Privacy-Preserving Fingercode Authentication[*]

Mauro Barni
Dipartimento di Ingegneria dell'Informazione
Università di Siena
barni@dii.unisi.it

Tiziano Bianchi
Dipartimento di Elettronica e Telecomunicazioni
Università di Firenze
tiziano.bianchi@unifi.it

Dario Catalano
Dipartimento di Matematica e Informatica
Università di Catania
catalano@dmi.unict.it

Mario Di Raimondo
Dipartimento di Matematica e Informatica
Università di Catania
diraimondo@dmi.unict.it

Ruggero Donida Labati
Dipartimento di Tecnologie dell'Informazione
Università di Milano
ruggero.donida@unimi.it

Pierluigi Failla
Dipartimento di Ingegneria dell'Informazione
Università di Siena
pierluigi.failla@gmail.com

ABSTRACT

We present a privacy preserving protocol for fingerprint-based authentication. We consider a scenario where a client equipped with a fingerprint reader is interested into learning if the acquired fingerprint belongs to the database of authorized entities managed by a server. For security, it is required that the client does not learn anything on the database and the server should not get any information about the requested biometry and the outcome of the matching process. The proposed protocol follows a multi-party computation approach and makes extensive use of homomorphic encryption as underlying cryptographic primitive. To keep the protocol complexity as low as possible, a particular representation of fingerprint images, named Fingercode, is adopted. Although the previous works on privacy-preserving biometric identification focus on selecting the best matching identity in the database, our main solution is a generic identification protocol and it allows to select and report *all* the enrolled identities whose distance to the user's fingercode is under a given threshold. Variants for simple authentication purposes are provided. Our protocols gain a notable bandwidth saving (about $25 - 39\%$) if compared with the best previous work [1] and its computational complexity is still low and suitable for practical applications. Moreover, even if such protocols are presented in the context of a fingerprint-based system, they can be generalized to any biometric system that shares the same matching methodology, namely distance computation and thresholding.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public key cryptosystems; H.2.0 [**Database Management**]: General—*Security, integrity, and protection*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Algorithms, Security

1. INTRODUCTION

Biometric-based identification is receiving more and more attention as an extremely reliable way of identifying people. Such an interest is mainly due to the high reliability ensured by biometric identification, its universality (most biometric traits are such that every person owns them), uniqueness (it is very rare, if not impossible, that two persons own the same biometric trait), and permanence (good biometric traits are virtually time invariant) [2]. As a matter of fact, biometric templates are uniquely associated with each user and thus represent the strongest form of personally identifiable information. For the same reason, however, the possibility that a biometric template could be stolen or exchanged raises concerns on its uses and abuses. In this sense, a common concern is the possibility that a government agency or a company which maintains personal data might monitor and track the actions and the behavior of each individual. Another basic concern is about the anonymity loss implied by the massive use of biometrics for identification or authentication purposes. Hence the widespread use of biometric systems asks for a careful policy specifying to which party biometric data can be revealed. It is even more important to note that the biometric matching process may involve a central server or be adopted in partially untrusted environments. It is therefore clear that developing techniques to process biometric data in a privacy-preserving way would have a great impact on the use of biometric-based authentication systems in every-day life.

In this framework the use of multiparty computation and secure function evaluation techniques has been lately advanced as a viable way to process encrypted biometric data as they preserve the privacy of the parties involved in the computation. For example, the works in [3, 1] propose two privacy-preserving face recognition protocols that may be seen as the basis for a biometric-based identification service where faces are used as the underlying biometric trait. Although face images are widely used in many applications, they are known to be quite weak biometric traits. Therefore

[*]Sponsored by MIUR under project "*Priv-Ware*" (contract n. 2007JXH7ET)

MM&Sec'10, September 9–10, 2010, Roma, Italy.
Copyright 2010 ACM 978-1-4503-0286-9/10/09 ...$10.00.

more reliable traits like fingerprint, iris code or DNA are likely to be used in applications that need higher reliability.

In this paper, we propose a privacy-preserving system for fingerprint-based authentication. We consider the following typical scenario: a client $\mathcal{C}$ equipped with a specific-biometric device (fingerprint reader) is interested into learning if the just acquired fingerprint belongs to the database of authorized entities that is managed by a server $\mathcal{S}$. For privacy, we require that the client should trust the server to correctly perform the matching algorithm for the fingerprint recognition and also it should not learn anything about the database managed by the server, beyond the outcome of the matching process. On the other hand the server should not get any information about the requested biometry and the outcome of the matching process.

As to the template used to represent the user's fingerprint, we adopted the Fingercode representation introduced in [4]. While other kinds of representations, noticeably those based on minutiae [5], are more common in practical applications, we chose the fingercode representation as it is more suitable for being implemented in a multi-party computation setting. In fact, after the feature extraction step, its matching phase needs only distancepurposes computation and thresholding. Furthermore it keeps good accuracy when quantized to work with integer vectors.

Though the works in [3, 1] focus on selecting the best-matching identity in the database managed by the server (giving out to the client a specific identifier), our main solution is an identification protocol that allows to select and report the identifiers of *all* (if more than one are present) the enrolled identities whose distance to the user's fingercode is lower than a given threshold. We also propose the following variants for authentication purposes. The first one considers applications where the client is interested only into knowing if the users's fingerprint is in the database or not (without an identifier). The second one handles the case of a client who wishes to verify if a given alleged identity is in the database and if the just acquired fingerprint matches with such identity. Such scenarios are detailed in the next sections.

Our protocols are entirely based on the use of homomorphic cryptosystems and gain a notable bandwidth saving (about $25-39\%$), if compared with the best previous work [1]. The computational complexity is still low and suitable for practical applications (as shown in Section 5). Moreover, even if such protocols are presented in the context of a fingerprint-based system, they can be generalized to any biometric system that shares the same matching methodology, namely distance computation and thresholding.

The rest of the paper is organized as follows. In Section 2, we briefly describe the Fingercode approach for fingerprint-based authentication. In Section 3 we give a rigorous description of the scenario considered in the paper. In Section 4 we introduce our protocols, whose complexity and security is analyzed in Section 5. Few technical parts are discussed in appendices for lack of space.

2. FINGERCODE-BASED ID MATCHING

Various approaches for automatic fingerprint matching have been proposed in the literature. The most popular ones are based on the minutiae pattern of the fingerprint and are collectively called minutiae-based approaches [5].

Although rather different one from the other, most of these methods require extensive preprocessing operations in order to reliably extract the minutia features [6]. Another class of fingerprint matching approaches matches directly the fingerprint images [7], or tries to match features extracted from the image by means of certain filtering or transform operations [8].

The algorithm this paper focuses on, is based on a particular representation of the fingerprints which yields a relatively short, fixed length code, called Fingercode [4] suitable for matching as well as storage on a smartcard. The matching step is particularly simple since it boils down to the computation of the Euclidean distance between the to-be-matched fingercodes and its comparison against a threshold τ. The fingercode representation exploits some well-known peculiarities of fingerprints to generate a short fixed length code, maintaining a high recognition accuracy.

The fingercode representation was selected because its good performances in terms of accuracy and speed, and because the used template and the matching step are particularly suited for a secure multi-party implementation. The accuracy of the fingercode system is related to the quality of the input image and the experience of the user to be correctly enrolled. In the literature, implementations of the fingercode algorithm report Equal Error Rates in the range of 3-5% working on standard fingerprint datasets [4, 9]. Experiments show that filter-based matchers such as the fingercode tend to perform slightly worse than the state-of-the-art minutiae-based matcher on the same databases, but the fingercode matching function has a much lower computational complexity.

The feature extraction algorithm can be split into four main steps as shown in Figure 1. By referring to such a figure, given the fingerprint gray scale image, the matching algorithm works as follow:

1. determine a reference point;

2. tessellate the region of interest around the reference point;

3. filter the region of interest in eight different directions using a bank of Gabor filters (more details in [10]);

4. compute the average absolute deviation from the mean of gray values in individual sectors in filtered images to define the feature vector or the fingercode.

