Privacy-Preserving Face Recognition *

Zekeriya Erkin[1] Martin Franz[2,3] Jorge Guajardo[2]
Stefan Katzenbeisser[2,3] Inald Lagendijk[1] Tomas Toft[4,5]

[1] Technische Universiteit Delft [2] Philips Research Europe
[3]Technische Universität Darmstadt [4]Technische Universiteit Eindhoven
[5]CWI Amsterdam

Abstract. Face recognition is increasingly deployed as a means to unobtrusively verify the identity of people. The widespread use of biometrics raises important privacy concerns, in particular if the biometric matching process is performed at a central or untrusted server, and calls for the implementation of Privacy-Enhancing Technologies. In this paper we propose for the first time a strongly privacy-enhanced face recognition system, which allows to efficiently hide both the biometrics and the result from the server that performs the matching operation, by using techniques from secure multiparty computation. We consider a scenario where one party provides a face image, while another party has access to a database of facial templates. Our protocol allows to jointly run the standard Eigenfaces recognition algorithm in such a way that the first party cannot learn from the execution of the protocol more than basic parameters of the database, while the second party does not learn the input image or the result of the recognition process. At the core of our protocol lies an efficient protocol for securely comparing two Paillier-encrypted numbers. We show through extensive experiments that the system can be run efficiently on conventional hardware.

1 Introduction

Biometric techniques have advanced over the past years to a reliable means of authentication, which are increasingly deployed in various application domains. In particular, face recognition has been a focus of the research community due to its unobtrusiveness and ease of use: no special sensors are necessary and readily available images of good quality can be used for biometric authentication. The development of new biometric face-recognition systems was mainly driven by two application scenarios:

– To reduce the risk of counterfeiting, modern electronic passports and identification cards contain a chip that stores information about the owner, as well as biometric data in the form of a fingerprint and a photo. While this biometric data is not widely used at the moment, it is anticipated that the

* Supported in part by the European Commission through the IST Programme under Contract IST-2006-034238 SPEED.

digitized photo will allow to automatize identity checks at border crossings or even perform cross-matching against lists of terrorism suspects (for a recent Interpol initiative to use face recognition to mass-screen passengers see [5]).
– The increasing deployment of surveillance cameras in public places (e.g. [19] estimates that 4.2 million surveillance cameras monitor the public in the UK) sparked interest in the use of face recognition technologies to automatically match faces of people shown on surveillance images against a database of known suspects. Despite massive technical problems that render this application currently infeasible, automatic biometric face recognition systems are still high on the agenda of policy makers [26, 20].

The ubiquitous use of face biometrics raises important privacy concerns; particularly problematic are scenarios where a face image is automatically matched against a database without the explicit consent of a person (for example in the above-mentioned surveillance scenario), as this allows to trace people against their will. The widespread use of biometrics calls for a careful policy to which party biometric data is revealed, in particular if biometric matching is performed at a central server or in partly untrusted environments.

In this paper we propose for the first time strong cryptographic Privacy-Enhancing Technologies for biometric face recognition; the techniques allow to hide the biometric data as well as the authentication result from the server that performs the matching. The proposed scheme can thus assure the privacy of individuals in scenarios where face recognition is beneficial for society but too privacy intrusive.

In particular, we provide a solution to the following two-party problem. Alice and Bob want to privately execute a standard biometric face recognition algorithm. Alice owns a face image, whereas Bob owns a database containing a collection of face images (or corresponding feature vectors) from individuals. Alice and Bob want to jointly run a face recognition algorithm in order to determine whether the picture owned by Alice shows a person whose biometric data is in Bob's database. While Bob accepts that Alice might learn basic parameters of the face recognition system (including the size of the database), he considers the content of his database as private data that he is not willing to reveal. In contrast, Alice trusts Bob to execute the algorithm correctly, but is neither willing to share the image nor the detection result with Bob. After termination, Alice will only learn if a match occurred; alternatively, an ID of the identified person may be returned.

In a real world scenario Bob might be a police organization, whereas Alice could be some private organization running an airport or a train station. While it may be common interest to use face recognition to identify certain people, it is generally considered too privacy intrusive to use Bob's central server directly for identification, as this allows him to create profiles of travelers. Thus, the two parties may decide for a privacy-friendly version where the detection result is not available to the central party. As the reputation of both parties is high and because both parties are interested in computing a correct result, it is reasonable to assume that they will behave in a semi-honest manner. We

provide a complete implementation of the above-mentioned two-party problem using the standard Eigenface [35] recognition system, working on encrypted images. At the heart of our privacy-enhanced face recognition system lies a highly optimized cryptographic protocol for comparing two Pailler-encrypted values. The system is very efficient and allows matching of an encrypted face image of size 92×112 pixels against a database of 320 facial templates in approximately 40 seconds on a conventional workstation, despite the huge computational complexity of the underlying cryptographic primitives; using pre-computations for intermediate values which do not depend on the input image, recognition only takes 18 seconds. While there is a small constant overhead when performing a face-recognition, the time to perform the recognition is linear in the size of the database. For a large database containing M facial templates, time for one recognition increases only slowly and lies in $\mathcal{O}(0.054M)$ seconds for the conventional approach and $\mathcal{O}(0.031M)$ seconds when using pre-computations.

2 Cryptographic Tools

As a central cryptographic tool, we use two semantically secure additively homomorphic public-key encryption schemes, namely the Paillier and the DGK cryptosystem. In an additively homomorphic cryptosystem, given encryptions $[a]$ and $[b]$, an encryption $[a + b]$ can be computed by $[a + b] = [a][b]$, where all operations are performed in the algebra of the message or ciphertext space. Furthermore, messages can be multiplied with constants under encryption, i.e., given an encrypted message $[a]$ and a constant b in the clear, it is possible to compute $[ab]$ by $[ab] = [a]^b$.

Paillier cryptosystem. Introduced by Paillier in [30], its security is based on the decisional composite residuosity problem. Let $n = pq$ of size k, with p, q prime numbers and k from the range 1000-2048. Also let $g = n + 1$ [10]. To encrypt a message $m \in \mathbb{Z}_n$, the user selects a random value $r \in \mathbb{Z}_n$ and computes the ciphertext $c = g^m r^n \bmod n^2$. Note that due to our choice of g, encryption requires only one modular exponentiation and two modular multiplications, as $c = (mn + 1)r^n \bmod n^2$. We will write the encryption of a message m in the Paillier cryptosystem as $[m]$. Since all encryptions in the proposed protocol will be computed using one fixed public key, we do not specify the key explicitly. It is easy to see that Paillier is additively homomorphic and that for an encryption $[m]$ we can compute a new probabilistic encryption of m without knowing the private key (this will be denoted as *re-randomization*). We refer the reader to [30] for a description of the decryption operation and further details on the cryptosystem.

