Privacy-Preserving Biometric Matching Using Homomorphic Encryption

Gaëtan Pradel
INCERT, Luxembourg
Royal Holloway, University of London
Egham, United Kingdom
gpradel@incert.lu

Chris Mitchell
Royal Holloway, University of London
Egham, United Kingdom
me@chrismitchell.net

***Abstract*—Biometric matching involves storing and processing sensitive user information. Maintaining the privacy of this data is thus a major challenge, and homomorphic encryption offers a possible solution. We propose a privacy-preserving biometrics-based authentication protocol based on fully homomorphic encryption, where the biometric sample for a user is gathered by a local device but matched against a biometric template by a remote server operating solely on encrypted data. The design ensures that 1) the user's sensitive biometric data remains private, and 2) the user and client device are securely authenticated to the server. A proof-of-concept implementation building on the TFHE library is also presented, which includes the underlying basic operations needed to execute the biometric matching. Performance results from the implementation show how complex it is to make FHE practical in this context, but it appears that, with implementation optimisations and improvements, the protocol could be used for real-world applications.**

***Index Terms*—Privacy-Preserving, Multiparty Computation, Biometrics Homomorphic, Encryption**

I. Introduction

This paper proposes a privacy-preserving biometric-based authentication protocol based on fully homomorphic encryption (FHE), designed for use in the case where the biometric sample for a user is gathered by a local device but matched against a biometric template by a remote server. The goal is to enable this to occur without the remote server gaining access to any of the sensitive biometric data. The privacy-preserving and authentication properties of the protocol are formally established. A proof-of-concept C/C++ implementation building on the TFHE library due to Chillotti et al. [1] has also been developed, in which face matching is used as the biometric. Performance results from this implementation are presented. The results of the implementation confirm the difficulty of making FHE practical in such a scenario, but we suspect that, with optimisations and improvements, the protocol could be used for real-world applications.

As part of the proof-of-concept, all the elementary operations necessary to execute the protocol using FHE were implemented. Thus, as a side contribution, we have provided a set of elementary arithmetic routines in the ciphertext domain[1], which could be useful for other prototype implementations.

Supported by the Luxembourg National Research Fund (FNR) (12602667).

[1]The implementation is hosted here: https://github.com/lab-incert/threats.

a) Homomorphic encryption: Homomorphic encryption allows one to perform computations on encrypted data, without ever decrypting it. This enables users to perform operations in untrusted environments. The idea of performing computations on encrypted data was introduced in 1978 by Rivest, Shamir and Adleman [2]. While many homomorphic schemes have been proposed [3]–[6], it wasn't until 2009 that Gentry presented [7] the first FHE scheme, based on ideal lattices. Gentry's breakthrough rests on a technique called *bootstrapping*. An FHE scheme based on Gentry's blueprint enables an arbitrary number of additions and multiplications, i.e. any function, to be computed on encrypted data. Since then, many other schemes have been proposed [8]–[13], including schemes not using the bootstrapping technique. For example, in 2012, Brakerski, Gentry and Vaikuntanathan [14] presented a scheme based on the ring version of the Learning With Errors problem, introduced by Regev [15]. A second type of FHE scheme was introduced by Gentry, Sahai and Waters [16]. This scheme was further improved [17], [18], and most recently by Chillotti et al. [1], [19].

b) Biometric authentication: The use of biometrics for authentication has been discussed for several decades, and has seen growing use. International organisations suggest passwordless[2] systems for authentication, and biometrics can solve this issue. Advances mean that in some circumstances biometric recognition algorithms perform better than humans, even for face recognition [20]. Nonetheless, biometric authentication faces a range of challenges [21], in particular regarding the protection of users' sensitive data. Biometric data, such as a fingerprint, is fixed for a lifetime, meaning that its use gives rise to significant privacy concerns. Ideally, biometric data should not be processed without protection or anonymisation. Homomorphic encryption offers a possible solution to this problem [21], as it allows the authentication provider to perform biometric matching on (encrypted) data, while protecting the privacy of sensitive biometric data.

c) Related work: The use of homomorphic cryptography in the context of biometric matching is not new [22], [23]. However, most previous work uses partially homomorphic

[2]See for example the World Economic Forum: https://www.weforum.org/agenda/2020/04/covid-19-is-a-reminder-that-its-time-to-get-rid-of-passwords/.

encryption and not FHE. Some of this work has promising performance results, e.g. Blanton and Gasti [24] who calculate the Hamming distance between two iris feature vectors in only 150 ms. However, because of the additive-only (partially homomorphic) characteristic of the encryption schemes they use, they are not able to evaluate a circuit much more complex than for Hamming distance. Yasuda et al. [25] used a homomorphic scheme that also enables multiplications in the ciphertext domain, but still only compute the Hamming distance between two biometric vectors; moreover, the approach is vulnerable against malicious attackers [26]. Back in 2008, Bringer and Chabanne [27] proposed an authentication protocol based on the homomorphic properties of two partially homomorphic encryption schemes.