The result of the above procedure is the fingercode that is a k-dimensional feature vectors. By referring again to Figure 1, the top left disk represents the details for the 0° degree orientation of the Gabor filter, while the bottom right disk represents the 157.5° component. Each disk corresponds to one particular Gabor filter that enhances the details in given directions. In the original configuration, there are 5 bands and 16 sectors for each disk, which results in a total of 640 $(5 \times 16 \times 8)$ components for each fingercode.

Matching is easily achieved by computing the Euclidean distance between the to-be-matched fingercodes and comparing it against a matching threshold. The translation invariance in the fingercode is due to the proper choice of the reference point. However, features are not rotationally invariant. An approximate rotation invariance is achieved by cyclically rotating the features in the fingercode itself. To do so, for each fingerprint in the database, it is necessary

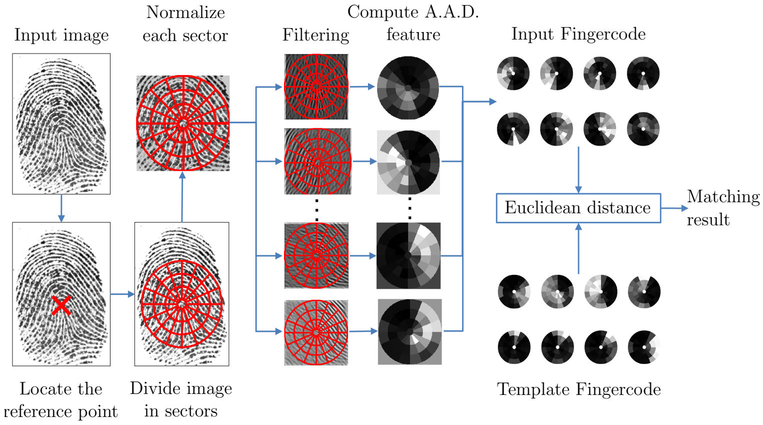


Figure 1: System diagram of the fingerprint authentication system.

to store m fingercode templates corresponding to a given set of m rotations of the fingerprint image (usually we have $m = 5$). The input fingercode is matched against the m templates stored in the database to obtain m different matching scores. Matching scores are compared with a threshold τ and if at least one of them is below τ the matching succeeds. To improve the matching accuracy, biometric systems may also use personalized threshold (τ_i) for each user so to ensure a better accuracy and a lower false positive rate.

To adapt the plain version of this algorithm to work on encrypted data, we need to quantize the data involved in the computation. This is necessary because the cryptographic technique works just only on integer number (i.e. $\mathbb{Z}_n$). Moreover, we can also change some parameters (e.g., number of bands, sectors) in order to reduce the length of the fingercode. There are several possible choices for such parameters, our approach is based on finding, via experiments, the solution that guarantees the lowest number of bits and components without significantly affecting the performance of the overall system. The subsequent privacy transformation will not affect the accuracy of the underling biometric identification system.

3. OUR SCENARIO

In the following we discuss our scenario as well as some known constructions of privacy-preserving identification using biometric measurements. In particular we focus on the works of Erkin *et al.* [3] and of Sadeghi *et al.* [1] as our solution employs some basic building blocks used already in [3, 1]. In a nutshell, both such works implement a secure face recognition protocol using standard Eigenfaces [11]. The main difference between the two solutions is that the former relies entirely on homomorphic encryption (HE) while the latter adopts an hybrid approach where Garbled Circuits (GC) are used in conjuction with HE.

In this paper we consider the following scenario: a client $\mathcal{C}$ is equipped with a biometric device (e.g. a fingerprint reader). The device is used to read some biometric data ID to be transmitted (in some encrypted form) to a server $\mathcal{S}$ where a database of authorized identities is stored. The privacy requirement that we impose is that $\mathcal{C}$ should not be able to get any information beyond the fact that ID is in the database or not (as stated later, in our work we also consider the support of more than one matching record). At the same time $\mathcal{S}$ should not get any information about ID (not even if it is in the database or not).

In our scenario $\mathcal{C}$ owns a pair of (matching) keys $(pk_\mathcal{C}, sk_\mathcal{C})$ for a public-key cryptosystem and we assume that the server has a certified copy of $pk_\mathcal{C}$[1]. As in [3, 1], our solutions adopt the following three steps approach:

- **vector extraction**: on a first stage the target biometry (i.e. the information acquired by the biometric device) is “converted” in a quantized characteristic feature vector $\bar{x}$; in our specific case, the fingerprint image is processed as described in Section 2 in order to extract the fingercode vector; similar processes are available in literature for other biometric systems;
- **distances computation**: the distances (with respect to some appropriate metric) between the target vector $\bar{x}$ and the vectors corresponding to each ID in the database are computed; in our case, we are going to use the Euclidean distance as required by the fingercode system;
- **selection of the matching identities**: one (or more) IDs matching the target ID are selected.

About the last step, in order to open new application scenarios, we slightly change the original semantic of the problem: instead of querying about the nearest matching enrolled identity in the database as in [3, 1], we are interested in getting all the matching enrolled identities. In other words: the required outcome for the client is the list of all the identities in the database whose characteristic feature vectors are “near enough” to be considered a successful match (i.e., the distance is lower than the threshold τ).

Are the two problems equivalent? With some biometric systems, if we assume well-chosen parameters (like the threshold τ), one may assume that a measure of a specific biometry matches with just one person: the owner. If, for some application-related reasons, the same person is enrolled in the database more than one time, it should be fine to return all these identities to the client. However, for specific

[1] Jumping ahead, the protocols of Section 4 require the use of two different encryption schemes (Paillier and EC-ElGamal, see below): to simplify the presentation we are assuming the $pk_\mathcal{C}$ contains the different public-keys of the required cryptosystems.

biometric systems or applications, it could not be equivalent and/or desirable.

Willing to adapt our main construction to the identification problem threated in [3, 1] we also propose two further variants in Section 4.4: one is suitable to applications where the client is only interested in knowing if a specific biometry is in the database or not (without an identifier); the second one allows a client to verify if a given alleged identity $\hat{id}$ is in the database and if a given biometric measure matches with that identity.

3.1 Parameters and Model

We will denote the symmetric security parameter by t and the asymmetric one (i.e., bit-length of RSA moduli) by T. Recommended parameters for short-term security are $t = 80$ and $T = 1024$, whereas for long-term security $t = 128$ and $T = 3072$ are recommended [12].

In all the scenarios we consider a server $\mathcal{S}$ with a database of n enrolled entities, where each of them is represented by a characteristic feature vector of k ℓ-bits integers. We will denote with τ the biometric-threshold that, given a specific metric, allows to say if two biometric measures match or not. In order to support the specific matching logic on the fingercode (see Section 2), we will assume that for each enrolled identity $m = 5$ different vectors are stored in the database as well as an eventual identity-specific threshold τ^i.

The values of the k, ℓ and m parameters must be tuned according to the current fingerprint dataset. For example, working with a dataset of $n = 900$ fingerprints captured with a standard fingerprint sensor, a proper parameters configuration can be the following: $2-5$ concentric bands, $4-16$ sectors, $2-8$ gabor filters, quantized with $4-8$ bits and stored with five different orientations ($k = 16-640$, $\ell = 4-8$ and $m = 5$). Typical bit lengths of the fingercode range from 64 to 5120 bits.

Finally, we work in the *honest-but-curious* model (as in [3, 1]), where parties are assumed to follow the protocol but may try to learn additional information from the protocol trace beyond what can be derived from the inputs and outputs of the algorithm when used as a black-box.

4. OUR CONSTRUCTIONS

In this section we present our proposals to efficiently solve the problems introduced in Section 3. Our constructions strongly rely on the notion of (additively) Homomorphic Encryption (HE) schemes.