Damgård, Geisler and Krøigaard cryptosystem (DGK). For efficiency reasons we use at a key point in our protocol another homomorphic cryptosystem, which was proposed by Damgård, Geisler and Krøigaard [8, 9]. As in Paillier, let $n = pq$ be a k-bit integer (with k chosen from the range 1000-2048), with p, q primes. The ciphertext c corresponding to a message $m \in \mathbb{Z}_u$ is computed as $c = g^m h^r \bmod n$, where u is a prime number and r is a randomly chosen integer. In practice (and more importantly in our application) u is from a very

small range, say 8-bit values, which results in a very small plaintext space $\mathbb{Z}_u$. Similarly to Paillier, DGK is also additively homomorphic and it is possible to re-randomize existing ciphertexts. Compared to Paillier, the scheme has substantially smaller ciphertexts and the smaller plaintext space results in a large performance gain. To note the difference between Paillier and DGK ciphertexts we will denote the encryption of m in the DGK cryptosystem as $[\![m]\!]$.

3 Face Recognition

In 1991, Matthew Turk and Alex Pentland proposed an efficient approach to identify human faces [35, 36]. This approach transforms face images into characteristic feature vectors of a low-dimensional vector space (the face space), whose basis is composed of *eigenfaces*. The eigenfaces are determined through Principal Component Analysis (PCA) from a set of training images; every face image is succinctly represented as a vector in the face space by projecting the face image onto the subspace spanned by the eigenfaces. Recognition of a face is done by first projecting the face image to the face space and subsequently locating the closest feature vector. A more detailed description of the enrollment and recognition processes is given below.

During enrollment, a set of M training images $\Theta_1, \Theta_2, \ldots, \Theta_M$, which can be represented as vectors of length N, is used to determine the optimal low-dimensional face space, in which face images will be represented as points. To do this, the average of the training images is first computed as $\Psi = \frac{1}{M}\sum_{i=1}^{M}\Theta_i$. Then, this average is subtracted from each face vector to form difference vectors $\Phi_i = \Theta_i - \Psi$. Next, PCA is applied to the covariance matrix of these vectors $C = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = \frac{1}{M}AA^T$ to obtain orthonormal eigenvectors and corresponding eigenvalues where A is matrix with each column corresponding to the image Θ_i for $i = 1$ to M. (As the size of C makes it computationally infeasible to directly run PCA, the eigenvectors are usually obtained by applying PCA to the much smaller matrix A^TA and appropriate post-processing). At most M of the eigenvalues will be nonzero. To determine the face space, we select $K \ll M$ eigenvectors $u_1, \ldots, u_K$ that correspond to the K largest eigenvalues. Subsequently, images $\Theta_1, \Theta_2, \ldots, \Theta_M$ showing faces to be recognized (not necessarily the training images) are projected into the subspace spanned by the basis $u_1, \ldots, u_K$ to obtain their feature vector representation $\Omega_1, \ldots, \Omega_M$.

During recognition, a new face image Γ is projected into the face space by calculating weights $\bar{\omega}_i = u_i^T(\Gamma - \Psi)$ for $i = 1, \ldots, K$. These weights form a feature vector $\bar{\Omega} = (\bar{\omega}_1, \bar{\omega}_2, \ldots, \bar{\omega}_K)^T$ that represents the new image in the face space. Subsequently, the distances between the obtained vector $\bar{\Omega}$ and all feature vectors $\Omega_1, \ldots, \Omega_M$ present in the database are computed,

$$D_i = \|(\bar{\Omega} - \Omega_i)\|. \tag{1}$$

A match is reported if the smallest distance $D_{min} = \min\{D_1, \ldots, D_M\}$ is smaller than a given threshold value T. Note that this basic recognition algorithm can be augmented with additional checks that reduce the number of false positives

and negatives during recognition; for the sake of simplicity, we stick to the basic eigenface recognition algorithm presented above.

4 Privacy-Preserving Eigenfaces

In this section, we present a privacy preserving realization of the Eigenface recognition algorithm which operates on encrypted images. We work in the two-party setting in the semi-honest attacker model. Informally, this assumes that the parties involved in the protocol follow it properly but keep a log of all the messages that have been exchanged (including their own) and try to learn as much information as possible from them. Alice's privacy is ensured against a computationally bounded attacker, while Bob's is unconditional—even a computationally unbounded Alice cannot compromise it. It is also assumed that the parties communicate over an authenticated channel (this can be achieved by standard mechanisms and is thus outside the scope of this paper).

4.1 Setup and Key Generation

Two parties Alice and Bob jointly run the recognition algorithm. We assume that Bob has already set up the face recognition system by running the enrollment process (in the clear) on all available training images to obtain the basis $u_1, \ldots, u_K$ of the face space and feature vectors $\Omega_1, \ldots, \Omega_M$ of faces to be recognized. Furthermore, we assume that all coordinates of the eigenfaces and feature vectors are represented as integers; this can always be achieved by appropriate quantization: non-integer values are first scaled by a fixed scale factor S and rounded to the nearest integer. This is necessary, as all values need to be integers in order to encrypt them with Paillier and process them using homomorphic operations. The effects of this quantization step on the detection reliability are experimentally analyzed in Section 6. Each feature vector in the database is further accompanied by a string Id_i that contains the identity of the person the feature vector belongs to; we assume that the identity is encoded as a *non-zero* element of the message space of the chosen encryption scheme.

During the interactive recognition protocol, Alice provides an encrypted face image $[\Gamma]$ as input. At the end of the protocol, Alice learns whether the face shown on her image matches one of the feature vectors $\Omega_1, \ldots, \Omega_M$ owned by Bob: Depending on the application, Alice either receives the identity Id_i of the best matching feature vector or only a binary answer (i.e. whether there was a match or not). Apart from this answer (and the number M), Bob keeps the database content secret. Bob learns nothing from the interaction, i.e. neither the face image Γ, nor its representation in the face space, nor the result of the matching process.