Biometric matching based on FHE has been previously proposed; perhaps the first example is the private face verification system of Troncoso-Pastoriza et al. [28]. Cheon et al. [29] proposed *Ghostshell*, a tool that works on iris templates, that is computationally costly. More recently, Boddeti [30] showed how to execute a secure face matching using the Fan-Vercauteren FHE scheme [31] and obtained practical results by packing the ciphertexts in a certain way.

d) Structure of the paper: Section II introduces the notions necessary for the rest of the paper. Sections III and IV are the core of the paper, presenting the design and security properties of the protocol. Finally, Sections V and VI give results from the protocol implementation and conclude the paper.

II. Preliminaries

$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{B}$ represent the sets of natural numbers, integers, reals and bits, respectively.

A. Security notions

We next introduce some formal security notions. For more complete versions of Definitions 1-5, see Goldreich [32].

Definition 1 (Negligible)**.** *We say a function $f : \mathbb{N} \mapsto \mathbb{R}$ is* negligible *if for every polynomial p there exists an N such that, for all $n > N$:*

$$f(n) < \frac{1}{p(n)}.$$

Definition 2 (Probability ensemble)**.** *Let I be a countable index set. A* probability ensemble *indexed by I is a sequence of random variables indexed by I. Namely, any $X = (X_i)_{i \in I}$, where each X_i is a random variable, is an ensemble indexed by I.*

Definition 3 (Polynomial-time indistinguishability)**.** *Suppose $X = (X_i)_{i \in \mathbb{N}}$ and $Y = (Y_i)_{i \in \mathbb{N}}$ are ensembles with index set $\mathbb{N}$, where $X_i, Y_i \in \mathbb{B}^n$ for all i. Then X and Y are said to be* indistinguishable in polynomial-time *if, for every probabilistic polynomial-time algorithm $D : \mathbb{B}^n \to \mathbb{B}$, every polynomial p, and all sufficiently large n:*

$$|Pr[D(X_n) = 1] - Pr[D(Y_n) = 1]| < \frac{1}{p(n)}.$$

We write $X \overset{c}{\approx} Y$.

Remark 1. *We follow common practice and refer to* computational indistinguishability *instead of indistinguishability in polynomial-time.*

Definition 4 (Statistical distance)**.** *Suppose $X = (X_i)_{i \in \mathbb{N}}$ and $Y = (Y_i)_{i \in \mathbb{N}}$ are ensembles with index set $\mathbb{N}$, where $X_i, Y_i \in \mathbb{B}^n$ for all i. Then the* statistical distance *function $\Delta : \mathbb{N} \to \mathbb{R}$ is defined as:*

$$\Delta(n) \overset{def}{=} \frac{1}{2} \sum_{\alpha \in \mathbb{B}^n} |Pr[X_n = \alpha] - Pr[Y_n = \alpha]|.$$

Definition 5 (Statistical indistinguishability)**.** *Suppose $X = (X_i)_{i \in \mathbb{N}}$ and $Y = (Y_i)_{i \in \mathbb{N}}$ are ensembles with index set $\mathbb{N}$, where $X_i, Y_i \in \mathbb{B}^n$ for all i. Then X and Y are said to be* statistically indistinguishable *if their statistical distance is negligible.*

We write $X \overset{s}{\approx} Y$.

Remark 2. *If the ensembles X and Y are statistically indistinguishable, then they are also computationally indistinguishable. The converse is not true.*

Definition 6 (Adversary)**.** *An* adversary $\mathcal{A}$ *for a cryptographic scheme is a polynomial-time algorithm (or a set of polynomial-time algorithms) that models a real-world attacker. It is equipped with defined computational resources and capabilities, and is designed to attack the security of the scheme typically as a participant in a security game.*

Definition 7 (Challenger)**.** *A* challenger *for a cryptographic scheme is a polynomial-time algorithm (or a set of polynomial-time algorithms) that models a real-world instance of the scheme. It is usually assumed to possess unlimited computational resources and capabilities, and is viewed as a 'black box' which responds to queries made by an adversary in a security game.*

Definition 8 (Security game)**.** *A* security game *models an attack on a cryptographic scheme involving an adversary and a challenger.*

Definition 9 (Advantage)**.** *In the context of a cryptographic scheme and a security game for this scheme, the* advantage *of an adversary is a function of the probability that the adversary wins the security game that measures the adversary's improvement over random choice.*

B. Homomorphic Encryption

We next formally introduce homomorphic encryption and certain associated notions. For more complete versions of these definitions, see Armknecht et al. [33].

