_1r_2}\cdot e(g,g^{\alpha q_2})^{r_1m_2}\cdot e(g,h)^{\alpha q_2r_1r_2}\cdot {h_1}^r \\
&= {g_1}^{m_1m_2}\cdot {h_1}^{m_1r_2}\cdot e(g,h)^{r_1m_2}\cdot {h_1}^{\alpha q_2r_1r_2}\cdot {h_1}^r \\
&= {g_1}^{m_1m_2}\cdot {h_1}^{m_1r_2}\cdot {h_1}^{r_1m_2}\cdot {h_1}^{\alpha q_2r_1r_2}\cdot {h_1}^r \\
&= {g_1}^{m_1m_2}\cdot {h_1}^{m_1r_2+r_1m_2+\alpha q_2r_1r_2+r} \\
&= {g_1}^{m_1m_2}\cdot {h_1}^{\hat{r}}
\end{aligned}$$

where $\hat{r} = m_1r_2 + r_1m_2 + \alpha q_2r_1r_2 + r$ is distributed uniformly in $\mathbb{Z}$. Thus, C denotes the encryption of m_1m_2 mod n, but in $\mathbb{G}_\mathbb{T}$ rather than $\mathbb{G}$. Clearly, the system is still additively homomorphic in $\mathbb{G}_\mathbb{T}$. Note that h^r is optional. More specifically, h^r is used for re-randomization. Avoiding blinding with h^r will make the homomorphic computation deterministic.

VIII. Security Analysis

In this section, we present the security analysis of *PANDORA*. Since *PANDORA* is based on the BGN scheme, our security analysis is also based on similar settings as in [7]. In general, a scheme is considered secure if no adversary can break the scheme with probability significantly greater than random guessing. The adversary's advantage in breaking the scheme should be a negligible function (defined below) of the security parameter.

Definition 1 (Negligible Function): A function f is negligible if for each polynomial $p(.)$, there exists K such that for all integers $k > K$ it holds that:

$$f(k) < \frac{1}{p(k)}$$

We consider a realistic adversary that is computationally bounded and show that our scheme is secure against such an adversary. We model the adversary as a randomized algorithm that runs in polynomial time and show that the success probability of any such adversary is negligible. An algorithm that is randomized and runs in polynomial time is called a Probabilistic Polynomial Time (PPT) algorithm.

The scheme relies on the existence of a pseudorandom function f. Intuitively, the output a pseudorandom function cannot be distinguished by a realistic adversary from that of a truly random function. Formally, a pseudorandom function is defined as:

Definition 2 (Pseudorandom Function): A function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is pseudorandom if for all PPT adversaries $\mathscr{A}$, there exists a negligible function *negl* such that:

$$|Pr[\mathscr{A}^{f_s(\cdot)} = 1] - Pr[\mathscr{A}^{F(\cdot)} = 1]| < negl(k)$$

where $s \to \{0,1\}^k$ is chosen uniformly randomly and F is a function chosen uniformly randomly from the set of function mapping n-bit string to n-bit string.

Our proof relies on two assumptions. The first assumption is that the Discrete Log (DL) problem is hard in $\mathbb{G}$, *i.e.,* given g and g^α, it is hard for an adversary to compute α. The second assumption is that the integer factorization of a large composite number is hard, *i.e.,* given n, it is hard compute (non-trivial) q_1 and q_2 such that $n = q_1 \cdot q_2$.

Without knowing the factorization of the group order n, it is hard to decide whether x is in a subgroup of $\mathscr{G}$. This problem is known as the *subgroup decision problem* and we formally define it as follows.

Definition 3 (Subgroup Decision Problem): Let $\mathscr{G}$ and $\mathbb{G}_\mathbb{T}$ are groups of order $n = q_1q_2$, where q_1 and q_2 are primes. Let g be a generator of $\mathscr{G}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathbb{T}$ be the bilinear map. Let x is an element of $\mathscr{G}$, *i.e.,* $x \in \mathscr{G}$. For an algorithm $\mathscr{A}$, the advantage of $\mathscr{A}$ in solving the subgroup decision problem is defined as:

$$\begin{aligned}
SD-Adv(\mathscr{A},\mathbb{G}) = |Pr[\mathscr{A}(n,\mathscr{G},\mathbb{G}_\mathbb{T},e,x) = 1] - \\
Pr[\mathscr{A}(n,\mathscr{G},\mathbb{G}_\mathbb{T},e,x^{q_2}) = 1]| < negl(k)
\end{aligned}$$

We say that $\mathscr{G}$ satisfies the subgroup decision problem if for all PPT adversaries $\mathscr{A}$, $SD-Adv(\mathscr{A},\mathbb{G}) < negl(k)$.