Note that the vectors u_i are directly computed from the set of training images; thus, they *do* carry information on the faces stored in Bob's database. Even though it is hard to quantify the exact amount of data leakage through the knowledge of the basis $u_1, \ldots, u_K$, our solution will treat it as sensitive data

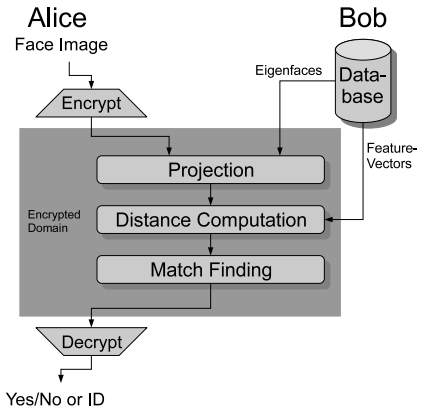

Fig. 1. Privacy-Preserving Face Recognition.

that will not be disclosed to Alice. In an alternative implementation, the basis $u_1, \ldots, u_K$ can be derived from a sufficiently large public face database so that they do not carry personal information; the proposed system can easily be changed to take advantage of public basis vectors, see Section 6 for details. Since Alice is the only party who receives an output, we can construct the protocol using any standard homomorphic public-key encryption algorithm; as stated in Section 2 we choose Paillier encryption for the implementation. In particular, we do *not* need a threshold homomorphic scheme, as it is widely employed in the construction of secure multiparty protocols. Before the interaction starts, Alice generates a pair of public and private keys and sends her public key to Bob over an authenticated channel. In the first step of the protocol, Alice encrypts all pixels of the image Γ separately with her public key and sends the result to Bob, who is unable to decrypt them. However, Bob can use the homomorphic property of the cipher to perform linear operations on the ciphertexts; for some operations (such as computing distances between vectors or finding a minumum), he will require assistance from Alice in the form of an interactive protocol. At the end of the protocol, Alice receives back an encryption containing the result of the biometric matching operation, which only Alice can decrypt. Appendix A gives a sketch of the security of our system in the semi-honest attacker model.

4.2 Private Recognition Algorithm

To match a face image against feature vectors in a database, three steps need to be performed. First, the image needs to be projected into the face space in order to obtain its corresponding feature vector representation. Subsequently, distances between the obtained vector and all feature vectors in Bob's database need to be computed. Finally, the one with minimum distance is selected; if this distance is smaller than a threshold, a match is reported. In the following, we show how these three steps can be realized in a privacy preserving manner.

Figure 1 shows an outline of the private face recognition protocol; the gray area denotes operations that need to be performed on encrypted values.

Projection. As a first step, the input image Γ has to be projected onto the low dimensional face space spanned by the eigenfaces $u_1, \ldots, u_K$. This can be performed by computing the scalar product of

$$\Phi = \Gamma - \Psi = \begin{pmatrix} \Gamma_1 - \Psi_1 \\ \vdots \\ \Gamma_N - \Psi_N \end{pmatrix}$$

and each eigenface vector u_i to obtain

$$\bar{\omega}_i = \Phi_1 \cdot u_{i1} + \ldots + \Phi_N \cdot u_{iN}$$

for each $i \in \{1, \ldots, K\}$.

These operations have to be performed in the encrypted domain by Bob, who receives the encrypted face image $[\Gamma]$ from Alice. As Bob knows the vector Ψ in plain, he can easily compute $-\Psi = (-1) \cdot \Psi$ and then encrypt each of its components. These encryptions can be pairwise multiplied with the encrypted components of $[\Gamma]$ in order to perform the componentwise subtraction of the vectors Γ and Ψ. Thus Bob computes

$$[\Phi] = [\Gamma - \Psi] = \begin{pmatrix} [\Gamma_1] \cdot [-\Psi_1] \\ \vdots \\ [\Gamma_N] \cdot [-\Psi_N] \end{pmatrix}.$$

Subsequently Bob performs the projection

$$[\bar{\omega}_i] = [\Phi_1 \cdot u_{i1} + \ldots + \Phi_N \cdot u_{iN}] = [\Phi_1]^{u_{i1}} \cdot \ldots \cdot [\Phi_N]^{u_{iN}}$$

for each $i \in \{1, \ldots, K\}$. This is done as follows. As Bob knows the vector u_i in plain, he can perform the required multiplications using the homomorphic property. For example, in order to multiply the first components of both vectors Bob has to compute $[\Phi_1]^{u_{i1}}$. To obtain the sum of all these products he just multiplies the encryptions with each other. Doing this for all $1 \leq i \leq K$, Bob obtains an encrypted feature vector description of the face image as $[\bar{\Omega}] := ([\bar{\omega}_1], \ldots, [\bar{\omega}_K])^T$. Note that every computation in the projection operation can be performed by Bob without interacting with Alice.

Calculating distances. After having obtained the encrypted feature vector $[\bar{\Omega}]$, encryptions of the distances $D_1, \ldots, D_M$ between $\bar{\Omega}$ and all feature vectors $\Omega \in \{\Omega_1, \ldots, \Omega_M\}$ from the database have to be computed. Since in the remainder of the protocol we are only concerned with the relative order of the obtained distances, it suffices to compute the square of the Euclidean distance,

$$\begin{aligned} D(\Omega, \bar{\Omega}) &= \|\Omega - \bar{\Omega}\|^2 = (\omega_1 - \bar{\omega}_1)^2 + \ldots + (\omega_K - \bar{\omega}_K)^2 \\ &= \underbrace{\sum_{i=1}^{K} \omega_i^2}_{S_1} + \underbrace{\sum_{i=1}^{K} (-2\omega_i \bar{\omega}_i)}_{S_2} + \underbrace{\sum_{i=1}^{K} \bar{\omega}_i^2}_{S_3}. \end{aligned}$$

Again, we need to evaluate this equation in the encrypted domain: Bob knows the encryption $[\bar{\Omega}]$ and needs to compute the encrypted distance $[D(\Omega, \bar{\Omega})]$, while he knows the feature vector Ω in the clear. To compute $[D(\Omega, \bar{\Omega})]$ it suffices to compute encryptions of the three sums $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$, as by the homomorphic property and Eq. (2),

$$[D(\Omega, \bar{\Omega})] = [\mathcal{S}_1] \cdot [\mathcal{S}_2] \cdot [\mathcal{S}_3].$$

