Fast and Accurate Likelihood Ratio-Based Biometric Verification Secure Against Malicious Adversaries

Amina Bassit, Florian Hahn, Joep Peeters, Tom Kevenaar, Raymond Veldhuis, *Senior Member, IEEE*, and Andreas Peter

***Abstract*— Biometric verification has been widely deployed in current authentication solutions as it proves the physical presence of individuals. Several solutions have been developed to protect the sensitive biometric data in such systems that provide security against honest-but-curious (a.k.a. semi-honest) attackers. However, in practice, attackers typically do not act honestly and multiple studies have shown severe biometric information leakage in such honest-but-curious solutions when considering dishonest, malicious attackers. In this paper, we propose a provably secure biometric verification protocol to withstand malicious attackers and prevent biometric data from any leakage. The proposed protocol is based on a homomorphically encrypted log likelihood-ratio (HELR) classifier that supports any biometric modality (e.g., face, fingerprint, dynamic signature, etc.) encoded as a fixed-length real-valued feature vector. The HELR classifier performs an accurate and fast biometric recognition. Furthermore, our protocol, which is secure against malicious adversaries, is designed from a protocol secure against semi-honest adversaries enhanced by zero-knowledge proofs. We evaluate both protocols for various security levels and record a sub-second speed (between 0.37s and 0.88s) for the protocol secure against semi-honest adversaries and between 0.95s and 2.50s for the protocol secure against malicious adversaries.**

Manuscript received April 14, 2021; revised August 13, 2021 and October 13, 2021; accepted October 15, 2021. Date of publication October 26, 2021; date of current version November 5, 2021. This work was supported in part by the Dutch Research Council [Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO)] and GenKey Netherlands B.V. for the Research Programme Kennis Innovatie Mapping (KIEM) under Project ENPPS.KIEM.018.001 and in part by the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie Grant under Agreement 860315 for PriMa Project. The associate editor coordinating the review of this manuscript and approving it for publication was Mr. Frederik Armknecht. *(Corresponding author: Amina Bassit.)*

Amina Bassit, Florian Hahn, and Joep Peeters are with the Data Management and Biometrics (DMB) Group and Services and CyberSecurity (SCS) Group, University of Twente, 7522 NB Enschede, The Netherlands (e-mail: a.bassit@utwente.nl; f.w.hahn@utwente.nl; joep@jjpeeters.nl).

Tom Kevenaar is with GenKey Netherlands B.V., 5656 AG Eindhoven, The Netherlands (e-mail: tom.kevenaar@genkey.com).

Raymond Veldhuis is with the Data Management and Biometrics (DMB) Group and Services and CyberSecurity (SCS) Group, University of Twente, 7522 NB Enschede, The Netherlands, and also with the Department of Information Security and Communication Technology, Norwegian University of Science and Technology, 2802 Gjøvik, Norway (e-mail: r.n.j.veldhuis@utwente.nl).

Andreas Peter is with the Data Management and Biometrics (DMB) Group and Services and CyberSecurity (SCS) Group, University of Twente, 7522 NB Enschede, The Netherlands, and also with the Computer Science Department, University of Oldenburg, 26129 Oldenburg, Germany (e-mail: a.peter@utwente.nl).

Digital Object Identifier 10.1109/TIFS.2021.3122823

***Index Terms*— Biometric verification, threshold homomorphic encryption, secure two-party computation, semi-honest and malicious models.**

I. Introduction

BIOMETRIC verification plays a pivotal role in current authentication technologies. Through measuring biometric modalities, such as faces, biometric verification provides evidence of the physical presence of individuals. Compared to passwords, PIN codes, and tokens, biometric data is irreversible and cannot be reissued once leaked or compromised. This categorizes it as highly sensitive data that is constantly subject to severe security threats. The major challenges encountered concerning biometric data are its storage and processing that tend to be performed in an unprotected manner. Real-life examples confirm the seriousness of these security threats. In August 2019, [1] reported a biometric data breach in the security platform BioStar2, exposing facial recognition data and fingerprint data of millions of users. In November 2020, [2] reported another biometric data breach in TronicsXchange's AWS S3 Bucket that was left unprotected, leaking approximately 10.000 fingerprints. These incidents show the urgency of protecting biometric data that is Personally Identifiable Information (PII). At the same time, many countries have legislations (e.g., the EU's GDPR) that govern how PII of their citizens should be handled, including the use of strong data protection technologies.

Biometric verification systems (e.g., multi-user access control) involve two protocols: enrollment and verification that include users, a client, and a server as main entities. The client represents the acquisition device, such as a biometric scanner. Its role is to capture the user's biometric reference data during the enrollment and the live *probe* during the verification. The server, on the other hand, stores the biometric reference data together with some auxiliary information in a *template* during the enrollment and compares it with the live probe during the verification. The aim of protecting the biometric data throughout the entire verification process implies secure storage and secure processing, which is achievable via homomorphic encryption. On the one hand, homomorphic encryption offers flexibility in manipulating encrypted data without decryption. However, on the other hand, this same flexibility makes tracking the computations a complicated task, especially when the parties may not be trusted.

From a security point of view in the context of biometrics, the client or the server could be compromised by an attacker

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

who tries not to bypass the authentication but to leak sensitive biometric data that is either stored (the template) or freshly captured (the probe). In an attempt to remain unnoticed, this attacker could either follow the biometric verification protocol as intended or arbitrarily deviate from it by following a specific strategy to achieve his desired adversarial goal, which is not only inferring knowledge about a template or probe but also attacking the protocol correctness using any information gained or injected during the protocol execution. The cryptography literature [3] describes the first type as a *semi-honest* attacker and the second type as a *malicious* attacker. The protection against malicious attackers is more challenging than the protection against semi-honest ones since a malicious attacker deviates from the protocol and hence cannot only access the sent and received messages (as also a semi-honest attacker can) but also alter the sent messages, e.g., by injecting messages, omitting messages or compromising the protocol's computation, with the goal of inferring sensitive information or even altering the protocol's outcome, which threatens its correctness.

The security of state-of-the-art biometric verification systems can be split in three categories: *semi-honest client and semi-honest server* [4]–[8], *malicious client and semi-honest server* [9]–[12] and *malicious client and malicious server* [13], [14]. In the first two categories, the existing systems run relatively fast; however, they show severe biometric information leakage when considering a malicious server as described in [15]. For instance, in the case of the system studied in [16], a malicious server can send encrypted computations of its own choice instead of the ones dictated by the protocol. Although the studied verification protocols in [16] employ encryption schemes based on the ring-LWE problem, this attack enables a server to learn the biometric template in at most $2N - \theta$ queries (where N the bit-length of a biometric template and θ the probe-template comparison threshold). Both [15] and [16] emphasize that a biometric verification assuming a semi-honest server or client puts the biometric data in peril. Although biometric authentication systems are vulnerable to hill-climbing and brute-force attacks [17], [18] which, in practice, are mitigated by limiting the number of authentication attempts per user. However, restraining the client/server (the party who owns the access disclosure right) to learn only the minimal functionality, that is access is granted or denied, is crucial. This motivates security against malicious attackers in the context of biometric authentication, which guarantees the protection of the biometric probes and templates in such a scenario.

Among the solutions that tried to address the problem of both malicious client and malicious server, there is the biometric verification system THRIVE [19]. While their overall protocol is only proven secure against attackers that follow the behavior defined in the semi-honest adversary model, the authors introduce a secret key per user, which they use jointly with the user's biometric probe in a two-factor authentication protocol. This makes it harder for an attacker to act maliciously as he would be required to compromise both factors first. In Section IX, we elaborate on why THRIVE does not achieve security against malicious adversaries. There is also [13] that proposed a continuous authentication three-party protocol secure in the malicious model; however, the template-probe distance is leaked to the server, which makes it vulnerable to hill-climbing and brute-force attacks [17], [18]. The most closely related work to ours is SEMBA [14] that is a client-server multimodal biometric verification protocol achieving security against malicious adversaries by using SPDZ [20], [21] that follows the offline/online paradigm. SEMBA was evaluated with an unrealistically low-security strength of 46 bits (while we consider security strengths of at least 96 bits), for which it runs nearly one order of magnitude faster than our protocol, between 0.109s and 0.120s,[1] but achieves a biometric performance lower than ours; EER between 0.98% and 1.15% from the fusion of iris features of EER between 2.51% and 2.08% and eigenfaces of EER 17.37%. For instance, in the case of faces, they achieve 17.37% while we achieve 0.27% EER. In Section VIII, we go into more detail on SEMBA and also explain its high storage requirements on the client-side as well as the implications of having offline/online phases in the authentication setting.

In this paper, we propose a practical biometric verification protocol that achieves both security in the malicious model and a low EER. We adopt the data-driven biometric recognition approach based on the log likelihood ratio (LLR) classifier [22], known for its optimality in the Neyman-Pearson sense. Our approach, called *homomorphically encrypted log likelihood-ratio* (HELR) classifier, allows us to speed up the biometric recognition by pre-computing the classifier and storing it into lookup tables. Thus when applying an encryption layer, the recognition performance does not degrade compared to the unprotected classifier. Our HELR classifier supports any biometric modality encoded as a fixed-length real-valued feature vector (such as faces) and does not support the one encoded as a binary-valued feature vector such as irises. Based on our accurate (EER between 0.25% and 0.27% for faces) HELR classifier, we first present a fast (between 0.37s and 0.88s depending on the desired bit security level) biometric verification protocol secure against a semi-honest client and server. Then we address the above-mentioned problem by proposing a practical (between 0.95s and 2.50s depending on the desired bit security level) biometric verification protocol secure against both a *malicious client and malicious server*. The template is encrypted using threshold homomorphic encryption (THE) such that neither the client nor the server can decrypt it on its own. The probe is encrypted by the client only using homomorphic encryption (HE). Encryption alone guarantees neither the outcome correctness nor the security in the presence of malicious adversaries. Therefore, we force the client and the server to follow our semi-honest construction protocol's steps by using zero-knowledge proofs (ZK-proofs) to check and keep track of the computations. To realize this, integer-oriented THE and HE schemes with compatible ZK-proofs are required. We use the additive homomorphic ElGamal encryption scheme and adapt three sigma protocols to suit our construction. The proposed protocol protects biometric information from leakage in the presence of malicious adversaries. Also, it imposes on both the client and server to follow the protocol honestly; if one of them tries to misbehave, the other entity will detect it and terminate the protocol (in the cryptography literature, this is called security with abort).

In summary, we make the following contributions:

- We introduce the HELR lookup tables that speed up and simplify the biometric recognition reducing it to

[1]These are the online phase runtime of one authentication attempt on different template sizes. However, the non-reusability of the preprocessed data makes the runtime of the offline phase necessary to be included in the overall runtime to fairly assess the efficiency of such a solution in practice.

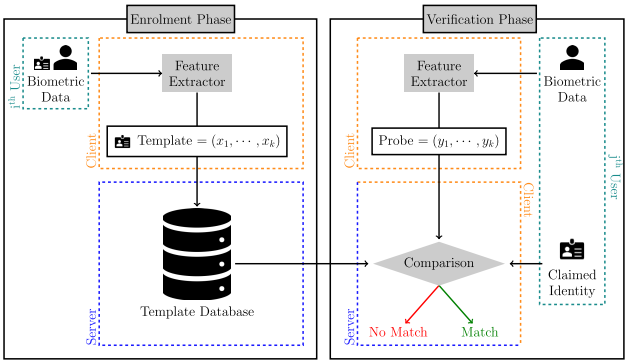

Fig. 1. Overview of a multi-user biometric verification system where the comparison is performed between the client and the server. Note that $(x_1, \cdots, x_k)$ and $(y_1, \cdots, y_k)$ represent the feature vectors of the template and a probe.

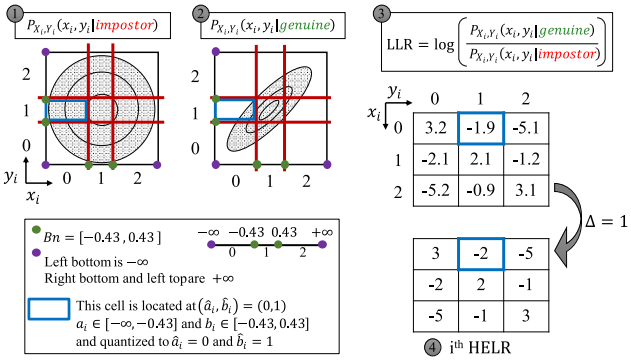

Fig. 2. Generation of the i^{th} HELR lookup table using a feature level $n = 3$. First the red $n \times n$ grid equiprobably partitions the impostor PDF with respect to B_n along both axes. Then the same grid is applied on the genuine PDF. Subsequently, the LLR is computed then quantized using $\Delta = 1$ and stored in the i^{th} HELR table. The blue cell is an example of a sample $(a_i, b_i) = (-0.25, 0.31)$.

three elementary operations (i.e., selection, addition, and comparison), paving the way for applying an encryption layer over these operations without degrading the biometric accuracy.
- We design two biometric verification protocols that perform the recognition under encryption preventing biometric information (i.e., template, probe, and score) from leakage in the presence of semi-honest and malicious adversaries.
- We prove the security of our protocols, evaluate their computational performance, and show that we achieve practical efficiency for widely accepted security levels.

II. Preliminaries

In this work, we denote by $\boldsymbol{x} = (x_1, \cdots, x_k)$ a $k-$dimensional feature vector, $\boldsymbol{X} = (X_1, \cdots, X_k)$ its corresponding multivariate random variable (from which the features are sampled), $f_{\boldsymbol{X}}$ its corresponding probability density function (PDF) and $f_{\boldsymbol{X},\boldsymbol{Y}}$ the joint probability density function of $\boldsymbol{X}$ and $\boldsymbol{Y}$.