Now, we can prove security of the BGN scheme.

Theorem 1: If the subgroup decision problem is hard relative to $\mathbb{G}$, the BGN scheme is semantically secure.

Proof (Sketch). Let us assume that the BGN scheme is not semantically secure and there exists a PPT $\mathscr{B}$ that can break the BGN scheme in *negl*(k). We assume there exists an adversary $\mathscr{A}$ that breaks the subgroup decision assumption with the same advantage. Given $(n,\mathscr{G},\mathbb{G}_\mathbb{T},e,x)$, $\mathscr{A}$ works as follows:

- $\mathscr{A}$ chooses a random generator $g \in \mathbb{G}$ and gives the public key $(n,\mathscr{G},\mathbb{G}_\mathbb{T},e,x)$ to $\mathscr{B}$.
- $\mathscr{B}$ outputs two messages m_0 and m_1 to which $\mathscr{A}$ responds with $C = g^{m_b}h^r \in \mathbb{G}$ for a random $b \overset{R}{\leftarrow} \{0,1\}$ and $r \overset{R}{\leftarrow} \mathbb{Z}_n$.
- $\mathscr{B}$ outputs $b' \in \{0,1\}$. If $b = b'$, $\mathscr{A}$ outputs 1 (meaning x is uniform in a subgroup of $\mathscr{G}$); otherwise, $\mathscr{A}$ outputs 0 (meaning x is uniform in $\mathscr{G}$).

As we know that x is uniform in $\mathscr{G}$, the challenge ciphertext C is uniformly distributed in $\mathscr{G}$. Thus,

$$Pr[b = b'] = 1/2$$

On the other hand, if x is uniform in the (q_1) subgroup of $\mathscr{G}$, the public key $(n,\mathscr{G},\mathbb{G}_\mathbb{T},e,x)$ and the challenge ciphertext C given to $\mathscr{B}$ are the ones that are given in the real game. By definition of $\mathscr{B}$, we know that:

$$Pr[b = b'] > 1/2 + negl(k)$$

$\mathscr{A}$ satisfies $SD-Adv(\mathscr{A},\mathbb{G}) > negl(k)$. This implies that $\mathscr{A}$ breaks the subgroup decision problem with *negl*(k).

Note that the proof is for $\mathbb{G}$. Without loss of generality, it also holds for $\mathbb{G}_\mathbb{T}$. For more details, an interested reader is referred to [7].

Also note that both homomorphic operations including addition (*i.e.,* **Add**) and multiplication (*i.e.,* **Mul**) functions are pseudorandom due to blinding with h^r. h^r makes both homomorphic operations semantically secure. If we remove it, then both homomorphic operations will not be semantically secure.

Theorem 2: If the BGN scheme is semantically secure, *PANDORA* is also semantically secure.

Proof (Sketch). In our solution, recall that we represent the camera fingerprint and the PRNU noise of the query image using two integer vectors (the conversion from a floating point number to an integer has been explained in Section VI-B),

$$F = \{F_1, F_2, \ldots, F_n\}$$

and

$$F' = \{F'_1, F'_2, \ldots, F'_n\}$$

respectively, where each vector is of length n. In our solution, we perform matching of the camera fingerprint with the PRNU noise of the query image. Since this matching should be performed without revealing the data, all the elements in each vector are encrypted using the BGN scheme. This matching is based on the correlation coefficient explained in Section VI-C, requiring addition and multiplication operations. Technically, the respective elements (*i.e.,* F_i and F'_i, where $1 \leq i \leq n$) in both vectors are added and multiplied in an encrypted manner. As we use semantically secure addition and multiplication operations, the resultant elements after running the correlation are also semantically secure. Thus, the correlation vector consisting of the resultant elements is also semantically secure.

IX. Results and Analysis

In this section, we discuss the experimental results of *PANDORA*. The experiments were performed by executing the Third-Party Expert, Match Maker Server, and the Match Maker in our Lab's infrastructure.

We simulated the TrustZone using Open-TEE [2], a virtual, hardware-independent software-based TrustZone, that was installed on a PC powered by Intel Core i5-3570 Quad-Core 3.4 GHz processor and 8 GB RAM. Our version of Open-TEE runs on Ubuntu 16.04 LTS operating system. In Open-TEE, we implemented the fingerprint and fingerprint encryption modules using C++ language. Our fingerprint module was based on the Winer filter, which was discussed in Section II. In this module, we also computed the fingerprint digest by considering 10000 highest elements of the fingerprint. In our fingerprint encryption module, we first converted the floating point numbers in the fingerprint digest to integers by rounding off the floats by two decimal places, and then encrypted the integers using BGN encryption. Our implementation of BGN encryption had a key length of 1024 bits.