The term $\mathcal{S}_1$ is the sum over the components of Ω known in the clear. Thus, Bob can compute $\mathcal{S}_1$ directly and encrypt it to obtain $[\mathcal{S}_1]$. $\mathcal{S}_2$ consists of the products $\omega_i\bar{\omega}_i$, where Bob knows ω_i in the clear and has $[\bar{\omega}_i]$ in encrypted form. In a first step the values ω_i can be multiplied with -2. The term $[(-2\omega_i)\bar{\omega}_i]$ can be computed by raising $[\bar{\omega}_i]$ to the power of $(-2\omega_i)$, using the homomorphic property. To obtain an encryption of $\mathcal{S}_2$, Bob finally computes $[\mathcal{S}_2] = \prod_{j=1}^{K}[(-2\omega_i)\bar{\omega}_i]$. Thus, the value $[\mathcal{S}_2]$ can again be computed by Bob without interacting with Alice. The term $\mathcal{S}_3$ consists of the squares of the encrypted values $[\bar{\omega}_i]$. Unfortunately, Bob cannot perform the required multiplication without help from Alice. Thus, Bob additively blinds the value $\bar{\omega}_i$ with an uniformly random element r_i from the plaintext space to obtain $[x_i] = [\bar{\omega}_i + r_i] = [\bar{\omega}_i] \cdot [r_i]$. Note that for every component $\bar{\omega}_i$ of the vector $\bar{\Omega}$ a fresh random value must be generated. Finally, he sends the elements $[x_i]$ to Alice who decrypts. Alice can now compute the values x_i^2 in plain as the square of the plaintext x_i and compute the value $\mathcal{S}_3' = \sum_{j=1}^{K} x_i^2$. She encrypts this value and sends $[\mathcal{S}_3']$ back to Bob, who computes

$$[\mathcal{S}_3] = [\mathcal{S}_3'] \cdot \prod_{j=1}^{K} ([\bar{\omega}_i]^{(-2r_i)} \cdot [-r_i^2]),$$

which yields the desired result because

$$[x_i^2] \cdot [\bar{\omega}_i]^{(-2r_i)} \cdot [-r_i^2] = [(\bar{\omega}_i + r_i)^2 - 2r_i\bar{\omega}_i - r_i^2] = [\bar{\omega}_i^2].$$

Note that this interactive protocol to compute the value $[\mathcal{S}_3]$ needs to be run only once. The value $[\mathcal{S}_3]$ depends only on the encrypted feature vector $[\bar{\Omega}]$ and can be used for computation of all distances $[D_1], \ldots, [D_M]$. Note further that due to the blinding factors, Alice does not learn the values $\bar{\omega}_i$.

Match finding. In the last step of the recognition algorithm, the feature vector from the database that is closest to $\bar{\Omega}$ must be found. This distance is finally compared to a threshold value T; if the distance is smaller, a match is reported and an encryption of the identity Id which corresponds to the best matching feature vector is returned to Alice.

As a result of the last step we obtained encrypted distances $[D_1], \ldots, [D_M]$, where D_i denotes the distance between $\bar{\Omega}$ and the i-th feature vector $\Omega_i \in \{\Omega_1, \ldots, \Omega_M\}$ from the database. To find the minimum we employ a straightforward recursive procedure: in the first step, we compare the $k = \lfloor \frac{M}{2} \rfloor$ encrypted distances $[D_{2i+1}]$ and $[D_{2i+2}]$ for $0 \leq i \leq k-1$ with each other, by using a cryptographic protocol that compares two encrypted values; a re-randomized encryption of the smaller distance is retained (re-randomization is necessary to

prevent Bob from determining the outcome of the comparison by inspecting the ciphertexts). After this step, there will be $\lceil \frac{M}{2} \rceil$ encryptions left. In a second run we repeat this procedure for the remaining encryptions, and so forth. After $\lceil \log_2(M) \rceil$ iterations there will only be one encryption left, the minimum.

As we need to return the identity of the best matching feature vector, we also have to keep track of the IDs during the minimum computation. This is done by working with pairs $([D_i], [Id_i])$ of distances and their corresponding identities, where the recursive minimum finding algorithm is applied to the distances only, but re-randomized encryptions of both the smaller distance and its identity are retained for the next round. An efficient implementation of the required comparison protocol is described in Section 5.

To check if the minimum distance is smaller than a threshold T, we can treat the value T as one additional distance that has the special identity 0. Together with the distances $D_1, \ldots, D_M$ we run the algorithm to find the minimum as described above. After $\lceil \log_2(M+1) \rceil$ iterations, Bob receives the minimum distance and the corresponding identity $([D], [Id])$, where $D \in \{T, D_1, \ldots, D_M\}$ and $Id \in \{0, Id_1, \ldots, Id_M\}$. Thus, if a face image could be recognized the value Id contains the corresponding identity. If no match could be found Id is equal to 0. The value $[Id]$ is finally sent to Alice as the result of the private face recognition protocol.

Note that there is an easy way to modify the protocol to make it terminate only with a binary output: rather than using actual IDs, Bob may assign a second special identity, the integer 1, to all images. In this case Alice will either receive a 1 or a 0, with the former indicating that a match was found.

5 Comparison Protocol

The only missing block is a protocol for selecting the minimum of two encrypted ℓ-bit values $[a]$ and $[b]$ along with the encrypted ID of the minimum. (Note that the bit-length ℓ can be determined by knowing the bit-length of the input data and the scale factor S used to quantize eigenfaces).

At the core of our protocol is a comparison protocol due to Damgård, Geisler and Krøigaard [8, 9]. Their setting differs from ours as follows: one input is public while the other is held (bitwise) in encrypted form by one party; moreover the output is public. They note several variations, but in order to provide a solution for the present setting some tweaking is needed. This section presents the protocol in a top-down fashion.