A. Biometric Background

1) Overview: Biometric verification systems check the authenticity of a claimed user identity by exploiting that biometric traits discriminatively characterize individuals. Figure 1 depicts the main phases of those systems. The captured biometric raw measurement goes through a feature extraction step to yield a feature vector. During the enrollment phase, the extracted feature vector represents the *template* and is stored along with the user's identity in the system's database. Later in the verification phase, a user claims to have a certain identity. The extracted feature vector, in this phase, is called a *probe*. Subsequently, the system compares the template obtained in the enrollment with the probe and measures the similarity between both feature vectors as a score. In case the score exceeds a preset threshold θ, the system considers the user genuine and outputs *match*; otherwise, it considers the user an impostor and outputs *no match*.

2) Log Likelihood Ratio Classifier: In [22], the authors show the optimality in the Neyman-Pearson sense of the log likelihood ratio as a similarity measure comparing two fixed-length feature vectors (one representing the template and the other representing the probe). The decision comparison is made based on the hypothesis that the system is dealing with the same person (i.e., *genuine verification*) versus the hypothesis that it is dealing with a different person (i.e., *impostor verification*). Consider $\boldsymbol{x}$ (resp. $\boldsymbol{y}$) a biometric feature vector from the enrollment (resp. verification) and $\boldsymbol{X}$ (resp. $\boldsymbol{Y}$) its corresponding multivariate random variable where their X_i (resp. Y_i) are assumed to be independent and normally distributed. For the i^{th} feature, the distribution of the genuine verification is defined by

$$P_{X_i,Y_i}(x_i, y_i|\text{gen}) = f_{X_i,Y_i}(x_i, y_i) \tag{1}$$

a cigar-shaped 2D Gaussian distribution (see Figure 2), whereas the distribution of the impostor verification is defined by

$$P_{X_i,Y_i}(x_i, y_i|\text{imp}) = f_{X_i}(x_i) \cdot f_{Y_i}(y_i) \tag{2}$$

a circular-shaped 2D Gaussian distribution (see Figure 2). The similarity between x_i and y_i is measured by calculating the log likelihood ratio (LLR) score from these distributions.

$$s(x_i, y_i) = \log\left(\frac{P_{X_i,Y_i}(x_i, y_i|\text{gen})}{P_{X_i,Y_i}(x_i, y_i|\text{imp})}\right) \tag{3}$$

The LLR classifier is based on the data-driven approach since it requires the knowledge of f_{X_i}, f_{Y_i} and f_{X_i,Y_i} that are in practice estimated from a dataset representative of the relevant population. Also, this approach assumes that the features were extracted in a statistically independent and identically distributed (i.i.d.) manner. In practice, this can be achieved by applying a combination of principal component analysis (PCA) and linear discriminant analysis (LDA) as in [22] and [23]. As a consequence, the final similarity score between two feature vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by the sum of the individual LLR scores $s(x_i, y_i)$ since the independence between features is assumed.

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{k} s(x_i, y_i) \tag{4}$$

The verification system defines a threshold θ based on which only the final scores that are above θ are counted as a *match* whereas those below are counted as a *no match*.

3) Performance Assessment: The performance of biometric verification systems is tightly related to the performance of their core comparison algorithm (called comparator). It is expressed in terms of False Non-Match Rate (FNMR), that is,

the probability that the comparator decides no match for two samples coming from the same individual; and False Match Rate (FMR) that is the probability that the comparator decides match for two samples coming from two different individuals. An infinitely high threshold θ results in FNMR $= 1$ and FMR $= 0$, lowering the threshold decreases (increases) the FNMR (FMR), respectively towards an infinitely low threshold θ for which FNMR $= 0$ and FMR $= 1$. This trade-off can be graphically illustrated by a decision error trade-off (DET) curve representing the FNMR as a function of the FMR. The Equal Error Rate (EER) denotes the point on the curve where FNMR and FMR are equal. FNMR@FMR $= 0.1\%$ denotes the point on the curve where FMR$= 0.1\%$. The system's threshold θ is set to meet an amount of acceptable FMR, often at FMR $= 1\%$ or 0.1%.

B. Additively Homomorphic ElGamal

We briefly recall the additively homomorphic ElGamal and its $(2, 2)$-threshold version [24]. Let q be a large prime, $\mathbb{G}$ a group of order q and generator g. Let $k = g^s$ be the public key corresponding to the private key s. The encryption of a message $m \in \mathbb{Z}_q$ is $[m] \stackrel{\text{def}}{=} (g^r, g^m \cdot k^r)$ where $r \in \mathbb{Z}_q$ is random. The decryption of the ciphertext $[m]$ is the discrete log of $g^m \cdot k^r \cdot (g^r)^{-s}$. The additively homomorphic ElGamal is secure against Indistinguishable Chosen-Plaintext Attack (IND-CPA) [24] under the Decisional Diffie-Hellman (DDH) assumption. It supports the following operations: ciphertext multiplication $[m_1] \cdot [m_2] = [m_1 + m_2]$, re-randomizing with randomness r_0: $[m] \cdot [0] = \left(g^{r+r_0}, g^{m+0} \cdot k^{r+r_0}\right) = [m]$, blinding with blinding value r_d: $[m]^{r_d} = (g^{r_d \cdot r}, g^{r_d \cdot m} \cdot k^{r_d \cdot r}) = [r_d \cdot m]$ and subtraction of two ciphertexts $[m_1] - [m_2] \stackrel{\text{def}}{=} [m_1] \cdot [m_2]^{-1} = [m_1 - m_2]$.

For implementing a $(2, 2)$-threshold additively homomorphic ElGamal, we use the technique described in [25]. Given the key pair (pk_i, sk_i) for $i \in \{1, 2\}$ such that $pk_i = g^{sk_i}$; $pk_{joint} \stackrel{\text{def}}{=} pk_1 \cdot pk_2$ is the joint public key. The encryption is performed under pk_{joint} and denoted as $[\![m]\!] \stackrel{\text{def}}{=} (g^r, g^m \cdot pk_{joint}^r)$. The decryption of $[\![m]\!]$ comes into two stages. First, each party i, using its private key sk_i, produces a partial decryption $[m]_i \stackrel{\text{def}}{=} (g^r, g^m \cdot pk_{joint}^r \cdot (g^r)^{-sk_i})$. Then by combining the exchanged partial decryptions, the final decryption is the discrete log of $g^m \cdot pk_{joint}^r \cdot (g^r)^{-sk_1} \cdot (g^r)^{-sk_2}$. Notice here that a partial decryption $[m]_1$ is a non-threshold ElGamal ciphertext encrypted under pk_2 and vice versa for $[m]_2$. Threshold ElGamal is also IND-CPA secure [26] under the DDH assumption and supports all the operations mentioned above.

C. Zero-Knowledge Proofs

ZK-proofs allow proving a statement without revealing the secret. In the literature [27], ZK-proofs constructed from Σ-protocols are efficient and flexible to fit the desired proof. It is also possible to combine them to prove the conjunction of several statements (called AND proofs). In Table I, we recall, from [28], the Σ-protocol, denoted as Σ_{Plain}, that proves the plaintext knowledge of an ElGamal ciphertext $[m] = (u, v)$. It can be enhanced by the generic construction in [27] to transform it into a zero-knowledge proof of knowledge; that we denote as $\text{ZKPoK}_{\text{Plain}}$ and use later in Protocol Figure 4. In the same protocol, we also use non-interactive ZK-proofs constructed from Σ-protocols using the Fiat-Shamir transformation [29].

TABLE I

Σ_{PLAIN} PROTOCOL THAT PROVES THE PLAINTEXT KNOWLEDGE OF AN ELGAMAL CIPHERTEXT $[m] = (u, v)$

Commitment	Challenge	Response	Verification
$U = g^{r'}$ $V = g^{m'} \cdot k^{r'}$	$e \in \{0,1\}^t$	$z_r = r' + e \cdot r$ $z_m = m' + e \cdot m$	$g^{z_r} =^? U \cdot u^e$ $g^{z_m} =^? V \cdot v^e$

III. SECURITY MODEL

In this work, we follow Canetti's security model [30] for malicious static adversaries in the special case of two parties. We use the notations and extensions from [31] where each party P_i receives a secret input $x_i^{(s)}$ and a public input $x_i^{(p)}$ and returns a secret output $y_i^{(s)}$ and a public output $y_i^{(p)}$. Also, the adversary receives the public input and output of all parties.

A. Real-World Model

Let π be a two-party protocol. Let $\vec{x} = \left(x_1^{(s)}, x_1^{(p)}, x_2^{(s)}, x_2^{(p)}\right)$ be the parties' inputs, $\vec{r} = (r_1, r_2, r_{\mathcal{A}})$ be the parties' and the adversary's $\mathcal{A}$ random inputs and $a \in \{0, 1\}^*$ be the adversary's auxiliary input. We assume that only one party, P_j where $j \in \{1, 2\}$, is corrupted at the time. $\text{ADVR}_{\pi,\mathcal{A}}(k, \vec{x}, \{j\}, a, \vec{r})$ denotes the output of the adversary and $\text{EXEC}_{\pi,\mathcal{A}}(k, \vec{x}, \{j\}, a, \vec{r})_i$ the output of the party P_i after a real-world execution of π in the presence of the adversary $\mathcal{A}$ corrupting the party P_j.

$$\begin{aligned}\text{EXEC}_{\pi,\mathcal{A}}(k, \vec{x}, \{j\}, a, \vec{r}) = (&\text{ADVR}_{\pi,\mathcal{A}}(k, \vec{x}, \{j\}, a, \vec{r}),\\ &\text{EXEC}_{\pi,\mathcal{A}}(k, \vec{x}, \{j\}, a, \vec{r})_1,\\ &\text{EXEC}_{\pi,\mathcal{A}}(k, \vec{x}, \{j\}, a, \vec{r})_2)\end{aligned}$$

$\text{EXEC}_{\pi,\mathcal{A}}(k, \vec{x}, \{j\}, a)$ denotes the random variable $\text{EXEC}_{\pi,\mathcal{A}}(k, \vec{x}, \{j\}, a, \vec{r})$ where $\vec{r}$ is chosen uniformly random and $\text{EXEC}_{\pi,\mathcal{A}} = \left\{\text{EXEC}_{\pi,\mathcal{A}}(k, \vec{x}, \{j\}, a)\right\}_{k,\vec{x},\{j\},a}$ the distribution ensemble indexed by the security parameter $k \in \mathbb{N}$, the input $\vec{x}$, the corrupted party P_j and the auxiliary input a.

B. Ideal Model

Let f be a probabilistic two-party function computable in PPT defined as $f\left(k, x_1^{(s)}, x_1^{(p)}, x_2^{(s)}, x_2^{(p)}, r\right) = \left(y_1^{(s)}, y_1^{(p)}, y_2^{(s)}, y_2^{(p)}\right)$ where k is the security parameter and r is the random input. In the ideal model, each party P_i sends its input $\left(x_i^{(s)}, x_i^{(p)}\right)$ to the trusted party $\mathbb{T}$ that computes f on the inputs and a uniformly chosen random r then returns to each party P_i its output $\left(y_i^{(s)}, y_i^{(p)}\right)$. Note that the malicious static adversary $\mathcal{S}_{\mathcal{A}}$, operating in the ideal execution, uses the real-world adversary $\mathcal{A}$, which corrupts the party P_j, as a subroutine. At the beginning of the execution, $\mathcal{S}_{\mathcal{A}}$ sees the public values of both parties and the secret values of the corrupted party P_j and also substitutes P_j's input by values of his choice. Again, we denote by $\text{IDEAL}_{f,\mathcal{S}_{\mathcal{A}}}(k, \vec{x}, \{j\}, a, \vec{r})_i$ the output of the party P_i after an ideal execution of f in the presence of the adversary $\mathcal{S}_{\mathcal{A}}$.

$$\begin{aligned}\text{IDEAL}_{f,\mathcal{S}_{\mathcal{A}}}(k, \vec{x}, \{j\}, a, \vec{r}) = (&\text{ADVR}_{f,\mathcal{S}_{\mathcal{A}}}(k, \vec{x}, \{j\}, a, \vec{r}),\\ &\text{IDEAL}_{f,\mathcal{S}_{\mathcal{A}}}(k, \vec{x}, \{j\}, a, \vec{r})_1,\\ &\text{IDEAL}_{f,\mathcal{S}_{\mathcal{A}}}(k, \vec{x}, \{j\}, a, \vec{r})_2)\end{aligned}$$

We denote by $\text{IDEAL}_{f,\mathcal{S}_\mathcal{A}}(k,\vec{x},\{j\},a)$ the random variable $\text{IDEAL}_{f,\mathcal{S}_\mathcal{A}}(k,\vec{x},\{j\},a,\vec{r})$ where $\vec{r}$ is chosen uniformly random and $\text{IDEAL}_{f,\mathcal{S}_\mathcal{A}} = \left\{\text{IDEAL}_{f,\mathcal{S}_\mathcal{A}}(k,\vec{x},\{j\},a)\right\}_{k,\vec{x},\{j\},a}$ the distribution ensemble indexed by the security parameter $k \in \mathbb{N}$, the input $\vec{x}$, the corrupted party P_j and the auxiliary input a.

C. Hybrid Model

In the $(g_1,\cdots,g_l)$-Hybrid model, the execution of a protocol π proceeds as in the real-world model, except that the parties have access to a trusted party $\mathbb{T}$ for evaluating the two-party functions $g_1,\cdots,g_l$. These ideal evaluations proceed as in the ideal-model. As above, we define the following distribution ensemble

$$\text{EXEC}^{g_1,\cdots,g_l}_{\pi,\mathcal{A}} = \left\{\text{EXEC}^{g_1,\cdots,g_l}_{\pi,\mathcal{A}}(k,\vec{x},\{j\},a)\right\}_{k,\vec{x},\{j\},a}$$

The security in this model is defined by requiring that a real-world execution or $(g_1,\cdots,g_l)$-Hybrid execution of a protocol π for computing a function f should reveal no more information to the adversary than what the ideal evaluation of f does, namely the output.