Definition 10 (Homomorphic Encryption scheme)**.** *A* homomorphic encryption scheme $\mathcal{E}$ *for a circuit family* Π *consists of four PPT algorithms* (KeyGen, Enc, Dec, Eval) *with the following properties.*

- $(sk, pk, evk) \leftarrow \mathsf{KeyGen}(1^\lambda)$. *Given the security parameter $\lambda \in \mathbb{N}$,* KeyGen *outputs a key triple made up of a*

secret key sk, a public key pk and an evaluation key evk. The plaintext space $\mathcal{M}$ and the ciphertext space $\overline{\mathcal{M}}$ are determined by pk.

- $\overline{m} \leftarrow \mathsf{Enc}(pk, m)$. *Given a public key pk and a plaintext $m \in \mathcal{M}$,* Enc *outputs a ciphertext $\overline{m} \in \overline{\mathcal{M}}$.*
- $\begin{cases} m \\ \perp \end{cases} \leftarrow \mathsf{Dec}(sk, \overline{m})$. *Given a secret key sk and a ciphertext $\overline{m}$,* Dec *outputs either the plaintext $m \in \mathcal{M}$ if $\overline{m} \leftarrow \mathsf{Enc}(pk, m)$ or $\perp$.*
- $\overline{m}' \leftarrow \mathsf{Eval}(evk, \pi, \overline{m})$. *Given an evaluation key evk, a circuit $\pi \in \Pi$, where Π is a circuit family (see Appendix A for details) and a ciphertext $\overline{m} \in \overline{\mathcal{M}}$,* Eval *outputs another ciphertext $\overline{m}' \in \overline{\mathcal{M}}$.*

Remark 3. *Depending on the scheme, the evaluation key evk might be part of, or equal to, the public key pk. For simplicity of presentation, here and throughout we assume that the circuit input to* Eval *has input size corresponding to the size of the input ciphertext(s).*

Definition 10, and those below, holds for a range of types of plaintext, including both bit strings and vectors of plaintexts. Some algorithms, such as KeyGen, take as input a security parameter λ, which will be denoted as such throughout this paper unless stated otherwise. This input is usually written in unary representation 1^λ because we want an algorithm that runs in time polynomial in the size of λ to be considered as efficient. We refer to the outputs of Enc as 'fresh ciphertexts' and those of Eval as 'evaluated ciphertexts'.

Definition 11 (Correctness)**.** *Suppose $\mathcal{E} =$ (KeyGen, Enc, Dec, Eval) is a homomorphic encryption scheme with security parameter λ. We say $\mathcal{E}$ is* correct *for a circuit family Π if $\mathcal{E}$ correctly decrypts both fresh and evaluated ciphertexts, namely, for all $\lambda \in \mathbb{N}$, the following two conditions hold.*

- *Suppose $(sk, evk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$. If $m \in \mathcal{M}$ and $\overline{m} \leftarrow \mathsf{Enc}(pk, m)$ then $m \leftarrow \mathsf{Dec}(sk, \overline{m})$. Else $\perp \leftarrow \mathsf{Dec}(sk, \overline{m})$.*
- *For any key triple $(sk, evk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, any circuit $\pi \in \Pi$, any plaintext $m \in \mathcal{M}$ and any ciphertext $\overline{m} \in \overline{\mathcal{M}}$ with $\overline{m} \leftarrow \mathsf{Enc}(pk, m)$, if $\overline{m}' \leftarrow \mathsf{Eval}(evk, \pi, \overline{m})$ then $\mathsf{Dec}(sk, \overline{m}') \rightarrow \pi(m)$.*

Definition 12 (Indistinguishability under Chosen-Plaintext Attacks security game)**.** *Suppose $\mathcal{E} =$ (KeyGen, Enc, Dec, Eval) is a homomorphic encryption scheme with security parameter λ. Suppose also that $\mathcal{A}$ is a PPT adversary. The* indistinguishability under chosen-plaintext attacks *(*IND-CPA*)* security game *is as follows.*

