SCiFI – A System for Secure Face Identification

Margarita Osadchy
University of Haifa

Benny Pinkas
University of Haifa

Ayman Jarrous
University of Haifa

Boaz Moskovich
University of Haifa

***Abstract*—We introduce SCiFI, a system for Secure Computation of Face Identification. The system performs face identification which compares faces of subjects with a database of registered faces. The identification is done in a secure way which protects both the privacy of the subjects and the confidentiality of the database. A specific application of SCiFI is reducing the privacy impact of camera based surveillance. In that scenario, SCiFI would be used in a setting which contains a server which has a set of faces of suspects, and client machines which might be cameras acquiring images in public places. The system runs a secure computation of a face recognition algorithm, which identifies if an image acquired by a client matches one of the suspects, but otherwise reveals no information to neither of the parties.**

Our work includes multiple contributions in different areas:

- **A new face identification algorithm which is unique in having been specifically designed for usage in secure computation. Nonetheless, the algorithm has face recognition performance comparable to that of state of the art algorithms. We ran experiments which show the algorithm to be robust to different viewing conditions, such as illumination, occlusions, and changes in appearance (like wearing glasses).**
- **A secure protocol for computing the new face recognition algorithm. In addition, since our goal is to run an actual system, considerable effort was made to optimize the protocol and minimize its online latency.**
- **A system – SCiFI, which implements a secure computation of the face identification protocol.**
- **Experiments which show that the entire system can run in near real-time: The secure computation protocol performs a preprocessing of all public-key cryptographic operations. Its online performance therefore mainly depends on the speed of data communication, and our experiments show it to be extremely efficient.**

***Keywords*-Secure computation, face recognition, privacy.**

I. Introduction

Video and camera based surveillance is very common, and is found to be useful for fighting crime. On the other hand, the ubiquity of such surveillance is a major concern for the public, that feels that its privacy is being violated. Our work focuses on face recognition systems, which can automatically identify if some known suspects appear in a large set of images. Such systems can be useful, for example, for automatically searching for suspects in a stream of images coming from public places. On the other hand, these systems can be misused to track people regardless of suspicion, and a rogue operator can even combine data from these systems with a universal database linking faces to identities, such as a database of drivers' license photos.

A simple solution to the privacy concern might be to provide each camera with the list of images of suspects, perform the identification task locally at the camera, and report to the operator only in the rare cases where a match is found. This solution might not be acceptable, though, if the list of suspects is confidential, as is often the case. There is therefore a need for a solution which protects both the privacy of the public and the confidentiality of the data held by authorities.

We developed the SCiFI system which matches images taken by a client camera to a list of images (of potential suspects) which are held by a server. Face identification in SCiFI is based on a novel face recognition algorithm that performs very well in terms of applicability to real-life images and robustness to unseen conditions (e.g., different illumination conditions). The matching is done in a privacy preserving way, using efficient methods of secure computation, and does not reveal any information to the parties, except for whether a match was found. Furthermore, to further protect the database of suspects it is possible to distribute the operation of the server to run on several machines, such that each of these machines knows only part of each suspect's face. (The details of a system utilizing this distributed approach are given in the full version of the paper.)

System overview: SCiFI is composed of two major components, a server, which stores a set of face representations of subjects from a confidential list, and a client, whose input is a single face. In a typical setting the server might have a list of faces of suspected persons, while the client might be a camera which takes photos of passersby.[1] The system must find out if the face held by the client matches any of the faces in the server's list. The system must also satisfy two other requirements: (1) As with any biometric data, it is unlikely that there will be an exact match between the image acquired by the client and the image of the same person that exists in the list. Therefore a fuzzy face identification algorithm must be used. (2) The matching must be done in a privacy preserving manner. Namely, the server

[1]We assume that the recognition module of the client receives images of faces preprocessed for face recognition. We do not discuss in this paper methods for face detection or localization that perform this preprocessing. See, e.g., [1], [2] for related work.

and client must not learn any information except for whether there is a match between the client's input and a face in the server's list.

Achieving this goal required designing a new face recognition algorithm which has very good performance in terms of the robustness of recognition, and can also support secure computation protocols. (The main challenge in this respect is that face recognition algorithms typically use data representations over the Real numbers, whereas secure protocols operate over finite fields, and a simple conversion of existing face recognition methods to finite fields results in degradation of recognition.) We also designed secure protocols, based on homomorphic encryption and oblivious transfer, computing the face recognition algorithm. We conducted experiments showing the accuracy of the face recognition algorithm and the nearly real-time performance of the secure protocols.

Face Recognition and Privacy: Face recognition is an inexpensive and non-intrusive technique, and its usage for user verification is more convenient than using passwords, hardware devices, or other biometric identification methods. Thus previous research in incorporating face technology with security focused on the *verification* task [3], where the user identifies himself to the system and the system verifies his identity by comparing the current image of his face with a representation of his face stored in the system. Such applications assume a controlled environment, rely on user's cooperation, and usually use several images of a person under controlled conditions in the registration phase.

We address a different recognition task, denoted in the face recognition literature as *identification*. This is a one-to-many recognition task where a single image is compared with a list of stored images. This task is more useful in surveillance applications, like the detection of criminals or terrorists in public places, or a search for a missing person. It has several distinguishing characteristics which make it much harder to implement than the verification task:

- The registration of persons into the system is based on very few images of them (usually a *single* image per person) which might have been taken without the person's cooperation, under arbitrary conditions of illumination and pose, and might be of poor quality.
- The recognition routine is also different: Given a novel image of a face under arbitrary, and not necessarily ideal, viewing conditions, the system must recognize if this is an image of one of the individuals stored in the system or otherwise reject it.

In addition, unlike the one-to-one verification task, the one-to-many identification task, which is done without the subject's cooperation, must be robust to different changes that are likely to occur between the original picture stored in the database, and the image taken by the client camera. These can include, for example, changes in illumination, i.e. in the amount and direction of light; changes in the viewpoint of the camera, and in the pose of the subject; different occlusions which hide parts of the subject's face; difference in facial expressions; and changes in local appearance due to facial hair, makeup, glasses, etc., which might be used by the suspect to avoid being identified.

Figure 1: Examples of variation in test images : top row – illumination direction and glasses on/off, bottom row – three subjects with variation in pose, lighting, and facial expression

Figure 1 describes some examples of variation in test images. These images are taken from the FERET [4] and PIE [5] databases of images that we use in order to test the recognition performance of SCiFI.

Face recognition algorithms and identification: As with any biometric data, two images of the same person are never identical. On the other hand, authentication based on passwords or cryptographic keys always expects the user to enter the same password or use the same key. Representations used in recognition must be designed to produce the same results for similar, but not necessarily identical, inputs. In cryptographic algorithms only identical inputs enable successful authentication, and therefore they cannot be applied to biometric recognition. (To bridge this gap, there have been attempts in cryptography to develop noise resistant one-way hash functions [6], [7], [8]. These are discussed in Section I-A.)

The novel algorithm that we use in SCiFI performs well in the one-to-many identification task since it can generalize to unseen conditions. There are other face recognition algorithms which are robust to changes in the environment in which photos are taken. The algorithm of SCiFI is unique, however, in that it lends itself easily to secure computation, which is inherently based on the usage of discrete mathematics. Other effective face recognition algorithms employ continuous face representations which are compared by complex measures of similarity that in some cases are not even metric. Such representations are not easily supported by cryptographic algorithms. A naive conversion from the continuous comparison methods used in face recognition to a discrete measure, using, e.g., simple quantization, affects the accuracy of recognition, and result in degraded performance. This issue is elaborated in more detail in Section I-A.

An overview of the face representation used in SCiFI: We propose and use an index-based face representation that was specifically designed for usage with secure computation. The representation is based on only a single known image per person. It is robust to illumination changes, occlusions, shadows, highlights, and local changes in appearance (such as wearing glasses). We do not address pose in this paper, but in principle the face representation can be extended to variation in pose.

The representation is based on the idea of facial composite (aka photo-robot), where a face is formed as a collection of fragments taken from vocabularies of facial features. (A similar system is typically used by police departments to record eyewitness's memory of a face.) The vocabularies of facial features contain typical appearances of facial fragments obtained from a set of people unrelated to the face that should be reconstructed. (To exemplify and simplify the system, assume that the vocabulary contains a set of typical images for each facial component, such as the nose, eyes, eyebrows, etc.) Choosing the best match for each part and placing those parts in the spatial configuration similar to the sought face is likely to produce an output close to the original face. The reconstruction process is very fast, and although the result will not be photo-realistic, it can still suffice for recognition.

In SCiFI we use more parts than are commonly used by humans, and they do not necessarily correspond to the semantic parts of a face. We propose to represent a face by a vector which is composed of (1) indices into part vocabularies, and of (2) quantized relative distances of the parts to the center of the face. We assign to each part a *set* of words from the corresponding vocabulary, instead of a single match. (E.g., the nose might be represented by the indices of the four noses from the vocabulary which are most similar to it. Such flexibility in the representation makes it robust to image conditions and local noises.) Any two representations are essentially sets that can be compared by a secure computation of their set difference.

The face recognition part is very fast, it does not require dense correspondence between the input and database faces (but rather only 3-5 points for initial alignment of faces), and it does not use 3D models or any extensive training. The secure computation of this algorithm has a relatively small overhead, which means that it can be deployed in real systems.