5.1 A High-level View of the Protocol

Initially Bob, who has access to both $[a]$ and $[b]$, computes

$$[z] = [2^\ell + a - b] = [2^\ell] \cdot [a] \cdot [b]^{-1}.$$

As $0 \leq a, b < 2^\ell$, z is a positive $(\ell+1)$-bit value. Moreover, z_ℓ, the most significant bit of z, is exactly the answer we are looking for:

$$z_\ell = 0 \Leftrightarrow a < b.$$

If Bob had an encryption of $z \bmod 2^\ell$, the result would be immediate: z_ℓ could be computed as

$$z_\ell = 2^{-\ell} \cdot (z - (z \bmod 2^\ell)).$$

Correctness is easily verified; the subtraction sets the least significant bits to zero, while the multiplication shifts the interesting bit down. As only z and $z \bmod 2^\ell$ are encrypted, this is a linear combination in the encrypted domain, which can be computed by Bob.

Once Bob has an encryption of the outcome $[z_\ell] = [a < b]$, an encryption of the minimum m, is easily obtained using arithmetic, as $m = (a < b) \cdot (a - b) + b$. The multiplication requires assistance of Alice, but is easily performed through a (short) interactive protocol. Determining an encryption of the ID is analogous, $(a < b) \cdot (Id_a - Id_b) + Id_b$. Thus, it remains to describe how Bob obtains the encryption of $z \bmod 2^\ell$.

5.2 Computing $[z \bmod 2^\ell]$

The value z is available to Bob only in encrypted form, so the modulo reduction cannot easily be performed. The solution is to engage in a protocol with Alice, transforming the problem back to a comparison.

First, Bob generates a uniformly random $(\kappa + \ell + 1)$-bit value r, where κ is a security parameter, say 100, and $\kappa + \ell + 1 \ll \log_2(n)$. This will be used to additively blind z,

$$[d] = [z + r] = [z] \cdot [r];$$

$[d]$ is then re-randomized and sent to Alice who decrypts it and reduces d modulo 2^ℓ. The obtained value is then encrypted, and returned to Bob.

Due to the restriction on the bit-length of r, Bob can now *almost* compute the desired encryption $[z \bmod 2^\ell]$. The masking can be viewed as occurring over the integers, thus we have $d \equiv z + r \bmod 2^\ell$ and

$$\big(z \bmod 2^\ell\big) = \big(\big(d \bmod 2^\ell\big) - \big(r \bmod 2^\ell\big)\big) \bmod 2^\ell.$$

Alice has just provided $[d \bmod 2^\ell]$ and r is known to Bob. Thus, he can compute

$$[\tilde{z}] = [(d \bmod 2^\ell) - (r \bmod 2^\ell)] = [d \bmod 2^\ell] \cdot [(r \bmod 2^\ell)]^{-1}.$$

Had the secure subtraction occurred modulo 2^ℓ, $\tilde{z}$ would be the right result; however, it occurs modulo n. Note, though, that if $d \bmod 2^\ell \geq r \bmod 2^\ell$, $\tilde{z}$ is the right result. On the other hand, if $r \bmod 2^\ell$ is bigger, an underflow has occurred; adding 2^ℓ in this case gives the right result. So, if Bob had an encryption $[\lambda]$ of a binary value indicating whether $r \bmod 2^\ell > d \bmod 2^\ell$, he could simply compute

$$[z \bmod 2^\ell] = [\tilde{z} + \lambda 2^\ell] = [\tilde{z}] \cdot [\lambda]^{2^\ell},$$

which adds 2^ℓ exactly when $r \bmod 2^\ell$ is the bigger value. This leaves us with a variant of Yao's millionaires problem: Bob must obtain an encryption $[\lambda]$ of a binary value containing the result of the comparison of two private inputs: $\hat{d} = d \bmod 2^\ell$ held by Alice and $\hat{r} = r \bmod 2^\ell$ held by Bob.

5.3 Comparing Private Inputs

The problem of comparing private inputs $\hat{d}$ and $\hat{r}$ is a fundamental one, which has been studied intensively (see e.g. [39, 29, 14, 3, 4, 15, 8]). For efficiency reasons, we solve this problem using a *different* homomorphic encryption scheme, namely the one proposed by Damgård et al. [8, 9], which has a very small plaintext space $\mathbb{Z}_u$ for some prime u. This allows very efficient multiplicative masking; in contrast to the Paillier scheme, the exponents are small.

Though the basic setting of Damgård et al. considers one public and one secret value, they note how to construct a solution for private inputs. They also note how to obtain a secret output. However, they obtain this output as an additive secret sharing, while in our setting Bob must receive a *Paillier encryption* $[\lambda]$ at the end of the protocol. Naturally Alice must not see this encryption as she knows the secret key.

We assume that Alice has run the DGK key-generation algorithm and has sent the public key to Bob. This key pair can be re-used whenever the comparison protocol will be run. Inertially, Alice sends Bob encryptions of the bits of her input, $[\![\hat{d}_{\ell-1}]\!], \ldots, [\![\hat{d}_0]\!]$. Bob then chooses $s \in_R \{1, -1\}$ and computes

$$[\![c_i]\!] = [\![\hat{d}_i - \hat{r}_i + s + 3 \sum_{j=i+1}^{\ell-1} w_j]\!] = [\![\hat{d}_i]\!] \cdot [\![-\hat{r}_i]\!] \cdot [\![s]\!] \cdot \left(\prod_{j=i+1}^{\ell-1} [\![w_j]\!] \right)^3 ,$$

where $[\![w_j]\!] = [\![\hat{d}_j \oplus \hat{r}_j]\!]$, which he can compute as Bob knows $\hat{r}_j$. For technical reasons (to avoid the case $\hat{d} = \hat{r}$), we append differing bits to both $\hat{d}$ and $\hat{r}$, i.e., we compare the values $2\hat{d} + 1$ and $2\hat{r}$ instead.

Eq. (2) differs from the one proposed by Damgård et al. in order to efficiently hide the output, but the core idea remains. Consider the case of $s = 1$; if $\hat{d}$ is bigger, then all c_i will be non-zero. (The modulus u is chosen such that there is no overflow.) However, if $\hat{r}$ is bigger, then exactly one c_i will equal zero, the one at the most significant differing bit-position. Both claims are easily verified. For $s = -1$ we have exactly the same situation, except that the zero occurs if $\hat{d}$ is bigger. The factor of 3 ensures that the values are non-zero once even a single w_j is set.