Definition 1: Let f be a two-party function and π be a two-party protocol. We say that π securely evaluates f in the $(g_1,\cdots,g_l)$-Hybrid model if for any malicious static $(g_1,\cdots,g_l)$-Hybrid adversary $\mathcal{A}$ corrupting one party, there exists an adversary $\mathcal{S}_\mathcal{A}$ operating in the ideal world such that

$$IDEAL_{f,\mathcal{S}_\mathcal{A}} \overset{c}{\approx} EXEC^{g_1,\cdots,g_l}_{\pi,\mathcal{A}}$$

where $\overset{c}{\approx}$ means the computational indistinguishability of ensembles, see Definition 3 in [30].

IV. Pre-Computed HELR Classifier

Recall from Section II-A the biometric comparison of two feature vectors using the LLR classifier is a data-driven approach where the parameters of the genuine and impostor distributions are estimated from a given dataset representative of the relevant population. For each feature, we draw the PDFs in Equation (1) and Equation (2) as estimated from the training dataset. Assuming that the extracted features follow the Gaussian distribution, they are rendered i.i.d. by applying a combination of PCA and LDA as shown in [22] and [23]. Then we compute the LLR per feature as in Equation (3) in order to produce the final score in Equation (4) for each comparison.

The LLR in Equation (3) can be visualized as a function with two inputs (the i$^{\text{th}}$ feature, one from the template and one from a probe) and one output (individual score). This function can be arranged into a lookup table where the rows' indexes represent the possible values of the first input (features from the template), the columns' indexes represent the possible values of the second input (features from a probe) and the cells contain the output (individual scores). In order to produce such a lookup table, a mapping from a continuous domain to a finite set is needed to limit the possible feature values allowing the storage of a representative score per cell. This respective score is then quantized to an integer in order to facilitate the application of homomorphic encryption. An example of generating the HELR lookup table of one feature is given in Figure 2.

A. Feature Quantization

We describe the feature quantization procedure for the i$^{\text{th}}$ feature; the same is applied for the remaining features. Assuming the PDFs are zero-mean, which is achievable by subtracting the mean, implies that the impostor PDF has a unit variance. Recall, that X_i and Y_i are normally distributed, hence we get $X_i \sim \mathcal{N}(0,1)$ and $Y_i \sim \mathcal{N}(0,1)$. To perform a feature quantization on n levels (called *feature level*), we divide the 2D impostor PDF in an equiprobable manner so that all bins will have the same probability; thus, an arbitrary feature observation is just as likely to land on any of those bins. This is done by determining the bins' borders following Algorithm 1 where $\text{ICDF}(p,0,1)$ is the inverse cumulative distribution function of a $\mathcal{N}(0,1)$ at the cumulative probability p and it returns the value associated with p.

Algorithm 1 Procedure to Determine the Bins' Borders

Input: n feature quantization level
Output: Bn array containing the bins' borders
Bn array of size $n-1$;
for $j \leftarrow 1$ **to** $n-1$ **do**
  $p = j/n$;
  $Bn[j] = \text{ICDF}(p,0,1)$;
end

$a_i \in Bn_{a_i}$ (resp. $b_i \in Bn_{b_i}$) denotes the measured value for feature x_i (resp y_i) from the first (resp. second) sample and is quantized to $\hat{a_i}$ (resp. $\hat{b_i}$) following Algorithm 2 using the same Bn bins' borders array of the i$^{\text{th}}$ feature.

Algorithm 2 Feature Quantization on n Feature Levels

Input: a_i raw feature value of the i$^{\text{th}}$ feature and Bn array containing the bins' borders of the i$^{\text{th}}$ feature
Output: $\hat{a_i}$ quantized value
for $j \leftarrow 1$ **to** $n-1$ **do**
  if $a_i < Bn[j]$ **then return** $j-1$
end
return $n-1$

B. Score Quantization

As we are dealing with 2D distributions, we partition the impostor PDF and the genuine PDF according to an $n \times n$ grid using Bn for both axes x_i and y_i; see the red grid Figure 2. For a cell located at $(\hat{a}_i,\hat{b}_i)$, where a_i and b_i the measured values, we compute the genuine probability distribution (see Equation (5)) inside that cell by calculating the area under the curve delimited by its borders Bn_{a_i} and Bn_{b_i}, see the dotted surface depicted in Figure 2. Based on Equation (1) we hence get:

$$P_{X_i,Y_i}(\hat{a}_i,\hat{b}_i|\text{gen}) = \int_{Bn_{b_i}}\int_{Bn_{a_i}} f_{X_i,Y_i}(x_i,y_i)\,dx_i\,dy_i \quad (5)$$

Note that Bn_{a_i} and Bn_{b_i} have one of the three forms $]-\infty, Bn[1][$, $[Bn[j], Bn[j+1][$ or $[Bn[n-1], +\infty[$. For the impostor distribution, all cells have identical probability since it was equiprobably divided:

$$P_{X_i,Y_i}(\hat{a}_i,\hat{b}_i|\text{imp}) = \frac{1}{n\times n} \quad (6)$$

After that, we calculate the LLR for that cell $(\hat{a}_i, \hat{b}_i)$ and place the resulted non-quantized score in a lookup table at row $\hat{a}_i$ and column $\hat{b}_i$. As described in Section II-B, homomorphic encryption requires integers and the resulted LLR scores are real-valued. We perform a second quantization to map each real-valued non-quantized score to an integer that we call *quantized score* by dividing the real-valued score by a quantization step Δ and rounding the result to the nearest integer to yield the quantized score $s(\hat{a}_i, \hat{b}_i)$.

C. HELR Lookup Tables

For each feature, we generate an $n \times n$ HELR lookup table where its cells contain the quantized score resulted from a row (resp. column) that refers to the quantized feature value of the first (resp. second) feature vector. To calculate the similarity score of two feature vectors $\boldsymbol{a} = (a_1, \cdots, a_k)$ and $\boldsymbol{b} = (b_1, \cdots, b_k)$ using the HELR tables, we map each feature a_i (resp. b_i) to its quantized value $\hat{a}_i$ (resp. $\hat{b}_i$) and select its corresponding score $s(\hat{a}_i, \hat{b}_i)$ from the i$^{\text{th}}$ HELR table, the value at location row $\hat{a}_i$ and column $\hat{b}_i$. Based on Equation (4) we calculate the final score as $S = \sum_{i=1}^{k} s(\hat{a}_i, \hat{b}_i)$. Recall that we can sum the individual scores since the features are assumed to be independent. The dimension of an HELR table is bounded by the Signal-to-Noise ratio, or within-subject variation, under the Gaussian assumption [32] which depends on how much identity information the features carry. Note that this dimension impacts only the template size that is formed by choosing one row per table. Since the HELR lookup tables are generated from a dataset representative of the relevant population, we assume that they are public and accessible by any party, namely the client and the server. The comparison outcome is determined by the final score that is calculated using sensitive biometric data. Thus, all values that are involved in this calculation are sensitive. Hence, row and column positions as well as the individual scores and the final score must be protected. In the following sections, we aim to perform biometric verification under encryption using the HELR tables where the biometric data is kept encrypted throughout the process and only the comparison outcome (match or no match) is revealed.

V. Proposed Verification Protocols

Our final goal is to achieve security against both malicious client and malicious server. We first design a biometric verification protocol that ensures zero-biometric information leakage secure against semi-honest client and server. Then, we modify this construction to force them to behave honestly. Thus, we obtain a protocol secure against malicious client and server, ensuring both the correctness of the comparison outcome and zero-biometric information leakage. In both scenarios, the server must not learn the probe, the unprotected template, the individual scores, the final score, and the comparison outcome. The same requirements apply to the client except for the probe and the comparison outcome that it should be able to learn.

In the following, we suppose that the client and the server respectively hold the key pairs (pk_{clt}, sk_{clt}) and (pk_{ser}, sk_{ser}) from which the threshold ElGamal public key pk_{joint} is calculated; see Section II-B. The HELR lookup tables, the comparison threshold θ, and the maximum score $S_{\max}$ are public knowledge. We assume that the initial enrollment process is performed in a fully controlled environment.

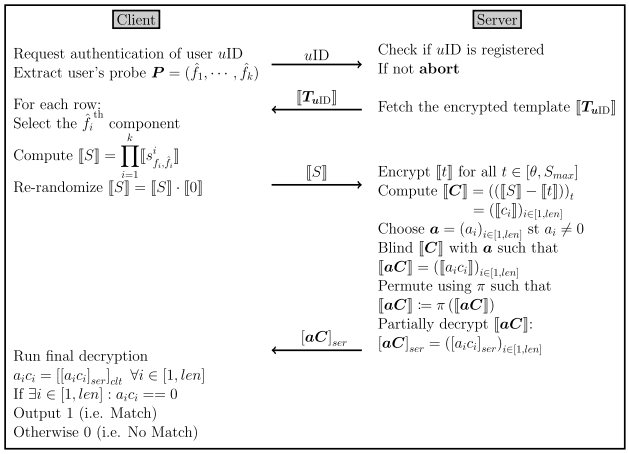

Fig. 3. Biometric Verification Protocol Secure Against Semi-Honest Adversaries.

A. Protocol Secure Against Semi-Honest

Prior to a biometric verification, a user should enroll in order to register his template. In this semi-honest construction, a new user uID presents his biometric modality to the client that first extracts a k-dimensional feature vector $(f_1, \cdots, f_k)$. Then for the i-th feature, the client selects the f_i-th row from the i-th HELR lookup table to form the user's template $\boldsymbol{T}_{u\text{ID}}$; which can be seen as a vector of vectors.

$$\boldsymbol{T}_{u\text{ID}} = \left(\left(s^i_{f_i,j} \right)_{j\in[1,n]} \right)_{i\in[1,k]} \tag{7}$$

where $s^i_{f_i,j}$ is the score at the intersection of row f_i and column j from the HELR lookup table i. Finally, the client encrypts $\boldsymbol{T}_{u\text{ID}}$ using pk_{joint} then along with uID sends the encrypted template $[\![\boldsymbol{T}_{u\text{ID}}]\!]$, that is of size $n \cdot k$ ciphertexts, to the server who stores them for later retrieval.

$$[\![\boldsymbol{T}_{u\text{ID}}]\!] = \left(\left([\![s^i_{f_i,j}]\!] \right)_{j\in[1,n]} \right)_{i\in[1,k]} \tag{8}$$

Figure 3 describes our verification protocol secure against semi-honest adversaries where the veracity of a user claiming an identity uID is assessed. After extracting a k-dimensional feature vector $\boldsymbol{P}$ from the acquired live biometric modality, the client requests, from the server, the user's corresponding encrypted template $[\![\boldsymbol{T}_{u\text{ID}}]\!]$. Recall, from Section II-B, that in the additively homomorphic ElGamal, multiplication under encryption is equivalent to addition in the plain domain. That is, the client selects and multiplies the encrypted individual scores to form the final score $[\![S]\!]$ then re-randomizes it before sending it back to the server. This re-randomization prevents the server from guessing the selected individual scores from the encrypted template since the multiplication of two ciphertexts twice yields the same ciphertext.

The server needs to perform the comparison under encryption to determine whether the encrypted final score is above or below the threshold θ. One would think of subtracting only $[\![\theta]\!]$ from $[\![S]\!]$; however, this yields a zero only when both are equal and non-zero value when S is above or below. To solve this, one should compare S with the integers between θ and $S_{\max}$. If $\theta \leq S$ then exactly one single subtraction results in 0, and if $S < \theta$ then all the subtractions are unequal to 0. In addition, a multiplicative blinding must be applied to these subtractions to protect the final score S when a decryption

occurs. Note that the multiplicative blinding preserves only the 0 in contrast with the values unequal to 0, which become random.

Applying this, the server encrypts the integers between θ and $S_{\max}$ under pk_{joint} then computes the blinded comparison vector $[\![\boldsymbol{aC}]\!]$ from the comparison vector $[\![\boldsymbol{aC}]\!]$ and $\boldsymbol{a}$ a vector of random values. After that, it applies a random permutation π to $[\![\boldsymbol{aC}]\!]$ and partially decrypts the blinded-permuted comparison vector and sends $[\boldsymbol{aC}]_{ser}$. The client runs the final decryption on $[\boldsymbol{aC}]_{ser}$ to retrieve the plain values of the blinded-permuted comparison vector $\boldsymbol{aC}$. If it finds a value $a_i c_i == 0$ in $\boldsymbol{aC}$ that means S is equal or greater than the threshold θ. In this case, the client outputs *match*. If all values in $\boldsymbol{aC}$ are different from zero, that means S is strictly below the threshold θ. In this case, the client outputs *no match*.

Protocol Figure 3 requires two rounds and a communication complexity of $O(n{\cdot}k){+}O(len){+}1$. For the local computational complexity, the client requires $O(k) + O(len)$ and the server requires $O(len)$, where $len = S_{\max} - \theta + 1$, k features and n feature levels.

Limitations: This verification protocol is suitable for a client and a server that trust each other regarding the correctness of the exchanged messages. However, in an untrusted setting, this protocol leads to serious security threats. Consider the case of a malicious client that can arbitrarily deviate from the protocol. Since the system's threshold θ is public, it can encrypt θ using the joint public key pk_{joint} to receive a blinded-permuted comparison vector $\boldsymbol{aC}$ that contains a zero; thus, it succeeds in forcing a *match*. For the case of a malicious server, instead of sending the actual encrypted template, it can craft a template of the form $((0,\cdots,0),\cdots,(1,\cdots,n),\cdots,(0,\cdots,0))$, fixing the individual scores of the i$^{\text{th}}$ feature to $(1,\cdots,n)$ and the individual scores of the remaining features to zero. He then encrypts the crafted template with his public key pk_{ser} and sends it to the client, who will send back a sum of the encrypted individual scores. By decrypting the sum, the server learns the value of the i$^{\text{th}}$ component of the probe. Repeating this for all k components reveals the probe. Another attack could be: instead of sending the partial decryption of the permuted comparison vector $\boldsymbol{aC}$, it sends a non-zero values vector of the same length as $\boldsymbol{aC}$ and encrypted with the client's public key pk_{clt}. Thus it forces a *no match* to a genuine user since the decryption of $\boldsymbol{aC}$ yields a non-zero vector.