Our contributions: The SCiFI system is the first secure face identification system that is suited for real-life applications. The advantages of SCiFI over existing identification methods can be summarized as follows:

- As is demonstrated by our experiments, the SCiFI system uses a face identification algorithm which provides results that are comparable to the state-of-the-art in illumination invariant face recognition, and achieves very good robustness to occlusions. The algorithm can be based on computing the Hamming distance, and is therefore a natural candidate for secure computation. The results are superior to those of the Eigenfaces algorithm which, to our knowledge, is the only other face recognition algorithm for which secure computation was applied.
- SCiFI uses efficient secure computation techniques to identify, without leaking any other information, whether a given image is in a suspects list.
- We implemented the complete SCiFI system. This implementation computes the face representation at the client and then runs a secure computation with the server. We ran experiments of both the accuracy of face identification, and of the performance timings of the entire system (including communication between the parties). Even though we implemented the cryptographic part in Java, its performance is remarkably fast, thanks in part for optimizations which perform a precomputation of computation intensive tasks, and enable near real-time identification of suspects.

Another attractive property of SCiFI is that face identification is done through an interactive protocol which reveals to the client an upper bound on the number of items in the server's database, and therefore the client can be assured that the server does not use the system for large scale image identification (e.g., for comparing the client's input with images of all people who have a criminal record). Another advantage is that the server cannot store the client's input for future use. This prevents the server from, say, comparing those who pass by the client's camera today to suspects whose faces will become known in the future.

A. Related Work in Face Recognition

Different aspects of face recognition have been addressed in numerous papers. Here we focus on the robustness of the representation, and on protection of the biometric data.

Robust Face Representations: An ideal face representation should be robust to illumination changes, occlusions, and other variations of the image. In particular, illumination variations are known to greatly influence the appearance of human faces. Adini et al. [9] have shown that images of different faces under similar illumination vary less than the same face under different illumination. This is a great obstacle to any face identification system that must work in real-life conditions.

Most existing systems for robust face cognition use Real-valued representations of faces, and therefore cannot be used as is in secure computation. Straightforward attempts to quantize the values of the representations result in poor recognition results. A previous design of a privacy preserving face identification system was based on the Eigenfaces algorithm (see discussion in Section I-B). The Eigenfaces algorithm is a well known basic face recognition algorithm [10]. However, its recognition performance is poor

if the training and suspect images were taken in different conditions (see our experiments in Section VI).

Protection of Biometric Data: It is known that different readings of the same biometric data of the same person are usually similar, but not necessarily identical. Therefore the biometric representations used in recognition are designed to produce the same results for similar inputs. An easy way to do that is to store a copy of the original biometric data and compare it to the reading of the user's data. This, however, enables attackers to steal this data. A more secure approach would be to store a one-way hash of the biometric data, as is used with password based authentication. In the case of biometric data this approach requires the usage of noise resistant one-way hash functions, referred to as fuzzy hashing (or fuzzy commitments), or as secure sketches. Such functions were described in [6], [7], [8].

Recent attempts in integrating these schemes in face recognition [3] focus on the simpler, one-to-one, verification task. These methods are limited to controlled environments, and use a large number of images of the subject for registration. In SCiFI the biometric data is stored at the server, which is assumed to be much more secure than users' machines, and therefore we do not use these methods to protect it.

Another approach for securing biometric data uses revocable biometrics, see e.g. [11], [12]. It involves transforming the biometric data using a one-way hash function such that it is impossible to restore the original data from the result. Different persons use different transformation functions and therefore their images cannot be compared. If the stored data is compromised, the person can simply re-enroll using another transformation function.

Computer vision and privacy research: Recent work on "blind vision" [13] investigates a setting which is different than ours. In that setting one party wishes to detect faces in a collection of sensitive images it owns, and the other party has a confidential face detection *algorithm*. The work shows how to use cryptographic secure computation to privately perform the recognition task.

There has also been a series of results on obscuring private information in digital surveillance videos [14], [15], [16], [17], [18]. The setting is one where the persons watching the surveillance videos can detect suspected behavior without being able to examine the faces of the subjects. The challenge is to identify the private information and obscure it in real-time, while being able to recover it if needed.

B. Related Work on Secure computation

The problem we discuss is that of secure computation (or Secure Function Evaluation – SFE). It involves several parties with private inputs that wish to compute a function of their joint inputs, without revealing to an adversarial party (or a coalition of such parties) any information that cannot be computed using the input of the adversary and the output of the function.

There exist well known solutions for secure computation of any function (see e.g. [19] for the two party case, or [20] for a detailed discussion of the subject). The general method employed by most of these solutions is to construct a binary circuit that computes the required function, and run a distributed protocol that securely evaluates the circuit gate by gate. The communication overhead of these generic protocols is linear in the size of the circuit, and the computation involves an oblivious transfer for each input bit. It seems hard to apply these methods to compute continuous functions or represent Real numbers, since the methods inherently work over finite fields.

Secure computation of Eigenfaces: Secure computation of a face recognition algorithm was previously applied to the Eigenfaces algorithm. A secure protocol for computing this algorithm was presented in [21], and a considerable improvement in efficiency was shown in [22]. The secure protocol in that work computes a quantized version of Eigenfaces. The quantization did not affect the recognition performance so much, but as we show in Section VI, the original Eigenfaces algorithm is not very good in recognizing images taken in unseen conditions.

The Eigenfaces algorithm is based on computing the Euclidean distance, whose secure computation is more complicated than that of the Hamming distance. It was unknown before our work, how to translate the face recognition problem to a computation of the Hamming distance, which lends itself more efficiently to secure computation.

To exemplify the efficiency of SCiFI, we note that the secure computation of Eigenfaces must send a homomorphic encryption of every pixel of the image. The experiments conducted in [21], [22] use images of $92 \times 112 = 10304$ pixels. (It is hard to imagine that smaller images could be used for meaningful recognition.) This image size translates to more than $10,000$ homomorphic encryptions, and any increase in the image size (which is needed in order to improve the quality of recognition) will result in an increase in this number. In SCiFI, on the other hand, a face is always represented by a vector of 900 bits, independently of the size of the image. The system sends a homomorphic encryption per each of these 900 bits. Moreover, this communication can be done in a preprocessing phase, while the online communication requires sending only a single 900 bit representation of the face.

The papers discussing secure computation of Eigenfaces provide timing results for the implementation of the secure computation part alone (as well as an analysis of the number of bytes that must be communicated). We report on experiments which time also the communication layer, including socket opening, etc.

II. System Architecture

The SCiFI system is composed of a server and a client. The operation of the system can be separated into an offline (or preprocessing) part, and an online part.

The structure of the SCiFI system is described in Table I. The offline part prepares the face recognition database, by computing representations of the faces that are in the server's list. This stage is also used to execute some initializations and preprocessing of the cryptographic algorithms.

The online part is executed after the client obtains an image. This part decides whether the image is of a person who appears in the list, and can be separated into two distinct stages. In the first stage the client prepares a representation of the face that it acquired. In the second stage the two parties execute a cryptographic algorithm which compares the client's representation with those in the server's list, and decides whether there is a match between them (without revealing any other information).

	server	**client**
offline	prepare representations of faces in server's list	
	initialize cryptographic algorithms	
online		- acquire face - generate representation s of the acquired face
	run a secure protocol, checking if there is a match between s and list	

Table I: The operation of the system.

This modular structure of the system enables us to separate the description of the system into a description of the face recognition part, in Section III, and a description of the cryptographic parties, in Section IV.

III. The Face Recognition Algorithm

We assume that facial features have a number of typical appearances and almost every face can be generated by combining such components. Let X denote a set of people enrolled in the recognition system. Assume that we have a (possibly public) database Y of faces unrelated to X. While this database is public, we want to protect the data in X. We divide a face into p parts and build vocabularies of typical appearances (that we call *words*) per part using Y. Let $V = \{V_1, \ldots, V_p\}$ denote the part vocabularies (where, for example, in a simplified way, V_1 might include 20 options for the nose, V_2 includes 20 options for the mouth, etc.).

Let I be an image of a person from X. We represent I as a vector of indices, denoted $\mathbf{s}$, of words from the part vocabularies which are most similar to the parts in I. Our goal is to produce almost identical vectors of indices from different images of the same person. Different face representation vectors can now be compared by computing their set difference (the set difference of two sets A and B is defined as the difference between the size of their union and the size of their intersection; if $A = B$ then their set difference is 0).

Previous research shows that *locations* of facial features in a face have good discriminative power. Thus our representation takes these locations into account as well, and includes quantized distances from each part of the face to the center of the face. (An exact description of the representation appears in Section III-C.)

The proposed representation has a number of advantages: (1) The model is tailored for computing the set difference and the Hamming distance, which are discrete metrics that can be used in secure computation. (2) The vocabularies of parts are constructed from a set Y of people unrelated to the set X of enrolled people, and therefore there is no need to store the original data of the persons enrolled in the system. (3) The representation also makes it possible to use only a single image per person for recognition, which is an important feature for practical systems (where, say, only a single photo of a suspect is available). (4) The vocabularies are constructed from Y, and therefore they stay fixed whether X changes or not, and thus no retraining is needed when a new subject is added to the system. (5) The proposed model is more flexible than the existing part-based face representations, because each part in our model is represented by an unordered set of appearances. The use of set difference for comparison allows for partial similarity between the representations of the same individual. This contributes to the robustness against occlusions, shadows, highlights, and other local changes in appearance. (6) The proposed representation uses facial components which are small patches, which allows to assume their planarity and use illumination insensitive local descriptor of patches (e.g., SIFT [23]) in order to cope with varying illumination. To summarize, the proposed representation is very robust to environmental changes and is designed to be incorporated with privacy preserving mechanisms. These features make it an excellent choice for security applications.

