A Practical System for Privacy-preserving Video Surveillance

Elmahdi Bentafat, M. Mazhar Rathore, and Spiridon Bakiras

Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar
{ebentafat,mrathore,sbakiras}@hbku.edu.qa

Abstract. Video surveillance on a massive scale can be a vital tool for law enforcement agencies. To mitigate the serious privacy concerns of wide-scale video surveillance, researchers have designed secure and privacy-preserving protocols that obliviously match live feeds against a suspects' database. However, existing approaches provide stringent privacy guarantees and, as a result, they do not scale well for ubiquitous deployment. To this end, we introduce a system that relaxes the underlying privacy requirements by giving away some information when a face is compared against the law enforcement's database. Specifically, our protocol reveals a random permutation of obfuscated similarity scores, where each obfuscated score discloses minimal information about the actual similarity score. We show that, despite the relaxed security definitions, our system protects the privacy of the underlying faces, while offering significant improvements in terms of performance. In particular, our protocol necessitates a single round of communication between the camera and the server and, for a database of 100 suspects, the online computation time at the camera and the server is 155 ms and 34 ms, respectively, while the online communication cost is only 12 KB.

Keywords: video surveillance · biometric privacy · homomorphic encryption

1 Introduction

Video surveillance is being deployed in numerous countries around the world. An effective video surveillance system automatically monitors all available data feeds, extracts the individual faces (feature vectors), compares them against a suspects' database, and raises an alarm when a match is found. Nevertheless, this approach raises significant privacy concerns, because all individuals with known feature vectors can be tracked on a daily basis. Analyzing such information-rich datasets has the potential to reveal sensitive personal information, including home and work locations, health issues, religious affiliations, etc. Even if we trust the law enforcement authorities to protect the location privacy of their citizens, the stored location data may still be accessed by malicious users, such as rogue insiders or hackers.

As a result, the research community has proposed several methods [9, 17, 19] that perform privacy-preserving face recognition. In particular, these methods execute a secure two-party protocol between the camera and the database server, which lets the camera learn in zero knowledge whether a captured face matches one of the suspects in the database. If a match is found, the id number of the suspect is revealed; otherwise, the protocol discloses no information to either party. These protocols first compute an encrypted similarity score (Euclidean or Hamming distance) for each suspect in the database, using an additively homomorphic cryptosystem. Then, a variety of techniques are employed to identify the matching suspect, if and only if the underlying similarity distance is below a certain threshold. These techniques involve standard cryptographic primitives for secure computations, such as homomorphic encryption, garbled circuits, and oblivious transfer.

Nevertheless, all the aforementioned systems suffer from high computational and communication costs that render them impractical for wide-scale deployment. For instance, the Eigenfaces implementation by Sadeghi et al. [19] necessitates 40 sec of online computations to match a single face against a database of 320 suspects. In addition, the online communication cost is over 5 MB. Similarly, SCiFI [17] reports 31 sec of online computations for a database of 100 suspects. Another significant limitation of these protocols is their reliance on offline computation and communication that has to be performed for every face that is captured by the camera. While the offline tasks reduce the overall online cost dramatically, it is not feasible to process and store the underlying data for potentially millions of detected faces on a daily basis.

To this end, our work introduces the first practical system for privacy-preserving video surveillance on a large scale. The efficiency of our approach stems mainly from two design decisions. First, the suspects' database is distributed to all cameras, after it is encrypted with the public key of the law enforcement agency. As such, the expensive operations for computing the encrypted similarity scores are performed at the surveillance cameras, thus alleviating the server's computational load. The local database copy also allows the cameras to precompute most values that are involved in the encrypted distance computations. More importantly, unlike existing approaches, the offline computations are performed only once, during the system's initialization.

Our second decision is to relax the zero knowledge requirement of the face recognition protocol. In particular, for each captured face, the database server will learn a random permutation of *obfuscated* similarity scores between the captured face and all suspects in the database. As a result, the protocol involves a single round of communication for the server to learn the (binary) result of the identification. If a match is found, an additional verification protocol is invoked, where the server learns the suspect's id and optionally receives the image of the potential suspect. The relaxed privacy requirements also facilitate the use of an efficient elliptic curve cryptosystem (ElGamal) that reduces significantly the computational and communication costs. Nevertheless, as we will show, the

permuted and obfuscated scores are not sufficient for the server to infer any meaningful information about the underlying individuals.

We built our system on top of the OpenFace [2] platform that implements the face recognition layer. OpenFace is one of the most accurate open-source face recognition systems that employs Google's FaceNet [22] algorithm. Besides its high accuracy, a notable advantage of FaceNet over other approaches is its compact feature vector (just 128 bytes) that speeds up considerably the encrypted distance computations. We performed an extensive experimental evaluation of our system and demonstrated its applicability in a wide-scale video surveillance environment. Specifically, matching one face against a database of 100 suspects entails 190 ms of compute time and 12 KB of communication between the camera and the server. These costs are orders of magnitude lower compared to the current state-of-the-art systems.

The rest of the paper is organized as follows. Section 2 presents a literature review on privacy-preserving video surveillance and face recognition systems, and Section 3 discusses the main tools that we utilized in our implementation. Section 4 introduces our problem definition and describes the underlying threat model. Section 5 discusses in detail the operation of our system and Section 6 evaluates its security. A performance comparison against other approaches is introduced in Section 7, while the implementation details are presented in Section 8. Section 9 summarizes our experimental results and Section 10 concludes our work.

2 Related Work

2.1 Face recognition

Face recognition systems face several challenges–such as brightness, face position, and facial expression–that can highly influence the appearance of an individual. The first industrial face recognition applications were based on the Eigenfaces [26] technique. This groundbreaking study by Turk and Pentlandin in 1991 is the milestone for many other methods that have been introduced since then. The original Eigenfaces approach employed Principal Component Analysis (PCA) to generate the eigenvectors. Following that, other transformations were adopted, including Linear Discriminant Analysis (LDA) [23], Independent Component Analysis (ICA) [15], and Support Vector Machines (SVM) [13]. Additionally, some approaches combined multiple classifiers for dimensionality reduction and feature extraction [6, 25].