4.1 Homomorphic Encryption

A public-key encryption cryptosystem is said to be additively homomorphic if, given the encryptions[2] $[\![a]\!]$ and $[\![b]\!]$, the ciphertext $[\![a+b]\!]$ can be easily computed as $[\![a+b]\!] = [\![a]\!] \circ [\![b]\!]$, where $\circ$ denotes some efficiently computable operator (e.g. plain multiplication on the underlying ring). As a consequence, it is also possible to compute the multiplication of an encryption $[\![a]\!]$ for a constant c in clear as $[\![c \cdot a]\!] = [\![a]\!]^c$.

In our protocol we will make extensive use of semantically secure[3] additively homomorphic encryption schemes. In particular we will adopt Paillier's encryption scheme and a variant of the well known-variant of the ElGamal encryption scheme (the latter is detailed in Appendix A).

4.1.1 *Paillier cryptosystem.*

Paillier presented in [13] the following efficient scheme. Let $N = pq$ be a T-bits RSA modulus, with p, q primes, and g is an element whose order is multiple of N in $\mathbb{Z}^*_{N^2}$. The probabilistic encryption of a message $x \in \mathbb{Z}_N$ is computed as $[\![x]\!] = g^x r^N \bmod N^2$, where $r \in \mathbb{Z}_N$ is chosen at random. Such a scheme is clearly additively homomorphic: given $[\![x]\!]$ and $[\![y]\!]$ we have that $[\![x+y]\!] = [\![x]\!] \cdot [\![y]\!]$. The homomorphic operation is deterministic and, for security reasons, it will be occasionally necessary to "re-randomize" the resulting ciphertext $[\![z]\!]$ using a fresh random value $r \in \mathbb{Z}_N$ as $[\![z]\!] \cdot r^N$. Pailliers scheme is efficient enough to be used in practice but the ciphertext is twice as long as the original plaintext. The semantic security of the scheme is proven under the decisional composite residuosity assumption (DCRA) [13].

4.2 Vector Extraction

During a preliminary phase the acquired fingerprint image is converted into a quantized fingercode vector. We assume that this phase is done in clear (i.e. not in the encrypted domain) by the client. Notice that this is not an issue in our honest-but-curious setting where the client (i.e. the biometric device) *already* has the fingerprint data.

Moreover, given our current state of knowledge, such an assumption seems to be necessary for our protocol to be practical. Indeed, for many biometric systems (e.g. fingerprint, iris,...) the analysis of the biometric measures, and their corresponding quantization process, are too complex to be efficiently done on the encrypted domain.

4.3 An HE-based Solution for the Identification of Matching Identities

Here we outline the details of our main construction. As stated above, we assume that the client $\mathcal{C}$ has already processed the fingerprint image to get a characteristic feature vector (fingercode) $\bar{x}$. On the other side, the server $\mathcal{S}$ manages a database of n pairs $(id^i, \bar{y}^i)$, where id^i is a unique numeric identifier associated to the specific enrolled identity and $\bar{y}^i$ is the related precomputed feature vector. Our solution requires the use of specific values for these identifiers: $id^i = 2^i$ (powers of 2).

In this phase we deliberately ignore some technical details that are fingercode-specific. In particular we do not consider here the presence of m different fingercodes for each identity and the use of identity-specific thresholds τ^i. In this way we get a more general protocol that could be used for other biometric systems as well. Fingercode specific aspects are discussed in Section 4.5.

4.3.1 *Secure vector submission.*

Following Erkin *et al.*'s approach, in the first step the client $\mathcal{C}$ sends an encrypted element-by-element version of the integer vector $\bar{x}$ to the server $\mathcal{S}$: more specifically, k Paillier encryptions $[\![x_0]\!], \ldots, [\![x_{k-1}]\!]$ jointly with a further encryption $[\![\sum_{j=0}^{k-1} x_j^2]\!]$. The latter value will be used to complete the computation of the distances in the ciphertexts domain as described later.

[2] In the rest of the paper we will denote with $[\![x]\!]$ the encryption of the plaintext x; the public-key used for the encryption is generally deducible from the context.

[3] A cryptosystem is said to be *semantically secure* if it is infeasible for a "passive" adversary to derive significant information about the plaintext when it is given only the corresponding ciphertext and the used public-key.

4.3.2 Computation of distances.

In this step, it is required to compute the distances between the target vector $\bar{x}$ and characteristic feature vectors $\bar{y}^i$ extracted from each identity in the database. The fingercode system, as well as other biometric systems, uses the Euclidean distance as underlying metric. In particular we consider squared distance to reduce the complexity of the protocol[4]. Denoting with D^i the square of the Euclidean distance between $\bar{x}$ and the stored vector $\bar{y}^i$, the server can non-interactively compute $[\![D^i]\!]$ by exploiting the homomorphic properties of the Paillier cryptosystem, its knowledge of $\bar{y}^i$ and the ciphertexts received by $\mathcal{C}$ as follows:

$$
\begin{aligned}
[\![D^i]\!] &= \left[\!\!\left[\sum_{j=0}^{k-1}(x_j - y_j^i)^2\right]\!\!\right] \\
&= \left[\!\!\left[\sum_{j=0}^{k-1} x_j^2\right]\!\!\right] \cdot \left[\!\!\left[-2\sum_{j=0}^{k-1} x_j y_j^i\right]\!\!\right] \cdot \left[\!\!\left[\sum_{j=0}^{k-1}(y_j^i)^2\right]\!\!\right] \\
&= \left[\!\!\left[\sum_{j=0}^{k-1} x_j^2\right]\!\!\right] \cdot \prod_{j=0}^{k-1}[\![x_j]\!]^{-2y_j^i} \cdot \left[\!\!\left[\sum_{j=0}^{k-1}(y_j^i)^2\right]\!\!\right] \quad \forall i \in \{1,\dots,n\}
\end{aligned}
$$

4.3.3 Identities selection.

This step uses a protocol, that we call *bit-MIN*, that allows to obliviously[5] compute an encryption of the *predicate bit* of "$X < Y$": i.e. $[\![b]\!] = \text{bit-MIN}([\![X]\!],[\![Y]\!])$ where b is such that $(b = 0 \Leftrightarrow X < Y)$. Here we assume that the server $\mathcal{S}$ has got the encryptions of the two inputs $[\![X]\!],[\![Y]\!]$ and that is interested in receiving the outcome $[\![b]\!]$. The security of our protocol requires that the client $\mathcal{C}$ should not learn anything about X, Y and that the returned predicate bit b should not be revealed to $\mathcal{S}$. For this task we simplify a protocol given in [3], which in turn builds from a solution originally given by Damgård *et al.* [14]. In these schemes, in order to gain bandwidth, a protocol-tailored variant of the Benaloh-Fischer cryptosystem [15] is adopted[6] as underlying, additively homomorphic, encryption scheme.

Our solution is simpler than [3] in that we can allow the client to know b[7]. For lack of space, it is described in Appendix A.

Once the distance computation phase is over, $\mathcal{S}$ gets the distances $[\![D^1]\!],\dots,[\![D^n]\!]$. Let's consider the following n pairs: $(id^1,[\![D^1]\!]),\dots,(id^n,[\![D^n]\!])$. The server randomly permutes[8] these pairs as: $(id^{j_1},[\![D^{j_1}]\!]),\dots,(id^{j_n},[\![D^{j_n}]\!])$ and then it computes, using parallel executions of bit-MIN, the values $[\![b^{j_i}]\!] = \text{bit-MIN}([\![\tau]\!],[\![D^{j_i}]\!])$ for $i \in \{1,\dots,n\}$. The following invariant holds:

$$b^i = 1 \Leftrightarrow D^i < \tau \text{ for } i \in \{1,\dots,n\} \qquad (1)$$

Finally, the server computes and returns to the client the following encrypted value:

$$[\![R]\!] = \left[\!\!\left[\sum_{i=1}^{n} b^i \cdot id^i\right]\!\!\right] = \prod_{i=1}^{n}[\![b^i]\!]^{id^i}$$

As a consequence of invariant (1), it is easy to check that the final value R will consist in the sum of the numeric identifiers associated to the enrolled identities that match the target biometry. In other words: the bit at position i in R is set to 1 if and only if the i-th identity matches. The client can easily extract R and reconstruct the list of matching identities. The complete protocol is shown in Figure 2.