The Third-Party Expert was executed in a cluster of 64 nodes, to recreate a large deployment on a cloud environment. Each node was powered by Intel Xeon E5-2680 8 Core 2.7GHz processor and 128G RAM, and was running Red Hat Enterprise Linux 6.3 OS. The nodes were connected using QDR Infiniband (40Gb/s). We used MPI (Message Passing Interface) message passing system to maintain communication among the nodes. The Third-Party Expert was implemented using C++ language. Note that for lower computation cost, we used a variant of Pearson correlation coefficient[3] in our implementation.

[2]https://github.com/Open-TEE

We executed the Match Maker Server and the Match Maker on a single node that is powered by Intel Xeon E5-2680 8 Core 2.7GHz processor and 128G RAM. As we have clarified before, the Match Maker does two jobs: computation and encryption of the PRNU noise of the query image. On the other hand, the Match Maker Server decrypts the partially-computed Pearson correlation coefficient, completes the computation of the Pearson correlation coefficient, and matches the Pearson correlation coefficient with a threshold. The computation of the PRNU noise and the encryption of PRNU computed noise were implemented in C++ language on Ubuntu 16.04 LTS platform. The PRNU noise was computed using a Winer filter-based method discussed in Section II, and the PRNU noise was also encrypted using BGN encryption. The decryption of the partially-computed Pearson correlation coefficient, and the remaining computation of the decrypted Pearson correlation coefficient were also implemented using C++ language.

TABLE III: List of cameras used in our experimentation.

Name	**No.**	**Name**	**No.**
HUAWEI H30	0	Casio EX-Z150	5
iPadMini	1	Kodak M1063	6
Bobi LD700	2	Nikon CoolPix S710	7
SONY C6903	3	Olympus Mju 1050SW	8
Samsung Galaxy S4	4	Samsung L74	9

Our test environment consisted of 10 cameras of 10 different brands as shown in Table III. From each camera, we took 10 images for calculating the fingerprint and 40 query images for the correlation. The chosen images had not gone through geometric processing, such as scaling, cropping, and rotation. The generated Pearson correlation coefficient was matched against a threshold that was set per each camera.

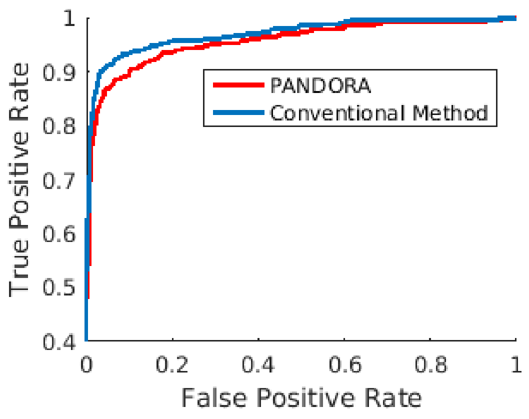

Fig. 4: ROC curve: *PANDORA* vs conventional scheme.

To study the practicality of *PANDORA*, we compared *PANDORA* with the plaintext domain camera attribution that was implemented using the above setup. In the plaintext domain scheme, we computed fingerprint in Open-TEE and computed Pearson correlation coefficient in the cluster of

[3]http://www.stat.wmich.edu/s216/book/node122.html

nodes (*i.e.,* Third-Part Expert). Since the fingerprint and PRNU noise were not encrypted (floating point numbers were also not rounded off), the Pearson correlation coefficient was fully computed by the Third-Part Expert without any involvement of the Match Maker Server. The Match Maker computed the PRNU noise of the query image, and compared the Pearson correlation coefficient with the threshold.

Figure 4 shows the comparison of ROC curve of *PANDORA* (in red) with the ROC curve of the plaintext-based conventional scheme (in blue). These curves were obtained by averaging the error rates experimentally obtained from 10 different cameras. As can be seen in the graph, *PANDORA* has comparable error rates with respect to the conventional scheme. This means that even if we round off values to perform the encryption, the lack of precision does not affect too much the validity of the results obtained with *PANDORA*.

1) Performance Analysis: In this section, we analyze the computational and storage overheads incurred by *PANDORA*. Clearly, because our scheme uses encryption it introduces more overhead when compared to a plaintext scheme. Our goal here is to study and quantify such overhead.