Section organization: The following sections describe in detail the construction of face representations. Readers who are only interested in the security applications can jump to Section III-C which describes the representation which is used by the secure computation.

A. Preprocessing

Definition of Parts: We define a regular grid, corresponding to the centers of the parts, over facial area with higher variance, namely eyes, eyebrows, nose and mouth. Patch sizes were chosen to be relatively small (20% of face width) in order to have many patches with minimum overlap.

Part Vocabularies: The construction of part vocabularies consists of three steps: (1) Normalization of images of subjects in Y to a canonical size. (2) Extracting patches from images of subjects in Y. Prior to extraction, patches are localized, by searching a corresponding template from

an average face in images from Y. (3) Selection of words for part vocabularies. In this step, patches corresponding to the same part are clustered and only a single patch per cluster is chosen to be a word in the part vocabulary. This aims to remove similar words from the vocabularies.

At the end of the process p vocabularies of parts are formed, one for every face part. Each vocabulary containing N words, which are given unique indices in the range $[0, N-1]$ to be used in the face representation.

Distance Vocabularies: The spatial information is modelled by the distance from the center of a part to the center of the face. During the preprocessing stage, we estimate the distance distributions of the parts and quantize them into a number of bins. Each bin is given a unique index. The estimation is done on the subjects from the public set Y.

B. Input to the system

We assume that the input to the system is an image of a face in which the the positions of the eyes and mouth are known. The positions of the eyes and mouth can be marked manually by an operator of the client's module or determined using automatic methods for detection of facial features [24], [25]. These positions are used for alignment purposes.

C. Face Representation

The representation of a face has the following form. We denote by N the number of words in each part vocabulary, and by p the number of parts in the face representation. A full face representation is in the format $s = (s^a, s^s)$ and contains the following components:

- *Appearance component:* s^a is the appearance component and is composed of p sets $s^a_1, \ldots, s^a_p$, one set per facial part, where each set contains the indices of n out of N words of the part vocabulary. To select a set s^a_i for the part i we define a window centered at the corresponding grid point in the input image. Every word from the part vocabulary is matched against the defined window. The indices of the n words that have the most resemblance with the part i are selected for the set s^a_i.
- *Spatial component:* s^s is the spatial component of the representation. Since we want to use discrete values, the representation uses quantized distances. The spatial part is therefore composed of p sets $s^s_1, \ldots, s^s_p$, where each set contains z indices of bins of quantized distance from the center of the face (namely, the set s^s_i is a quantized representation of the distance of the ith part from the center of the face). Denote the total number of these bins by Q. The ith part of the input face is localized by matching the same part from the average face against the local window centered at the grid point.

Our experiments use $p = 30$ face patches, with a vocabulary of $N = 20$ visual words and $Q = 10$ quantized distance bins per patch. We empirically found that it is best to set the number n of indices in each set s^a_i to 4. The number z of indices in each set s^s_i is 2. This means that the appearance component contains $p = 30$ sets, where each set contains $n = 4$ words out of $N = 20$ options. The spatial component contains $p = 30$ sets of $z = 2$ words out of $Q = 10$ options.

D. Recognition

The recognition task in SCiFI consists of deciding if a face in the image acquired by the client matches one of the faces in the database stored by the server. Two images are said to match if the set difference between their two representations is below some predefined threshold. To increase the accuracy of recognition, the system learns, in a preprocessing stage, an individual threshold for every person in the database. Then, in the real time phase, the representation of the client's face is compared with every face representation in the database and is said to match it if their set difference is smaller than the corresponding threshold. If the client's face passes one of the individual thresholds it is identified as a match.

Learning individual thresholds: Learning individual thresholds is a hard task because these thresholds depend on variations in different images of the same face, while we assume that SCiFI has only a single image of a suspect for registration. The invariance of our face representation to illumination changes and its robustness to occlusions reduces the intra-user variance, but does not cancel it completely. There are two possible solutions to this problem. One is based on the assumption that intra-user variation is independent from the identity of the user, meaning that the variance of the representation depends on the viewing conditions and occlusions and not on the user himself. Then we can learn this variation using a public database. Such an assumption, however, is rather simplistic, since some people tend to change their appearance more then others (makeup, facial hair etc). An alternative solution is to determine a threshold on the set difference for each person that will discriminate him/her from an ensemble of people which includes individuals registered in the system and can also include other subjects unrelated to the system (which should ideally represent typical inputs to the system). An initial threshold for the ith user is set based on the smallest set difference between him and the rest of the people in the ensemble and is corrected according to the viewing conditions in the image which can be determined in the client's module and sent to the server without revealing any information about the face. We haven't yet implemented the correction part and used the initial thresholds in reported experiments.

IV. The Cryptographic Algorithms

This section describes the cryptographic algorithms used for secure computation of face recognition, with an emphasis on pushing as much as possible of the computation to a

preprocessing stage which can be performed before the client obtains its inputs. This is done in order to optimize the time it takes to securely compute a face recognition after obtaining an image of a face.[2]

A. Cryptographic Tools

We only consider security against semi-honest adversaries (also known as passive, or honest-but curious adversaries). Namely, it is assumed that corrupt parties follow the protocol but might try to learn additional information. For lack of space we do not provide formal definitions of security in this workbut rather refer the reader to [20] and note that we follow the definitions in that book (for the semi-honest case). Informally, we remark that security can be defined by requiring that the entire view of each party during the protocol can be simulated given only the input and output of the party. Therefore the protocol execution itself does not add any new information to the party.

A major tool that we use is an additively homomorphic encryption. This is public-key encryption which enables, given two encryptions $E(m_1)$, $E(m_2)$ and without knowledge of the private key, to compute $E(m_1 + m_2)$, or compute $E(c \cdot m_1)$ for any known constant c. We require the encryption system to be semantically secure. Namely, an adversary which does not know the private key and which is given a message which is an encryption of one of two plaintexts, where these plaintexts might even be chosen by the adversary, cannot distinguish which one of them is encrypted in the message. In particular, this implies that the encryption scheme must be probabilistic and different encryptions of the same plaintext will be different. We specifically use the Paillier cryptosystem [26].

1-out-of-N Oblivious transfer, denoted OT_1^N is a two party protocol, run between a sender with N inputs $X_0, \ldots, X_{N-1}$, and a receiver with an input $i \in \{0, \ldots, N-1\}$. The receiver learns X_i and nothing else, and the sender learns no information. There are different variants of OT and a rich research on this subject. OT is the basic tool of secure computation, and it was shown how to base secure computation on OT alone. It was shown in [27] how to implement OT_1^N using $\log N$ invocations of OT_1^2 and N symmetric encryptions. OT_1^2 can be implemented using several public-key operations (i.e., exponentiations), using, say, El Gamal encryption. We will also use the fact that it is easy to preprocess the public-key operations: In the preprocessing stage the server prepares N random pads and the parties run a OT_1^N of these values, where the client's input is a random index i'. In the online stage, the client, which wants to learn item i, sends the difference between i and i' to the server, which shifts the order of its pads accordingly. The server then computes the exclusive-or of each pad with the corresponding input X, and sends the results to the client (who can decrypt only one of these values). As a result, online computation consists only of efficient exclusive-or operations. We will use this variant of OT in our implementation.

[2] An alternative approach to our protocols would have been to apply Yao's generic secure [19] two-party protocol to the recognition algorithm. This would have required expressing the algorithm as a circuit which computes and compares many Hamming distances, and then sending and computing that circuit. The protocol would have had to perform 900 oblivious transfers, instead of 9 oblivious transfers per item in the server's database, as is described for the $F_{\mathrm{threshold}}$ protocol below. We therefore believe that the performance of our protocols is significantly better than that of applying generic protocols.

B. Functionality

The cryptographic algorithm of SCiFI receives its input from the face recognition part of the system. We define the functionality of this algorithm in terms of its input and output.

Input: The input of the client, as well as each entry in the server's list, contain a representation of face, in the format defined in Section III-C. We will translate this representation to an equivalent representation as a binary vector, which will be more convenient for applying the cryptographic algorithm. The new representation is defined as follows:

- Every set s_i^a is represented as a binary vector v_i^a of length $N = 20$. Each bit of v_i^a corresponds to a different index of an entry in the part vocabulary. The vector v_i^a has exactly $n = 4$ bits set to 1, in the locations of the n indices in the set s_i^a.
- Every set s_i^s is represented as a binary vector v_i^s of length $Q = 10$. Each bit of v_i^s corresponds to a different bin of quantized distances. The vector v_i^s has exactly two bits set to 1, in the locations of the indices in the set s_i^s.
- A face is represented as a vector $v = v_1^a | \cdots | v_p^a | v_1^s | \cdots | v_p^s$. The length of V is $p \cdot (N + Q) = 30 \cdot (20 + 10) = 900$ bits.