Lately, neural networks have also been employed in the face recognition domain to improve the classification accuracy [8, 20]. The current state-of-the-art algorithms are Facebook's DeepFace [24] and Google's FaceNet [22], both of which are based on Convolutional Neural Networks (CNNs). OpenFace [2] is a face recognition library written in Python that leverages the aforementioned CNN systems to provide better accuracy. In 2018, Tadas et al. introduced OpenFace 2.0 [3] as a C++ toolkit for facial behavior analysis. The implementation

targets a real-time environment and is, therefore, optimized in terms of computational cost. OpenFace 2.0 presents a more accurate facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation.

2.2 Privacy-preserving video surveillance

The first privacy-preserving face recognition protocol is due to Erkin et al. [9] in 2009. It leverages the Eigenfaces algorithm for face recognition, but is very inefficient in terms of online performance. Specifically, the protocol requires $O(\log M)$ rounds of online communication (M is the number of suspects in the database) and heavy public key homomorphic operations over the ciphertexts. Sadeghi et al. [19] improved the performance of Erkin's work by shifting some computations into a precomputation phase, and using garbled circuits [28] to compute the *Minimum* function. In a recent work, Xiang et al. [27] further improved upon the aforementioned protocols [9, 19] by outsourcing the expensive server computations to the cloud.

SCiFI [17] is the only protocol in the literature that is not based on the Eigenfaces representation. Instead, the authors proposed a novel face recognition method that takes into account the appearance of certain facial features. In SCiFI, each face is represented with a 900-bit vector, while the similarity score is simply the Hamming distance between two vectors. After the Hamming distance is computed, the result of the suspect identification is revealed through a 1-out-of-$d_{max}+1$ oblivious transfer protocol [16], where d_{max} is the maximum theoretical Hamming distance. One advantage of this approach is that Hamming distance computations on the ciphertext space are significantly faster than the Euclidean ones. Finally, various studies have used similar cryptographic tools, mainly garbled circuits and oblivious transfer, in the context of biometric identification. In particular, researchers have proposed several efficient protocols to compute the similarity scores, including Hamming distance, Euclidean distance, Mahalanobis distance, and scalar product [5, 10, 11, 14, 29].

3 Tools

3.1 OpenFace

OpenFace [2] is an open-source face verification and recognition system that maps face images to a compact Euclidean space. It is a deep convolutional network trained method for face recognition that achieves an accuracy of 92.95% on the Labeled Faces in the Wild (LFW) benchmark, one of the largest publicly-available datasets. OpenFace matches very well the performance of FaceNet [22] and DeepFace [24], despite the small size of the trained network. The advantage of OpenFace is the face representation efficiency that consists of 128 features. This vector can be reduced to just 128 bytes with a very small loss in recognition accuracy. The similarity score between two faces is represented by the Euclidean distance of the two feature vectors, and ranges between 0 (for the same image) and 4. A threshold $t = 0.9$ has been set empirically by the system developers of OpenFace, such that a distance less than t indicates a positive match.

3.2 Homomorphic encryption

Homomorphic cryptosystems [1] allow for the evaluation of certain arithmetic operations directly on the ciphertext domain. Fully homomorphic encryption (FHE) [12] supports both addition and multiplication operations and can, thus, be used to evaluate any circuit over encrypted data. Nevertheless, FHE schemes are still very inefficient to be used in real-time applications, such as video surveillance. Instead, similar to previous work, we built our protocol on top of additively homomorphic cryptosystems, such as Paillier [18] or ElGamal [7]. More specifically, we opted for an implementation of ElGamal's cryptosystem over elliptic curves, due to its computational efficiency and compact ciphertexts (128 bytes). The cryptosystem consists of the following functions:

- **Key generation**: Instantiate an elliptic curve group of prime order q with generator P. Choose a *private* key x uniformly at random from $\mathbb{Z}_q^*$ and set the *public* key $Q = x \cdot P$.
- **Encrypt**: Let m be the secret message. Choose r uniformly at random from $\mathbb{Z}_q^*$ and compute ciphertext $\mathsf{Enc}(m) = \langle r \cdot P, (m + r) \cdot Q \rangle$.
- **Decrypt**: Compute $m \cdot Q = (m + r) \cdot Q - x \cdot r \cdot P$ and solve the discrete log to recover m.

ElGamal's scheme is semantically secure and its security is based on the decisional Diffie-Hellman assumption. Note that, in our implementation, we utilized a look-up table of precomputed $m \cdot Q$ values (for all theoretically possible values of m) in order to speed up the discrete log computations at the database server.

4 Problem Definition and Threat Model

We assume a wide-scale surveillance environment, where a large number of cameras, equipped with moderate computational, storage, and communication capabilities, are deployed throughout a city. The database server (law enforcement) holds a database $\mathbb{S} = \{S_1, S_2, \ldots, S_M\}$ of M suspects, where each suspect S_i is represented by an N-th dimensional feature vector x_i. More specifically, the feature vectors are generated from OpenFace's deep learning model and consist of $N = 128$ values. During the system initialization, the database server shares an encrypted version of $\mathbb{S}$ (to be discussed later) with all cameras, using its own public key Q that is also known to all cameras. Every camera will then capture all passing-by faces and, for each candidate face C_j, compute its feature vector y_j using OpenFace's model. What follows, is a two-party protocol between the camera and the database server, where

- The server learns a random permutation π_j of obfuscated similarity scores between y_j and $x_i, \forall i \in \{1, 2, \ldots, M\}$.
- The camera learns nothing.

When the protocol's output is revealed to the server, it will immediately disclose the identification result. In particular, if all similarity scores are positive,

then the candidate face does not match any suspect in the database; otherwise, a negative score is a potential match for a suspect, whose identity is unknown due to the underlying permutation. In that case, the camera and the server invoke a separate two-party protocol, where the camera verifies that the similarity score is indeed negative. During that protocol,

- The server learns the actual id of the matching suspect and (optionally) receives the captured image from the camera.
- The camera learns the actual similarity score and id of the matching suspect.