Comparing our approach with that of Erkin *et al.* [3], our identities-selection methodology leads to a solution that is more efficient both in terms of the round complexity (our protocol requires a constant number of rounds, like in [1]) and in terms of the overall bit (bandwidth) complexity (see Section 5.1). Moreover, further bandwidth saving comes from the fact that, in the bit-MIN protocol, we use additive ElGamal (over Elliptic Curves) as underlying homomorphic encryption scheme, which is perfectly suited for our application. Additive ElGamal differs from standard one [16] in that it is additively homomorphic. This comes at the cost of requiring a relatively small messages space (for our application, however, this limitation is not a problem at all as we use additive ElGamal to encrypt bits). Moreover, the scheme becomes extremely bandwidth-efficient when implemented over suitably chosen Elliptic Curve[9]. In the following we will refer to this scheme as *additively homomorphic EC-ElGamal* (additional details are deferred to Appendix A).

4.4 Variants for Specific Scenarios

Here we collect a few variants of the novel protocol of Figure 2 suitable for some specific application scenarios.

4.4.1 Simple authentication.

In applications where the client is willing to accept a simple boolean outcome, like "*authenticated/rejected*", our solution is functionally equivalent to the prior works in [3, 1]. It is sufficient to change the way the value $[\![R]\!]$ is computed:

$$[\![R]\!] = r \cdot \left[\!\!\left[\sum_{i=1}^{n} b^i\right]\!\!\right] = (\prod_{i=1}^{n}[\![b^i]\!])^r$$

where r is a fresh random integer. The client $\mathcal{C}$ will output *rejected* if $R = 0$, *authenticated* otherwise.

4.4.2 Authentication with identity confirmation.

Let's think about the following high-security authentication scenario: the person who is going to authenticate is doubly checked through some kind of hardware token (or a simple card with a bar-code) and some specific biometry (e.g., fingercode). In this case the client (the biometric reader) is able to send to the server an alleged identity $\hat{id}$ read from the hardware token. The final boolean outcome will be positive (*authenticated*) if and only if the submitted

[4] We are of course using the fact that the square function is monotonically increasing function on positive inputs. We implicitly assume that the threshold values τ, τ^i are properly adapted to accommodate this.

[5] Here by oblivious we mean that the output of the protocol is hidden to at least one of the two parties.

[6] We will not discuss the details of this cryptosystem here. The interested reader is referred to [14].

[7] The functionality of the bit-MIN sub-protocol is exploited in the work of Erkin *et al.* [3] in the identity selection phase. The security of their final solution requires an even stronger variant of it: the protocol has to be oblivious to both the parties (i.e. not even the client can learn the bit b)

[8] The randomization is only used to hide to the client the relation among the positions of the identities in the DB and the order of querying. It is strictly required a fresh permutation at each new session.

[9] For instance using the curves from the SECG standard [17, 18].

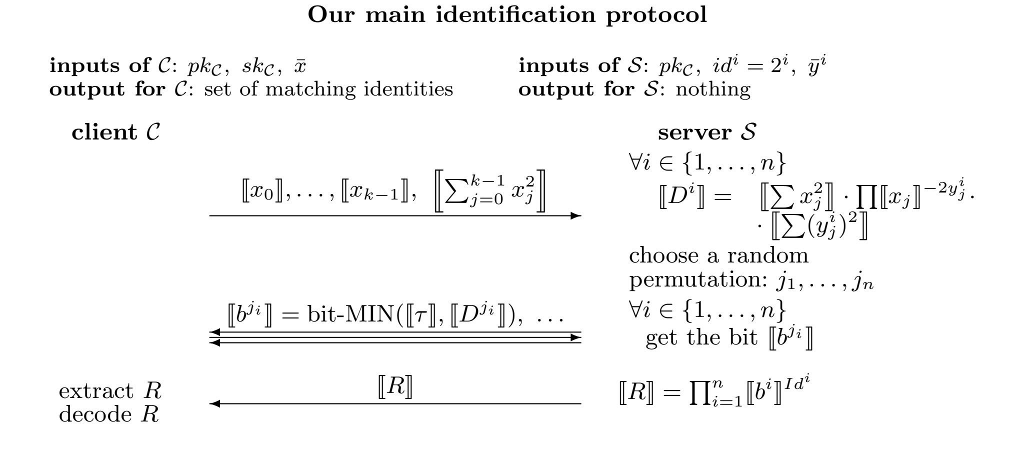

Our main identification protocol
inputs of $\mathcal{C}$: $pk_{\mathcal{C}},\ sk_{\mathcal{C}},\ \bar{x}$
output for $\mathcal{C}$: set of matching identities
inputs of $\mathcal{S}$: $pk_{\mathcal{C}},\ id^i = 2^i,\ \bar{y}^i$
output for $\mathcal{S}$: nothing
client $\mathcal{C}$
server $\mathcal{S}$
$\llbracket x_0 \rrbracket, \dots, \llbracket x_{k-1} \rrbracket,\ \left\llbracket \sum_{j=0}^{k-1} x_j^2 \right\rrbracket$
$\forall i \in \{1, \dots, n\}$
$\llbracket D^i \rrbracket = \llbracket \sum x_j^2 \rrbracket \cdot \prod \llbracket x_j \rrbracket^{-2y_j^i} \cdot$
$\cdot \llbracket \sum (y_j^i)^2 \rrbracket$
choose a random
permutation: $j_1, \dots, j_n$
$\llbracket b^{j_i} \rrbracket = \text{bit-MIN}(\llbracket \tau \rrbracket, \llbracket D^{j_i} \rrbracket),\ \dots$
$\forall i \in \{1, \dots, n\}$
get the bit $\llbracket b^{j_i} \rrbracket$
extract R
decode R
$\llbracket R \rrbracket$
$\llbracket R \rrbracket = \prod_{i=1}^{n} \llbracket b^i \rrbracket^{Id^i}$

Figure 2: An HE-based for the identification of matching identities

biometry matches one of the enrolled identities as well as the alleged identity $\hat{id}$.

The suitable protocol is shown in Figure 3. After the computation of the encrypted distances $\llbracket D^i \rrbracket$, the server will compute the auxiliary values:

$$\llbracket m^i \rrbracket = \llbracket r^i \cdot (\hat{id} - id^i) \rrbracket = (\llbracket \hat{id} \rrbracket \cdot \llbracket id^i \rrbracket^{-1})^{r^i}$$

where r^i are fresh random integers. All the values m^i will be different than zero except for the alleged identity $\hat{id}$.

The values $\llbracket m^i \rrbracket$ will be sent to the client during the executions of the sub-protocol bit-MIN: $\mathcal{C}$ will return the exact outcome $\llbracket b^i \rrbracket$ of bit-MIN only if the corresponding m^i is not null, otherwise a dummy outcome $\llbracket 0 \rrbracket$ is sent. In this way only a single bit b^i can be not null and only if it matches the alleged identity $\hat{id}$.

4.5 Fingercode-specific Adaptation

The fingercode biometric system has few peculiarities that have to be specifically addressed in our proposals. The main issue comes from the biometric matching algorithm that, in order to keep low the error rate, requires to store m different fingercodes (at different inclinations) for each enrolled identity. The algorithm considers the m Euclidean distances between the target fingercode and all these m vectors: the final reference distance will be the minimum one.

If we consider the case of a successful match with a given biometry, it is quite probable that more than one of the m (quite similar) fingercodes present a specific distance under the threshold τ. Given that, if we use the same identifier id^i for all the m fingercode variants, in our identification protocol of Figure 2 we could have as final outcome $R = k_{i_1} \cdot id^{i_1} + \cdots + k_{i_q} \cdot id^{i_q}$, where $id^{i_1}, \dots, id^{i_q}$ are q different matching identities and $k_{i_1}, \dots, k_{i_q}$ are specific constants that denote the number of matching fingercodes for each identity. Such R would not allow a reliable decoding. To overcome the problem we can use $(n \cdot m)$ different identifiers for each of the stored fingercodes: each biometry will have m different identifiers with a publicly known relation. For example, the i-th biometry could use the following identifiers: $id^{mi} = 2^{mi}, id^{mi+1} = 2^{mi+1}, \dots, id^{mi+(m-1)} = 2^{mi+(m-1)}$. Given a matching identifier $id^w = 2^w$, a unique identifier for the biometry is (w mod m).