It is important to note that the set difference between the representations s, s' of two faces, is exactly equal to the Hamming distance of the vectors v, v'. The Hamming distance of these 900 bit vectors can be at most $p \cdot 2 \cdot (n+2) = 30 \cdot 2 \cdot 6 = 360$, since each v_i^a component has only $n = 4$ bits set to 1, and every v_i^s component has only 2 bits set to 1. Furthermore, in our experiments we identified that the maximum Hamming distance between two face representations is even smaller. We denote the bound on the value of the Hamming distance as $d_{\max}$. In our experiments we found it to be $d_{\max} = 180$. We use this fact to further optimize the cryptographic algorithm.

Output: The goal of the system is to identify a match between the client's input and an item in the database. (Typically, it is expected that only a single match will be found, since each entry corresponds to a different face.) There are different options for identifying a match based on the Hamming distance of the face representations. We

describe here two functions which can be used for this purpose.

- $F_{\text{threshold}}$. This functionality has an additional input, a threshold t_i, for each face in the server's database. The functionality computes the Hamming distance between the representation of the client's input and each of the representations of the items in the server's list. The output is the index of the item (or items) in the server's list whose Hamming distance with the client's input is smaller than the corresponding threshold t_i.
- $F_{\text{min+t}}$. The output is the index of the item in the list whose Hamming distance with the client's input which is minimal. However, if this distance is larger than the threshold, i.e. if no database item is closer to the input than the threshold, then no output is given.

Choosing the right functionality: The outputs of the functionalities $F_{\text{threshold}}$ and $F_{\text{min+t}}$ only differ when the basic algorithm finds the client's image to be similar to more than a single face in the server's database. Ideally this would not happen since a person should only be similar to himself. Still, if the similarity thresholds t_i are not accurately calibrated then a single image might be considered by the algorithm to be close to two or more database images (i.e., the set differences will be smaller than the corresponding t_i values). In that case the $F_{\text{threshold}}$ functionality outputs the identities of all database items which are close to client's input, whereas $F_{\text{min+t}}$ only outputs the identity of the closest item. While $F_{\text{min+t}}$ provides more privacy, one could argue that it is more reasonable to use $F_{\text{threshold}}$, since similarity to any of the suspects must be investigated. We also note that secure computation of $F_{\text{min+t}}$ is harder than that of $F_{\text{threshold}}$. For both of these reasons, we only implemented the latter in the SCiFI system (although we discuss the computation of both functionalities in this paper).

Learning the output: It is possible to let only the client, only the server, or both parties, learn the output of the computation. We will describe protocols for all these cases.

C. Cryptographic Protocols for Face Recognition

We start with a description of a protocol for secure computation of the $F_{\text{threshold}}$ functionality, and then describe how to optimize the implementation of the protocol. We then describe the $F_{\text{min+t}}$ protocol. Both protocols are based on extensions of ideas used in [28]. As described in Section IV-B, each face representation is of length $\ell = p \cdot (N+Q) = 900$ bits. The Hamming distance between two representations is known to be at most $d_{\max} = 180$.

The secure protocol computing $F_{\text{threshold}}$, where only the client learns an output, is described in Figure 2. In the protocol the client and server first use homomorphic encryption to count the number of locations in which their two input words differ. The result is in the range $[0, d_{\max}]$. None of the parties learns this value, but the client learns the sum of the Hamming distance and of a random number r chosen by the server. Next, the two parties use 1-out-of-$(d_{\max}+1)$ oblivious transfer to map the result to the appropriate output value: The sender is the server, and it sets the OT inputs to be $X_0, \ldots, X_{d_{\max}+1}$ where X_j is equal to 1 if $j + r \bmod (d_{\max}+1)$, is between 0 and the threshold t_i. The receiver is the client. Its input to the OT is the sum of the Hamming distance and r.

$F_{\text{threshold}}(w, (w^1, \ldots, w^N))$

INPUT: The client's input is a binary word $w = (w_0, \ldots, w_{\ell-1})$, the server's input is a list (database) of N binary words, $w^1, \ldots, w^N$, where $w^i = (w_0^i, \ldots, w_{\ell-1}^i)$. The server has additional inputs $t_1, \ldots, t_N$, which are the thresholds associated with each database word.
In addition, the parties know an upper bound $d_{\max}$ on the Hamming distance between two words. In general, $d_{\max} = \ell$, but in our application we know that $d_{\max} \leq 180$.
OUTPUT: The client learns the indices i for which $d_H(w, w^i) \leq t_i$. The server learns nothing.
The protocol uses a homomorphic encryption function, $E_{pk}(\cdot)$. The plaintexts are in a ring or a field $\mathcal{F}$. pk is a public key that both parties know, but only the client knows the corresponding private key and can decrypt messages.

1. The client sends the homomorphic encryption of each bit of the binary representation of $w = w_0, \ldots, w_{\ell-1}$.
   The server receives the encrypted representation $\{E_{pk}(w_0), \ldots, E_{pk}(w_{\ell-1})\}$. The following steps are repeated for each database item $w^i \in \{w^1, \ldots, w^N\}$:
2. For each bit location j the server calculates $E_{pk}(\vartheta_j)$, where $\vartheta_j = w_j \oplus w_j^i$. The calculation can be done in the following way (although a more efficient computation method is described in the text below): $E_{pk}(\vartheta_j) = E_{pk}(w_j) \cdot_{pk} (1 - w_j^i) +_{pk} (1 -_{pk} E_{pk}(w_j)) \cdot_{pk} w_j^i$.
3. Using the homomorphic properties, the server sums the results of the previous step and computes $E_{pk}(d_H) = \sum_0^{\ell-1} E_{pk}(\vartheta_j)$. The value d_H is in the range $\{0, 1, \ldots, d_{\max}\}$ and is equal to the Hamming distance between w and w^i.
   The server chooses a random value $r_i \in \mathcal{F}$, computes the value $E_{pk}(d_H + r_i)$, and sends it to the client. Note that with overwhelming probability of $1 - d_{\max}/|\mathcal{F}|$, this addition operation does not involve a modular reduction.
4. The client receives $E_{pk}(d_H + r_i)$ and decrypts the result.
5. Next, the parties map the result to the appropriate output value, by invoking a $\text{OT}_1^{d_{\max}+1}$ protocol where the client is the receiver and the server is the sender:
   - The input of the client is $(d_H + r_i) \bmod (d_{\max}+1)$.
   - The server has inputs $X_0, \ldots, X_{d_{\max}}$, where

$$X_j = \begin{cases} 1 & \text{if } 0 \leq (j - r_i) \bmod d_{\max} \leq t_i \\ 0 & \text{otherwise} \end{cases}$$

Figure 2: The $F_{\text{threshold}}$ protocol where the client learns the output.

Correctness: In Step 3 of the protocol the server computes the value $E(d_H + r_i)$, where r_i was chosen at random. If $r_i < |\mathcal{F}| - d_{\max}$ (which happens with overwhelming probability) the computation of $d_H + r_i$ in $\mathcal{F}$ does not involve a modular reduction and has the same result as adding them

over the integers. In that case, reducing the result modulo $d_{\max}+1$ (in Step 5) is equal to $(d_H + r_i) \bmod (d_{\max}+1)$. The client uses this result as its input to the 1-out-of-$(d_{\max}+1)$ OT protocol, while the server sets the sender's inputs in the OT to be shifted by r_i locations. As a result, the output of the client in the OT protocol is as required.

Security: The security of the protocol is proved assuming that the oblivious transfer protocol is secure. Namely, the proof is described in a hybrid model where oblivious transfer is implemented by an oracle which receives the inputs of the two parties and sends the output to the receiver. Recall also that the setting is one where the two parties are assumed to be semi-honest. According to [20], the proof can be carried out according to the simulation paradigm, where we need to show that it is possible to simulate the view of each party (i.e., simulate the distribution of all messages that this party receives and sends) given only the party's input and output. Given this simulation it is clear that the view reveals nothing that cannot be computed from the input and output alone.

Let us first examine the client, whose input, consisting of the private key and of w, and output, which consists of a bit for each input of the server, are given to the simulator. For each index i of an input w^i of the server, the simulator generates the first message from the client to the server, which is an encryption of each of the bits of the client's input. The simulator then simulates the message received by the client in Step 4, which is an encryption of a random value $R = d_H + r_i$. Finally, the simulator sets the input of the client to the OT, which is $R \bmod (d_{\max}+1)$, and its output of the OT which is the output bit corresponding to the ith input of the server.

Consider now the server. The server's set of input words is given to the simulator. For each word, the simulator generates the first message sent by the client, which is an encryption of ℓ bits, all equal to 0. Since we assume that the encryption scheme is semantically secure, the server cannot distinguish these encryptions from encryptions of the actual input of the client. Now, in the rest of the protocol the server sends a message in Step 4, and sets its input to the OT in Step 5. These values are a function, defined by the protocol, of the server's inputs (known to the simulator), of the ciphertexts the server receives from the client, and of a value r_i chosen uniformly at random in $\mathcal{F}$. The simulator can therefore easily compute these messages by itself.

D. Optimizations and Overhead

Our goal was to implement a system computing this protocol in near real-time, so that face identification can be performed sufficiently quickly to let the system operators take the required steps if a suspect is found. We therefore aimed to minimize the online overhead of the system (after all, any improvement by a constant factor can be of great importance in practice).