We assume that the server and all cameras are semi-honest players. In other words, they will follow the protocols correctly, but try to infer some non-trivial information about the other party's input from the communication transcript. For example, the camera might want to learn the plaintext content of the suspects' database, while the server might want to infer some information about the captured faces that do not produce a database match. We also allow the server to act maliciously after the initial identification result, by falsely claiming that a certain similarity score is negative. Such behavior will be discovered during the subsequent verification protocol. Finally, we should note that our protocol cannot protect against illegitimate inputs from any of the parties. For instance, the server can insert into their database $\mathbb{S}$ an innocent civilian that it wants to track, while the camera can test whether a specific individual is part of $\mathbb{S}$ by using their feature vector in the identification protocol. However, none of the existing privacy-preserving protocols can protect against such attacks, since they are not cryptographic in nature.

5 System Description

In this section, we present in detail the operation of our privacy-preserving video surveillance system. We begin by introducing the offline/initialization phase of the protocol, and then proceed to describe the various elements involved in the online face identification process.

5.1 Offline phase

The server first instantiates an elliptic curve group of prime order q (as described in Section 3) and generates its public and private keys. The public key is distributed securely to all surveillance cameras in the city. Next, the server employs OpenFace to generate the feature vectors x_i for every suspect $S_i \in \mathbb{S}$. By default, OpenFace operates over floating point numbers, so we first had to convert the vectors into integers before applying any homomorphic operations. We empirically computed the normalization parameters for a floating point representation f as follows: $\lfloor f \times 400 + 128 \rfloor$, where $f \in \mathbb{Q} : -0.32 < f < 0.32$. With this transformation, every element in a feature vector is an integer in the range $[0, 256)$, thus allowing us to represent a vector with just 128 bytes. Furthermore, the transformation does not result in a significant loss of accuracy, as

illustrated in Table 1. Specifically, the table depicts the accuracy results from various state-of-the-art face recognition algorithms, and also quantifies the loss of accuracy due to the normalization of the features values. For our system, we trained the CNN model using integers in the range $[0, 256)$ instead of floats and, out of 13,233 images, we had only one misidentification compared to the original OpenFace implementation. Note that FaceNet and DeepFace are more accurate than OpenFace because they are trained on much larger datasets.

Table 1: Accuracy results on the LFW benchmark [2]

Model	**Accuracy**
Human	97.53%
EigenFaces	$60.02\% \pm 0.79$
FaceNet	$99.64\% \pm 0.9$
DeepFace	$97.35\% \pm 0.25$
OpenFace	$92.95\% \pm 1.34$
OpenFace, normalized	$92.92\% \pm 1.36$

Given suspect S_i's feature vector x_i and a potential candidate's vector y_i, the first step of FaceNet's face recognition algorithm is to compute the Euclidean distance between the two feature vectors. In the ciphertext domain, it is only feasible to compute the squared Euclidean distance, i.e.,

$$d_i^2 = \sum_{j=1}^{N} (x_{i,j} - y_{i,j})^2 = \sum_{j=1}^{N} (x_{i,j}^2 + y_{i,j}^2 - 2x_{i,j}y_{i,j}) \tag{1}$$

where $N = 128$ is the vector dimensionality. In the ciphertext domain over elliptic curves, this is equivalent to

$$\mathsf{Enc}(d_i^2) = \sum_{j=1}^{N} \mathsf{Enc}(x_{i,j}^2) + \sum_{j=1}^{N} \mathsf{Enc}(y_{i,j}^2) + \sum_{j=1}^{N} y_{i,j} \cdot \mathsf{Enc}(-2x_{i,j}) \tag{2}$$

Therefore, for the cameras to correctly compute $\mathsf{Enc}(d_i^2)$, the server will send them an encrypted version of the database $\mathbb{S}$, consisting of

- $\sum_{j=1}^{N} \mathsf{Enc}(x_{i,j}^2), \forall i \in \{1, 2, \ldots, M\}$.
- $\mathsf{Enc}(-2x_{i,j}), \forall i \in \{1, 2, \ldots, M\}, j \in \{1, 2, \ldots, N\}$.

As such, the offline communication cost of our protocol is $(N+1) \times M \times T$ bytes, where T is the size of an ElGamal ciphertext (typically 128 bytes). Due to the semantic security of the cryptosystem, the cameras cannot infer any information regarding the feature vectors of the suspects.

After a camera receives the encrypted database, it performs a series of offline precomputations, in order to speed up the online computation of the similarity

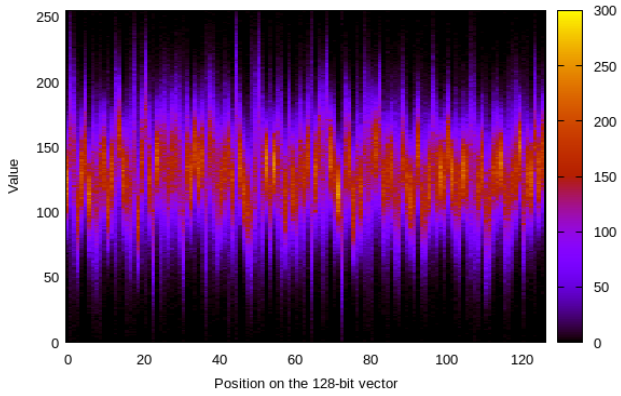


Fig. 1: Heat map of feature vector coefficients for 13095 faces

scores. In particular, the camera will precompute all possible values for the second and third terms of Equation 2, which is feasible due to the limited range of $y_{i,j}$ (just 256 distinct values). The computational cost involves $256 \times N \times M$ elliptic curve point multiplications and 256 encryption operations. Additionally, the storage requirements at the camera (for the database and all precomputed values) is $(256 + M + 256 \times N \times M) \times T$ bytes. Even for large databases (e.g., $M = 1000$), the storage cost is approximately 4GB, which is very reasonable for a low-cost camera.

Nevertheless, if a camera does not possess the storage capacity to hold the entire set of precomputed values, we may still gain a lot in performance if we store partial information. (We will illustrate this in our experimental results.) As shown in Fig. 1, the coefficients of a feature vector are not uniformly distributed over the entire range, but instead, values ranging from 64 to 191 tend to occur more frequently. As such, the camera may only precompute, say, 50% of the values (for $y_{i,j} \in [64, 191]$) and perform the remaining elliptic point multiplications, i.e., $y_{i,j} \cdot \mathsf{Enc}(-2x_{i,j})$, on the spot.