The two variants presented in Section 4.4 do not require the use of $n \cdot m$ different identifiers: the former does not use them at all, and in the latter the use of the same identifier for all the m fingercodes does not arise specific issues.

As stated in Section 2, another peculiarity of the quantized version of the fingercode system is the possibility to use a personalized threshold τ^i for each one of the enrolled identities. This can be easily handled in our selection proposals: the threshold τ^i will be stored in the database record and used in the invocation: $\llbracket b^i \rrbracket = \text{bit-MIN}(\llbracket \tau^i \rrbracket, \llbracket D^i \rrbracket)$. It is important to note that the τ^i values (or any related information) for the non-matching identities are not revealed to the client.

4.6 More Scalability

The main proposal in Figure 2 strictly requires the use of powers of 2 as identifiers: in real application scenarios with a wide set of enrolled people this fact could limit its scalability. Indeed, the maximum number of different identifiers is equal to the bit-length of the Paillier plaintext (T). For a security level of $t = 128$, we can handle at most $T = 3072$ different identifiers[10]. This can be handled clustering the $n > T$ identifiers and using multiple outcomes R_j; more specifically, for $j = 0, \dots, \lceil \frac{n}{T} \rceil - 1$ the server computes $\llbracket R_j \rrbracket = \prod_{i=1}^{n_j} \llbracket b^{jT+i} \rrbracket^{2^i}$, where $n_0 = T, n_1 = T, \dots, n_{\lceil \frac{n}{T} \rceil - 1} = n \bmod T$ are the cluster cardinalities. In this way the i-th bit in R_j is associated to the identity id^{jT+i}.

The ciphertexts $\llbracket R_j \rrbracket$ are sent to the client in the last exchange in the protocol. These changes on the protocol do not imply any further leakage of information.

As stated above, we notice that such scalability issue on the number of identifiers does not apply at all to both variants presented in Section 4.4.

5. ANALYSIS OF THE PROPOSALS

5.1 Bandwith Usage

In order to evaluate the performances of our new proposals, we carry on an analytics comparison of the bandwidth

[10] The fingercode system reduces this availability of a factor of m.

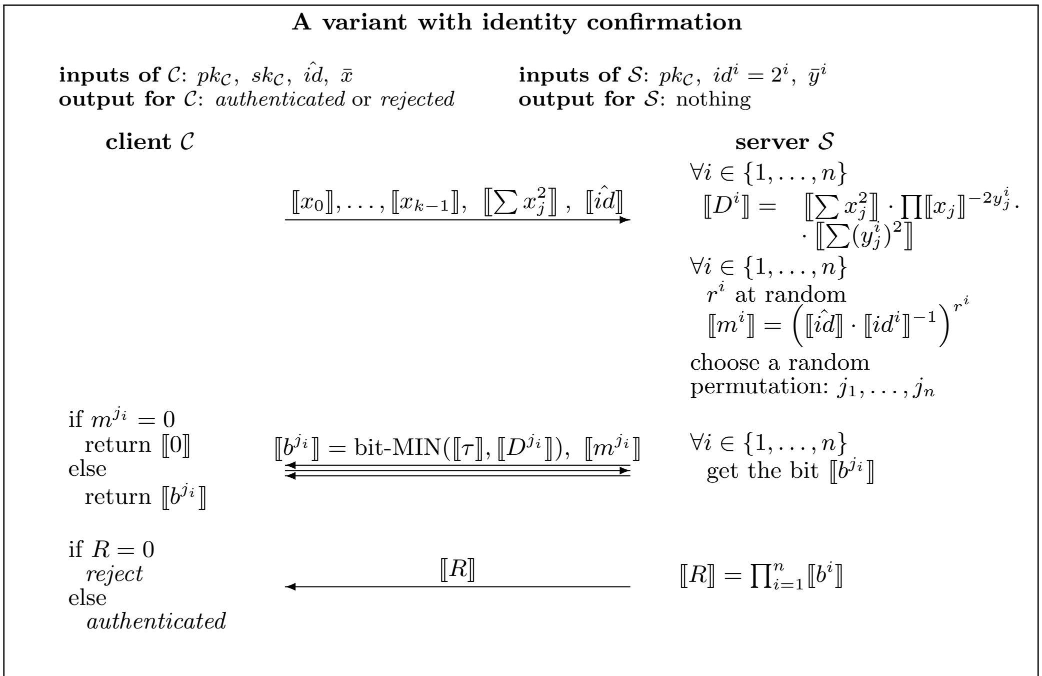

A variant with identity confirmation
inputs of $\mathcal{C}$: $pk_{\mathcal{C}},\ sk_{\mathcal{C}},\ \hat{id},\ \bar{x}$
output for $\mathcal{C}$: *authenticated* or *rejected*
inputs of $\mathcal{S}$: $pk_{\mathcal{C}},\ id^i = 2^i,\ \bar{y}^i$
output for $\mathcal{S}$: nothing
client $\mathcal{C}$
server $\mathcal{S}$
$\forall i \in \{1, \ldots, n\}$
$[\![D^i]\!] = [\![\sum x_j^2]\!] \cdot \prod [\![x_j]\!]^{-2y_j^i} \cdot \cdot [\![\sum (y_j^i)^2]\!]$
$[\![x_0]\!], \ldots, [\![x_{k-1}]\!],\ [\![\sum x_j^2]\!],\ [\![\hat{id}]\!]$
$\forall i \in \{1, \ldots, n\}$
r^i at random
$[\![m^i]\!] = \left([\![\hat{id}]\!] \cdot [\![id^i]\!]^{-1}\right)^{r^i}$
choose a random permutation: $j_1, \ldots, j_n$
if $m^{j_i} = 0$
return $[\![0]\!]$
else
return $[\![b^{j_i}]\!]$
$[\![b^{j_i}]\!] = \text{bit-MIN}([\![\tau]\!], [\![D^{j_i}]\!]),\ [\![m^{j_i}]\!]$
$\forall i \in \{1, \ldots, n\}$
get the bit $[\![b^{j_i}]\!]$
if $R = 0$
reject
else
authenticated
$[\![R]\!]$
$[\![R]\!] = \prod_{i=1}^n [\![b^i]\!]$

Figure 3: A protocol for authentication with identity confirmation

usage of all the available solutions. More specifically, we consider our protocol in Figure 2 as well as the solutions in [3, 1]. Due to the different underling biometric systems used in the prior works, we are excluding the preliminary phase related to vector extraction. We consider a scenario where the client $\mathcal{C}$ has got an already-computed vector $\bar{x}$ with k ℓ-bits integers and the server manages a DB with n identities. Later we use the maximum bit-length of the square of Euclidean distance computed on such vectors: $\ell' = 2\ell + \lceil log_2 k \rceil + 1$.

In the following we detail on the respective analysis:

- **our protocol**: it requires the exchange of $k + 2$ Paillier encryptions and n invocations of our bit-MIN sub-protocol; each invocation of this last implies the use of $2\ell'$ EC-ElGamal ciphertexts and of 3 Paillier encryptions (see Section 4 and Appendix A);

- **Erkin *et al.***: their HE-based solution exchanges $k+3$ Paillier ciphertexts, makes n invocations of their fully-oblivious bit-MIN sub-protocol and n of a specific multiplication sub-protocol: the former requires $2\ell' + 1$ DGK cyphertexts and 3 Paillier ones; the latter makes use of 2 Paillier encryptions (see [3] for details);

- **Sadeghi *et al.***: for such work we are considering all the optimizations[11] described in [1]. Here the preliminary delivery of the encrypted vector uses $k+1$ Paillier encryptions, then a sub-protocol for the data conversion is invoked: it exchanges n Paillier ciphertexts and makes use of $n\ell'$ parallel executions of an Oblivious-Transfer (OT) protocol. The use of GC evaluation requires the exchange of the garbled values for the server's inputs ($n \cdot \ell'$ bits), $n \cdot 3\ell'$ table entries for the conversion protocol as well as $8\ell'(n-1) + 4(n+1)$ entries for the minimum selection. Each garbled value and each entry table requires $t + 1$ bits. For the OT protocol we are considering the use of the optimal implementation in [22].