B. Protocol Secure Against Malicious

To address the limitations mentioned in Section V-A, we transform the construction in Figure 3 into a protocol secure against both malicious client and malicious server using adapted ZK-proofs.

Adapted ZK-Proofs: In order to check and track the correctness of the computations over ElGamal encrypted data, we construct three ZK-proofs from three Σ-protocols and provide their proofs in Appendix A. Table II presents the corresponding non-interactive ZK-proofs using the Fiat-Shamir transformation where $\mathcal{H} : \mathbb{G} \rightarrow \{0,1\}^t$ is a hash function. In our construction, we use an AND proof of ZK-proofs, as shown in [27], using the same challenge for all individual ZK-proofs.

We introduce a trusted enrollment party called *enrollment server* which is merely involved during the enrollment and offline during the verification. In real-world biometric applications, such a party is needed to guarantee the validity of the identity claim and the quality of biometric templates that are

TABLE II

ADAPTED NON-INTERACTIVE ZERO-KNOWLEDGE PROOFS (NIZKS)

	$\text{NIZK}_{\text{DecZero}}$ for $[m_1]-[m_2]=(u,v)$	$\text{NIZK}_{\text{Blind}}$ for $[m]^r=(u^r,v^r)=(a,b)$	$\text{NIZK}_{\text{Partial}}$ for $[m]_i=(u,v{\cdot}u^{-sk_i})=(u,c)$
Commitment	$X=g^{r'},U=u^{r'}$	$U=u^{r'},V=v^{r'}$	$X=g^{r'},U=u^{r'}$
Challenge	$e=\mathcal{H}(k,X,U)$	$e=\mathcal{H}(k,U,V)$	$e=\mathcal{H}(pk_i,X,U)$
Response	$z=r'+e{\cdot}s$	$z=r'+e{\cdot}r$	$z=r'+e{\cdot}sk_i$
Verification Equations	$e=^?\mathcal{H}(k,X,U)$ $g^z=^?X{\cdot}k^e$ $u^z=^?U{\cdot}v^e$	$e=^?\mathcal{H}(k,U,V)$ $u^z=^?U{\cdot}a^e$ $v^z=^?V{\cdot}b^e$	$e=\mathcal{H}(pk_i,X,U)$ $g^z=^?X{\cdot}pk_i^e$ $u^z=^?U{\cdot}(v{\cdot}c^{-1})^e$

to be stored (in a protected form) on the verification server. To reflect this, we assume that the enrollment server holds a signature key pair (v_{enr}, s_{enr}) to sign the encrypted templates. Unlike the enrollment in our protocol Figure 3, the client has an additional key pair (k_{clt}, s_{clt}) for an additively homomorphic ElGamal encryption and generates k pseudo-random permutations (PRP) $(\pi_1,\cdots,\pi_k)$. The template is formed differently as well. After selecting the corresponding rows as in Equation (7), the client performs the following modifications on Equation (8). The component $[\![s^i_{f_i,j}]\!]$ becomes a vector that contains the encryption, under k_{clt}, of its column position (i.e. $[j]$) and an index $r_{i,j} = \pi_i(j)$ by which this component will be located later. The components of the i$^{\text{th}}$ vector are ordered according to the indexes $r_{i,j}$. Thus $[\![\boldsymbol{T'}_{\boldsymbol{u}\text{ID}}]\!]$ becomes

$$[\![\boldsymbol{T'}_{\boldsymbol{u}\text{ID}}]\!] = \left(\left(r_{i,j}, [j], [\![s^i_{f_i,j}]\!]\right)_{j\in[1,n]}\right)_{i\in[1,k]} \tag{9}$$

The client then sends $[\![\boldsymbol{T'}_{\boldsymbol{u}\text{ID}}]\!]$ to the enrollment server who appends to each component two signatures: $\sigma_{i,j} = \text{Sign}\left(s_{enr}, r_{i,j}, [j], u\text{ID}\right)$ that binds the index $r_{i,j}$ with encryption of the column position j and uID; and $\alpha_{i,j} = \text{Sign}\left(s_{enr}, [j], [\![s^i_{f_i,j}]\!], u\text{ID}\right)$ that binds $[j]$ with $[\![s^i_{f_i,j}]\!]$ and uID. Those signatures ensure the authenticity of the template during the verification. The final protected template (10) is of size $n \cdot k$ components where each of them comprises an index, two ciphertexts, and two signatures.

$$[\![\boldsymbol{T}_{\boldsymbol{u}\text{ID}}]\!] = \left(\left(r_{i,j}, [j], [\![s^i_{f_i,j}]\!], \sigma_{i,j}, \alpha_{i,j}\right)_{j\in[1,n]}\right)_{i\in[1,k]} \tag{10}$$

Besides, the enrollment server generates a permutation π_{thr} to form $\boldsymbol{\Theta}$ the *permuted-encrypted threshold vector* under pk_{joint}.

$$\boldsymbol{\Theta} = ([\![\pi_{thr}(i)]\!])_{i\in[\theta,S_{\max}]} \tag{11}$$

Note that only the enrollment server knows the order of the plain values of $\boldsymbol{\Theta}$. Finally, it sends $\boldsymbol{\Theta}$ to the client and uID, $[\![\boldsymbol{T}_{\boldsymbol{u}\text{ID}}]\!]$ and $\boldsymbol{\Theta}$ to the server that stores them for later retrieval. This template's structure facilitates the transition from a semi-honest construction to a more suitable one that withstands malicious adversaries.

Figure 4 describes our biometric verification protocol secure against malicious adversaries. Unlike in our protocol Figure 3, in step ②, the client sends its probe encrypted $[\boldsymbol{P}]$ and proves the knowledge of the underlying plain probe. He also sends the corresponding indexes $\boldsymbol{R}$ to allow the server to locate the desired components. The server, in step ③, sends the first half of the components to allow the client to prove that the requested components correspond to the ones that appear in the protected template. Once the server is convinced, it then

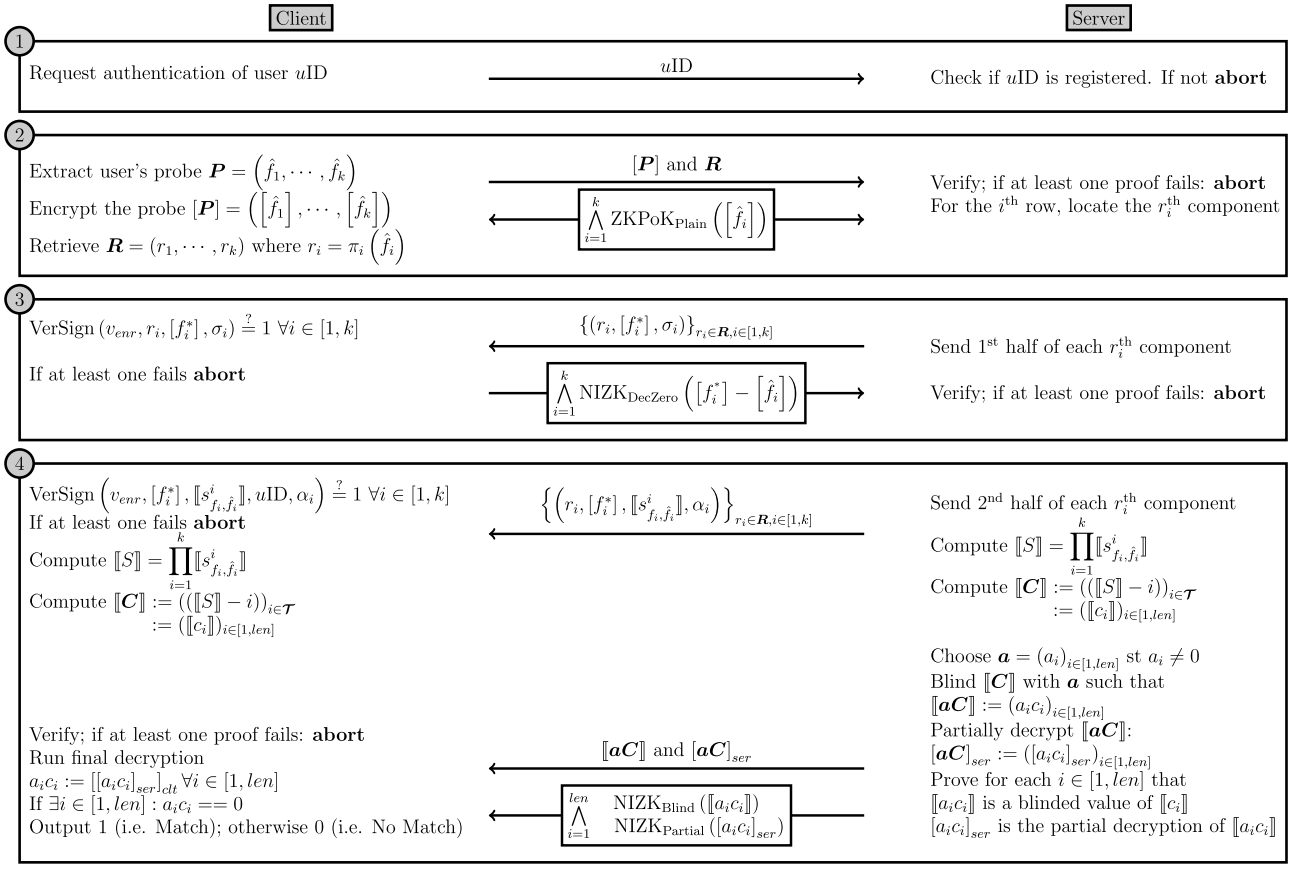

Fig. 4. Biometric Verification Protocol Secure Against Malicious Adversaries.

sends, in step ④, the second half that contains the encrypted scores. Before engaging in any proof, the client checks the authenticity of the received data by verifying the signatures σ_i and α_i, in steps ③ and ④. In step ④, with respect to the same order determined by $\boldsymbol{R}$ as well as $\boldsymbol{\Theta}$, both the client and the server compute $[\![S]\!]$ and $[\![\boldsymbol{C}]\!]$. Note that here they must have the same resulted values with the same randomness. Next only the server blinds the comparison vector $[\![\boldsymbol{C}]\!]$ using a vector of random values $\boldsymbol{a}$ to get $[\![\boldsymbol{aC}]\!]$ and partially decrypts it as $[\boldsymbol{aC}]_{ser}$. He then proves to the client that $[\![\boldsymbol{aC}]\!]$ is a blind version of $[\![\boldsymbol{C}]\!]$ and that $[\boldsymbol{aC}]_{ser}$ is its partial decryption. If the client is convinced, it runs the final decryption and parses the blinded comparison vector $\boldsymbol{aC}$. If there is a zero, it outputs a *match* otherwise outputs a *no match*.

Protocol Figure 4 requires four rounds and a communication complexity of $O(k) + O(len)$. For the local computational complexity, both the client and the server require $O(k) + O(len)$, where $len = S_{\max} - \theta + 1$ and k is the number of features.

VI. Security Analysis

As mentioned in Section V, the enrollment is performed in a fully controlled environment. Therefore, we only discuss the security of verification protocols by separately analyzing the case of compromised client and the case of compromised server. Regardless of noise measurement, the correctness of both protocols is straightforward. For a perfectly captured user's probe, an honest client who is interacting with an honest server will yield a *match* if the user is genuine and a *no match* if the user is an impostor.

A. Security Proof of Protocol 3

1) Semi-Honestly Compromised Client: it receives first the template (8) that is encrypted under threshold ElGamal, which means that the client can not decrypt on its own to learn the scores. However, it may tend to use the public HELR lookup tables and the encrypted template to learn which rows were encrypted. Thanks to the IND-CPA property of ElGamal, the client is unable to link the i$^{\text{th}}$ vector of $[\![\boldsymbol{T}_{u\text{ID}}]\!]$ to any row of the i$^{\text{th}}$ public HELR lookup table. In the second round, the client receives the partial decryption of the comparison vector that was blinded and permuted by the server. Again the client fails in attempting to infer any significant information from the comparison vector. Given that $\boldsymbol{aC}$ was blinded and permuted by the server, the only leaked information here is the comparison outcome. In the case of a match, the client will find a value zero in a random position of the comparison vector. In case of a no match, all values it will obtain are random; thus, in both cases, it can not learn the final score S. Moreover, the client may try to combine both the encrypted $[\![\boldsymbol{T}_{u\text{ID}}]\!]$ and the blinded-permuted comparison vector $\boldsymbol{aC}$; again, nothing can be inferred thanks to the robustness of threshold ElGamal, the applied permutation and the randomness introduced by the blinding.

2) Semi-Honestly Compromised Server: regardless of the user's identity, it receives only one message from the client that is the threshold encrypted final score $[\![S]\!]$, so the server

can not decrypt on its own. Given the fact that $[\![S]\!]$ is only a multiplication of the chosen, accordingly to the probe, encrypted individual scores from the template, the server may try to select some encrypted values from the template and compare their multiplication with the received $[\![S]\!]$. This comparison is meaningless since the client re-randomizes $[\![S]\!]$ before sending it, and thanks to the IND-CPA property of ElGamal, the server is unable to learn any information about the probe.

Both cases demonstrate that no biometric information is leaked except the comparison result, which is the protocol's output. Therefore, our protocol in Figure 3 is secure in the semi-honest model.