At the server side, the offline cost to compute the encrypted database is $2 \times N \times M$ encryption operations plus $N \times M$ elliptic curve point additions, which is trivial for a powerful multi-core server. On the other hand, in order to speed up the decryption operations that constitute the bottleneck of the online identification protocol, we need to precompute a large number of elliptic curve points (lookup table), as explained in Section 3. Assuming a maximum bit-length of k bits for the obfuscated similarity scores, the server will precompute and store 2^k 32-byte values with a computational cost of 2^k additions (that are relatively cheap). In our implementation, we set $k = 30$, which necessitates 32 GB of main memory.

Finally, it is worth noting that, unlike existing approaches, the offline costs of our method are incurred only once and are independent of the number of faces that are captured by the camera.

5.2 Similarity score computation

Following the offline phase, our system is ready for real-time video surveillance. A camera will capture all passing-by faces and, for each face C_i, it will generate the plaintext feature vector y_i. Based on the generated $y_{i,j}$, the camera selects the corresponding ciphertexts from the precomputed values and evaluates the encrypted squared Euclidean distances $\mathsf{Enc}(d_i^2)$ for every suspect i, as given in Equation 2. This task entails, for all M suspects, $(2 \times N + 1) \times M$ elliptic curve point additions. Each distance is then adjusted by subtracting the normalized similarity threshold t, thus generating an encrypted similarity score s_i that is (i) positive for a non-match or (ii) negative for a match. Therefore, the encrypted similarity score for suspect i is computed as

$$\mathsf{Enc}(s_i) = \mathsf{Enc}(d_i^2 - t) = \mathsf{Enc}(d_i^2) + \mathsf{Enc}(-t) \tag{3}$$

By precomputing the encrypted threshold value (constant), the computational cost of this step is M point additions. To summarize, the overall cost for computing the similarity score is $2 \times (N+1) \times M$ point additions. Finally, we should mention that, based on the normalization parameters given is Section 5.1, the normalized similarity threshold is set to $t = (0.9 \times 400)^2 = 129,600$.

5.3 Similarity score obfuscation

To further strengthen the privacy of our system, each similarity score is obfuscated, in order to increase the uncertainty at the database server. In particular, for every similarity score s_i, the camera selects two uniformly random numbers $r_1, r_2 \in [0, 2^\ell)$ and masks the score as $\delta_i = s_i \cdot r_1 + r_2$. In the ciphertext domain this is computed as

$$\mathsf{Enc}(\delta_i) = r_1 \cdot \mathsf{Enc}(s_i) + \mathsf{Enc}(r_2) \tag{4}$$

To avoid reversing the sign of the similarity score, we always choose $r1 > r_2$. The exact value of ℓ depends on the main memory specifications of the server. In our experiments, we empirically determined the max value for the similarity score to be $< 2^{19}$ and set $\ell = 11$, which limits the obfuscated scores to values $< 2^{30}$. Note that, since $r_2 < 2048$, we may precompute all possible encryptions of r_2 and reduce the computational cost of this step to M point multiplications and M point additions.

5.4 Matching

When all obfuscated scores δ_i are computed for a captured face C_j, the camera applies a random permutation π_j and sends the permuted score vector to the database server. The server then decrypts all ciphertexts with its private key and, if all M are positive, it infers that C_j is not a potential suspect. On the other hand, if a score is negative, a verification protocol is invoked in order for the server to learn the id of the potential suspect and (optionally) receive the image of the captured face. This step is necessary to prevent a malicious server

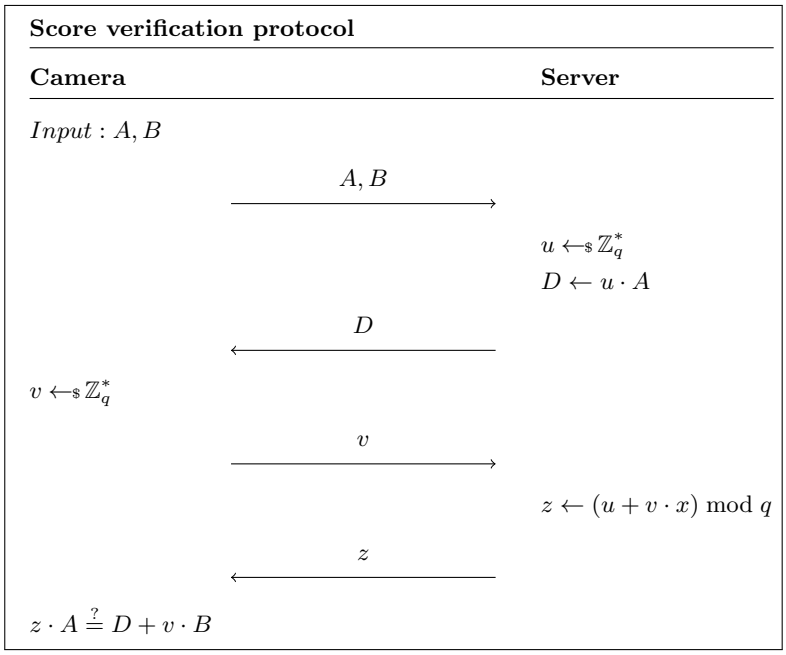


Fig. 2: Score verification protocol

from requesting footage of random individuals that did not actually produce a database match.