The remaining global parameters are chosen according to a plausible fingercode-based scenario[12]: a DB with $n = 900 \cdot m$ fingercodes ($m = 5$ fingercodes for each identity) and vectors of $k = 16$ integers of $\ell = 7$ bits. The total bandwidth usage (in kilobytes) for the recommended [12] short/mid/long-term security parameters ($t = 80, 112, 128$) are reported in Table 1. The round complexity is also reported in the last column.

Table 1: Bandwidth usage (in kb) and rounds

protocol	$t = 80$	$t = 112$	$t = 128$	rounds
our protocol	10100.9	16152.3	20867.7	$\mathcal{O}(1)$
Sadeghi *et al.*	16502.0	23712.8	27882.7	$\mathcal{O}(1)$
Erkin *et al.*	27567.2	55134.5	82701.7	$\mathcal{O}(\log n)$

Our protocol provides a notable bandwidth saving: about $63 - 75\%$ on Erkin *et al.* and $25 - 39\%$ on Sadeghi *et al.*. Moreover the constant rounds complexity allows to get a better communication latency during the protocol executions.

5.2 Time Efficiency

Our solution is mainly designed to save on bandwidth and round complexity, nevertheless its computational efficiency

[11] The analysis takes account of the following techniques: "free XOR" gates [19] to do "free" evaluation of XOR gates (no communication and negligible computation), *garbled row reduction* on non-XOR gates [20] and *point-and-permute* [21] to fast GC evaluation. They also consider the possibility to carry on an heavy on-line preprocessing phase among the client and the server. We are excluding such feature in our analysis: it restricts the range of possible applicatory scenarios.

[12] Our advantage on the bandwidth usage is not strictly related to these fingercode-specific parameters: we still enjoy optimal bandwidth usage even using different parameters.

is quite practical and comparable with the others. Its time complexity is clearly linear in the number of identities in the DB. We fully implemented our main protocol in order to gather information about its real efficiency: we consider a local run of the authentication protocol on a database of 64 identities with $m = 5$ fingercodes for each ($n = 320$) with parameters $t = 80$, $k = 16$ and $\ell = 7$. Exploiting the use of offline precomputation one execution requires about 16 seconds on a common PC (Intel Core 2 Duo at 2.4GHz). Willing to force a comparison with the previous works: Erkin *et al.* report an experiment where the distance computation and minimum extraction on 320 pre-computed feature vectors requires about 18 seconds; the efficient Sadeghi *et al.*'s solution requires about 8 seconds in such setting.

5.3 Security Argument

In this section we sketch a security argument for our protocol in Figure 2. In particular we want to argue that, in the honest-but-curious setting, no party should be able to get any information about the other party's input. In other words, this means that the client $\mathcal{C}$ should not be able to get anything about the database held by $\mathcal{S}$ (beyond what revealed by the functionality implemented by the protocol) whereas $\mathcal{S}$ should not get anything about the fingercode and outcome of the authentication process.

We discuss each phase of the protocol separately. The vector extraction phase is done entirely by $\mathcal{C}$ so no information is leaked to $\mathcal{S}$. Security of the distance computation phase can be proved easily following the same approach used by Erkin *et al.* in [3] (recall that our distance computation protocol is the same as that used in [3]). It remains to discuss the selection of the matching identities phase.

Intuitively it is clear that the protocol is private for the server as all the messages it receives are encrypted with respect to $\mathcal{C}$'s public key (using a semantically secure cryptosystem). Things are a bit trickier for the client as the latter knows the private key corresponding to the public key with respect to which the ciphertexts are created. Still, we argue that this does not allow $\mathcal{C}$ to get more information than what prescribed by the protocol. This is because, whenever $\mathcal{C}$ receives a ciphertext, the encrypted message is altered by $\mathcal{S}$ via an information theoretic secure mask. For instance, in the bit-MIN protocol (see Appendix A) $\mathcal{C}$ receives an encryption of d which is statistically indistinguishable from a uniformly distributed $101 + \ell$ random integer. As a final note we point out that even though $\mathcal{C}$ gets b[13] in the clear at the end of the protocol bit-MIN this is not an hazard (in an honest-but-curious setting) because of the fact that all the couples are randomly permuted by $\mathcal{S}$ before executing the bit-MIN protocol.

The two variants in Section 4.4 enjoy the same security level: the client does not get any information about the others identities in server's DB.

6. ADDITIONAL AUTHORS

Additional authors: Dario Fiore (Ècole Normale Supèrieure – CNRS-INRIA, email: `dario.fiore@ens.fr`), Riccardo Lazzeretti (Dip. di Ing. dell'Informazione – Università di Siena, email: `riccardo.lazzeretti@gmail.com`), Vincenzo Piuri (Dip. di Tecnologie dell'Informazione – Università di Milano, email: `vincenzo.piuri@unimi.it`), Fabio Scotti (Dip. di Tecnologie dell'Informazione – Università di Milano, email: `fabio.scotti@unimi.it`), Alessandro Piva (Dip. di Elettronica e Telecomunicazioni – Università di Firenze, email: `alessandro.piva@unifi.it`).

7. REFERENCES

[1] A. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *ICISC '09: Proceedings of the 12th Annual International Conference on Information Security and Cryptology*, ser. LNCS, vol. 5984. Springer-Verlag, December 2-4, 2009, pp. 235–253, full version available at http://eprint.iacr.org/2009/507.

[2] R. Bolle and S. Pankanti, *Biometrics: Personal Identification in Networked Society*, A. K. Jain, Ed. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[3] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *PETS '09: Proceedings of the 9th International Symposium on Privacy Enhancing Technologies.* Berlin, Heidelberg: Springer-Verlag, 2009, pp. 235–253.

[4] A. Jain, S. Prabhakar, L. Hong, and S. Pankanti, "Filterbank-based fingerprint matching," *Image Processing, IEEE Transactions on*, vol. 9, no. 5, pp. 846–859, May 2000.

[5] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition.* Springer Publishing Company, Incorporated, 2009.

[6] A. Jain, L. Hong, S. Pankanti, and R. Bolle, "An identity-authentication system using fingerprints." *Proc. IEEE, 85, (9)*, pp. 1364–1388, 1997.

[7] C. L. Wilson, C. Watson, , and E. Peak, "Effect of resolution and image quality on combined optical and neural network fingerprint matching," *Putt. Recognit., 33, (Z)*, pp. 317–331, 2000.

[8] C. Lee and S. Wang, "Fingerprint feature extraction using gabor filters," *Eleclron. Lett., 35, (4)*, pp. 288–290, 1999.

[9] H.-W. Sun, K.-Y. Lam, M. Gu, and J.-G. Sun, "An efficient algorithm for fingercode-based biometric identification," in *OTM Workshops (1)*, 2006, pp. 469–478.

[10] A. Jain, S. Prabhakar, and L. Hong, "A multichannel approach to fingerprint classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 4, pp. 348–359, Apr 1999.

[11] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, 1991, pp. 586–591.

[12] "Nist recommendation for key management," ser. NIST Special Publication, vol. 800-57, August 2005.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT'99*, ser. LNCS, J. Stern, Ed., vol. 1592. Springer-Verlag, Berlin, Germany, May 1999, pp. 223–238.

[14] I. Damgård, M. Geisler, and M. Krøigard, "Efficient and secure comparison for on-line auctions," in

[13] More specifically, the client $\mathcal{C}$ does not directly get b but instead a related information: the bit λ.