B. Security Proof of Protocol 4

Theorem 1: Assume that $\mathcal{H}$ is a collision resistant hash function and the signature scheme is EUF-CMA secure then the protocol in Figure 4 is secure in the ($f_{ZKPoK}^{R_{Plain}}$, $f_{NIZK}^{R_{DecZero}}$, $f_{NIZK}^{R_{Blind}}$, $f_{NIZK}^{R_{Partial}}$)-Hybrid world in the presence of malicious adversary.

Proof: Note that the enrollment server is offline during the verification phase; thus, it is not modeled in this proof.

1) Maliciously Compromised Client: Let $\mathcal{A}$ be an adversary operating in the ($f_{\text{ZKPoK}}^{R_{\text{Plain}}}$, $f_{\text{NIZK}}^{R_{\text{DecZero}}}$, $f_{\text{NIZK}}^{R_{\text{Blind}}}$, $f_{\text{NIZK}}^{R_{\text{Partial}}}$)-Hybrid world that is controlling the client, we construct $\mathcal{S}_{\mathcal{A}}$ an adversary operating in the ideal world that uses $\mathcal{A}$ as a subroutine. $\mathcal{S}_{\mathcal{A}}$ is given the description of $\mathcal{A}$, the probe $\boldsymbol{P}$, the user's identity uID, permuted-encrypted threshold vector $\boldsymbol{\Theta}$ (11), v_{enr}, k_{clt}, pk_{clt}, pk_{joint}, sk_{clt} and k_{clt}. Moreover, $\mathcal{S}_{\mathcal{A}}$ is also given the public input of the server, i.e. the set of templates $\{[\![\boldsymbol{T}_{u\text{ID}}]\!]\}_{u\text{ID}}$ (10) and pk_{ser}. During the simulation, $\mathcal{S}_{\mathcal{A}}$ will be playing the role of the honest server and the trusted party that computes the functionalities used in the hybrid world. We describe $\mathcal{S}_{\mathcal{A}}$ as follows:

① $\mathcal{S}_{\mathcal{A}}$ receives uID from $\mathcal{A}$ and verifies if uID has been enrolled before if not **abort**.

② $\mathcal{S}_{\mathcal{A}}$ receives $[\boldsymbol{P}_{\boldsymbol{\mathcal{A}}}]$ and $\boldsymbol{R}_{\boldsymbol{\mathcal{A}}}$ from $\mathcal{A}$ and a request containing the statements $[\boldsymbol{P}_{\boldsymbol{\mathcal{A}}}]$ and the witness $\boldsymbol{P}_{\boldsymbol{\mathcal{A}}} = \left(\widehat{f_{i,\mathcal{A}}}\right)_{i\in[1,k]}$ to be sent to $f_{\text{ZKPoK}}^{R_{\text{Plain}}}$. At this stage, $\mathcal{S}_{\mathcal{A}}$ sends uID and $\boldsymbol{P}_{\boldsymbol{\mathcal{A}}}$ to the trusted party and receives back $b \in \{0, 1\}$ the public output of the client. Here 0 means *no match* and 1 means *match*. $\mathcal{S}_{\mathcal{A}}$ verifies whether the statements are consistent with the witness if not **abort**. According to $\boldsymbol{R}_{\boldsymbol{\mathcal{A}}}$ and user's uID template, $\mathcal{S}_{\mathcal{A}}$ sends to $\mathcal{A}$ the first half of the requested components extracted from the set of templates $\{[\![\boldsymbol{T}_{u\text{ID}}]\!]\}_{u\text{ID}}$.

③ $\mathcal{S}_{\mathcal{A}}$ receives the statements $[\boldsymbol{P}_{\boldsymbol{\mathcal{A}}}]$ and $\left([f_i^*]\right)_{i\in[1,k]}$ and witnesses that $\mathcal{A}$ sends to $f_{\text{NIZK}}^{R_{\text{DecZero}}}$. $\mathcal{S}_{\mathcal{A}}$ verifies whether the statements are consistent with the witness if not **abort**. $\mathcal{S}_{\mathcal{A}}$ sends the second half of the requested components to $\mathcal{A}$.

④ At this stage, there are the following two cases:

Case $b = 0$**:** $\mathcal{S}_{\mathcal{A}}$ generates a random vector $\boldsymbol{aC}_{\boldsymbol{\mathcal{S}_A}} = (t_i)_{i\in[1,len]}$ such that $\forall i \in [1, len]$ $t_i \neq 0$ then encrypts it first using pk_{joint} to get $[\![\boldsymbol{aC}_{\boldsymbol{\mathcal{S}_A}}]\!]$ then encrypts it using pk_{clt} to get $\left[\boldsymbol{aC}_{\boldsymbol{\mathcal{S}_A}}\right]_{ser}$ so that $\mathcal{A}$ will succeed in decrypting it and the resulted vector will contain non-zero values. This is possible since in ElGamal, a partial decryption under sk_{ser} yields an encryption under pk_{clt} and the final decryption of an ElGamal threshold encryption is the decryption of an ElGamal non-threshold encryption.

Case $b = 1$**:** $\mathcal{S}_{\mathcal{A}}$ generates a random vector $\boldsymbol{aC}_{\boldsymbol{\mathcal{S}_A}} = (t_i)_{i\in[1,len]}$ such that $t_j = 0$ and $\forall i \neq j : \ t_i \neq 0$ then encrypts it first using pk_{joint} to get $[\![\boldsymbol{aC}_{\boldsymbol{\mathcal{S}_A}}]\!]$ then encrypts it using pk_{clt} to get $\left[\boldsymbol{aC}_{\boldsymbol{\mathcal{S}_A}}\right]_{ser}$ so that $\mathcal{A}$ will succeed in decrypting it and the resulted vector will contain exactly one zero. This is also possible for the same reasons mentioned in case $b = 0$.

Since the simulator can not partially decrypt on behalf of the server, the only alternative for him is to encrypt using pk_{clt}. Then sends both vectors $[\![\boldsymbol{aC}_{\boldsymbol{\mathcal{S}_A}}]\!]$ and $\left[\boldsymbol{aC}_{\boldsymbol{\mathcal{S}_A}}\right]_{ser}$ to $\mathcal{A}$ who requests the answer of $f_{\text{NIZK}}^{R_{\text{Blind}}}$ and $f_{\text{NIZK}}^{R_{\text{Partial}}}$ on these statements. $\mathcal{S}_{\mathcal{A}}$ hands 1 to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ does. Given the fact that $\mathcal{S}_{\mathcal{A}}$ outputs whatever $\mathcal{A}$ does, what remains to be proven is the indistinguishability of $\mathcal{A}$'s view in both worlds: ($f_{\text{ZKPoK}}^{R_{\text{Plain}}}$, $f_{\text{NIZK}}^{R_{\text{DecZero}}}$, $f_{\text{NIZK}}^{R_{\text{Blind}}}$, $f_{\text{NIZK}}^{R_{\text{Partial}}}$)-Hybrid and ideal (simulation). $\mathcal{A}$'s view consists of sent and received messages. Here the difference lies in the fact that the received messages were generated by the honest server in the real execution while, during the simulation, they were generated by the simulator $\mathcal{S}_{\mathcal{A}}$.

The steps ①, ② and ③ are identical to the real execution since the only thing $\mathcal{S}_{\mathcal{A}}$ sends is some components from the template. In step ④, in the real execution, the honest server always forms valid proofs. However, in the simulation, $\mathcal{S}_{\mathcal{A}}$ hands $\mathcal{A}$ fake valid proofs according to the output that it has received from the trusted party. As a result, the indistinguishability is maintained since $\mathcal{A}$ in both worlds receives the blinded vector encrypted using pk_{joint} (IND-CPA property of the threshold ElGamal) and a ciphertext decryptable under its own private key sk_{clt} (IND-CPA property of non-threshold ElGamal) and 1 as an answer from $f_{\text{NIZK}}^{R_{\text{Blind}}}$ and $f_{\text{NIZK}}^{R_{\text{Partial}}}$. Therefore, ($f_{\text{ZKPoK}}^{R_{\text{Plain}}}$, $f_{\text{NIZK}}^{R_{\text{DecZero}}}$, $f_{\text{NIZK}}^{R_{\text{Blind}}}$, $f_{\text{NIZK}}^{R_{\text{Partial}}}$)-Hybrid and simulation are indistinguishable.

2) Maliciously Compromised Server: Let $\mathcal{A}$ be an adversary operating in the ($f_{\text{ZKPoK}}^{R_{\text{Plain}}}$, $f_{\text{NIZK}}^{R_{\text{DecZero}}}$, $f_{\text{NIZK}}^{R_{\text{Blind}}}$, $f_{\text{NIZK}}^{R_{\text{Partial}}}$)-Hybrid world controlling the server, we construct $\mathcal{S}_{\mathcal{A}}$ an adversary operating in the ideal world using $\mathcal{A}$ as a subroutine. $\mathcal{S}_{\mathcal{A}}$ is given the description of $\mathcal{A}$, set of templates $\{[\![\boldsymbol{T}_{u\text{ID}}]\!]\}_{u\text{ID}}$ (10), permuted-encrypted threshold vector $\boldsymbol{\Theta}$ (11), v_{enr}, pk_{ser}, pk_{joint} and sk_{ser}. Moreover, $\mathcal{S}_{\mathcal{A}}$ is also given the public input of client uID the user's identity, k_{clt} and pk_{clt}. During the simulation, $\mathcal{S}_{\mathcal{A}}$ will be playing the role of the honest client and the trusted party that computes the functionalities used in the hybrid world. We describe $\mathcal{S}_{\mathcal{A}}$ as follows:

① $\mathcal{S}_{\mathcal{A}}$ verifies if uID has been enrolled before if not **abort**.

② $\mathcal{S}_{\mathcal{A}}$ generates a random probe $\boldsymbol{P}_{\boldsymbol{\mathcal{S}_A}} = \left(\hat{f}_i^*\right)_{i\in[1,k]}$ as a $k-$ dimensional feature vector and encrypts it using k_{clt} to get $\left[\boldsymbol{P}_{\boldsymbol{\mathcal{S}_A}}\right]$. $\mathcal{S}_{\mathcal{A}}$ generates another random vector $\boldsymbol{R}_{\boldsymbol{\mathcal{S}_A}} = (r_i^*)_{i\in[1,k]}$ where $r_i \in [1, n]$. Then sends uID, $\left[\boldsymbol{P}_{\boldsymbol{\mathcal{S}_A}}\right]$ and $\boldsymbol{R}_{\boldsymbol{\mathcal{S}_A}}$ to $\mathcal{A}$. $\mathcal{S}_{\mathcal{A}}$ receives the statement $[\boldsymbol{P}_{\boldsymbol{\mathcal{S}_A}}]$ that $\mathcal{A}$ sent to $f_{\text{ZKPoK}}^{R_{\text{Plain}}}$ and hands 1 to $\mathcal{A}$.

③ $\mathcal{S}_{\mathcal{A}}$ receives from $\mathcal{A}$ the first half of the requested components and verifies the signatures σ_i. If at least one of them fails **abort** otherwise **continue**. $\mathcal{S}_{\mathcal{A}}$ receives the statements $[\boldsymbol{P}_{\boldsymbol{\mathcal{S}_A}}]$ and $([f_i^*])_{i\in[1,k]}$ that $\mathcal{A}$ sent to $f_{\text{NIZK}}^{R_{\text{DecZero}}}$ then hands 1 to $\mathcal{A}$.

④ $\mathcal{S}_{\mathcal{A}}$ receives the second half of the requested components and verifies the signatures α_i. If at least one of them fails **abort** otherwise **continue**. $\mathcal{S}_{\mathcal{A}}$ receives the statements $[\![S]\!]$, $[\![\boldsymbol{C}]\!]$, $[\![\boldsymbol{aC}]\!]$, $[\boldsymbol{aC}]_{ser}$ along with their proper witnesses that

$\mathcal{A}$ sent to $f_{\text{NIZK}}^{R_{\text{Blind}}}$ and $f_{\text{NIZK}}^{R_{\text{Partial}}}$. $\mathcal{S}_{\mathcal{A}}$ checks the statements and their witnesses, if at least one of them is not correct **abort** otherwise $\mathcal{S}_{\mathcal{A}}$ outputs whatever $\mathcal{A}$ does.

Since $\mathcal{S}_{\mathcal{A}}$ outputs whatever $\mathcal{A}$ does, what remains to be proven is the indistinguishability of $\mathcal{A}$'s view in both worlds: $(f_{\text{ZKPoK}}^{R_{\text{Plain}}}, f_{\text{NIZK}}^{R_{\text{DecZero}}}, f_{\text{NIZK}}^{R_{\text{Blind}}}, f_{\text{NIZK}}^{R_{\text{Partial}}})$-Hybrid and ideal (simulation). $\mathcal{A}$'s view consists of sent and received messages. Here the difference lies in the fact that the received messages were generated by the honest client in the real execution while, during the simulation, they were generated by the simulator $\mathcal{S}_{\mathcal{A}}$.