The verification protocol works as follows. The server first informs the camera of the suspect's position and score on the permuted vector, and the camera then looks up the suspect's real id and encrypted score in the permutation π_j that is temporarily stored in its local storage. Assume that the stored copy of the encrypted score is equal to $\mathsf{Enc}(s) = \langle r_1 \cdot P, (s + r_1) \cdot Q \rangle$. The camera will then generate an encryption of the score s' that the server claims to be true: $\mathsf{Enc}(s') = \langle r_2 \cdot P, (s' + r_2) \cdot Q \rangle$. If $s = s'$, then a subtraction of the two ciphertexts will produce an encryption of the value zero. Therefore, the camera will compute

$$\mathsf{Enc}(s) - \mathsf{Enc}(s') = \langle (r_1 - r_2) \cdot P, (s - s' + r_1 - r_2) \cdot Q \rangle \tag{5}$$

which is supposedly equal to $\langle A, B \rangle = \langle r \cdot P, r \cdot Q \rangle$ for some unknown random r. As such, it suffices to prove that $x \cdot A = B$, where x is the server's private key. Essentially, the server has to prove to the camera that it knows the value x that satisfies the above equation. This is trivially done with Schnorr's identification protocol [21], as shown in Fig. 2. D represents the server's commitment in the protocol, while v is the challenge posed by the camera. The server's response z can only be computed by the party who knows x, and the camera accepts the result if and only if the last equation holds.

Under normal conditions, the overwhelming majority of captured faces will not produce a database match, so the cost of the matching protocol is dominated

by the M decryption operations, each requiring one point multiplication and one point addition (due to the stored lookup table). The communication cost involves the transmission of M ciphertexts and is, thus, equal to $M \times T$ bytes.

6 Security

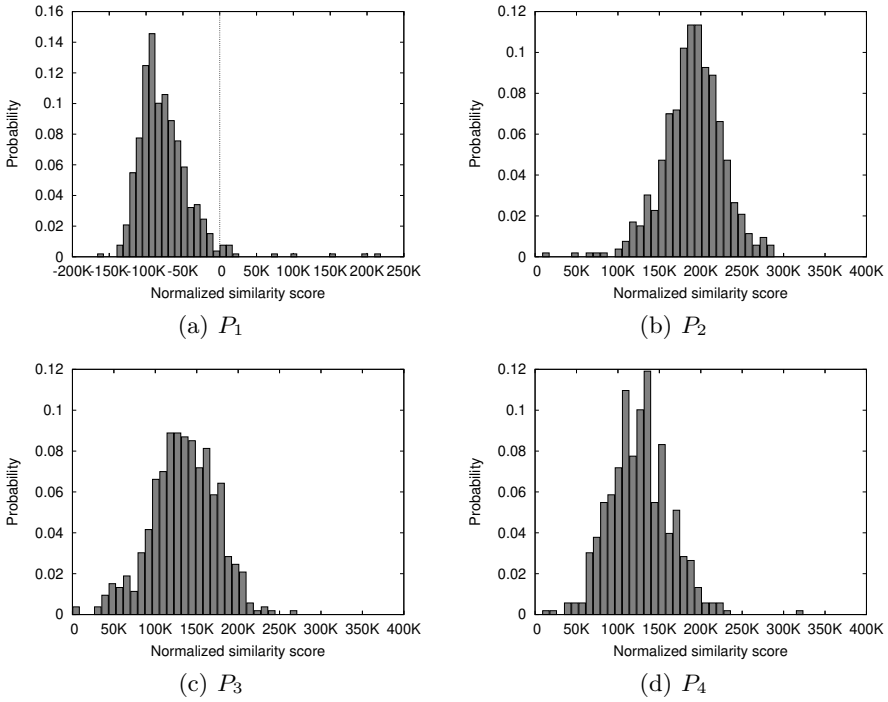
We consider two types of attacks against our system. The first one is a complete privacy break, where the server is able to retrieve the plaintext version of the feature vector for some captured face. This is only possible if the server is able to correctly inverse the camera's permutation and obfuscation steps and solve the underlying non-linear equations with N unknowns (assuming $M \geq N$). Nevertheless, this is infeasible due to (i) the exponential number $M!$ of possible permutation outcomes and (ii) the unpredictability of FaceNet's deep learning approach to feature vector generation that makes it very difficult to link a similarity score to a specific face-suspect pair.

To illustrate the second point above, we analyzed the similarity scores generated by our system for four random faces from the LFW dataset. We selected 500 images from person P_1 and computed the (non-obfuscated) similarity scores against one image of P_1, P_2, P_3, P_4. The results are shown in Fig. 3, where it is evident that the obtained scores follow a Gaussian-like distribution with a large overlap among the different faces. In particular, for the non-matching faces, the large majority of similarity scores lie within the interval $[50K, 200K]$, thus preventing the server from inferring any non-trivial information about the underlying permutation.

The second type of attack is less severe and pertains to the ability of the database server to distinguish an unknown individual across multiple cameras. For example, suppose that a captured face generates an identical feature vector across a series of cameras. While the probability of that event is negligible, it is worth investigating the effect of the obfuscation step on the generated score distribution. Fig. 4 depicts the probability distribution of the obfuscated score bit-lengths against a database of 1000 suspects. P_1 is indistinguishable across two different obfuscations (for an identical feature vector) and all four distributions are very similar to each other with large overlaps. Note that, we are not interested in the distribution of negative scores, since a match will trigger the verification protocol that reveals the suspect's identity.

7 Performance Comparison

Compared to the current state-of-the-art approaches, such as Eigenfaces [19] and SCiFI [17], our system offers tremendous improvements (multiple orders of magnitude) in terms of online cost, while still maintaining strong privacy guarantees. We were able to obtain these results because of (i) the efficiency of FaceNet's feature vectors and (ii) the storage of the suspects' database at the surveillance cameras. Indeed, the feature vector of Eigenfaces is equal to the number of image