1) *A challenger runs $(sk, pk, evk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and shares pk with $\mathcal{A}$.*
2) *$\mathcal{A}$ generates two distinct plaintexts $\{m_0, m_1\}$ and submits a query to the challenger to request the encryption of one of them with pk.*
3) *The challenger chooses $i \in \mathbb{B}$ uniformly at random, computes $\overline{m} \leftarrow \mathsf{Enc}(m_i, pk)$ and sends $\overline{m}$ to $\mathcal{A}$.*
4) *$\mathcal{A}$ outputs a pair $(m_j, \overline{m})$, where $j \in \mathbb{B}$, and wins the game if $i = j$.*

We denote this security game by IND-CPA$^{\mathcal{A}}_{\mathcal{E}}(1^\lambda)$ and a win in an instance of this security game by IND-CPA$^{\mathcal{A}}_{\mathcal{E}}(1^\lambda) = 1$.

Definition 13 (Advantage for the IND-CPA security game)**.** *Suppose $\mathcal{E} =$ (KeyGen, Enc, Dec, Eval) is a homomorphic encryption scheme with security parameter λ. Suppose $\mathcal{A}$ is an adversary in the IND-CPA security game. The advantage of $\mathcal{A}$ with respect to $\mathcal{E}$, denoted $Adv^{\mathcal{E}}_{\mathcal{A}}(\lambda)$, is defined to be:*

$$Adv^{\mathcal{E}}_{\mathcal{A}}(\lambda) \stackrel{def}{=} \left| 2 \cdot Pr\left[\textit{IND-CPA}^{\mathcal{A}}_{\mathcal{E}}(1^\lambda) = 1\right] - 1 \right|.$$

Definition 14 (IND-CPA security)**.** *Suppose $\mathcal{E}$, $\mathcal{A}$ and λ are as in Definition 13. $\mathcal{E}$ is* IND-CPA secure *if the advantage $Adv^{\mathcal{E}}_{\mathcal{A}}(\lambda)$ for $\mathcal{A}$ in the IND-CPA security game is negligible.*

III. A NOVEL PRIVACY-PRESERVING PROTOCOL

We now describe the privacy-preserving biometric matching protocol. In fact we give two descriptions: in §III-A we give an informal introduction, explaining the motivation for the design, and then in §III-B we give a formal description which we use as the basis for the analysis in Section IV. For simplicity of presentation we suppose that the public key pk and the evaluation key evk are equal.

A. Informal description of the protocol

We describe a protocol involving two parties, a client C and a server S, where C is acting on behalf of user U. C wishes to access a certain service, not offered by S, which requires an initial authentication of the user U associated with C to S. The process of authentication uses sensitive biometric data such as face images or iris information for U that is gathered by C. If S successfully authenticates U, S sends an ID token τ, to C. C can now use τ to access the requested service.

Note that C is trusted by S to correctly gather a fresh biometric sample from U. In the protocol, S verifies that the gathered sample matches the appropriate user template, and also authenticates C to S. Note that the protocol neither provides authentication of S to C nor provides encryption of transferred messages; it is implicitly assumed that these properties are provided by the communications channel, e.g. using a server-authenticated TLS session.

In the description below, Step 0 (registration) is performed once before use of the protocol. Steps 1-4 of the protocol are performed every time the user U wishes to be authenticated to S (via C).

Step 0: Registration

C generates a key pair (sk_C, pk_C) for a homomorphic encryption scheme $\mathcal{E}$, and obtains by some means a biometric template t for its associated user U. C then encrypts t as $\overline{t} \leftarrow \mathsf{Enc}(pk_C, t)$ and sends $\overline{t}$ to S via a trusted channel. S stores $\overline{t}$, and subsequently uses it for biometric matching when the protocol is executed (see Step 2). In the remainder of this description we suppose that S, by some means, is assured of the identity of U and that the encrypted biometric template $\overline{t}$ for U is genuine.

Step 1: Initialisation

C takes a fresh biometric sample s from U and, using $\mathcal{E}$, computes an encrypted version $\overline{s} \leftarrow \mathsf{Enc}(pk_C, s)$ and sends it to S.