ACISP, ser. Lecture Notes in Computer Science, J. Pieprzyk, H. Ghodosi, and E. Dawson, Eds., vol. 4586. Springer, 2007, pp. 416–430.

[15] J. Cohen and M. Fischer, "A robust and verifiable cryptographically secure election scheme," in *26th FOCS*. IEEE Computer Society Press, Oct. 1985.

[16] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *CRYPTO'84*, ser. LNCS, G. R. Blakley and D. Chaum, Eds., vol. 196. Springer-Verlag, Berlin, Germany, Aug. 1985, pp. 10–18.

[17] "Standard for efficient cryptography, SEC1: Elliptic curves cryptography," Technical report, Certicom Research, 2000, available at http://www.secg.org.

[18] "Standard for efficient cryptography, SEC2: Recommended elliptic curves domain parameters," Technical report, Certicom Research, 2000, available at http://www.secg.org.

[19] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free xor gates and applications," in *ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part II.* Berlin, Heidelberg: Springer-Verlag, 2008, pp. 486–498.

[20] B. Pinkas, T. Schneider, N. Smart, and S. Williams, "Secure two-party computation is practical," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, M. Matsui, Ed., vol. 5912. Springer, 2009, pp. 250–267.

[21] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay—a secure two-party computation system," in *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium.* Berkeley, CA, USA: USENIX Association, 2004, pp. 20–20.

[22] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001, pp. 448–457.

APPENDIX

A. THE SUB-PROTOCOL BIT-MIN

In this section we recall the sub-protocol bit-MIN used in Section 4, which is a variant of the one proposed in [3]. As in rest of the paper: we consider a client $\mathcal{C}$ and a server $\mathcal{S}$. The latter has got the encryption of two ℓ-bit integers $[\![X]\!]$ and $[\![Y]\!]$[14]. The protocol bit-MIN allows $\mathcal{S}$ to compute the encrypted bit $[\![b]\!]$ such that $b = 0 \Leftrightarrow X < Y$ and uses as building block a variant of the comparison protocol proposed by Damgård *et al.* in [14] (see the next section). The protocol bit-MIN is given in Figure 4 and works as follows.

As first step the server homomorphically computes $[\![z]\!] = [\![2^\ell + X - Y]\!]$. Since X and Y are ℓ-bits long, z is an $\ell + 1$ bits integer. Moreover one can interestingly see that the most significant bit of z (which we denote z_ℓ) is 0 if and only if $X < Y$. Thus in order to learn if $X < Y$ it suffices to compute z_ℓ. This can be done as follows.

$\mathcal{S}$ additively blinds z with a suitable random value r, obtaining $[\![d]\!]$. $\mathcal{S}$ sends $[\![d]\!]$ to $\mathcal{C}$ and then they run the sub-protocol DGK (see next section) after which $\mathcal{S}$ will learn $[\![\lambda]\!]$ such that $\lambda = 0 \Leftrightarrow \hat{d} < \hat{r}$ (where $\hat{d}$ and $\hat{r}$ are, respectively, $d \bmod 2^\ell$ and $r \bmod 2^\ell$). We notice that the information about "$\hat{d} < \hat{r}$" is useful to compute z_ℓ. In fact observe that:

$$b = z_\ell = 2^{-\ell}(z - \hat{z}) = 2^{-\ell}(z - ((d - r) \bmod 2^\ell))$$

where it is possible to compute $(d-r) \bmod 2^\ell = (d \bmod 2^\ell) - (r \bmod 2^\ell) + \lambda \cdot 2^\ell$. Since $\lambda = 0 \Leftrightarrow \hat{d} < \hat{r}$ it is easy to see the correctness of z_ℓ.

A.1 The sub-protocol DGK

The DGK comparison protocol of [14] allows both parties (i.e. the client $\mathcal{C}$ and the server $\mathcal{S}$) to learn the bit λ of the predicate "$d < r$" where d and r are two ℓ-bit integers owned by $\mathcal{C}$ and $\mathcal{S}$ respectively.

The original DGK protocol is given in Figure 5 and works as follows. As in the other protocols, the client $\mathcal{C}$ has a pair of keys $(pk_\mathcal{C}, sk_\mathcal{C})$ for an additively homorphic cryptosystem: the original protocol uses the DGK [14] cryptosystem, we will use a different scheme as stated later. We are going to use another notation for such ciphetexts: $[x]$. The inputs for the parties are, respectively, an ℓ-bits integer d for the client and another ℓ-bit integer r for the server. After the run of the original protocol, the server (as well as the client) will learn the decision bit λ of "$d < r$" (i.e. $\lambda = 0 \Leftrightarrow d < r$) while d and r will remain hidden to the server and the client respectively. In our bit-MIN we use a slightly different version of this protocol where the client sends $[\![\lambda]\!]$ (encrypted with his Paillier public-key) instead of λ: in this way the value of the decision bit remains hidden to the server.

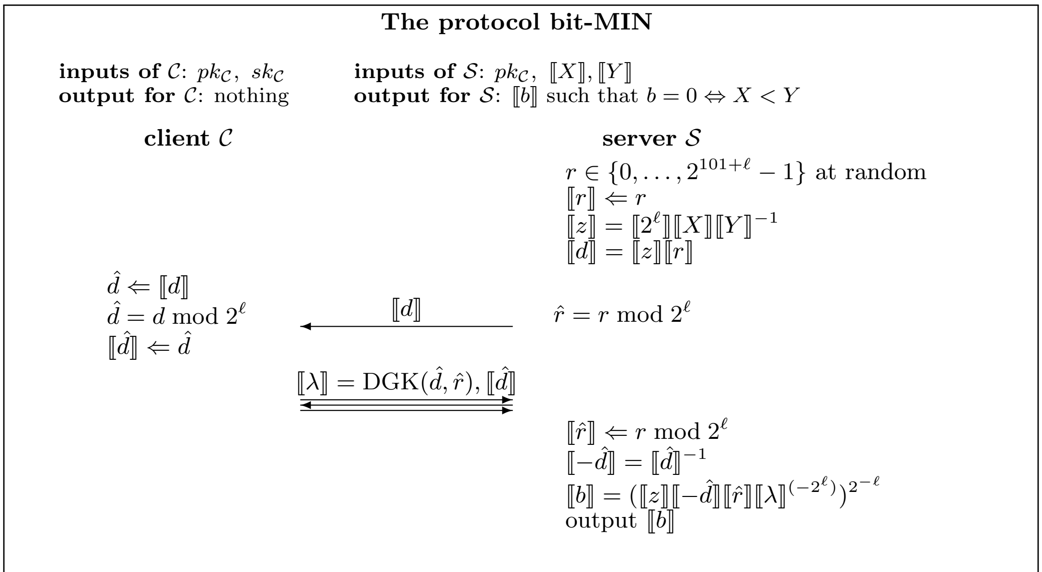
The protocol consists of three rounds during which 2ℓ ciphertexts are exchanged. More in detail, the server computes the values $[w_i] = [d_i \oplus r_i]$ and $[c_i] = [d_i - r_i + 1 + \sum_{j=i+1}^{\ell-1} w_j]$. The values c_i carry on the information whether or not $d < r$, in particular we have that one of the c_i's will be 0 if and only if $d < r$. To see the correctness of this, consider all possible cases. If $d = r$, then we clearly have $c_i = 1$ for all $i = 0, \ldots, \ell - 1$. If $d \neq r$, assume that the m-th bit (starting from the most significant) is the first one where they differ. Then $c_{\ell-1}, \ldots, c_{m+1}$ are equal to 1 while $c_m = d_m - r_m + 1$ (as $\sum_{j=m+1}^{\ell-1} w_j = 0$). Moreover since $w_m = 1$, we have $\sum_{j=i+1}^{\ell-1} w_j \geq 1$ and $c_i \geq 1$ $\forall i \in \{0, \ldots, m-1\}$. Thus c_m depends only on d_m and r_m and it will be 0 only if $d_m < r_m$.