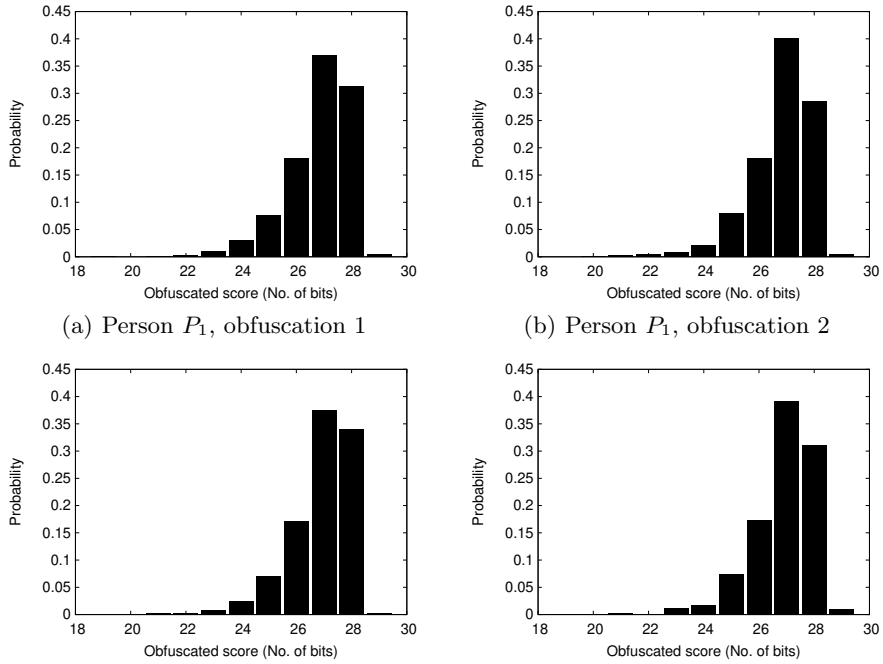
(a) P_1 (b) P_2

(c) P_3 (d) P_4

Fig. 3: Distribution of non-obfuscated similarity scores for 500 images of P_1 against one image of P_1, P_2, P_3, P_4

pixels, which cannot be less than 10,000; SCiFI also suffers from high vector dimensionality, as it utilizes 900-bit vectors. On the other hand, FaceNet achieves very accurate image classification with only 128 features. Feature vector dimensionality is a major factor that affects the online cost, since every feature has to be encrypted at the camera and transmitted to the database server (for Eigenfaces and SCiFI). These methods utilize precomputations to reduce that cost, but they do not come for free, as they have to be performed for every captured face, potentially millions per day.

This brings us to our major contribution, i.e., the distribution the suspects' encrypted database to the entire network of cameras. There are significant advantages with this approach. First, the cameras do not need to encrypt the feature vectors, thus alleviating the computational burden at these low-cost devices. Second, the static database facilitates the (one-time) precomputation of all intermediate results that reduces the computation of the encrypted similarity scores to a series of cheap elliptic curve point additions. Finally, the server is no longer involved in the expensive homomorphic operations that compute the similarity scores (as in Eigenfaces and SCiFI), which is a much more scalable approach for wide-scale video surveillance.



(a) Person P_1, obfuscation 1

(b) Person P_1, obfuscation 2

(c) Person P_2

(d) Person P_3

Fig. 4: Distribution of obfuscated score bit-lengths against a database of 1000 suspects ($r_1, r_2 < 2048$)

Our final contribution towards practical privacy-preserving video surveillance is the relaxation of the strict privacy guarantees of previous methods without any measurable consequences on user privacy. This eliminates an expensive zero knowledge protocol that reveals the id of the potential suspect only if the similarity score is below a certain threshold. Such protocols include garbled circuits [19] and oblivious transfers [17], which incur a lot of overhead, especially in terms of communication. Precomputations certainly improve the online costs, but they have to be performed for every captured face. On the other hand, our protocol necessitates a single communication round (with the exception of infrequent positive matches) and a few KB of communication cost.

8 Implementation Details

We implemented our system on two machines, one to emulate the law enforcement server and the other to simulate the camera operations. The server is a Ubuntu desktop machine equipped with Intel Xeon CPU E5-2620 2.10 GHz×16, 64 GB of RAM, and a 512 GB SSD. The other machine is a Ubuntu laptop with Intel Core i7-6500U CPU 2.50 GHz×4 and 8 GB of RAM (it is also equipped

with a front camera). The two machines are connected via a TCP/IP4 LAN over Gigabit Ethernet. We also tested our system on a more realistic environment, with regards to the surveillance cameras, by implementing our code on a Raspberry Pi 3 device. This device has limited computing and storage capabilities, featuring 1 GB of RAM and a 4ARM Cortex-A53 1.2 GHz CPU. The face recognition layer was based on the original implementation of OpenFace[1]. It employs *shape_predictor_68_face_landmarks* as face predictor and *nn4.small2.v1.t7* as the network model. The package is written in Python version 2.7 and, with the aforementioned configuration, face recognition and normalization takes about 600 ms on laptop.

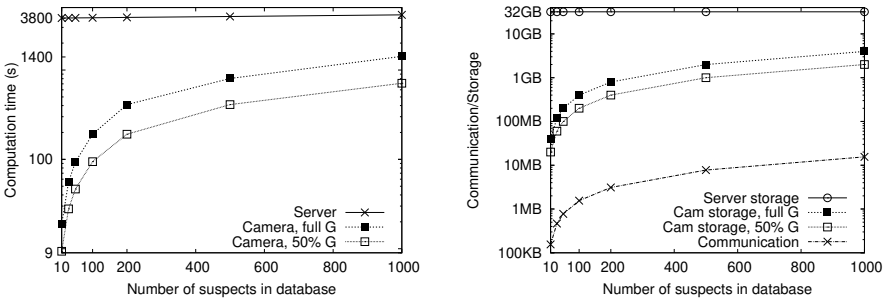
The cryptographic layer (elliptic curve ElGamal) was implemented in C, using the BIGNUM library of OpenSSL (version 1.1.0g). We also used SWIG to connect C with Python (version 4.0.1). We set the order of the elliptic curve to be a 256-bit prime number, as per NIST's recommendations [4]. As a result, all ciphertexts, which consist of two elliptic curve points, require 128 bytes of storage/communication. Under this C/Python environment, the average time for encryption, decryption, and point multiplication (with 256-bit scalars) is about 0.3 ms. On the other hand, point addition takes only about 0.02 ms. For each reported result, we run the experiment 4 times and plot the average time. Finally, our implementation leverages the parallel computing abilities of the two multi-core machines, since all our algorithms are easily parallelizable. The source code of our implementation is available online[2].