Step 2: Construction of the Matching Token

Phase 1: Matching Computation

We suppose that S is equipped with a biometric matching function $f\colon \mathcal{M} \times \mathcal{M} \to \mathbb{B}$ which inputs a biometric template and a biometric sample and outputs an indication of whether there is a sufficiently close match between them. Suppose $\pi_f \in \Pi_f$, where Π_f is the circuit family associated with $\mathcal{E}$ which implements f. S now computes

$$\overline{b} \leftarrow \mathsf{Eval}(pk_C, \pi_f, \langle \overline{s}, \overline{t} \rangle),$$

where $\overline{b}$ is the encrypted version of a boolean b indicating the success or not of the biometric matching, i.e. $\overline{b} \leftarrow \mathsf{Enc}(pk_C, f(s,t))$.

In a naïve version of the protocol, S now sends C the encrypted matching result $\overline{b}$; C decrypts it to obtain $b \leftarrow \mathsf{Dec}(sk_C, \overline{b})$, and sends b to S. S can now use b to decide whether not to generate the ID Token τ. For obvious reasons this is not secure (b is not authenticated), and hence we need a slightly more elaborate protocol.

In order to enable S to authenticate C, we introduce the notion of a *Matching Token*, denoted by y. In Phase 2 this token is constructed by S as a function of b (whilst still encrypted) in such a way that S can, when provided by C with a decrypted version of the token in Step 4, (a) verify its authenticity, and (b) determine the value of b.

Phase 2: Signature Computation We suppose S has an implementation of the function

$$g(b, r_0, r_1) = (1-b) \cdot r_0 + b \cdot r_1.$$

S first selects two random numbers $r_0 \xleftarrow{\$} \mathbb{B}^\lambda$ and $r_1 \xleftarrow{\$} \mathbb{B}^\lambda$, and stores them for use in Step 4. S next computes

$$\overline{r_0} \leftarrow \mathsf{Enc}(pk_C, r_0) \quad \text{and} \quad \overline{r_1} \leftarrow \mathsf{Enc}(pk_C, r_1).$$

In the encrypted domain of $\mathcal{E}$ (under pk_C), S now uses $\overline{b}$, $\overline{r_0}$ and $\overline{r_1}$ to compute the encrypted matching token $\overline{y}$ as:

$$\overline{y} \leftarrow \mathsf{Eval}(pk_C, \pi_g, \langle \overline{b}, \overline{r_0}, \overline{r_1} \rangle),$$

where $\pi_g \in \Pi_g$, the circuit family associated with $\mathcal{E}$ which implements g. That is, S obtains $\overline{y} \leftarrow \mathsf{Enc}(pk_C, g(b, r_0, r_1))$ although, of course, S does not have access to b; i.e. at this stage S does not know whether or not the biometric matching succeeded. S sends now $\overline{y}$ to C.

Note that this part of the protocol requires S to retain the random values r_0 and r_1 until Step 4, and hence the protocol is stateful.

Step 3: Decryption of $\overline{y}$

C receives $\overline{y}$ from S and computes

$$y \leftarrow \mathsf{Dec}(sk_C, \overline{y}).$$

At this point it is still the case that neither C nor S know whether the biometric matching succeeded. C only possesses a string which looks random, and S cannot decrypt any data encrypted with pk_C. C now sends y to S.

Step 4: Authentication of C

Phase 1: Verification

S receives y from C, and checks whether it is equal to r_0 or r_1. If so, S has successfully authenticated C; if not S rejects C.

Phase 2: Token generation

S generates an ID Token τ where $\tau \leftarrow$ ACCEPT if $y = r_1$, and $\tau \leftarrow$ REJECT otherwise, and sends it to C. As a result, C has a valid ID Token, which can be used to access the desired service, if and only if the biometric matching was successful and S has authenticated C.

B. Formal description of the protocol

We now formally present the protocol, referred to as $\mathcal{P}$. The protocol is summarised in Figure 1, where $\lambda_\mathcal{E}$ is the security parameter of $\mathcal{E}$. Protocol initialisation, described immediately below, assumes Step 0 has been successfully completed.

Input to C:

C has a biometric sample s, and a key pair (sk_C, pk_C) generated with a homomorphic encryption scheme $\mathcal{E}$. This is represented by the tuple (s, sk_C). We denote the plaintext space and the ciphertext space associated with $\mathcal{E}$ by $\mathcal{M}_\mathcal{E}$ and $\overline{\mathcal{M}_\mathcal{E}}$ respectively.

Input to S:

S has an encrypted biometric template $\overline{t} \leftarrow \mathsf{Enc}(pk_C, t)$ generated by C in a pre-computation phase. This is represented by the tuple $(\overline{t})$.

The following functions are used by S.