Bob now multiplicatively masks the $[\![c_i]\!]$ with a uniformly random $r_i \in \mathbb{Z}_u^*$

$$[\![e_i]\!] = [\![c_i \cdot r_i]\!] = [\![c_i]\!]^{r_i},$$

re-randomizes and permutes the encryptions $[\![e_i]\!]$ and sends them to Alice. Note that e_i is uniformly random in $\mathbb{Z}_u^*$ except when $c_i = 0$, in which case e_i also equals zero, i.e. the existence of a zero is preserved.

Alice now decrypts all e_i and checks whether one of them is zero. She then encrypts a bit $\tilde{\lambda}$, stating if this is the case. At this point she switches back to Paillier encryptions, i.e. Alice sends $[\tilde{\lambda}]$ to Bob. Given the knowledge of s, Bob can compute the desired encryption $[\lambda]$: while $[\tilde{\lambda}]$ only states whether there was a zero among the values decrypted by Alice, s explains how to interpret the

result, i.e. whether the occurrence of a zero means that $\hat{r} > \hat{d}$ or $\hat{d} \geq \hat{r}$. In the former case, Bob negates the result $[\tilde{\lambda}]$ under encryption, otherwise he directly takes $[\tilde{\lambda}]$ as output $[\lambda]$.

6 Implementation

The privacy-preserving face recognition system, as described in this paper, has been implemented in C++ using the GNU GMP library version 4.2.4, in order to determine its performance and reliability. Tests were performed on a computer with a 2.4 GHz AMD Opteron dual-core processor and 4GB of RAM running Linux. Both sender and receiver were modeled as different threads of one program, which pass messages to each other; thus, the reported performance data does not include network latency.

For testing purposes, we used the "ORL Database of Faces" from AT&T Laboratories Cambridge [1], which is widely used for experiments and contains 10 images of 40 distinct subjects, thus 400 images in total. All images in this database have a dark background with the subject in upright, frontal position. The size of each image is 92×112 pixels with 256 grey levels per pixel (thus $N = 92 \cdot 112 = 10304$). We use 5-fold cross validation for the experiments such that for each subject we use 8 images in the enrollment phase and 2 images for testing (thus, the database consists of 320 feature vectors). The security parameter k for both Paillier- and DGK-cryptosystem was set to 1024 bits (see Section 2 for details). Furthermore we set $\ell = 50$ (see Section 5 for details).

Reliability. During reliability testing, we assured that our privacy-preserving implementation of the Eigenface algorithm does not degrade the reliability when compared to a standard implementation which achieves approximately 96% correct classification rate. Reliability losses may occur due to the use of scaled and quantized feature vectors and eigenfaces. This scale factor has both an influence on the accuracy of the result and the performance of the scheme. Figure 2 shows the detection rates of the implementation for different scale factors, plotted on a logarithmic scale. It can be seen that scale factors below the value 1000 significantly degrade detection performance, while scale factors larger than 1000 do not improve the results. Hence, it suffices to set $S = 1000$ to achieve the same reliability as a reference implementation operating on floating point values. Another parameter that influences both the detection rate and the performance is the number K. Turk and Pentland [35] advised to set $K = 10$; experiments with our implementation demonstrate that values of $K > 12$ do not yield a significant gain in the detection rate; thus we set $K = 12$ in subsequent tests.

Computation complexity. We measure the computation complexity of the full recognition protocol, thus the efforts of both Alice and Bob. Table 1 depicts the average runtime of a single query (wall clock time) with respect to the size of the database M (second column) in seconds. Thus, matching an image against a database of size 320 takes roughly 40 seconds; this time includes all steps of the protocol of Section 4: computing the encrypted face image by Alice, projecting it into the face space, computing distances and selecting the minimum.

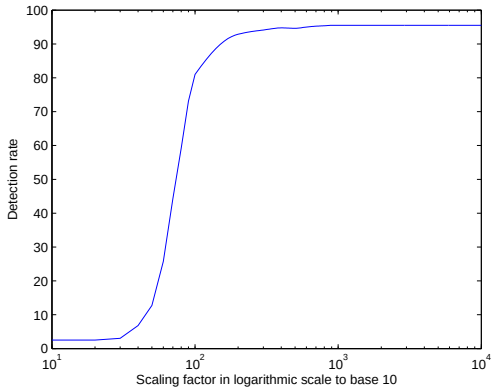

Fig. 2. Relation between scale factor and detection rate.

One can note that a major part of the computation efforts comes from computing encryptions, since they require one rather complex modular exponentiation. The time required to run the protocol can be largely reduced if these computationally expensive operations, which do *not* depend on the input image of Alice, can be computed in advance, during idle times of a processor or on a separate processor dedicated to this task. With this optimization in place, computing one encryption requires only two modular multiplications. The third column of Table 1 shows the execution time of the recognition algorithm under the assumption that *all* randomization factors r^n (Paillier) and h^r (DGK) can be pre-computed for free during idle times. In this case, matching an image against 320 feature vectors takes only 18 seconds; furthermore, the computations performed by Alice become much more lightweight, as nearly all of Alice's efforts is spent in computing encryptions.

In a third test we assume that Alice knows the eigenfaces u_i. As noted in Section 4.1, this might be the case if a (sufficiently large) public database of faces can be used to compute the eigenfaces, or if Bob explicitly decides to reveal these values to Alice. In this case Alice performs the projection and distance computation steps and sends an encrypted feature vector to Bob. The results of this experiment are depicted in the fourth column of Table 1. Observe that compared to a standard query (second column) only a small constant factor can be saved.

Communication complexity. The communication complexity highly depends on the size of Paillier and DGK encryptions; in our implementation, the size of a Paillier ciphertext is 2048 bits, whereas a DGK encryption requires only 1024 bits. Sending the encrypted image and performing the distance computations requires communication efforts independent of M; in particular, this part of the protocol requires transmission of $N + K + 1$ Paillier encrypted values (roughly 2580 kilobytes). The rest of the communication is linear in M: more precisely, the

minimum searching step requires transmission of $6M$ Pailler and $M(2\ell+1)$ DGK encryptions, which in our setting amounts to roughly 14.5 kilobytes per feature vector in the database. Table 2 shows the average amount of data in kilobytes transmitted in one run of the privacy-preserving face recognition protocol for several database sizes M (second column) and the communication complexity in case that a public basis of Eigenfaces can be used (third column). The overall communication complexity for matching an image against 320 feature vectors is thus approximately 7.25 MB.

Table 1. Computation complexity(sec.).