9 Experimental Results

In this section, we present the results of our experimental evaluation. The reported times correspond to actual measurements collected from the two implementations on the separate devices (laptop and workstation). Starting with the offline phase, we first evaluate the computation and communication/storage costs at both the camera and the server, as a function of the database size M. To this end, Fig. 5(a) illustrates the CPU time at all parties. G represents the database of precomputed values, so the bottom curve of the plot corresponds to the cost where only 50% of the precomputations are actually performed. The cost at the camera is clearly linear in M and is dominated by the computation of the terms $y_{i,j} \cdot \mathsf{Enc}(-2x_{i,j})$, as explained in Section 5.1. This is, by any means, an acceptable cost, as it is incurred only once and can terminate within a few minutes.

At the server-side, the offline cost includes the generation of the suspects' feature vectors from the corresponding images (OpenFace), the normalization of their representations (our algorithm), and the generation of the encrypted database that is sent to the cameras. Nevertheless, these costs are not evident in Fig. 5(a), as they are dominated by the cost of the precomputations for the discrete log lookup table. This operation necessitates over an hour of compute

[1] `https://github.com/cmusatyalab/openface`
[2] `https://github.com/mahdihbku/BlindGuardian`



(a) Computation (b) Communication/Storage

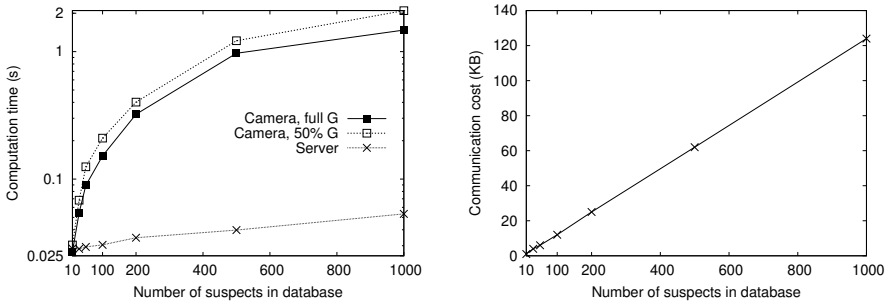
Fig. 5: Offline cost

time, but is crucial in our system because it speeds up considerably the decryption operations at the server. More importantly, this is a one-time cost that is incurred before the system becomes operational.

Fig. 5(b) depicts the offline communication/storage cost at the two devices. The compact representation of elliptic curve points makes it feasible to store the entire database G at the camera with only 4 GB of main memory. On the other hand, the cost at the server is again dominated by the discrete log lookup table, whose size is equal to 32 GB. However, this is a trivial requirement for today's state-of-the-art servers. Finally, the offline communication cost entails the transmission of the encrypted database and remains under 10 MB, even for a database of 1000 suspects.

In the next set of experiments, we evaluate the online cost of our approach, as a function of the database size M. First, Fig. 6(a) shows the online CPU time at all parties. Clearly, the cameras absorb most of the computational cost, since they have to compute the encrypted similarity scores for every suspect in the database. Nevertheless, the online cost is order of magnitudes lower compared to existing approaches, and remains below 1 sec for databases of up to 500 suspects. A notable observation that motivates the partial storage of G (as explained in Section 5.1) is that the performance penalty from storing 50% of G is not significant. In particular, for $M = 100$, the CPU time at the camera when the full G is available is 155 ms, and it only increases by 35% (to 210 ms) when 50% is available. Finally, a very promising result of our implementation is the online computation cost at the database server. For $M = 100$ the cost is just 34 ms, while for $M = 1000$ it only raises to 50 ms. As mentioned previously, the database server is the bottleneck in a wide-scale video surveillance system, because it may potentially process thousands of captured faces every second.

Fig. 6(b) illustrates the online communication cost for our system. It involves a single round of communication, where the camera transmits M encrypted similarity scores to the server. For a database of 1000 suspects, this entails a communication cost of just 128 KB.



(a) Computation (b) Communication

Fig. 6: Online cost

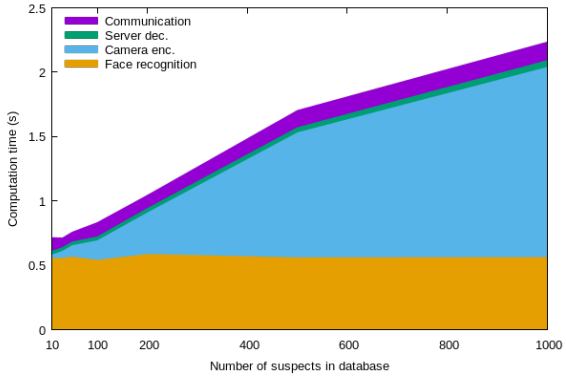
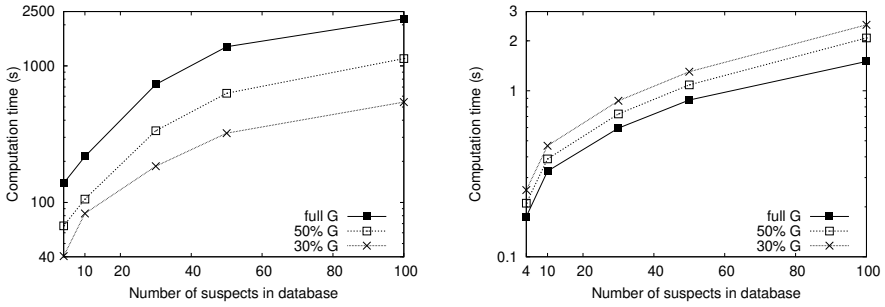


Fig. 7: Round Trip Time to detect a suspect

Our previous experiments focused only on the cryptographic overhead of the privacy-preserving face recognition system. Alternatively, Fig. 7 illustrates the true Round Trip Time (RTT) for detecting a suspect. It includes face recognition and detection at the camera, all the cryptographic operations at both the camera and the server, and the required communication that includes sending the suspect's image from the camera to the server. For a database of 100 suspects, the RTT is less than 0.8 seconds (with a precomputation of the entire database G), while for $M = 1000$ the RTT is approximately 2.2 seconds. Nevertheless, a large portion of the RTT (around 0.6 sec) is consumed on non-cryptographic operations, namely the face recognition and detection by the OpenFace software. A better combination of hardware/software at the surveillance cameras could improve that cost considerably.