Oblivious transfer. As it turns out, it is possible to move most of the computation and communication overhead of the OT to an preprocessing stage. A 1-out-of-$d_{\max}+1$ OT protocol can be implemented using $\log(d_{\max}+1)$ invocations of 1-out-of-2 OT [27] (namely, 8 invocations of 1-out-of-2 OT since $d_{\max} = 180$). Moreover, as is described in Section IV-A the public-key operations of the OT can be precomputed before the inputs are known. The online phase requires sending symmetric encryptions of the sender's inputs. In the $F_{\text{threshold}}$ protocol of Figure 2 the sender's inputs X_i are single bits. Therefore the online step of the OT consists of sending a $(\log d_{\max})$-bit long offset from the receiver to the sender, and a string of $d_{\max} = 180$ bits from the sender to the receiver.

Homomorphic operations. The main online computational overhead is thus that of computing the homomorphic operations which are required for computing the Hamming distance. In our experiments, we found that homomorphic encryption takes about 38 msec, addition takes less than 1 msec, while subtraction takes about 62 msec. Computing a subtraction of homomorphic encryptions is costly, as it requires computing the additive inverse. (These results were using a certain implementation in Java, as is detailed in Section VI-B, but most implementations will have the same ratio between the performance of the different operations). It is therefore beneficial to minimize the number of subtraction operations that must be computed.

Consider the computation of the Hamming distance of w and w^i. For each bit location j, the server must add $E(w_j)$ to the sum, if its own bit w^i_j is 0, or add to the sum the value $(1 - E(w_j))$ if its own bit w^i_j is 1. The computation is expected to perform ℓ additions and $\ell/2$ subtraction operations. The server can improve the overhead by separately computing the values $E(s_0) = \sum_{w^i_j=0} E(w_j)$, and $E(s_1) = \sum_{w^i_j=1} E(w_j)$. Let also use n_1 to denote the number of bits in w^i which are equal to 1. Then the encryption of the Hamming distance can be computed as $E(s_0) - E(s_1) + E(n_1)$. The computation according to this method requires ℓ homomorphic additions and only a single subtraction per database item.

Reducing online communication. The basic protocol requires the client to send a homomorphic encryption of each of the $\ell = 900$ bits of the face representation that is has. Recall that the length of a Paillier encryption is at least 2048 bits, and is more than 3000 bits is a reasonable security is required. These encryptions must be sent after the client receives its input. It is possible, however, to send these encryptions in a preprocessing stage, thus reducing the online communication overhead. In order to do that, the client chooses a random ℓ bit binary string $v_0, \ldots, v_{\ell-1}$ and sends the encryptions of these bites to the server in the preprocessing stage. Later, when the client receives its input, it sends the ℓ bit string $w_0 \oplus v_0, \ldots, w_{\ell-1} \oplus v_{\ell-1}$ to

the server. The server modifies its operation in Step 2 of the protocol according to the correction bit $v_j \oplus w_j$ that it received. Namely, instead of using its bit w_j^i to decide what operation to apply to the homomorphic encryption $E(w_j)$, it uses the bit $w_j^i \oplus (w_j \oplus v_j)$ to decide which operation to apply to the encryption $E(v_j)$. It is not hard to see that the optimization listed above can also be modified to use $E(v_j)$ and $(w_j \oplus v_j)$ instead of $E(w_j)$.

Further reducing the number of homomorphic operations. Let us first assume that the optimization of the online communication, described above, is not used, and the server receives the encryptions $E(w_0), \ldots, E(w_\ell - 1)$. Consider the ith item in the server's database and examine pairs of the bits pf its representation. If $\langle w_o^i, w_1^i \rangle = \langle 0, 0 \rangle$, then the value $E(w_0 + w_1)$ must be added by the server to the Hamming distance. If $\langle w_o^i, w_1^i \rangle = \langle 0, 1 \rangle$, then the value $E(w_0 + (1 - w_1))$ must be added to the Hamming distance, etc. Let the server first compute the four possible combinations $E(w_0 + w_1), E(w_0 + 1 - w_1), E(1 - w_0 + w_1)$ and $E(2 - w_0 - w_1)$. Then for each of the N items in the server's database it now only needs to perform a single addition operation, instead of two, in order to add the right value to the sum. It is not hard to see that the same method can be also be applied when the optimization of online communication is used. In that case the server receives in the preprocessing phase the values $E(v_0), \ldots, E(v_{\ell-1})$, and precomputes the four possible combinations of each pair of successive bits. Then, after receiving the correction string, it decides which of the four options it must add to the sum.

In the preprocessing phase this method computes 7 homomorphic additions and 2 subtractions for each of the $\ell/2$ pairs of input bits. The gain is in the online phase, where for each of the N database items the server needs to compute only $\ell/2$ homomorphic additions (instead of ℓ additions).

While we only implemented this optimization with pairs of input bits, it can be extended to handle sets of three or more consecutive bits. By processing sets of three bits, for example, the overhead of the preprocessing phase increases to $16/3 \cdot \ell$ additions and ℓ subtractions. The online overhead is reduced to $\ell/3$ addition operations.[3]

Parallelizing the computation. After the server receives the homomorphic encryptions from the client, it runs, for each item in its database, a computation which is independent of the other elements in the database. It is therefore easy to parallelize the operation of the server such that each processor or core handles a different set of faces in the server's database.

Overall online overhead. The overall online overhead is minimal. Let us summarize the operations that must be performed after the client receives its input. In Step 1 of the protocol, the client sends ℓ bits to the server. Then Steps 2-5 are repeated for every item in the server's database (and can be parallelized). In Step 2, the server performs $\ell/2$ homomorphic additions. In Step 3 it sends to the client a single homomorphic encryption. In Step 4 the client computes a single decryption. In Step 5 the client sends a $\log d_{\max} = 8$ bit offset to the server, and the server sends back ℓ bits to the client.

[3]We also note that the algorithm essentially computes different sums of the same set of homomorphic encryptions, and therefore the "Four Russians" method [29] can be used to further minimize the total number of addition operations. We have not implemented this improvement, though.

E. A threshold protocol where the server receives the output

A simple way to convert the basic $F_{\text{threshold}}$ protocol to one where the server, rather than the client, receives an output, is for the server to encrypt the values that the client learns in the final OT. Namely, with probability $1/2$ the server keeps its inputs to the OT as before, and with probability $1/2$ it decides to flip all inputs (from 0 to 1, and vice versa). As a result the client learns nothing from its output in the OT. The protocol then requires the client to send back its output to the server, which can decrypt it and learn the correct output value.

The modified protocol described above requires adding an additional communication step to the protocol. It is possible to achieve the same goal without adding this step, as we next describe.

The protocol where the client receives the result, i.e. the protocol of Figure 2, uses an OT where the client is the receiver, to check if the value that the server receives in the Step 4, $d_H + r_i$, is greater than the threshold t_i. To enable the *server* to receive the result, both parties must invoke an OT protocol with opposite roles, where the server is the receiver. This protocol is described in Figure 3. The only part in which it differs from the previous protocol is in Step 5:

Before the OT protocol begins, the client knows $d_H + r_i$, which is computed as in the previous protocol. The server knows r_i and t_i. It is also known that $d_H, t_i \leq d_{\max}$, and therefore $d_H - t_i$ is in the range $[-d_{\max}, d_{\max}]$. Let us assume for a minute that $r_i = 0$. In that case the parties can run an OT where the server is the receiver and its input is t_i, and the client has $2d_{\max} + 1$ inputs, $X_{-d_{\max}}, \ldots, X_{d_{\max}}$ such that $X_i = 0$ if $i \leq d_H$ and is equal to 1 otherwise. The server learns 1 if, and only if, $d_H - t_i < 0$. Now, in actuality the value r_i is random and is unknown to the client who knows only $d_H + r_i$. The parties can now check if $(d_H + r_i) - (t_i - + r_i) < 0$. The check can be done by reducing both values modulo $2d_{\max} + 1$, since it is known that $-d_{\max} \leq (d_H + r_i) - (t_- + r_i) \leq d_{\max}$. The server therefore uses the value $t_i + r_i \mod (2d_{\max} + 1)$ as its input in the OT. The client prepares $2d_{\max} + 1$ items, where $X_{(d_H+R-d_{\max}) \ mod \ 2d_{\max}+1}, \ldots, X_{d_H+R \ mod \ 2d_{\max}+1} = 1$ and $X_{d_H+R+1 \ mod \ 2d_{\max}+1}, \ldots, X_{d_H+R+d_{\max} \ mod \ 2d_{\max}+1} = 0$.

As for efficiency, note that the server's input $t_i + r_i$, i.e. the sum of the threshold and a random value, is known to the server even in the preprocessing phase. Thus, in the

preprocessing step of the OT the server can learn the exact keys that it needs in the online phase. This saves one round in the online stage, compared to the protocol where the client learns the output.

$F_{\mathrm{threshold}}(w, (w^1, \ldots, w^N))$

INPUT: The input of the protocol is as in the protocol of Figure 2.

OUTPUT: The server learns if $d_H(w, w^i) > t_i$. The client learns nothing.

1-4. Steps 1-4 are run as before. In Step 4 the client receives $E_{pk}(d_H + r_i)$ and decrypts the result.