- $f\colon \mathcal{M} \times \mathcal{M} \longrightarrow \mathbb{B}$ indicates whether or not two biometric values match, where $\mathcal{M}$ is the set of possible biometric values and an output of 1 indicates a match.
- $g\colon \mathbb{B} \times \mathbb{B}^\lambda \times \mathbb{B}^\lambda \longrightarrow \mathbb{B}^\lambda$ creates a matching token y from a boolean b and two random numbers, where

$$g\colon (b, r_i, r_j) \longmapsto (1-b) \cdot r_i + b \cdot r_j, \quad \text{where } i, j \in \mathbb{N}.$$

The above two initialisations are expressed formally as $\mathcal{P} : C(s) \leftrightarrow S(\overline{t})$.

Common input:

Both parties know the homomorphic encryption scheme $\mathcal{E}$ and the public key pk_C generated by C.

Protocol transcript:

(i) [C Pre-computation]:
  a) $(sk_C, pk_C) \leftarrow \mathsf{KeyGen}(1^{\lambda_\mathcal{E}})$;
  b) Take a fresh biometric sample t from U to be used as template;
  c) $\overline{t} \leftarrow \mathsf{Enc}(pk_C, t)$.

(ii) [$C \longrightarrow S$ Pre-computation]:
  a) Send $\overline{t}$ to S.

1) [$C \longrightarrow S$] C executes the following:
  a) Take a fresh biometric sample s from U;
  b) Compute $\overline{s} \leftarrow \mathsf{Enc}(pk_C, s)$;
  c) Send $\overline{s}$ to S.

2) [$S \longrightarrow C$] S executes the following:

a) (*Phase 1*) Compute $\overline{b} \leftarrow \mathsf{Eval}(pk_C, \pi_f, \langle \overline{s}, \overline{t} \rangle)$;
b) (*Phase 2*) Generate $r_0, r_1 \xleftarrow{\$} \mathbb{B}^n$;
c) Compute $\overline{r_0} \leftarrow \mathsf{Enc}(pk_C, r_0)$;
d) Compute $\overline{r_1} \leftarrow \mathsf{Enc}(pk_C, r_1)$;
e) Compute $\overline{y} \leftarrow \mathsf{Eval}(pk_C, \pi_g, \langle \overline{b}, \overline{r_0}, \overline{r_1} \rangle)$;
f) Send $\overline{y}$ to C.

3) $[C \longrightarrow S]$ C executes the following:
a) Compute $y \leftarrow \mathsf{Dec}(sk_C, \overline{y})$;
b) Send y to S.

4) $[S \longrightarrow C]$ S executes the following:
a) If $y \neq r_0$ and $y \neq r_1$, S terminates execution;
b) Compute $\tau \leftarrow \begin{cases} \text{ACCEPT} & \text{if } y = r_1, \\ \text{REJECT} & \text{if } y = r_0; \end{cases}$
c) Send τ to C.

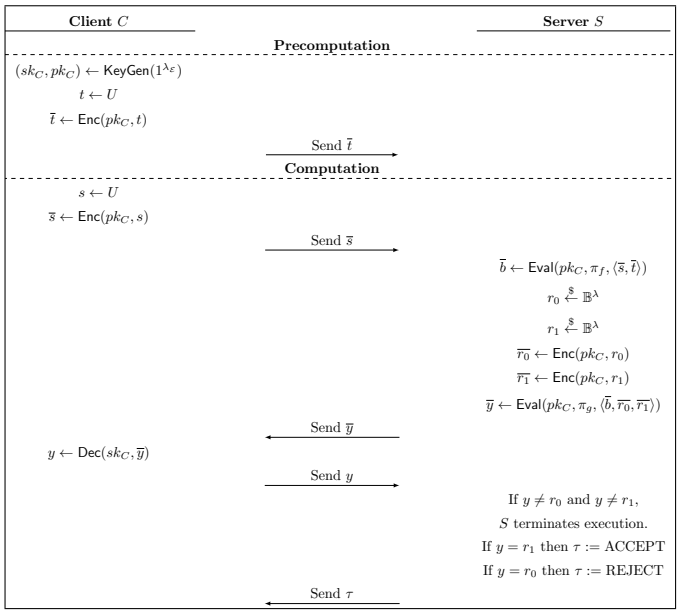

Fig. 1. Protocol summary

C. Proof of knowledge

The protocol $\mathcal{P}$ is an instance of an *interactive proof system*, as defined by Menezes et al. [34, Chapter 10]. We next show that $\mathcal{P}$ is a *proof of knowledge*, i.e. it has the properties of *completeness* and *soundness*. The following definition is adapted from [34, Chapter 10].