M	Query	With pre-computations	Public Eigenfaces
10	24	8.5	1.6
50	26	10	3.4
100	29	11.5	6
150	31.6	13	8.6
200	34.2	14.5	11.4
250	36.6	16	14.4
300	39.6	17.5	18
320	40	18	18.2

Table 2. Communication Complexity(kB).

M	Full Query	Public Eigenfaces
10	2725	149
50	3310	734
100	4038	1461
150	4765	2189
200	5497	2921
250	6228	3652
300	6959	4382
320	7249	4674

Round complexity. The round complexity of our protocol is very low. Sending the face image and receiving the result of the protocol takes one round. Another round is spent for distance computation. As the comparison protocol (see Section 5) runs in three rounds, finding the minimum of $M+1$ values takes at most $3\lceil \log_2(M+1) \rceil$ rounds. Therefore the round complexity of our protocol is $\mathcal{O}(\log_2(M))$.

7 Related Work

The problem considered in this paper is an instance of a secure two-party problem; thus standard methods of Secure Multiparty Computation [39, 7] can be applied. Secure Function Evaluation is a special case of Secure Multiparty Computation (SMC) [18, 17] where a set of players evaluate a given function on their private inputs. Both concepts were introduced by Yao [39]. Subsequently, various approaches to securely evaluating a function have been developed for different function representations, namely combinatorial circuits [18, 21], ordered binary decision diagrams [25], branching programs [28, 27], or one-dimensional look-up tables [27]. Nevertheless, these solutions tend to be impractical due to their high computational complexity for functions as the biometric matching process considered in this paper. Thus, specific protocols must be developed.

Recently there has been an increasing interest in the use of SMC for data-intensive problems, like clustering [16, 22], filtering [6] or statistical analysis [11]

of sensitive private data. Furthermore, the combination of signal processing with cryptographic techniques in order to protect privacy is an active area of research [13]; among others, solutions for recognizing speech on encrypted signals [34] or image classification and object recognition on encrypted images [38, 2] have been proposed. The latter work describes a solution to a problem that is complementary to the one discussed in the present paper (and can be used in conjunction with our solution): locating rectangular regions on an encrypted image that show human faces.

Some authors have proposed different complementary techniques for making surveillance cameras more privacy friendly, e.g. [33, 12, 40]. However, they do not consider face recognition. These approaches use methods from signal processing and pattern recognition to wipe out sensitive regions of a surveillance video automatically, based on access permissions of the surveillance personnel.

There were a few attempts to make other biometric modalities privacy-preserving, most notably fingerprints and iris codes [37, 31, 24]. However, these works consider a different setting, where the biometric measurement is matched against a hashed template stored on a server. The server that performs the matching gets to know both the biometric and the detection result (the aim is only to secure storage of templates). In contrast, our scenario even allows to hide this information. There are only a few works that apply cryptographic secure multiparty computation to the problem of securing iris codes and fingerprint templates (most notably [23, 32]); to the best of our knowledge there is no prior solution to the much more data-intensive problem of securing face biometrics.

8 Conclusions and Future Work

In this paper we have presented for the first time strong cryptographic Privacy-Enhancing Technologies for biometric face recognition systems. In particular, we provided an efficient protocol that allows to match an encrypted image showing a face against a database of facial templates in such a way that the biometric itself and the detection result is hidden from the server that performs the matching. Through extensive tests, we showed that our privacy-preserving algorithm is as reliable as a reference implementation in the clear, and that the execution of the protocol is feasible on current hardware platforms.

In this paper we used Eigenfaces, which provides a detection rate of about 96%, as core face recognition algorithm. Biometric algorithms that achieve better detection rates are known in the literature; however, these schemes are much more complex and thus more difficult to implement on encrypted images. We leave this, as well as further optimizations, as future work.

9 Acknowledgments

We would like to thank Berry Schoenmakers and Thijs Veugen for helpful discussions and the anonymous reviewers of PETS 2009 for their comments and suggestions.

References

1. The Database of Faces, (formerly 'The ORL Database of Faces'). Available at `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`. AT&T Laboratories Cambridge.
2. S. Avidan and M. Butman. Blind vision. In *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision*, volume 3953 of *LNCS*, pages 1–13. Springer, 2006.
3. I. F. Blake and V. Kolesnikov. Strong Conditional Oblivious Transfer and Computing on Intervals. In P. Joong Lee, editor, *Advances in Cryptology — ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 515–529. Springer, December 5-9, 2004.
4. I. F. Blake and V. Kolesnikov. Conditional Encrypted Mapping and Comparing Encrypted Numbers. In G. Di Crescenzo and A. D. Rubin, editors, *Financial Cryptography and Data Security — FC 2006*, volume 4107 of *LNCS*, pages 206–220. Springer, February 27-March 2, 2006.
5. O. Bowcott. Interpol wants facial recognition database to catch suspects. Guardian, October 20, 2008, `http://www.guardian.co.uk/world/2008/oct/20/interpol-facial-recognition`.
6. J. F. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57, 2002.
7. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, May 6-10, 2001.
8. I. Damgård, M. Geisler, and M. Krøigaard. Efficient and Secure Comparison for On-Line Auctions. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *Australasian Conference on Information Security and Privacy — ACSIP 2007*, volume 4586 of *LNCS*, pages 416–430. Springer, July 2-4, 2007.
9. I. Damgård, M. Geisler, and M. Krøigaard. A correction to "Efficient and secure comparison for on-line auctions". Cryptology ePrint Archive, Report 2008/321, 2008. `http://eprint.iacr.org/`.
10. I. Damgård and M. Jurik. A Generalization, a Simplification and some Applications of Paillier's Probabilistic Public-Key System. Technical report, Department of Computer Science, University of Aarhus, 2000.
11. W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA*, pages 222–233. SIAM, April 22-24, 2004.
12. F. Dufaux and T. Ebrahimi. Scrambling for video surveillance with privacy. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*. IEEE Press, 2006.
13. Z. Erkin, A. Piva, S. Katzenbeisser, and et al. Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing. *EURASIP Journal on Information Security*, 2007, Article ID 78943.
14. M. Fischlin. A Cost-Effective Pay-Per-Multiplication Comparison Method for Millionaires. In D. Naccache, editor, *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *LNCS*, pages 457–472. Springer, April 8-12, 2001.
15. J. A. Garay, B. Schoenmakers, and J. Villegas. Practical and Secure Solutions for Integer Comparison. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography — PK 2007*, volume 4450 of *LNCS*, pages 330–342. Springer, April 16-20, 2007.