Finally, since the c_i's might contain information about d and r, they are randomized (creating e_i) so that when the client decrypt e_i he will obtain either 0 (if $c_i = 0$) or a random value[15]. Therefore $\mathcal{C}$ will set $\lambda = 0$ if one of the e_i's decrypts to 0.

We defer the interested reader to [14] for more details about this protocol.

[14] We assume that encrypted input and output values of this protocol are made using the Paillier public-key of the client. Furthermore we note that, in the context of our protocol, bit-MIN is usually applied on inputs with a bit-length of $\ell' = 2\ell + \lceil log_2 k \rceil + 1$. In this section we assume ℓ-bits integers in order to simplify the protocol description.

[15] It is sufficient to check if the plaintext is equal to 0: the DGK cryptosystem, as well as one that we adopt, has a decryption procedure that is based on an exhaustive search in the plaintext space.



The protocol bit-MIN

inputs of $\mathcal{C}$: $pk_{\mathcal{C}},\ sk_{\mathcal{C}}$
output for $\mathcal{C}$: nothing

inputs of $\mathcal{S}$: $pk_{\mathcal{C}},\ [\![X]\!], [\![Y]\!]$
output for $\mathcal{S}$: $[\![b]\!]$ such that $b = 0 \Leftrightarrow X < Y$

client $\mathcal{C}$ — **server** $\mathcal{S}$

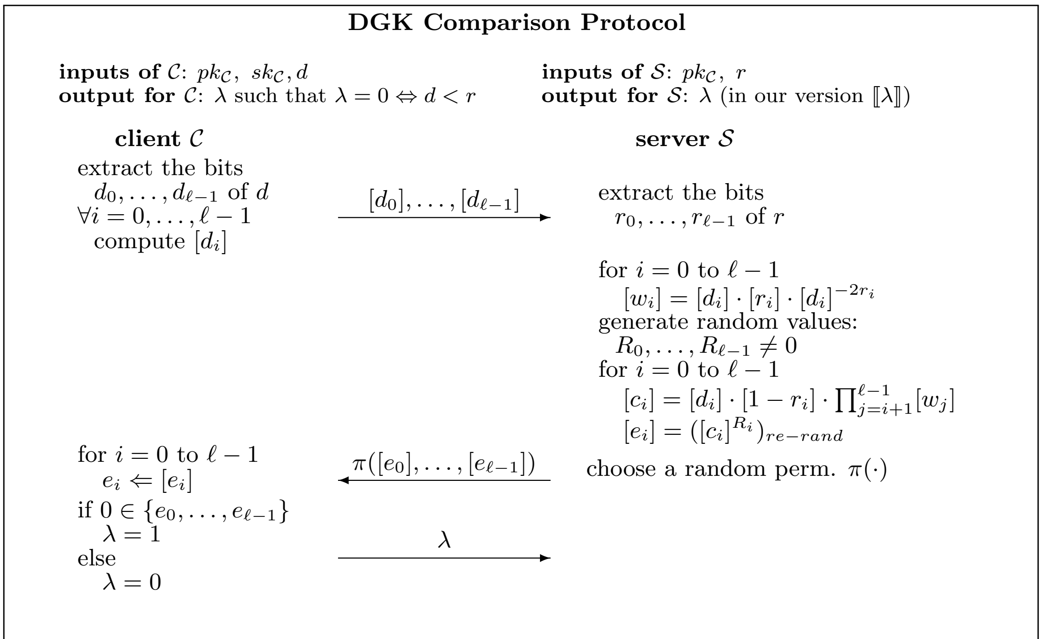
$r \in \{0, \ldots, 2^{101+\ell} - 1\}$ at random
$[\![r]\!] \Leftarrow r$
$[\![z]\!] = [\![2^\ell]\!][\![X]\!][\![Y]\!]^{-1}$
$[\![d]\!] = [\![z]\!][\![r]\!]$

$\hat{d} \Leftarrow [\![d]\!]$
$\hat{d} = d \bmod 2^\ell$ $\xleftarrow{[\![d]\!]}$ $\hat{r} = r \bmod 2^\ell$
$[\![\hat{d}]\!] \Leftarrow \hat{d}$

$\xrightarrow[\longrightarrow]{[\![\lambda]\!] = \mathrm{DGK}(\hat{d}, \hat{r}), [\![\hat{d}]\!]}$

$[\![\hat{r}]\!] \Leftarrow r \bmod 2^\ell$
$[\![-\hat{d}]\!] = [\![\hat{d}]\!]^{-1}$
$[\![b]\!] = ([\![z]\!][\![-\hat{d}]\!][\![\hat{r}]\!][\![\lambda]\!]^{(-2^\ell)})^{2^{-\ell}}$
output $[\![b]\!]$

Figure 4: A protocol that outputs the encrypted predicate bit of "$X < Y$"



DGK Comparison Protocol

inputs of $\mathcal{C}$: $pk_{\mathcal{C}},\ sk_{\mathcal{C}}, d$
output for $\mathcal{C}$: λ such that $\lambda = 0 \Leftrightarrow d < r$

inputs of $\mathcal{S}$: $pk_{\mathcal{C}},\ r$
output for $\mathcal{S}$: λ (in our version $[\![\lambda]\!]$)

client $\mathcal{C}$ — **server** $\mathcal{S}$

extract the bits
$d_0, \ldots, d_{\ell-1}$ of d
$\forall i = 0, \ldots, \ell - 1$
compute $[d_i]$

$\xrightarrow{[d_0], \ldots, [d_{\ell-1}]}$

extract the bits
$r_0, \ldots, r_{\ell-1}$ of r

for $i = 0$ to $\ell - 1$
$[w_i] = [d_i] \cdot [r_i] \cdot [d_i]^{-2r_i}$
generate random values:
$R_0, \ldots, R_{\ell-1} \neq 0$
for $i = 0$ to $\ell - 1$
$[c_i] = [d_i] \cdot [1 - r_i] \cdot \prod_{j=i+1}^{\ell-1} [w_j]$
$[e_i] = ([c_i]^{R_i})_{re-rand}$

for $i = 0$ to $\ell - 1$
$e_i \Leftarrow [e_i]$ $\xleftarrow{\pi([e_0], \ldots, [e_{\ell-1}])}$ choose a random perm. $\pi(\cdot)$
if $0 \in \{e_0, \ldots, e_{\ell-1}\}$
$\lambda = 1$
else $\xrightarrow{\lambda}$
$\lambda = 0$

Figure 5: A protocol that publicly computes the predicate bit of "$d < r$".

A.2 Our Changes

As stated in Section 4, we changed the protocol in Figure 5 in few points: the final bit λ is encrypted as $[\![\lambda]\!]$ in our solution, so that the bit value is oblivious to the server $\mathcal{S}$.

Furthermore we use a different cryptosystem instead of the one used in [14, 3]. The chosen cryptosystem is a known variant of the well-known ElGamal cryptosystem [16]. Such a scheme differs from the original in two points: it is additively homomorphic and all the computation is carried over a suitably chosen EC. We name it as: *additively homomorphic EC-ElGamal.* Let p be the $2t$-bit prime order of the working field, g a generator: the private-key will be a random value $a \in \mathbb{Z}_p$ and the public-key will be $(p, g, h = g^a)$. The encryption of a plaintext $x \in \mathbb{Z}_p$ is computed as the pair $(g^x h^r, g^r)$, where $r \in \mathbb{Z}_p$ is chosen at random. Given a ciphertext (c_1, c_2) the original plaintext can be recovered by exhaustive search, like in DGK, on the value $\frac{c_1}{c_2^a}$. The final cryptosystem is additively homomorphic.

The use of EC allows to obtain a great bandwidth saving, indeed, exploiting the point compression [17], the ciphertext can be transmitted using $2 \cdot (2t + 1)$ bits. For example, for a security parameter $t = 80$, the ciphertext sizes for the considered cryptosystems would be: Paillier 2048 bits, DGK [14] 1024 bits and EC-ElGamal 322 bits. On the other side, the use of EC usually requires slightly more complex computations. These are the same EC groups exploited in the efficient OT implementations used in the GC-based protocols [1].