Definition 15 (Completeness)**.** *An authentication protocol is* complete *if, given an honest client C and an honest server S, the protocol succeeds with overwhelming probability (i.e. S accepts C's claim).*

Theorem 1 (Completeness)**.** *The protocol $\mathcal{P}$ is* complete*.*

Proof: Suppose $\mathcal{P}$ is run with an honest client C that sends a validly constructed value $\overline{s} \leftarrow \mathsf{Enc}(pk_C, s)$ for a sample s to server S. We consider two cases.

(a) Suppose the sample s matches the template t, i.e. suppose $f(s,t) = 1$. Then, by definition, $\overline{b} = \mathsf{Enc}(pk_C, 1)$, and thus $\overline{y} = \overline{r_1}$. Hence, if $y \leftarrow \mathsf{Dec}(sk_C, \overline{y})$ then $y = r_1$. Thus, S accepts C.

(b) Suppose the sample s does not match the template t, i.e. suppose $f(s,t) = 0$.
Then, by definition, $\overline{b} = \mathsf{Enc}(pk_C, 0)$, and thus $\overline{y} = \overline{r_0}$. Hence, if $y \leftarrow \mathsf{Dec}(sk_C, \overline{y})$ then $y = r_0$. Thus, S does not accept C.

That is, S accepts C if and only if the sample s matches the template t. ■

The following definition is adapted from [34, Chapter 10].

Definition 16 (Soundness)**.** *An authentication protocol is* sound *if there exists an expected polynomial time algorithm $\mathcal{A}$ with the following property: if a dishonest client C' (impersonating C) can with non-negligible probability successfully execute the protocol with S, then $\mathcal{A}$ can be used to extract from C' knowledge (essentially equivalent to C's secret) which with non-negligible probability allows successful subsequent protocol executions.*

We first need the following preliminary result.

Lemma 1. *Suppose a client C^* engages in the protocol $\mathcal{P}$ with the server S, using sample s_{C^*}, and that S accepts C^*. It follows that:*

(a) the sample s_{C^*} matches the template t held by S;
(b) C^* has access to the value r_1 chosen by S in Step 2b of $\mathcal{P}$.

Proof:

(a) Since S accepts C^*, it immediately follows from Theorem 1 that the sample s_{C^*} matches the template t.

(b) In Step 4 of $\mathcal{P}$, S accepts C^* if and only if the value y sent by C^* to S in Step 3 equals r_1. The result follows.

■

We can now give our main result.

Theorem 2. *The protocol $\mathcal{P}$ is* sound*.*

Proof: Suppose $\mathcal{P}$ is run with a dishonest client C', impersonating an honest client C, that sends a validly constructed value $\overline{s_{C'}} \leftarrow \mathsf{Enc}(pk_C, s_{C'})$ for a sample $s_{C'}$ to server S in Step 1 of $\mathcal{P}$. Suppose also that there is a non-negligible probability that C' is accepted. We need to establish that C' can, with non-negligible probability, engage in further successful protocol executions with S.

Since S accepts C' in the protocol execution with non-negligible probability, by Lemma 1 we know that C' with non-negligible probability has access to r_1, which was provided to C' in encrypted form in Step 2 of $\mathcal{P}$. Hence C' must have access to an oracle $\mathcal{O}$ that, given an input encrypted using C's public key, with non-negligible probability returns its decrypted version.

Assume a subsequent instance of the same protocol $\mathcal{P}$.

1) In Step 1, C' uses the sample $s^*_{C'} = s_{C'}$, computes $\overline{s^*_{C'}}$ using the public key of C, and sends it to S.
2) Step 2 is executed as specified by S, where the two random values chosen by S are denoted by r^*_0 and r^*_1.

Clearly $s^*_{C'}$ matches t (from (a) above), and hence the value $\overline{y^*}$ sent to C' will satisfy $y^* = r^*_1$.

3) In Step 3, C' uses oracle $\mathcal{O}$ which will, with non-negligible probability, correctly decrypt $\overline{y^*}$; that is, the value y^* output by $\mathcal{O}$ will satisfy $y^* = r^*_1$ with non-negligible probability. C' then sends y^* to S.
4) In Step 4, since $y^* = r^*_1$ with non-negligible probability, S will accept C' with non-negligible probability.