5. Next, the parties compute the appropriate output value, by invoking a $\mathrm{OT}_1^{2d_{\max}+1}$ protocol where the server is the receiver and the client is the sender:
   - Let $LD = 2d_{\max}$. The input of the server is $(t_i + r_i) \bmod (LD + 1)$.
   - The client has inputs $X_0, \ldots, X_{LD}$, where

$$X_j = \begin{cases} 1 & \text{if } d_H + r_i - d_{\max} \bmod (LD+1) \leq j \\ & \text{and } j \leq d_H + r_i \bmod (LD+1) \\ 0 & \text{otherwise} \end{cases}$$

Figure 3: The $F_{\mathrm{threshold}}$ protocol where the server learns the output.

The security of the protocol is proved similarly to the security of the protocol of Figure 2. The details are given in the full version of the paper.

F. Computing $F_{\min+\mathrm{t}}$

The $F_{\min+\mathrm{t}}$ functionality finds the item in the server's database whose distance from the client's input is minimal, as long as this distance is below the threshold. This functionality can be implemented in a rather straightforward manner using a generic method for secure computation, such as Yao's protocol. We describe in the Appendix a specific protocol for this task, which is more efficient and more simple to implement, as it does not use a circuit representation of the function, and is based on oblivious transfer as the previous protocols that we have described.

G. Security against malicious adversaries

The protocols that we described are secure against semi-honest adversaries. A malicious adversary can deviate from the prescribed protocols, and can definitely change the function that is computed and learn information about the other party's input.

There are known generic transformations of any semi-honest protocol to a protocol secure against malicious adversaries [20], but these are not efficient in practice. More efficient protocols with this level of security were presented for specific applications. There are, however, several obstacles that need to be overcome in the case of the applications that we discuss: (1) The protocol must ensure that the inputs w, w' of the parties are in a format of a face representation, rather than being arbitrary binary vectors (namely, that the number of "1" bits and their locations are as defined in Section III-C). (2) The client must send in Step 1 encryptions of bits rather than of arbitrary values. (3) The server must send back an encryption of the Hamming distance (plus a random value), rather than of another function of the messages it receives. (4) The inputs to the OT must be according to the protocol.

Some of these issues can be solved rather efficiently (see for example [28] for a method for verifying the inputs to the OT stage, by replacing the OT with oblivious polynomial evaluation). However, ensuring in an efficient way that the entire protocol computes the desired functionality seems challenging. Another, possibly simpler, option is to design protocols which provide security only against covert adversaries, roughly meaning that adversaries which do not behave according to the protocol are caught with some non-negligible probability. This level of security might be sufficient to deter most attacks in our setting. See [30] for a discussion of security against covert adversaries.

V. An Example of a Real-Time Security System

The proposed algorithms can be combined in different configurations depending on the application. In this section we describe an example of a security system for recognition of suspected individuals using a remote camera module installed in a public place.

As was described in Section II, the system is composed of a server and a client. During the preprocessing phase, the server generates face representations of suspects, as described in Section III-C, and converts them to binary vectors as was shown in Section IV-B. The individual thresholds are computed as is described in Section III-D. The binary representations and individual thresholds are stored in the server. The cryptographic protocol is initialized by the client, which sends encryptions of the bits of a random binary vector, and performs the preprocessing step of the OTs. The server computes the sum of consecutive pairs of bits, as is required by the optimized protocol.

The client obtains an image from a real-time face detection algorithm (e.g., [1], [2]), builds a binary face representation (Sections III-C, IV-B) and sends homomorphic encryptions of its bits to the server. For each subject i in the database, the parties executes the $F_{\mathrm{threshold}}$ cryptographic algorithm. The output, which can be learnt by one party or both parties, according to the variant of the protocol that is used, is a set of binary bits, one for every database entry. If all bits are equal to 0 (which is the most likely case, since most of the people should not match any suspect), the client continues with the next image. If one or more of the bits are 1 then the operator of the system is notified.

VI. Experiments

As was detailed in Section II, the proposed system can be separated into a face recognition part and a secure computation part. The face recognition part generates representations of the faces in the server's database and of the face acquired by the client, and is run independently by each party. In the secure computation part the two parties jointly execute a secure protocol which checks if there is a match between the acquired face and the server's database.

In light of this architecture we separated our experiments into two parts. We first examined the face recognition algorithm used in SCiFI, with an emphasis on examining its accuracy. (Our current implementation of the algorithm in Matlab and takes about 10 seconds to process a face; it is clear that an implementation in C will be faster, probably by a factor of 4-5.) Then we examined the performance, i.e. latency, of the secure computation protocol.

A. Face Recognition Experiments

The face recognition experiments consist of two parts. The first part (presented in Section VI-A1) includes tests that simulate a real security systems with server and client. The second part (presented in Section VI-A2) includes experiments conducted according to the protocols used in the face recognition community. These tests are performed on benchmark sets which allows direct comparison with the state of the art in face recognition.

1) Real security system experiment: Our tests simulate a real security systems that stores a list of subjects in the server and decides whether an input image obtained by a client matches one of the faces on the list. To determine a threshold on the Hamming distance for each person we constructed an ensemble of people which included other individuals from the server's list and images of unrelated people which represent typical inputs to the system. An individual threshold for the ith subject is set based on the smallest Hamming distance between him and the rest of the people in the ensemble.

We constructed the public set Y of faces from which the part vocabularies will be taken, by rendering images with frontal illumination using a subset of 34 3D models of faces supplied by USF.[4]

We tested the proposed face representation on two benchmark databases, checking its robustness to various factors that influence the intra-user variation, and comparing it to the Eigenfaces method, which was the only other face recognition algorithm for which a secure protocol is known. (For the Eigenfaces method we pre-aligned all images and normalized them for brightness.)

[4]USF HumanID 3D Face Database, Courtesy of Prof. Sudeep Sarkar, University of South Florida, Tampa, FL.

Large illumination Variation: We tested the robustness of the representation to large illumination changes on the frontal pose subset of CMU-PIE database [5] that contains images of white, black, and asian faces of males and females, in total 68 subjects under 43 illuminations (see Figure 1, top row for a few examples from this data set).

The server's list included 12 persons under frontal illumination. The client's set (a stream of images) contained 2912 images (which is equivalent to an hour of video, with a processing rate of 1 image per sec.) of 68 subjects, from which 504 belonged to the subjects from the server's list. All of the client's images contained faces in a frontal pose under *large* illumination changes, which make the face identification task much harder. About third of the subjects on the server's list wear glasses, but then remove them in half of the client's images. The results are shown (Figure 4) in a form of a recognition rate plotted as a function of false positive rate. For example, our method spots suspects in about 93% of images with 15% false alarms. This is dramatically better than the Eigenface performance, which is less than 50% recognition at this false alarm rate. This result was, however, expected, since Eigenface cannot generalize well to unseen conditions, such as changes in lighting and local occlusions such as glasses.

Near-frontal changes in pose, mild facial expressions and mild illumination changes: Although the current implementation of the system does not allow large variations in pose or facial expression, it can still handle some variation in these factors. To test our representation in a more realistic setting, namely, near-frontal variation in pose and mild changes in facial expressions and illumination, we ran our system on the gallery and the **fc** probe set (i.e. thet set of test images) of the FERET [4] database. The probe set includes 194 images taken with a different camera and under different illumination than the images in the gallery set which includes 1196 subjects. The bottom row of Figure 1 shows some of the variations present between the gallery set and the probe set. We took 100 subjects from the **fc** probe set for the server's list, and used all 1196 gallery images as a client's set. Figure 5 shows the results of our method compared to the Eigenface, which again shows the benefit of our approach. For example, with a false alarm rate of 5%, our algorithm has 97% recognition success while Eigenface succeeds with probability of about 87%.

Robustness to illumination changes and partial occlusions: One of the advantages of using a part-based representation is its robustness to partial occlusions. We tested the effects of partial occlusions in eye, nose and mouth areas separately. To simulate occlusions we used a square area with size of 20% of the image width filled with random noise (Figure 8 shows some examples of partial occlusions used in the test). Occlusion was applied to images obtained from the client and not to images in the server's list. Table II summarizes the recognition results, tested on

images of 68 people from the CMU-PIE database under 10 illuminations and partial occlusions. The right column of the table describes the number of patches with at least half of their area occluded. For example, hiding the nose hides 3 of the 30 patches, and yet the recognition rate shows almost no degradation (92.8% compared to 93% for a 15% false positive rate).

2) Identification Performance: In order to test the recognition power of the proposed binary representation we used it in an *identification task in a closed universe*, meaning that all probes are in the gallery (see [4])[5]. The tests were conducted on benchmark data sets to allow comparison with the existing methods.

Following the FERET [4] evaluation of identification in a closed universe, we report the performance statistics by a graph of cumulative match. The horizontal axis of the graph is a rank (rank=c corresponds to the c persons who have the smallest distance to the probe among all persons in the gallery), and the vertical axis is the probability of identification. These graphs are obtained as follows: for each probe the galleries are ordered by their distance to the probe (from minimum to maximum). The rank corresponds to the threshold on the number of galleries considered. The recognition is defined to be successful for rank **r**, if the correct person is found within the **r** galleried in the ordered list corresponding to the probe. Our plots show the percentage of probes that are of a particular rank or less. For example, in Figure 6, 80% of the probes have the correct identity as their closest match, and 95% of the probes have the correct identity among the 6 closest matches.