In both executions, the steps ① and ② are indistinguishable. Indeed, the IND-CPA property of ElGamal ensures the indistinguishability between the random encrypted probe $[\boldsymbol{P}_{\boldsymbol{\mathcal{S}_A}}]$ and the real encrypted probe $[\boldsymbol{P}]$. The permutation used by the honest client is a PRP permutation. So in both worlds, $\mathcal{A}$ is unable to distinguish between $(r_i)_{i\in[1,k]}$ received from the honest client and $(r_i^*)_{i\in[1,k]}$ received from $\mathcal{S}_{\mathcal{A}}$. $\mathcal{S}_{\mathcal{A}}$ has generated then encrypted the vector $\boldsymbol{P}_{\boldsymbol{\mathcal{S}_A}}$ correctly (as the honest client would do). As a consequence, it gives $\mathcal{A}$ the value 1 instead of querying the ideal functionality $f_{\text{ZKPoK}}^{R_{\text{Plain}}}$. In step ③, $\mathcal{S}_{\mathcal{A}}$ checks the signatures σ_i, and if at least one of them fails, it aborts as the honest client would do. Then $\mathcal{A}$ always receives 1 from $\mathcal{S}_{\mathcal{A}}$ as an answer for $f_{\text{NIZK}}^{R_{\text{DecZero}}}$ which is always the case in the hybrid execution where the client is assumed to be honest. As a result, step ③ is indistinguishable in both executions. In step ④, $\mathcal{S}_{\mathcal{A}}$ checks the signatures α_i and if the honest client would abort $\mathcal{S}_{\mathcal{A}}$ will also abort. Then it checks the statements and the corresponding witnesses for $f_{\text{NIZK}}^{R_{\text{Blind}}}$ and $f_{\text{NIZK}}^{R_{\text{Partial}}}$. If at least one of them is incorrect, it aborts as the honest client would do. If not, it outputs whatever $\mathcal{A}$ does. Therefore, $(f_{\text{ZKPoK}}^{R_{\text{Plain}}}, f_{\text{NIZK}}^{R_{\text{DecZero}}}, f_{\text{NIZK}}^{R_{\text{Blind}}}, f_{\text{NIZK}}^{R_{\text{Partial}}})$-Hybrid and simulation are indistinguishable. The cases client malicious and server malicious conclude our proof.

VII. Experimentation and Evaluation

We used a 64-bit computer Intel(R) Core i7-8650U CPU with 4 cores (8 logical processors) rated at 2.11GHz and 16GB of memory. We used Python 3.5.2 to implement the HELR classifier and generated the lookup tables on the cluster. For the protocols, we used Linux Ubuntu 18.04.4 LTS ran on Windows-SubLinux (WSL) on Windows 10. Our implementations are publicly available at.[2] [3]

Note that our baseline biometric comparison metric is the HELR classifier. Its performance is influenced by the number of features k, the chosen feature levels n (where $n \times n$ is the dimension of an HELR table) and the quantization step Δ; the higher Δ is, the smaller the distance between maximum score and the comparison threshold $S_{\max} - \theta$ gets. The following evaluation depends on the parameters mentioned above that are crucial to determine the efficiency of our verification protocols since it depends on k and the length of the comparison vector $len = S_{\max} - \theta + 1$ which depends on $S_{\max}$ and θ. The parameter values mentioned in Table III are not the optimal values. Determining the best parameters requires solving an optimization problem that is out of the scope of this paper.

A. Generation of HELR Lookup Tables

Our approach supports any biometric modality that can be encoded as a fixed-length real-valued feature vector. To show this we conducted experiments on *dynamic signatures* (DS2 of BMDB [33] only genuine signatures are considered and skilled forgeries are not) and *faces* (PUT [34] and FRGC2.0 (Experiment 1, mask II) [35] datasets).

For the BMDB dataset, the initial features were extracted by the algorithm described in [36] and were rendered i.i.d. using PCA and LDA as described in [22]. Their dimensionality was reduced to 36, and then they were split into a training set containing 1350 feature vectors (45 users) and a test set containing 4800 feature vectors (160 different users). For the PUT dataset, the initial features were extracted by the VGGFace network [37] (the first layers' weights were taken from [38] and the last layer was retrained for PUT to learn a projection from the 4096-dimensional last layer's output of the pre-trained model to a 64-dimensional latent space) and were rendered i.i.d. as above. Their dimensionality was reduced to 49, and then they were split into a training set containing 1100 feature vectors (50 users) and a test set containing 1095 feature vectors (50 different users). For the FRGC dataset, the initial features were extracted by the VGGFace network [37] and were rendered i.i.d. as above. Their dimensionality was reduced to 94, and then they were split into a training set containing 12776 feature vectors (222 users) and a test set containing 16028 feature vectors (466 users) with identity overlap between both sets of 153 users. Because of the identity overlap, we use FRGC only to evaluate our protocols' speed on different HELR parameters.

We generated the HELR lookup tables using the training set and measured the HELR classifier performance using the test set. Figure 5 depicts the DET curves of LLR, HELR, and Cosine similarity on the three datasets. The difference between LLR and HELR DET curves and the Cosine DET curve is very pronounced and confirms the optimality of those classifiers in the Neyman-Pearson sense. The LLR and HELR classifiers were generated based on the parameters in Table III. For those parameters, the HELR performs better than the LLR. This can be justified by the data-driven nature of LLR, which makes it prone to slightly overfitting on the data. HELR reduces the effect of this model mismatch by applying quantization and rounding. Plus, HELR achieves an EER of 2.4% for BMDB outperforming [36] and an EER of 0.27% for PUT and 0.25% for FRGC that is similar to the state-of-the-art 0.2% [39].

To further demonstrate the practicality of the HELR classifier, we evaluated its performance in the case where the HELR lookup tables are generated using a genuine distribution estimated from one dataset and then tested on a different dataset. On the same FRGC test set, Figure 6 compares the performance of the LLR and HELR classifiers (for which the genuine distribution was estimated from two different datasets: FRGC and CelebA) with Cosine similarity that requires no prior distribution knowledge. We used the FRGC training set and a subset [4] of the CelebA dataset [40] to estimate the genuine distribution for the LLR and generate the HELR lookup tables (with the same number of features 94, HELR dimension 64×64 and quantization step $\Delta = 1.5$) and then measure the LLR and HELR classifiers' performances on the same FRGC test set. In the blue DETs (the genuine distribution is estimated from FRGC) and red DETs (the genuine distribution is estimated from CelebA), we observe an identical effect as in the previous experiment, Figure 5,

[2] https://github.com/aminabassit/helr-classifier

[3] https://github.com/aminabassit/bvsma

[4] It contains 31351 samples of 2476 different users for which the features were extracted as those of FRGC.

TABLE III
PARAMETERS OF HELR LOOKUP TABLES GENERATED FROM BMDB, PUT AND FRGC DATASETS

Dataset	Modality	#Features after LDA	HELR dimension	Score step Δ	Threshold θ at 0.1% FMR	$S_{\max}$	#Genuine comparison	#Impostor comparison	EER	FNMR at 0.1% FMR
BMDB	Signature	36	16×16	0.5	14	99	6.96×10^4	1.14×10^7	2.4%	11.07%
PUT	Face	49	64×64	1	-53	82	1.14×10^4	5.87×10^5	0.27%	0.81%
FRGC	Face	94	64×64	1.5	-1	73	3.99×10^5	1.28×10^8	0.25%	0.49%

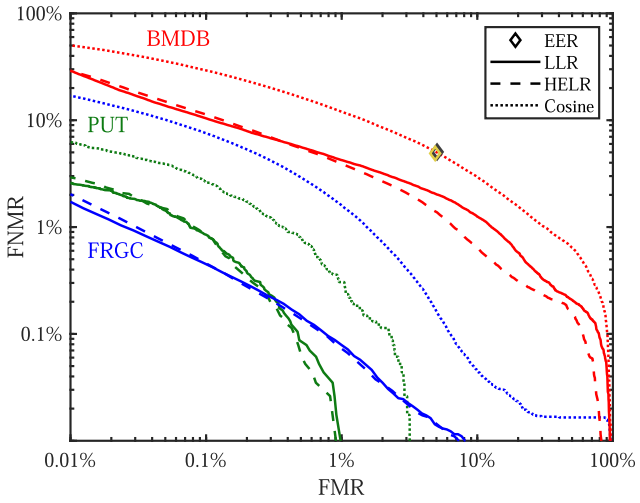

Fig. 5. LLR (solid line), HELR (dashed line), and Cosine similarity (dotted line) DET curves of BMDB (red), PUT (green), and FRGC (blue) datasets. From [36], EER of 5.05% for Cosine similarity (grey diamond) and EER of 4.89% for Euclidean distance (yellow diamond) both on BMDB with 5 signatures per subject in the enrollment. HELR is tested using one signature in the enrollment and achieves an EER of 2.4%.

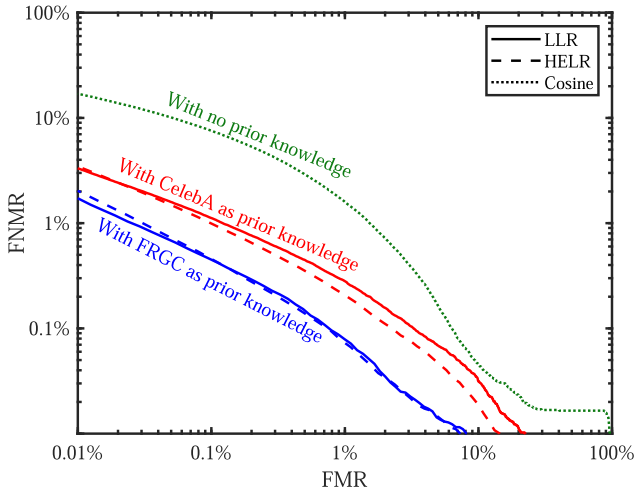

Fig. 6. Cross-validation experiment: LLR (solid line), HELR (dashed line), and Cosine similarity (dotted line) DET curves of FRGC dataset. In the blue curves, the genuine distribution was estimated from the FRGC dataset to estimate the LLR parameters and generate the HELR lookup tables, while in the red curves, it was estimated from the CelebA dataset to generate them. For the green curve, the Cosine similarity does not require a prior knowledge.

where HELR is slightly better than LLR, and both outperform Cosine similarity (the green DET curve Figure 6). However, we notice a performance degradation of LLR and HELR (the red DETs are above the blue DETs) when the classifiers have prior knowledge coming from the CelebA dataset, which can be due to the low quality of the facial images containing diverse variations (such as pose variations). The results of this cross-validation experiment demonstrate that although the LLR and HELR recognition performances have dropped, they are still outperforming the Cosine similarity.

B. Verification Protocols

We implemented the prototype of both protocols as described in Figure 3 and Figure 4 following the client/server architecture in C++ using libscapi library [41] for secure two-party computation and OpenMP [42] to parallelize the implementation.

Figure 7 shows the runtime, per elliptic curve, of 500 genuine verifications via protocols 3 and 4 tested on the three datasets and their median runtime is given in Table IV. The runtime increases as the size of the elliptic curve's prime increases implying the increase of the security strength. According to [43], the security strength of P192, P224 and P256 corresponds respectively to 96, 112 and 128 bits. We recall that in the last step of our protocols, the client decrypts a permuted vector and makes his comparison decision as soon as the first zero is found. The server does not learn the comparison outcome as it is not contacted again after sending this vector. The decryption of a permuted vector justifies the boxplots overall outliers in Figure 7. The runtime of a false non-match (genuine) or a true non-match (impostor) is slower, approximately the upper outliers of the boxplots in Figure 7, since the vector will be fully parsed searching for a zero.

Among HELR parameters (see Table III), the comparison vector length ($S_{\max}-\theta+1$) has more influence on the runtime since the test on FRGC shows the fastest runtime although it has the largest parameters except for the vector length. This can be justified by the expensiveness of the decryption operation.

Table IV summarizes the performance of our work and previous work, as reported in the respective papers, in terms of the runtime, tested modality, and security strength. It is important to emphasize that this table is not meant to compare detailed performance numbers but to indicate the order of magnitudes as each protocol was tested on different machines (i.e., mobile, laptop, workstation, etc.). The top half of the table reports results of protocols belonging to the first category (semi-honest client and semi-honest server), while its bottom half concerns the second category (malicious client and semi-honest server) and the third category (malicious client and malicious server). What stands out in this table is that the reported runtimes of the existing protocols are limited to a security strength of 80 bits, which is no longer recommended by [43]. We show here that

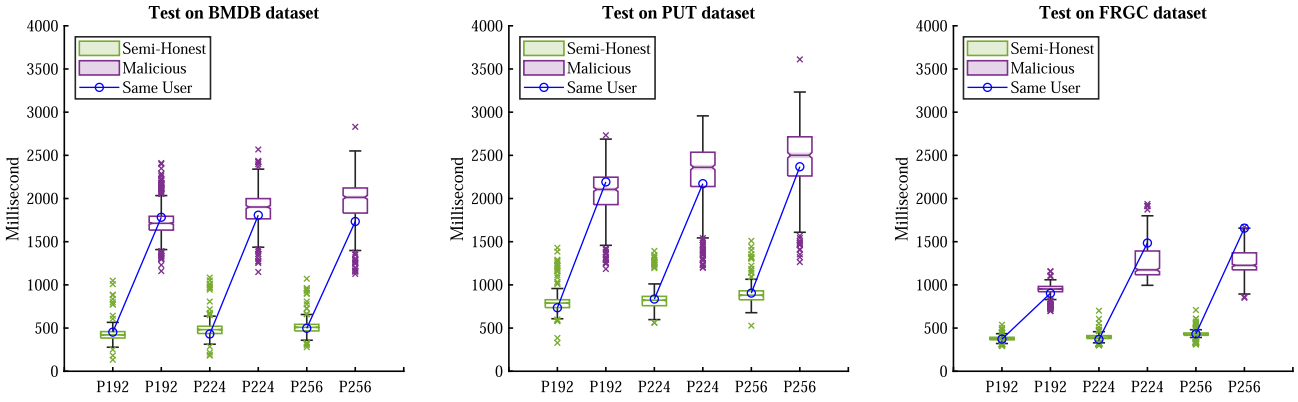

Fig. 7. Runtime of Protocol 3 (green) and Protocol 4 (purple) on 500 genuine verifications over three elliptic curves.