16. B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On secure scalar product computation for privacy-preserving data mining. In *7th ICISC*, volume 3506 of *LNCS*, pages 104–120. Springer, 2004.
17. O. Goldreich. *Foundations of Cryptography. Basic Applications*, volume Volume 2. Cambridge University Press, first edition, May 2004. ISBN 0-521-83084-2.
18. O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *ACM Symposium on Theory of Computing — STOC '87*, pages 218–229. ACM, May 25-27, 1987.
19. T. Grose. When surveillance cameras talk. Time Magazine, February 11, 2008, `http://www.time.com/time/world/article/0,8599,1711972,00.html`.
20. Interpol wants facial recognition database to catch suspects. Heise Online UK, March 20, 2008, `http://www.heise-online.co.uk/news/British-police-build-a-database-of-portrait-photos-for-facial-\ \-recognition--/110363`.
21. M. Jacobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT'00*, volume 1976 of *LNCS*, pages 162–177. Springer-Verlag, 2000.
22. G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 593–599, New York, NY, USA, 2005. ACM Press.
23. F. Kerschbaum, M. J. Atallah, D. M'Raïhi, and J. R. Rice. Private fingerprint verification without local storage. In *Biometric Authentication, First International Conference, ICBA 2004*, volume 3072 of *LNCS*, pages 387–394. Springer, 2004.
24. T. Kevenaar. *Security with Noisy Data*, chapter Protection of Biometric Information, pages 169–193. Springer Verlag, 2007.
25. L. Kruger, S. Jha, E.-J. Goh, and D. Boneh. Secure function evaluation with ordered binary decision diagrams. In *Proceedings of the 13th ACM conference on Computer and communications security CCS'06*, pages 410–420, Virginia, U.S.A., November 2006. ACM Press.
26. M. Magnier. Many eyes will watch visitors. Los Angeles Times, August 07, 2008, `http://articles.latimes.com/2008/aug/07/world/fg-snoop7`.
27. M. Naor and K. Nissim. Communication complexity and secure function evaluation. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(062), 2001.
28. M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *ACM Symposium on Theory of Computing*, pages 590–599, 2001.
29. M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *ACM Conference on Electronic Commerce*, pages 129–139, 1999.
30. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *LNCS*, pages 223–238. Springer, May 2-6, 1999.
31. N. Ratha, J. Connell, R. Bolle, and S. Chikkerur. Cancelable biometrics: A case study in fingerprints. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR, volume IV)*, pages 370–373. IEEE Press, 2006.
32. B. Schoenmakers and P. Tuyls. *Security with Noisy Data*, chapter Computationally Secure Authentication with Noisy Data, pages 141–149. Springer Verlag, 2007.
33. A. Senior, Oankanti A, and A. Hampapur et al. Enabling video privacy through computer vision. *IEEE Security and Privacy Magazine*, 3(3):50–57, 2005.
34. P. Smaragdis and M. Shashanka. A framwork for secure speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 15(4):1404–1413, 2007.

35. Matthew A. Turk and Alex P. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
36. Matthew A. Turk and Alex P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, pages 586–591, 1991.
37. P. Tuyls, A. H. M. Akkermans, T. A. M. Kevenaar, G. Jan Schrijen, A. M. Bazen, and R. N. J. Veldhuis. Practical biometric authentication with template protection. In *Audio- and Video-Based Biometric Person Authentication, 5th International Conference, AVBPA 2005*, volume 3546 of *LNCS*, pages 436–446. Springer, 2005.
38. J. Vaidya and B. Tulpule. Enabling better medical image classification through secure collaboration. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages IV–461–IV–464, 2007.
39. A. C.-C. Yao. Protocols for Secure Computations (Extended Abstract). In *Annual Symposium on Foundations of Computer Science — FOCS '82*, pages 160–164. IEEE, November 3-5, 1982.
40. X. Yu, K. Chinomi, and Koshimizu et al. Privacy protecting visual processing for secure video surveillance. In *IEEE International Conference on Image Processing (ICIP 2008)*. IEEE Press, 2008.

A Security (Sketch)

In this appendix we sketch why the face recognition protocol is privacy preserving. For semi-honest Alice and Bob, neither learns anything on the other's input—the database and the image—except the database size and what can be inferred from the output. As the parties are honest-but-curious, it suffices to demonstrate that no information is leaked by the messages seen.

Comparison protocol. The comparison protocol allows Bob to obtain a new encryption of the minimum of two encryptions he already possesses. On the intuitive level, security towards Bob is simple. All messages received are encrypted under Alice's public keys, and Bob cannot learn anything from these without breaking the semantic security of one of those schemes.

Alice on the other hand has access to the secret key. It must therefore be argued that no information is learned from the *contents* of the encryptions sent. But this is the case, as Alice only receives values that Bob has masked: this includes the messages sent for the secure selection of the minimal and ID, as well as $[d] = [z + r]$, which is statistically indistinguishable from a uniformly random $(\kappa + \ell + 1)$-bit value.

Treatment of the permuted $[\![e_i]\!]$ of Section 5.3 is only slightly more difficult. Alice either sees a list of uniformly random non-zero values, or an equivalent list, where one entry is replaced by a zero. A list of random values provides no information. Similarly, the zero does not cause any problems: Its position is random due to the permutation, and its existence also reveals nothing as it occurs with probability $1/2$; s can be viewed as a one-time-pad for the outcome. Thus, neither Alice nor Bob learn anything from the comparison protocol.

Complete Recognition Protocol. The proof of security of the full protocol is similar to that of the comparison. In addition to the comparisons, interaction is only needed to compute the distances $D_1, \ldots, D_M$. As above, the values

$x_1, \ldots, x_K$ that Alice receives are masked, in this case they are uniformly random over the whole plaintext space. Bob again receives only semantically secure encryptions, so he also learns nothing. This is also true when he receives Alice's input.

Based on the above intuition, a formal simulator proof is easily constructed. Given one party's input and the output, simulation of the other party is easy: Alice must be handed encryptions of random values, while Bob can be handed encryptions of 0, which are indistinguishable due to the semantic security.