Finally, Fig. 8 shows the computational costs of our Raspberry Pi 3 implementation. As we can see, the CPU time on the Raspberry device is around $10\times$ slower than the laptop. Even so, by storing the entire table of precomputations, we can match a face against a database with 100 suspects in under 2 seconds.



(a) Offline computation (b) Online computation

Fig. 8: Raspberry Pi 3 computational costs

Therefore, we argue that our system can be featured in a real-life implementation of privacy-preserving video surveillance, using cameras with moderate computational capabilities.

10 Conclusions

In this paper, we proposed the first near real-time privacy-preserving video-surveillance system that is based on the state-of-the-art face recognition algorithm. Our protocol diverges from the standard techniques employed by existing approaches by (i) replicating the suspects' database at the surveillance cameras and (ii) relaxing the stringent privacy requirements by disclosing some trivial information during the identification process. These design decisions facilitate the use of extensive precomputations that reduce significantly the computation of encrypted similarity scores. As a result, our system is able to match a single person against a database of 100 suspects in under 200 ms, while incurring a network transfer of just 12 KB of data. In our future work, we plan to incorporate specialized hardware (like GPUs) to further speed up the server computations, given the fact that our algorithms are highly parallelizable. We also plan to extend our framework to other domains as well, including biometric authentication and traffic surveillance.

References

1. Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys (CSUR) **51**(4), 79 (2018)
2. Amos, B., Ludwiczuk, B., Satyanarayanan, M.: Openface: A general-purpose face recognition library with mobile applications. CMU School of Computer Science **6** (2016)

3. Baltrusaitis, T., Zadeh, A., Lim, Y.C., Morency, L.P.: Openface 2.0: Facial behavior analysis toolkit. In: Proc. IEEE International Conference on Automatic Face & Gesture Recognition (FG). pp. 59–66 (2018)
4. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: NIST special publication 800-57. NIST Special publication, Recommendation for Key Management–Part 1: General (Revision 3) **800**(57), 1–142 (2012)
5. Bringer, J., Chabanne, H., Favre, M., Patey, A., Schneider, T., Zohner, M.: GSHADE: faster privacy-preserving distance computation and biometric identification. In: Proc. ACM Workshop on Information Hiding and Multimedia Security. pp. 187–198 (2014)
6. Dagher, I.: Incremental PCA-LDA algorithm. In: Proc. IEEE International Conference on Computational Intelligence for Measurement Systems and Applications. pp. 97–101 (2010)
7. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)
8. Er, M.J., Wu, S., Lu, J., Toh, H.L.: Face recognition with radial basis function (RBF) neural networks. IEEE Transactions on Neural Networks **13**(3), 697–710 (2002)
9. Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft, T.: Privacy-preserving face recognition. In: Proc. International Symposium on Privacy Enhancing Technologies (PETS). pp. 235–253 (2009)
10. Evans, D., Huang, Y., Katz, J., Malka, L.: Efficient privacy-preserving biometric identification. In: Proc. Network and Distributed System Security Symposium (NDSS). vol. 68 (2011)
11. Gasti, P., Šeděnka, J., Yang, Q., Zhou, G., Balagani, K.S.: Secure, fast, and energy-efficient outsourced authentication for smartphones. IEEE Transactions on Information Forensics and Security **11**(11), 2556–2571 (2016)
12. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proc. ACM Symposium on Theory of Computing (STOC). pp. 169–178 (2009)
13. Heisele, B., Ho, P., Poggio, T.: Face recognition with support vector machines: Global versus component-based approach. In: Proc. IEEE International Conference on Computer Vision (ICCV). vol. 2, pp. 688–694 (2001)
14. Karabat, C., Kiraz, M.S., Erdogan, H., Savas, E.: THRIVE: threshold homomorphic encryption based secure and privacy preserving biometric verification system. EURASIP Journal on Advances in Signal Processing **2015**(1), 71 (2015)
15. Liu, C., Wechsler, H.: Comparative assessment of independent component analysis (ICA) for face recognition. In: Proc. International Conference on Audio and Video Based Biometric Person Authentication (1999)
16. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. J. Cryptology **18**(1), 1–35 (2005)
17. Osadchy, M., Pinkas, B., Jarrous, A., Moskovich, B.: Scifi – a system for secure face identification. In: Proc. IEEE Symposium on Security and Privacy (SP). pp. 239–254 (2010)
18. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Proc. International Conference on the Theory and Applications of Cryptographic Techniques. pp. 223–238 (1999)
19. Sadeghi, A.R., Schneider, T., Wehrenberg, I.: Efficient privacy-preserving face recognition. In: Proc. International Conference on Information Security and Cryptology. pp. 229–244 (2009)

20. Sahoolizadeh, A.H., Heidari, B.Z., Dehghani, C.H.: A new face recognition method using PCA, LDA and neural network. International Journal of Computer Science and Engineering **2**(4), 218–223 (2008)
21. Schnorr, C.: Efficient signature generation by smart cards. J. Cryptology **4**(3), 161–174 (1991)
22. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 815–823 (2015)
23. Swets, D.L., Weng, J.J.: Using discriminant eigenfeatures for image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **18**(8), 831–836 (1996)
24. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1701–1708 (2014)
25. Toygar, Ö., Adnan, A.: Face recognition using PCA, LDA and ICA approaches on colored images. Istanbul University-Journal of Electrical & Electronics Engineering **3**(1), 735–743 (2003)
26. Turk, M., Pentland, A.: Eigenfaces for recognition. Journal of Cognitive Neuroscience **3**(1), 71–86 (1991)
27. Xiang, C., Tang, C., Cai, Y., Xu, Q.: Privacy-preserving face recognition with outsourced computation. Soft Computing **20**(9), 3735–3744 (2016)
28. Yao, A.C.C.: How to generate and exchange secrets. In: Proc. Symposium on Foundations of Computer Science (FOCS). pp. 162–167 (1986)
29. Zhou, K., Ren, J.: PassBio: Privacy-preserving user-centric biometric authentication. IEEE Transactions on Information Forensics and Security **13**(12), 3050–3063 (2018)