TABLE IV
PERFORMANCE SUMMARY OF OUR WORK AND PREVIOUS WORK AS REPORTED IN THE RESPECTIVE PAPERS

Adversarial model	Protocol	Modality		#Features	Classification method	Cryptographic technique	Security strength [43]	Runtime (s) ⋆	
		Supported †	Tested						
SH Client SH Server	[7]	Binary	Iris	2400	HD[1]	BGV and MAC	80	0.10 (Client) 0.47 (Server)	
	[19]	Binary	Face	192	HD[1]	THE Xor GM	80	0.25 (Client) 0.76 (Server)	
	Our work (Protocol 3)	Real-valued	Dynamic Signature	36	HELR[6]	THE ElGamal	96 112 128	0.42 0.48 0.50	
			Face	49 *and 94 ‡			96 112 128	0.79* 0.82 0.88	0.37 ‡ 0.39 0.43
MAL Client SH Server	[9]	Binary	User's profile	1	AAD[2]	Paillier	80	1.13 (Client) 0.04 (Server)	
	[10]	Real-valued	Touchscreen	10	MD[3]	GC, DGK and cut-and-choose	Unknown	3.70	
	[11]	Real-valued	Face	60	SVM[4]	ZK-proofs	80	15.42	
	[12]	Real-valued	Face	128	ED[5]	HE	80	1.07	
MAL Client MAL Server	[14]	Binary & Real-valued	Multimodal Iris & Face	(5760,2) ¶1 (6400,2) ¶2	HD[1](Iris) ED[5](Face)	SPDZ	46	0.109 ¶1 § 0.120 ¶2 §	
	Our work (Protocol 4)	Real-valued	Dynamic Signature	36	HELR[6]	THE ElGamal and ZK-proofs	96 112 128	1.71 1.90 2.01	
			Face	49 *and 94 ‡			96 112 128	2.10* 2.36 2.50	0.95 ‡ 1.17 1.22

[1] Hamming Distance [2] Absolute Average Deviation [3] Manhattan Distance [4] Support Vector Machine [5] Euclidean Distance
[6] Homomorphically Encrypted log Likelihood-Ratio classifier † Supported modality representation can be either binary feature vector or a real-valued feature vector. ⋆ This column represents, unless specified, the overall runtime of an authentication session without considering the feature extraction. The reported numbers are determined on different systems which makes them not directly comparable but only compared in terms of the orders of magnitude.
* PUT dataset ‡ FRGC dataset ¶1 5760 iris features have an EER of 2.51%, 2 eigenfaces have an EER of 17.37% while their fusion has an EER of 0.98% for fusion coefficients $\alpha = 0.80$ and $\beta = 1-\alpha$. ¶2 6400 iris features have an EER of 2.08%, 2 eigenfaces have an EER of 17.37% while their fusion has an EER of 1.15% for fusion coefficients $\alpha = 0.80$ and $\beta = 1-\alpha$. § This takes into account the online phase only.

even when considering security levels higher than what was considered in prior work, our solution for the semi-honest model (Protocol 3) runs similarly fast, in the order of hundreds of milliseconds, as existing work. Also, our Protocol 4 is the only solution that achieves security in the malicious model while maintaining a similar runtime as existing semi-honest solutions in terms of order of magnitude.

VIII. DISCUSSION

A. MPC-Based Biometric Verification Systems

These systems are based on generic MPC frameworks (such as SPDZ [20], [21]) that comprise an offline phase, where all the expensive computations are performed, and an online phase with only cheap operations. The offline phase is

independent of the inputs and the to-be-evaluated function. During that phase, the parties collectively produce a large amount of shared preprocessed data (input preparation masks, multiplication triples, square pairs, and shared bits) that will be consumed when evaluating the function's circuit during the online phase in an authenticated secret sharing manner. As also noted by other work (e.g., [44]), the shared preprocessed data is only used once (before being revealed). While the function still requires further evaluations, this makes the offline phase subject to a relaunch as soon as the preprocessed data is entirely consumed; for instance, an authentication function is meant to be evaluated several times. The efficiency of those MPC frameworks comes from the cheap nature of operations performed when using authenticated secret sharing. Each party has its input in the clear and shares it with the other parties before evaluating the function's circuit on its shares, and the other parties' input shares with the help of the shared preprocessed data. Subsequently, the parties perform the evaluation on their sides, and then they open their output share to reveal the output value for which they check a MAC before accepting it as a correct output. MPC-based biometric verification systems (such as SEMBA [14]) assume sharing the template between the client and the server. This implies that the client must store the remaining part of each shared template which is of equal size as the share of the template stored on the server. Hence, the client is required to have the same storage capacity as the server itself.

The application of MPC frameworks in the context of biometric verification requires: (1) the knowledge of the inputs (template and probe) in the clear, (2) secret sharing the template between the server and the client so that each one of them stores its part of the share to be used during the authentication, (3) the storage of a large amount of shared preprocessed data on the client and the server sides, and (4) its non-reusability once it is consumed; when a shared value is revealed to all parties, it cannot be reused as a share which makes its size depending on the number of authentication attempts. Thus, the relaunch of the offline phase is necessary [44] after spending all the preprocessed data on authentication attempts.[5] Contrary to (1) and (2), we assume that the server stores the encrypted template, and we require the client to learn no information about the clear-text template. During the verification, the server possesses an encrypted template as input, and the computations are performed over ciphertexts. Applying a generic MPC framework in our setting would involve performing the secret sharing under encryption. However, this would be counterproductive as the online phase of MPC frameworks is very efficient because all operations are performed on the clear-text shares and not under encryption; performing this phase on homomorphically encrypted shares would inherit the performance deficiency of homomorphic encryption. In opposition to (3) and (4), our setting protocol (as shown in Figures 3 and 4) is online-only with limited client-storage that only contains the keys, the PRPs, and the permuted-encrypted threshold vector, which are independent of the number of enrolled users and the number of authentication attempts.

For iris features of size 5760 (resp. 6400) with an EER of 2.51% (resp. 2.08%) and two eigenfaces with an EER of 17.37%, SEMBA generates a list of 17332 (resp. 19252) multiplication triples and two square pairs for which it runs in 0.109s (resp. 0.120s), that is nearly one order of magnitude faster than our protocol, with a low-security strength of 46 bits and a multimodel biometric performance EER of 0.98% (resp. 1.15%). Unlike our protocol (summarized in Figure 4) that for a security strength of 96 bits, it runs in 0.95s with an EER of 0.25% for 94 facial features and runs in 2.10s with an EER of 0.27% for 49 facial features.

[5]The communication cost of one offline phase in SPDZ for calculating a single two-party authentication based on Euclidean distance using a template of size 128 features is 21.19MB [12].

B. Flexibility of the Recognition Outcome Disclosure Right

The international standard ISO/IEC 24745 [45] defines several application models; some of them perform the biometric comparison on the client-side and others on the server-side. The disclosure right of the recognition outcome granted to either the client or the server depends on the application model. The biometric comparison in our protocol Figure 4 is a distributed comparison between the client and the server. The disclosure right is given to the client since the server performs the blinding $[\![\boldsymbol{aC}]\!]$ and the initial partial decryption $[\boldsymbol{aC}]_{ser}$ then sends $[\![\boldsymbol{aC}]\!]$ and $[\boldsymbol{aC}]_{ser}$ along with ZK-proofs to the client (the last two arrows of Step 4 of the same Figure). Note that our protocol can be modified with no effect on the total runtime by letting the client perform the blinding and the initial partial decryption then sends $[\![\boldsymbol{aC}]\!]$ and $[\boldsymbol{aC}]_{clt}$ along with ZK-proofs to the server (inversion of the last two arrows of Step 4 of the same Figure). Moreover, our protocol can be adjusted to allow the client and the server to learn the recognition outcome simultaneously. The client (resp. the server) chooses a blinding vector $\boldsymbol{a_{clt}}$ (resp. $\boldsymbol{a_{ser}}$), calculates the blinded comparison vector $[\![\boldsymbol{a_{clt}C}]\!]$ (resp. $[\![\boldsymbol{a_{ser}C}]\!]$), performs the initial partial decryption then sends $[\![\boldsymbol{a_{clt}C}]\!]$ and $[\boldsymbol{a_{clt}C}]_{clt}$ (resp. $[\![\boldsymbol{a_{ser}C}]\!]$ and $[\boldsymbol{a_{ser}C}]_{ser}$) along with ZK-proofs to the server (resp. client) who finishes the decryption and learns the recognition outcome. Therefore, our protocol is flexible in granting the recognition outcome disclosure right to the client or the server or both of them simultaneously.

IX. Related Work

Over the past decades, several approaches were developed to protect biometric data, known as *biometric template protection* (BTP) [45], most of them protect it at rest when it is stored in a biometric system. However, the increasing number of online services requires a more sophisticated biometric solution that protects the biometric data involved in the entire process for a given adversarial scenario. In this direction, various biometric systems have been proposed that can be categorized according to their adversarial model.

A. Semi-Honest Client and Semi-Honest Server

this category assumes that the parties properly follow the protocol steps and try to learn the biometric information only from the exchanged messages. To prevent such biometric information leakage, most of those systems perform the biometric comparison in the encrypted domain. For instance, [4] uses a support vector machine (SVM) classifier for the comparison. This classifier yields the final score as a linear combination of the template and the probe. The idea consists of storing the encrypted template on the server using a multiplicative homomorphic encryption scheme. Then, for the authentication, the client sends an encrypted probe to the server, which

calculates the linear combination under encryption and sends it randomized to the client, who decrypts it for the server. After canceling out the randomization, the server learns the final score and, based on it, makes its decision. To achieve the same goal, another system [6] follows a hybrid approach, which is a combination of additive homomorphic encryption and garbled circuits, to perform the comparison based on either Hamming distance or Euclidean distance. A common drawback of those systems is that a decryption is performed in the middle of the comparison process, creating a cheating opportunity for a compromised client to change the decrypted result. Ghostshell [7] addressed that by requiring the client to decrypt the final score along with an encrypted message authentication code (MAC) tag to ensure the integrity of the decrypted score. The server, then, makes its decision only for valid tags. Another limitation of those systems is that they expose the final score to the server, which leaks the closeness of a probe to the template as well as the quality of a user's probe. For instance, a server recording the final scores of its users after a successful match can determine which of them has the most stable biometric modality. This makes such a user an interesting target to attack.

B. Malicious Client and Semi-Honest Server

this category intensifies the client's adversarial model to the malicious setting, assuming that an adversary can arbitrarily deviate from the protocol not only to infer biometric information from the template but also to compromise the security of the protocol. Unlike the previous category, the client is asked to provide proofs of each interaction it makes with the server. For example, in [9] and [11], the client appends a ZK-proof to each interaction; either to prove its encrypted probe's consistency or the correctness of its computations. Also, [10] achieves this using garbled circuits empowered by the cut-and-choose technique but at the cost of the protocol's efficiency. Besides, [12] uses HE that supports quadratic operations and blinds the final score. They prove the security of their protocol by separating two types of malicious clients. Type 1 tries to obtain the clear-text values of the template, and type 2 aims at being successfully authenticated without having any biometric information of the legitimate user. Note that in their design, the server makes the comparison decision that delivers it to the client (representing a single user). In contrast to our design, this decision is made by both and known only by the client managing multiple users.

C. Malicious Client and Malicious Server

References [15] and [16] have emphasized that semi-honest security is insufficient in the context of biometric recognition since it leads to severe security risks. Systems belonging to both previously discussed categories store the biometric data encrypted on the server, assuming that it will follow the computations as prescribed by the protocol. The server, indeed, will learn nothing if it follows the protocol. However, an inevitable question arises: given the overwhelming cyber-security threats, can we guarantee that a server holding encrypted biometric data interacting with a client holding raw biometric data will not try to deviate arbitrarily from the protocol? Here, the threat is even worse due to cyberattacks on such servers: the server operator might not have bad intentions but can be manipulated by an external attacker. Thus, such servers are promising targets by themselves. This third category confronts any arbitrary deviation coming either from the server or the client.

Many studies have been approaching the problem of providing a biometric verification protocol for both client and server in the malicious model. In that sense, THRIVE [19] tried to solve this problem in the client/server architecture in a tailored manner. While the authors made some steps forward in this direction, the resulting protocol is unfortunately not secure in the malicious model as their proof follows the definition of the semi-honest model; see [27, Definition 2.2.1]. They construct a simulator that does not consider the adversary's input, which is insufficient since a malicious adversary may substitute the corrupted party's input to affect the correctness of the actual output. In the following, we explain why THRIVE does not achieve security against malicious adversaries: It performs the biometric recognition using Hamming distance between two binary vectors, namely the probe and the template. The server stores the template bit-wise encrypted using a $(2, 2)$-THE and signed by the client. For the authentication, the client sends the probe randomized and receives the encrypted template along with its signature. Subsequently, the client, homomorphically and bit-wise, combines the encrypted randomness used to randomize the probe with the encrypted template and partially decrypts the resulted encryption. Then, it sends to the server the encrypted randomness, the partial decryptions, and a signature that binds both. Note that this signature ensures to the server that the received messages (i.e., the encrypted randomness and the partial decryptions) were sent by the client (i.e., authenticity) and received without being modified during transmission (i.e., integrity). Hence, this signature does not guarantee to the server that the client has sent the correct encrypted randomness and partial decryptions as prescribed by the protocol. In other terms, the server is not sure about how the received messages were computed. On the other hand, the server sends its comparison decision to the client without employing any mechanism to prove the correctness of its decision. In this case, it could skip the last computations that lead to the comparison decision and simply send arbitrary decisions to the client. Thus, the simulator, in the proof of Theorem 1 provided in [19], does not capture the above-mentioned malicious behaviors. This design flaw enables a malicious client and a malicious server to send inconsistent messages that affect the correctness of the output. Therefore, THRIVE is not secure against malicious adversaries.

Later [13] proposed a three-party protocol for continuous authentication secure against malicious smartphone, cloud, and server based on the garbled circuit construction in [10] and without using the cut-and-choose technique. In their protocol, the server is allowed to learn the distance between the protected template and the protected probe. However, due to their use of a one-time-pad over a finite field, this distance is the same as the distance between the clear-text template and clear-text probe. Leaking the clear-text distance makes [13] vulnerable to hill-climbing and brute-force attacks [17], [18]. [13] runs in 0.72s (resp. 2s) for Manhattan distance using 8 (resp. 28) biometric features for a 80 bits security strength. They achieve a runtime similar to ours for a reduced number of features and security strength.