In the large gallery test, FERET reports the top 50 matches for each probe [4]. In our experiments we set the number of top matches relative to the number of subjects in the gallery.

CMU-PIE database: In the CMU-PIE frontal pose subset [5] we used frontal illumination with ambient light

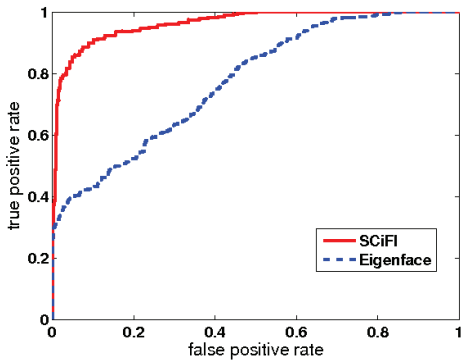

Figure 4: The illumination robustness test on CMU-PIE

[5]The gallery contains images of people stored in the database, probes are images that must be identified. Each probe image is compared against all images in the gallery for identification

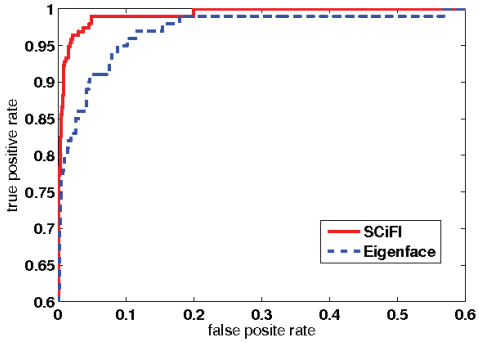

Figure 5: Robustness test to a combination of several factors. The test has been performed on the gallery and **fc** probe set of the FERET database.

as a gallery image and other 42 illuminations as probes, in total 2856 probe images. Note that our setting tests not only illumination variation, but also the effects of glasses, since 28 out of 68 subjects wear glasses in the gallery image and then remove them in the half of the probe images (without ambient light). The results are shown in Figure 6. A separate test in which all galleries and probes include ambient light shows 100% recognition. A test in which all galleries and probes have no ambient light and no glasses shows 99%. The recognition results in this test are comparable to the state of the art [31], [32].

FERET fc: We ran our tests on the gallery and **fc** probe sets of FERET [4]. The gallery of FERET contains 1196 subjects and the **fc** probe set includes images of 194 subjects taken with a different camera and under different illumination. Besides the illumination, some variation in facial expression (smiling and blinking) and near-frontal pose variation are present between the gallery and the probe sets. The results are reported in Figure 7. The recognition results on FERET are lower than CMU-PIE, but are comparable with previously reported results [4], [33]. The gallery in this test is about 17 times larger than in CMU-PIE, which explains the degradation in performance.

B. Secure Computation Experiments

A prototype of SCiFI was implemented in Java using Sun's JDK 1.6.0_12. Homomorphic encryption was implemented using Paillier's algorithm, with a modulus N of length 1024 bits.[6] The implementation of OT_1^2 was based on the Bellare-Miacli scheme and El Gamal encryption in a subgroup of order q of Z_p^*, where $|p| = 1024$ and $|q| = 160$. Symmetric encryption was done using AES.

The results, which are detailed below, are extremely fast, taking about 0.3 second to compare the vector representing the client's input with the vector representing an image in

[6]The implementation was based on the Java implementation in http://www.bricks.dk/~jurik/research.html.

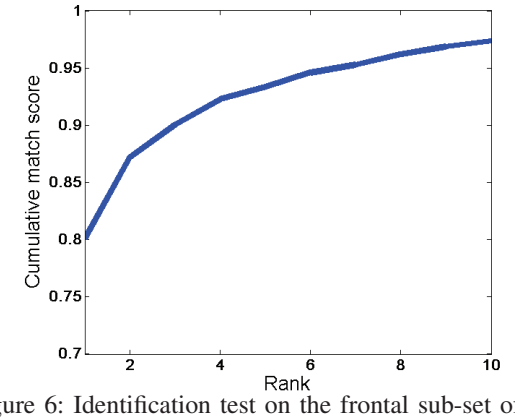

Figure 6: Identification test on the frontal sub-set of CMU-PIE.

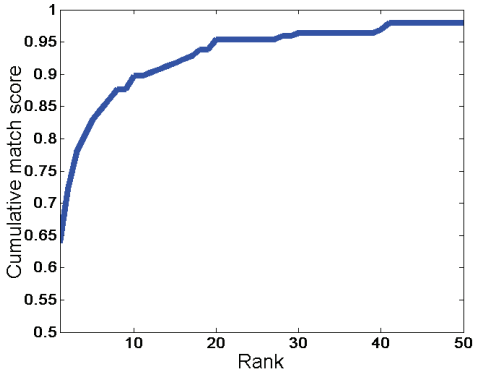

Figure 7: Identification test on the **fc** probe set of FERET.

the server's database. It is also clear that an implementation in the C language, with a faster cryptographic library, would have a considerably better performance.

The experiments were performed on two Linux machines. The machines were located in two different buildings and communicated using sockets through TCP/IP over the local network. We turned off Nagle algorithm in order to prevent the "ACK delay" (turning off Nagle algorithm indeed greatly improved the performance of the communication layer). The server machine was an 8 core machine of 2.6 GHz AMD Opteron processors and 1GB RAM. The client machine had a 2.8 GHz dual core Pentium D processor and 2GB RAM.

The protocol used in the SCiFI implementation is the $F_{\text{threshold}}$ protocol where the server learns the output. As we described earlier in the text, the computation is composed of two phases, a preprocessing phase which is run before the client's input is known, and an online execution phase. Our goal was to optimize the performance of online phase. Next, we detail the steps of each party.

Client preprocessing phase. In the preprocessing phase the client (1) chooses a random vector of 900 bits and sends the encryption of these bits to the server, and (2) runs the preprocessing phase of the OT, as is described in Section IV-A.

Server preprocessing phase. For every pair of consecutive bits sent by the client, i.e., $E(v_{2j}), E(v_{2j+1})$ the server precomputes the four possible values that can be added to the Hamming distance, namely $E(v_{2j}+v_{2j+1}), E(v_{2j}+1-v_{2j+1}), E(1-v_{2j}+v_{2j+1})$ and $E(2-v_{2j}-v_{2j+1})$. This computation is performed only once and used for all images in the database. The server also runs the preprocessing phase of the OT.

Client execution phase. After the client captures an image and computes its representative vector, it sends a correction vector which is the exclusive-or between this vector and the random vector chosen by the client in the preprocessing phase. This is a string of 900 bits. Afterwards the client decrypts the result that the server sends it, and both parties invoke the OT protocol where the client is the sender.

Server execution phase. After receiving the correction vector, the server computes the encryption of the Hamming distance between every image in the database and the client image. (The server can parallize this step.) It then sends to the client an encryption of the sum of the Hamming distance and a random value. Afterwards it runs the OT protocol with the client and learns the final result.

Results: We ran multiple experiments where the server stored a list of 100 face representations. Following are the average timing measurements results of the experiments.

Preprocessing. Offline preprocessing at the client took about 213 sec. Of this time, 38 seconds were spent on preparing 900 homomorphic encryptions, 71 seconds were used to send these encryption to the server, and 92 seconds were spent on running the preprocessing phase of the 1-out-of-180 OT (which is used since we set $d_{\max} = 180$). As can be seen, almost half of the preprocessing time is spent on preparing and sending the Homomorphic encryptions. As for the server, the offline preprocessing time of the server includes receiving the encryptions (this time was already counted by us at the client side); summing every possible combination of each pair of consecutive bits, a step which takes about 57 sec; and running the preprocessing of the OT (this step was also already counted by us at the client side).

Online performance. The previous preprocessing steps optimize the performance tremendously, and the resulting online execution time is minor for each image. The online execution time of the server for an image after receiving the correction binary vector is only about 0.31 second. This time is divided to (1) Computing the Hamming distance for the image, adding to it a random value and sending the result to the client; these steps require .28 sec. (2) Running the online step of the OT protocol, where the server learns the result, this step takes about .012 sec.

The run time is obviously linear in the size of the server's database. The total online time for comparing the client's input to 100 database images in only about $100 \cdot 0.31 = 31$ seconds. The bulk of the server's computation can be fully

Occluded Part	Recognition rate for 15% false positives	Num. of occluded parts
Left eye	89%	10/30
Mouth	91.5%	4/30
Nose	92.8%	3/30

Table II: Partial occlusion results on a subset of CMU-PIE.

parallelized, and therefore using, e.g., six processors, reduces the recognition time in this example to about 5 seconds.

Figure 8: Examples of tested occlusions.

REFERENCES

[1] M. Osadchy, Y. LeCun, and M. Miller, "Synergistic face detection and pose estimation with energy-based models," *Journal of Machine Learning Research*, vol. 8, pp. 1197–1215, May 2007.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2001, pp. 511–518.

[3] C. Chen, R. Veldhuis, T. Kevenaar, and A. Akkermans, "Biometric binary string generation with detection rate optimized bit allocation," in *CVPR Workshop on Biometrics*, 2008, pp. 1–7.

[4] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," *PAMI*, vol. 22, no. 10, pp. 1090–1104, 2000.

[5] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *PAMI*, vol. 25, pp. 1615–1618, 2003.