In *PANDORA*, the encryption of an element of the fingerprint or the encryption of an element of the PRNU noise involves one round-off operation, three exponential operations, and two multiplication operations. The encryption of the fingerprint is a one-time operation. This operation therefore can be performed offline. The encryption of the PRNU noise, however, needs to be performed by the Match Maker at runtime. The addition, scalar multiplication, and multiplication of the encrypted values need to be implemented using addition, multiplication, and exponentiation. Thus, in comparison to the conventional scheme, the Third-Party Expert needs more computation cost to partially compute the Pearson correlation coefficient. The partially-computed Pearson correlation coefficient needs to be decrypted by the Match Maker Server. This process requires 12 exponential operations, three discrete log operations, and one division operation. Note that the operation at the Third-Party Expert-end and Match Maker Server-end need to be performed at runtime.

Similar to the computational cost, *PANDORA* also increases the data overhead. Each floating point number is represented as a b-bit integer, where b is the number of bits in encryption key. Typically, $b = 1024$ and the float is represented as 32 bits. Thus, the data overhead in storing encrypted fingerprint and sending encrypted fingerprint to the Third-Party Expert is increased by 32 times. Similarly, the data overhead in sending encrypted PRNU is also increased by 32 times.

TABLE IV: Computation cost of *PANDORA* and conventional scheme at the Third-Party Expert-end.

Scheme	Time
Conventional scheme	0.54 ms
PANDORA (#1 node)	404.17 s
PANDORA (#4 nodes)	102.58 s
PANDORA (#16 nodes)	26.60 s
PANDORA (#64 nodes)	10.26 s

In addition to the above analysis, we also experimentally studied the computation cost of *PANDORA* and the conventional plaintext scheme. In our experiment, extra computation cost at the TrustZone is negligible. Therefore, we do not report it. The significant computation overhead at the Third-Party Expert-end, however, is reported in Table IV for a different number of nodes in the cluster. As expected, the computation cost decreases with the increase in the number of nodes used by the Third-Party Expert. The rate of decrease however is not strictly proportional to the rate of increase as the communication cost increases when the number of nodes increases. In our experiment, the computation cost at the Match Maker Server-end and Match Maker-end are 23.05s and 42.73s, respectively.

X. Conclusion

PRNU-based camera attribution is a widely used technique to identify the source camera of an anonymous questionable image. In this technique, a fingerprint of the camera is first computed from the PRNU noise of a set of images taken by the camera. Then this fingerprint is correlated with the PRNU noise of the query image to determine if the camera has taken the image. Although the PRNU-based method is very useful for digital forensics, privacy is one of the main concerns. Using this method, an adversary can unlawfully link a camera with an anonymous image, or vice versa. As a result, unethical linking of individuals to sensitive/anonymous information can occur. For example, anonymous social network accounts can be linked to known accounts if images of both the accounts belong to the same camera. In this article, we address this privacy concern by encrypting both the fingerprint and the noise, and performing the correlation in the encrypted domain. The PRNU noise and the camera fingerprint are computed in unencrypted form, but in trusted environment (*e.g.,* in the ARM TrustZone). *PANDORA* is such that the camera attribution can be performed by authorized users, who hold decryption keys. Experiment and analysis showed that *PANDORA* incurs reasonable overhead.

References

[1] J. Lukáš, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[2] A. E. Dirik, H. T. Sencar, and N. Memon, "Digital single lens reflex camera identification from traces of sensor dust," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 539–552, 2008.

[3] S. Bayram, H. T. Sencar, and N. Memon, "Seam-carving based anonymization against image & video source attribution," in *IEEE International Workshop on Multimedia Signal Processing*, 2013, pp. 272–277.

[4] A. E. Dirik, H. T. Sencar, and N. Memon, "Analysis of seam-carving-based anonymization of images against PRNU noise pattern-based source attribution," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 12, pp. 2277–2290, 2014.

[5] F. Bertini, R. Sharma, A. Iannì, and D. Montesi, *Smartphone verification and user profiles linking across social networks by camera fingerprinting*. Springer International Publishing, 2015, pp. 176–186.

[6] A. Karaküçük, A. E. Dirik, H. T. Sencar, and N. Memon, "Recent advances in counter PRNU based source attribution and beyond," *IS&T Electronic Imaging – Media Watermarking, Security, and Forensics*, vol. 9409, April 2015.

[7] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proceedings of the Second International Conference on Theory of Cryptography*, 2005, pp. 325–341.

[8] H. T. Sencar and N. Memon, *Digital image forensics: There is more to a picture than meets the eye*. New York, USA: Springer, 2013.

[9] S. Bayram, H. T. Sencar, and N. Memon, “Efficient techniques for sensor fingerprint matching in large image and video databases,” pp. 754 109–754 109, 2010.