SEMBA [14] proposed a multimodal biometric verification protocol based on SPDZ [20], [21] secure against malicious client and server. As mentioned in Section VIII, SEMBA assumes that the template is generated by the client and shared

between the client and the server. Thus, both store a share of the template of equal size, requiring the client to have a similar storage capacity as the server. This is different from our setting, where we assume that the template is only stored protected on the server-side. SEMBA has only been evaluated with an unrealistically low-security strength of 46 bits, for which it runs nearly one order of magnitude faster than our protocol, between 0.109s and 0.120s.[6] Doubling the size of the security parameters in SPDZ is known to increase the runtime by more than one order of magnitude [47]; it can hence be assumed that SEMBA would have a runtime comparable to the runtime of our approach when considering a 92 bits security strength.

Unlike our protocol Figure 4 that is online-only, SEMBA requires an expensive offline phase subject to a relaunch as soon as the shared preprocessed data is entirely consumed. This depends on how many authentications attempts the preprocessed data can support. For a single authentication attempt, SEMBA generates a list of 17332 (resp. 19252) multiplication triples and 2 square pairs to handle templates of size 5760 (resp. 6400) iris features and two eigenfaces. Moreover, increasing the number of iris features for a fixed number of eigenfaces and the same fusion coefficients degrades the biometric accuracy and the runtime of SEMBA. For two eigenfaces and fusion coefficients $\alpha = 0.80$ and $\beta = 1 - \alpha$, SEMBA runs in 0.109s with an EER of 0.98% for 5760 iris features while it runs in 0.120s with an EER of 1.15% for 6400 iris features. Contrary to our evaluation results, where increasing the number of features improves both the biometric accuracy and the runtime. For a security strength of 96 bits, our protocol 4 runs in 0.95s with an EER of 0.25% for 94 facial features and runs in 2.10s with an EER of 0.27% for 49 facial features.

X. Conclusion

In this paper, we propose a biometric verification protocol secure against malicious adversaries and evaluate its performance. We showed that the proposed HELR classifier is accurate (EER between 0.25% and 0.27% for faces) and diverse in terms of supported biometric modalities that can be encoded as a fixed-length real-valued feature vector. This approach allowed us to speed up the biometric recognition in the encrypted domain by pre-computing the classifier and organizing it into very fast lookup tables that are generated for the purpose of applying a homomorphic encryption layer. Further, we achieved security in the malicious model by strengthening our initial protocol that is secure in the semi-honest model using ZK-proofs. The evaluation of our protocols shows runtimes between 0.37s and 2.50s for security strengths varying from 96 to 128 bits. This makes us achieve a stronger security level for a runtime in the order of few seconds and demonstrates that the case of malicious client and malicious server is practical.

Appendix A
Proofs of Σ-Protocols

Σ-protocols Σ_{DecZero}, Σ_{Blind} and Σ_{Partial} corresponding to the NIZKs $\text{NIZK}_{\text{DecZero}}$, $\text{NIZK}_{\text{Blind}}$ and $\text{NIZK}_{\text{Partial}}$, Table II, are obtained by sampling the challenge $e \in \{0,1\}^t$ instead of calculating it via a hash function. Theorem 2 provides the proof for Σ_{Partial}. The remaining Σ_{DecZero} and Σ_{Blind} are demonstrated in the same way; thus, their proofs are omitted.

Theorem 2: The protocol $\Sigma_{Partial}$ satisfies the three requirements of a Σ-protocol, namely completeness, special soundness, and special honest verifier zero-knowledge.

Proof: **Completeness:** For a prover that knows the underlying partial private key sk_i of the joint key pk_{joint} also the underlying private key of the public key pk_i, a verifier will always accept the proof on the input $((u, v), (u, c))$ since:

$$g^z = g^{r'} \cdot (g^{sk_i})^e = X \cdot pk_i^e$$
$$u^z = u^{r'} \cdot (u^{sk_i})^e = U \cdot (v \cdot c^{-1})^e$$

This holds since $c = v \cdot u^{-sk_i}$.

Special soundness: Suppose that $(X, U; e_1; z_1)$ and $(X, U; e_2; z_2)$ are two valid transcripts such that $e_1 \neq e_2$. We have

$$\begin{cases} g^{z_1} = X \cdot pk_i^{e_1} \\ g^{z_2} = X \cdot pk_i^{e_2} \end{cases}$$

This implies that $g^{z_1 - z_2} = pk_i^{e_1 - e_2}$ and $g^{\frac{z_1 - z_2}{e_1 - e_2}} = pk_i$. Since $sk_i = log_g(pk_i)$ then we succeed in extracting $sk_i = \frac{z_1 - z_2}{e_1 - e_2}$.

Special honest verifier Zero-Knowledge: Consider $\mathcal{M}$ a simulator that is given an input $((u, v), (u, c), pk_i)$ and a challenge e. $\mathcal{M}$ operates as follows:

1) Chooses $z \in \mathbb{Z}_q$.
2) Computes $X = g^z / pk_i^e$ and $U = u^z / (v \cdot c^{-1})^e$.
3) Outputs the transcript $(U, V; e; z)$.

Thus, $\mathcal{M}$ outputs a transcript of the same probability distribution as transcripts between the honest prover and verifier on common input $((u, v), (u, c), pk_i)$.

Acknowledgment

The authors would like to thank Marta Gomez-Barrero, Una Kelly and Ali Khodabakhsh for providing us with the feature vectors for BMDB, PUT and FRGC databases respectively.

References

[1] G. Fawkes. (2019). *Report: Data Breach in Biometric Security Platform Affecting Millions of Users*. vpnMentor. [Online]. Available: https://www.vpnmentor.com/blog/report-biostar2-leak/

[2] B. Carl. (2020). *Report: Retail-Focused Used Electronics Business Leaks Customers' IDS & Fingerprints in Data Breach*. Website Planet. [Online]. Available: https://www.websiteplanet.com/blog/tronicsxchange-breach-report/

[3] O. Goldreich, *Foundations of Cryptography: Basic Applications*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[4] M. Upmanyu, A. M. Namboodiri, K. Srinathan, and C. V. Jawahar, "Blind authentication: A secure crypto-biometric verification protocol," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 2, pp. 255–268, Jun. 2010.

[5] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, "Packed homomorphic encryption based on ideal lattices and its application to biometrics," in *Proc. Int. Conf. Availability, Rel., Secur.* Berlin, Germany: Springer, 2013, pp. 55–74.

[6] H. Chun, Y. Elmehdwi, F. Li, P. Bhattacharya, and W. Jiang, "Outsourceable two-party privacy-preserving biometric authentication," in *Proc. 9th ACM Symp. Inf., Comput. Commun. Secur.*, Jun. 2014, pp. 401–412.

[7] J. H. Cheon, H. Chung, M. Kim, and K.-W. Lee, "Ghostshell: Secure biometric authentication using integrity-based homomorphic evaluations," *IACR Cryptol. ePrint Arch.*, vol. 4, p. 484, May 2016.

[8] J.-H. Im, J. Choi, D. Nyang, and M.-K. Lee, "Privacy-preserving palm print authentication using homomorphic encryption," in *Proc. IEEE 14th Int. Conf. Dependable, Autonomic Secure Comput., 14th Int. Conf. Pervas. Intell. Comput., 2nd Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Aug. 2016, pp. 878–881.

[6] For 5760 (resp. 6400) iris features and two eigenfaces, two eigenfaces and fusion parameters $\alpha = 0.80$ and $\beta = 1 - \alpha$.

[9] S. F. Shahandashti, R. Safavi-Naini, and N. A. Safa, "Reconciling user privacy and implicit authentication for mobile devices," *Comput. Secur.*, vol. 53, pp. 215–233, Sep. 2015.

[10] J. Sedenka, S. Govindarajan, P. Gasti, and K. S. Balagani, "Secure outsourced biometric authentication with performance evaluation on smartphones," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 2, pp. 384–396, Feb. 2014.

[11] H. Gunasinghe and E. Bertino, "PrivBioMTAuth: Privacy preserving biometrics-based and user centric protocol for user authentication from mobile phones," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 4, pp. 1042–1057, Apr. 2018.

[12] J.-H. Im, S.-Y. Jeon, and M.-K. Lee, "Practical privacy-preserving face authentication for smartphones secure against malicious clients," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2386–2401, 2020.

[13] P. Gasti, J. Šeděnka, Q. Yang, G. Zhou, and K. S. Balagani, "Secure, fast, and energy-efficient outsourced authentication for smartphones," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2556–2571, Nov. 2016.

[14] M. Barni, G. Droandi, R. Lazzeretti, and T. Pignata, "SEMBA: Secure multi-biometric authentication," *IET Biometrics*, vol. 8, no. 6, pp. 411–421, Nov. 2019.

[15] K. Simoens, J. Bringer, H. Chabanne, and S. Seys, "A framework for analyzing template security and privacy in biometric authentication systems," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 833–841, Apr. 2012.

[16] A. Abidin and A. Mitrokotsa, "Security aspects of privacy-preserving biometric authentication based on ideal lattices and ring-LWE," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Dec. 2014, pp. 60–65.

[17] R. M. Bolle, J. H. Connell, and N. K. Ratha, "Biometric perils and patches," *Pattern Recognit.*, vol. 35, no. 12, pp. 2727–2738, Dec. 2002.

[18] M. Martinez-Diaz, J. Fierrez-Aguilar, F. Alonso-Fernandez, J. Ortega-García, and J. A. Siguenza, "Hill-climbing and brute-force attacks on biometric systems: A case study in match-on-card fingerprint verification," in *Proc. 40th Annu. Int. Carnahan Conf. Secur. Technol.*, Oct. 2006, pp. 151–159.

[19] C. Karabat, M. S. Kiraz, H. Erdogan, and E. Savas, "THRIVE: Threshold homomorphic encryption based secure and privacy preserving biometric verification system," *EURASIP J. Adv. Signal Process.*, vol. 2015, no. 1, pp. 1–18, Dec. 2015.

[20] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proc. Annu. Cryptol. Conf.* Berlin, Germany: Springer, 2012.

[21] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical covertly secure MPC for dishonest majority–or: Breaking the SPDZ limits," in *Proc. Eur. Symp. Res. Comput. Secur.* Berlin, Germany: Springer, 2013, pp. 1–18.

[22] A. M. Bazen and R. N. J. Veldhuis, "Likelihood-ratio-based biometric verification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 86–94, Jan. 2004.

[23] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: A joint formulation," in *Computer Vision—ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Germany: Springer, 2012.

[24] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *Eur. Trans. Telecommun.*, vol. 8, no. 5, pp. 481–490, Sep. 1997.

[25] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Springer, 1991, pp. 522–526.

[26] P.-A. Fouque and D. Pointcheval, "Threshold cryptosystems secure against chosen-ciphertext attacks," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2001, pp. 351–368.

[27] C. Hazay and Y. Lindell, *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Berlin, Germany: Springer-Verlag, 2010. [Online]. Available: https://www.springer.com/gp/book/9783642143021, doi: 10.1007/978-3-642-14303-8.

[28] J. Camenisch, "Group signature schemes and payment systems based on the discrete logarithm problem," Ph.D. dissertation, Dept. Comput. Sci., ETH Zurich, Zürich, Switzerland, 1998.

[29] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Conf. Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1986, pp. 186–194.

[30] R. Canetti, "Security and composition of multiparty cryptographic protocols," *J. Cryptol.*, vol. 13, no. 1, pp. 143–202, Jan. 2000.

[31] R. Cramer, I. Damgard, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn. Adv. Cryptol. (EUROCRYPT)*, in Lecture Notes in Computer Science, vol. 2045. Berlin, Germany: Springer, 2001, pp. 280–300.

[32] F. M. J. Willems and T. Ignatenko, "Quantization effects in biometric systems," in *Proc. Inf. Theory Appl. Workshop*, Feb. 2009, pp. 372–379.

[33] J. Ortega-Garcia *et al.*, "The multiscenario multienvironment BioSecure multimodal database (BMDB)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 6, pp. 1097–1111, Jun. 2009.

[34] A. Kasinski, A. Florek, and A. Schmidt, "The put face database," *Image Process. Commun.*, vol. 13, nos. 3–4, pp. 59–64, 2008.

[35] P. J. Phillips *et al.*, "Overview of the face recognition grand challenge," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 947–954.

[36] M. Gomez-Barrero, J. Fierrez, J. Galbally, E. Maiorana, and P. Campisi, "Implementation of fixed-length template protection based on homomorphic encryption with application to signature biometrics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2016, pp. 191–198.

[37] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2015, pp. 41.1–41.12. [Online]. Available: http://www.bmva.org/bmvc/2015/papers/paper041/index.html

[38] R. C. Malli. *Vggface Implementation With Keras Framework*. Accessed: Mar. 27, 2017. [Online]. Available: https://github.com/rcmalli/keras-vggface

[39] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4690–4699.

[40] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.

[41] B. I. C. R. Group *et al.*, "LIBSCAPI: The secure computation api," Tech. Rep., 2016.

[42] L. Dagum and R. Menon, "OpenMP: An industry standard API for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan. 1998.

[43] D. Giry. (2020). *Keylength—Cryptographic Key Length Recommendation*. BlueKrypt. [Online]. Available: https://www.keylength.com/en/4/

[44] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1575–1590.

[45] ISO Central Secretary, *Information Technology—Security Techniques—Biometric Information Protection, International Organization for Standardization*, Standard ISO/IEC 24745, 2011. [Online]. Available: https://www.iso.org/standard/52946.html

[46] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain, "Biometric cryptosystems: Issues and challenges," *Proc. IEEE*, vol. 92, no. 6, pp. 948–960, Jun. 2004.

[47] D. Rotaru, N. P. Smart, T. Tanguy, F. Vercauteren, and T. Wood, "Actively secure setup for SPDZ," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1300, May 2021. [Online]. Available: https://eprint.iacr.org/2019/1300