[6] A. Juels and M. Sudan, "A fuzzy vault scheme," in *Symposium on Information Theory*, 2002.

[7] P. Tuyls and J. Goseling, "Capacity and examples of template-protecting biometric authentication systems," in *ECCV Workshop BioAW*, 2004.

[8] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, 2008.

[9] Y. Adini, Y. Moses, and S. Ullman, "Face recognition: the problem of compensating for changes in illumination direction," *PAMI*, vol. 19, no. 7, pp. 721–732, 1997.

[10] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[11] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle, "Generating cancelable fingerprint templates," *PAMI*, vol. 29, no. 4, pp. 561–572, 2007.

[12] T. Boult, "Robust distance measures for face-recognition supporting revocable biometric tokens," in *IEEE, 7th Intl. Conf. on Automatic Face and Gesture Recognition*, 2006, pp. 560–566.

[13] S. Avidan and M. Butman, "Blind vision," in *ECCV (3)*. Springer, 2006, pp. 1–13.

[14] A. Senior, S. Pankanti, A. Hampapur, L. Brown, Y.-L. Tian, A. Ekin, J. Connell, C. F. Shu, and M. Lu, "Enabling video privacy through computer vision," *IEEE Security and Privacy*, vol. 3, no. 3, pp. 50–57, 2005.

[15] F. Dufaux and T. Ebrahimi, "Scrambling for Video Surveillance with Privacy," in *IEEE Workshop on Privacy Research in Vision*. IEEE, 2006.

[16] E. M. Newton, L. Sweeney, and B. Malin, "Preserving privacy by de-identifying face images," *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, no. 2, pp. 232–243, 2005.

[17] T. Boult, "Pico: Privacy through invertible cryptographic obscuration," *Computer Vision for Interactive and Intelligent Environment, 2005*, pp. 27–38, 2005.

[18] J. Schiff, M. Meingast, D. Mulligan, S. Sastry, and K. Goldberg, "Respectful cameras: Detecting visual markers in real-time to address privacy concerns," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007, pp. 971–978.

[19] A. Yao, "How to generate and exchange secrets," in *FOCS*, 1986, pp. 162–167.

[20] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[21] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proc, of the 9th International Symposium on Privacy Enhancing Technologies (PET)*. Springer, 2009, p. 253.

[22] A. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient Privacy-Preserving Face Recognition," in *12th International Conference on Information Security and Cryptology (ICISC09), LNCS. Springer*, 2009.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[24] C. T. Yuen, M. Rizon, W. S. San, and M. Sugisaka, "Automatic detection of face and facial features," in *ISPRA'08*, 2008, pp. 230–234.

[25] N. Gourier, D. Hall, and J. L. Crowley, "Facial features detection robust to pose, illumination and identity," in *Int'l Conf. on Systems Man and Cybernetics*, 2004.

[26] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999, pp. 223–238.

[27] M. Naor and B. Pinkas, "Computationally secure oblivious transfer," *J. Cryptology*, vol. 18, no. 1, pp. 1–35, 2005.

[28] A. Jarrous and B. Pinkas, "Secure hamming distance based computation and its applications," in *Applied Cryptography and Network Security conf. (ACNS)*, 2009, pp. 107–124.

[29] V. Arlazarov, E. Dinic, M. Kronrod, and I. Faradzev, "On economical construction of the transitive closure of an oriented graph," in *Soviet Math. Dokl*, vol. 11, no. 1209-1210.

[30] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," in *TCC*, ser. Lecture Notes in Computer Science, S. P. Vadhan, Ed., vol. 4392. Springer, 2007, pp. 137–156.

[31] S. Romdhani, V. Blanz, and T. Vetter, "Face identification by fitting a 3d morphable model using linear shape and texture error functions," in *ECCV*, 2002, pp. 3–19.

[32] Y. Wang, Z. Liu, G. Hua, Z. Wen, Z. Zhang, and D. Samaras, "Face re-lighting from a single image under harsh lighting conditions," in *CVPR*, vol. 1, 2007, pp. 1–8.

[33] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *PAMI*, vol. 28, no. 12, pp. 2037–2041, 2006.

APPENDIX

The min *functionality:* The basic building block used for computing the $F_{\min+t}$ functionality is a protocol for outputting the minimum of two numbers (which will be used for comparing Hamming distances). More specifically, in this protocol, denoted min, the client has two inputs y_0, y_1, where $y_i = d_i + r_i$, $0 \leq d_0, d_1 \leq \ell$, and addition is performed in the field or ring over which the homomorphic encryption is defined. The server has inputs r_0, r_1. The server's output is a random number r'. The client's output is $b \in \{0,1\}$, which is the index of the smaller element among d_0, d_1, and the value $\min(d_0, d_1)+r'$, where the min operation is computed over the integers. (This basic protocol will later be used for computing the minimum among N numbers, by running a tournament between these N values.)

The basic observation behind the protocol is that the value $d_0 - d_1$ is in the range $[-d_{\max}, d_{\max}]$ and is negative iff $d_0 < d_1$. Therefore a $\text{OT}_1^{2d_{\max}+1}$ protocol, adapted to the use of the keys r_0, r_1, can compute the functionality.

The min protocol is described in Figure 9. Correctness follows from setting the server's inputs in the OT protocol such that the client retrieves $r' - r_0$ if $d_0 < d_1$, and $r' - r_1$ otherwise. Security is implied by the security of the OT protocol. The overhead is that of running a single $\text{OT}_1^{2d_{\max}+1}$ protocol, namely of executing $\log(2d_{\max}+1)$ invocations of OT_1^2, and encrypting and sending $2d_{\max}+1$ values.

The $F_{\min+t}$ functionality.: Running the basic $F_{\text{threshold}}$ protocol until Step 4 results in the client learning $D_H + r_i$ and the server learning r_i. This is exactly the input to the min protocol, and therefore the parties can run a tournament

$\underline{\mathsf{min}(y_0, y_1; r_0, r_1)}$

The protocol uses a ring or a field $\mathcal{F}$ over which a homomorphic encryption scheme is defined. The inputs d_0, d_1 are integers in the range $[0, d_{\max}]$. We note that the value of $d_{\max}$ is negligible compared to $|\mathcal{F}|$.

INPUT: The client's input is y_0, y_1, where $y_0 = d_0 + r_0$, $y_1 = r_1 + d_1$ and addition is done in $\mathcal{F}$. The server's input is r_0, r_1.

OUTPUT: The client receives $b, d_b + r'$, where $b \in \{0,1\}$ is such that $d_b < d_{1-b}$, where comparison is done over the integers. The server obtains r', which is chosen randomly in $\mathcal{F}$.

1. The client computes (over $\mathcal{F}$) the value $\alpha = y_0 - y_1 = (d_0 - d_1) + (r_0 - r_1)$. The client then computes (over the integers) $\beta = \alpha \bmod 2d_{\max} + 1$.
   Note that if $d_{\max} < r_0 - r_1 < |\mathcal{F}| - d_{\max} - 1$, an event which happens with overwhelming probability since $d_{\max} \ll \mathcal{F}$, then $0 \leq (d_0 - d_1) + (r_0 - r_1) \leq |\mathcal{F}| - 1$ and therefore β is equal to the result of computing $y_0 - y_1 = (d_0 - d_1) + (r_0 - r_1) \bmod 2d_{\max} + 1$ over the integers.
2. The parties run a $\text{OT}_1^{2d_{\max}+1}$ protocol where the client is the receiver and the server is the sender. The input of the client is β.
   The server chooses a random $r' \in \mathcal{F}$, and sets its inputs in the OT such that the client learns the value $0|(r' - r_0)$ if $y_0 < y_1$, and $1|(r' - r_1)$ otherwise. This is done by setting the inputs $X_0, \ldots, X_{2d_{\max}}$ such that $X_i = 0|(r' - r_0)$ if $i - (r_0 - r1) \bmod 2d_{\max} + 1$ is in the range $[d_{\max}+1, 2d_{\max}]$ and $X_i = 1|(r' - r_1)$ otherwise.
3. Denote the value learned by the client in the OT as $b|\gamma$. Then the client sets its output to be $y_b + \gamma = d_b + r_b + (r_b - r') = d_b + r'$. The server's output is r'.

Figure 9: The min protocol.

in which the minimal D_H value is computed: The first stage of the tournament compares adjacent values received from the $F_{\text{threshold}}$ protocol. The next stage compares the results of the previous stage. After $\log N$ stages the tournament finds a winner which is the minimal value. Note that the min protocol reveals which of the two items it compares is smaller, whereas the $F_{\min+t}$ functionality must not reveal any such intermediate result. Therefore the server must permute the order of its inputs each time $F_{\min+t}$ is run.

The protocol described above finds the minimal Hamming distance. More accurately, the server and client learn outputs out_s, out_c, respectively, such that the sum of these values is the minimal Hamming distance. However, the $F_{\min+t}$ functionality outputs the minimal Hamming distance only if it is smaller than the threshold. Therefore an additional step of the protocol must check if $out_s + out_c$ is smaller than the threshold, and outputs $out_c + out_s$ and the corresponding index if this is indeed the case. This step can be implemented using a solution to the millionaires problem [19]. (Some additional care must be taken if a different threshold is used for every face in the server's list. The details will be given in the full version of the paper.)