[10] Y. Sutcu, S. Bayram, H. T. Sencar, and N. Memon, “Improvements on sensor noise based source camera identification,” in *International Conference on Multimedia and Expo*, 2007, pp. 24–27.

[11] C. T. Li and Y. Li, “Color-decoupled photo response non-uniformity for digital image forensics,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 2, pp. 260–271, 2012.

[12] G. Chierchia, S. Parrilli, G. Poggi, C. Sansone, and L. Verdoliva, “On the influence of denoising in PRNU based forgery detection,” in *ACM Workshop on Multimedia in Forensics, Security and Intelligence*, 2010, pp. 117–122.

[13] C. T. Li, “Source camera identification using enhanced sensor pattern noise,” *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 280–287, 2010.

[14] R. Caldelli, I. Amerini, F. Picchioni, and M. Innocenti, “Fast image clustering of unknown source images,” in *IEEE International Workshop on Information Forensics and Security*, 2010, pp. 1–5.

[15] J. Lukáš, J. Fridrich, and M. Goljan, “Detecting digital image forgeries using sensor pattern noise,” in *Electronic Imaging 2006*. International Society for Optics and Photonics, 2006, pp. 60 720Y–60 720Y.

[16] G. Chierchia, G. Poggi, C. Sansone, and L. Verdoliva, “A Bayesian-MRF approach for PRNU-based image forgery detection,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 4, pp. 554–567, 2014.

[17] T. Gloe, M. Kirchner, A. Winkler, and R. Böhme, “Can we trust digital image forensics?” in *ACM International Conference on Multimedia*, 2007, pp. 78–86.

[18] A. Karaküçük and A. E. Dirik, “Adaptive photo-response non-uniformity noise removal against image source attribution,” *Digital Investigation*, vol. 12, no. C, pp. 66–76, 2015.

[19] A. E. Dirik and A. Karaküçük, “Forensic use of photo response non-uniformity of imaging sensors and a counter method,” *Optics Express*, vol. 22, no. 1, pp. 470–482, January 2014.

[20] R. Böhme and M. Kirchner, “Counter-forensics: Attacking image forensics,” in *Digital Forensics*, H. T. Sencar and N. Memon, Eds. Springer-Verlag, 2013, pp. 327–366.

[21] J. Entrieri and M. Kirchner, “Patch-based desynchronization of digital camera sensor fingerprints,” in *IS&T Electronic Imaging – Media Watermarking, Security, and Forensics*, 2016.

[22] M. Goljan and J. Fridrich, “Sensor fingerprint digests for fast camera identification from geometrically distorted images,” in *SPIE Media Watermarking, Security, and Forensics*, 2013.

[23] ——, “Camera identification from cropped and scaled images,” in *Electronic Imaging 2008*. International Society for Optics and Photonics, 2008, pp. 68 190E–68 190E.

[24] K. Rosenfeld, T. Sencar, and N. Memon, “A study of the robustness of PRNU-based camera identification,” in *SPIE Conference on Media Forensics and Security*, 2010, pp. 90–93.

[25] M. Goljan, J. Fridrich, and M. Chen, “Sensor noise camera identification: Countering counter-forensics,” in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2010, pp. 75 410S–75 410S.

[26] S. Taspinar, M. Mohanty, and N. Memon, “PRNU based source attribution with a collection of seam-carved images,” in *IEEE International Conference on Image Processing (Accepted)*, Phonix, USA, 2016.

[27] M. Mohanty, P. K. Atrey, and W. T. Ooi, “Secure cloud-based medical data visualization,” in *ACMMM*, Nara, Japan, 2012, pp. 1105–1108.

[28] M. Mohanty, W. T. Ooi, and P. K. Atrey, “Scale me, crop me, know me not: supporting scaling and cropping in secret image sharing,” in *IEEE ICME*, San Jose, USA, 2013.

[29] ——, “Secure cloud-based volume ray-casting,” in *IEEE CloudCom*, Bristol, UK, 2013.

[30] M. Mohanty, M. R. Asghar, and G. Russello, “2DCrypt: Image scaling and cropping in encrypted domains,” *IEEE Transactions on Information Forensics and Security*, no. 99, pp. 1–14, 2016.

[31] H. Farid, “Image forgery detection,” *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 16–25, 2009.

[32] M. Goljan, J. Fridrich, and T. Filler, “Managing a large database of camera fingerprints,” in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2010, pp. 754 108–754 108.

[33] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in *Advances in Cryptology - CRYPTO 2001*, ser. Lecture Notes in Computer Science, J. Kilian, Ed. Springer Berlin Heidelberg, 2001, vol. 2139, pp. 213–229.