That is, there exists a PPT algorithm $\mathcal{A}$, using $\mathcal{O}$ as a subroutine, that for any instance of $\mathcal{P}$ can be used to arrange that C' will be accepted by S with non-negligible probability. ■

IV. Security properties

A. Security model

We suppose the protocol $\mathcal{P}$ is carried out in the *real world* between a challenger and an adversary. In the real world, adversaries can play the role of the client or the server. We suppose adversaries are *static*, i.e. they cannot change their role within an instance of the protocol, and cannot play both roles at the same time.

B. Privacy of the biometric data

One of the main goals of $\mathcal{P}$ is to give C (and U) assurance regarding the privacy of biometric data shared with S, i.e. all samples and templates. As we next show, this property relies on the IND-CPA security (see Definition 14) of the homomorphic encryption scheme.

Definition 17 (Privacy-preserving)**.** *If a biometric authentication protocol preserves the privacy of the biometric data of the client against an adversary (a malicious server or external party), then the protocol is* privacy-preserving.

Definition 18 (Privacy-preserving game)**.** *Suppose the pre-computation phase of the protocol $\mathcal{P}$ is run with an honest client C that sends a validly constructed encrypted value $\overline{s} \leftarrow \mathsf{Enc}(pk_C, s)$ for template $\overline{t} \leftarrow \mathsf{Enc}(pk_C, t)$ to server S. Suppose also that $\mathcal{A}$ is a PPT adversary. The* privacy-preserving game *is as follows.*

1) *A challenger chooses $i \in \mathbb{B}$ uniformly at random and generates two distinct samples $\{s_0, s_1\}$ as follows.*
   (a) *$f(s_i, t) = 1$, and*
   (b) *$f(s_{1-i}, t) = 0$.*
2) *The challenger encrypts the two samples as $\overline{s_0} \leftarrow \mathsf{Enc}(pk_C, s_0)$ and $\overline{s_1} \leftarrow \mathsf{Enc}(pk_C, s_1)$.*
3) *The challenger sends $\{\overline{s_0}, \overline{s_1}, \overline{t}\}$ to $\mathcal{A}$.*
4) *$\mathcal{A}$ outputs a pair $(\overline{s_j}, \overline{t})$, where $j \in \mathbb{B}$, and wins the game if $i = j$.*

We denote this security game by PRI-PRE$^{\mathcal{A}}_{\mathcal{P}}(1^\lambda)$, where λ is the security parameter of the homomorphic encryption scheme $\mathcal{E}$ and a win in an instance of this security game by PRI-PRE$^{\mathcal{A}}_{\mathcal{P}}(1^\lambda) = 1$.

Definition 19 (Advantage for the PRI-PRE game)**.** *Suppose that $\mathcal{P}$, λ and $\mathcal{A}$ are as in Definition 18. The* advantage of the adversary with respect to $\mathcal{P}$*, denoted by $Adv^{\mathcal{P}}_{\mathcal{A}}(\lambda)$, is defined to be:*

$$Adv^{\mathcal{P}}_{\mathcal{A}}(\lambda) \stackrel{def}{=} \left| 2 \cdot Pr\left[\textit{PRI-PRE}^{\mathcal{A}}_{\mathcal{P}}(1^\lambda) = 1\right] - 1 \right|.$$

Definition 20 (PRI-PRE security)**.** *Suppose $\mathcal{P}$, $\mathcal{A}$ and λ are as in Definition 19. If the advantage $Adv^{\mathcal{P}}_{\mathcal{A}}(\lambda)$ for $\mathcal{A}$ in the PRI-PRE game is negligible, then $\mathcal{P}$ is* PRI-PRE*, i.e. privacy-preserving.*

Theorem 3. *The protocol $\mathcal{P}$ is privacy-preserving.*

Proof: Suppose that the protocol P is not privacy-preserving, i.e. by Definition 20, there exists an adversary $\mathcal{A}$ that has a non-negligible advantage in the privacy-preserving game. By definition this means that $\mathcal{A}$ has a distinguisher $\mathcal{D}$ that distinguishes, with non-negligible probability, which of two encrypted samples $\overline{s_0}$ and $\overline{s_1}$ will match an encrypted template $\overline{t}$.

We next construct an adversary $\mathcal{B}$ against the IND-CPA security of $\mathcal{E}$. Suppose $\mathcal{B}$ generates a triple of values (s, s', t) satisfying $f(s, t) = 1$ and $f(s', t) = 0$. $\mathcal{B}$ now submits the pair (s, s') to a challenger in the IND-CPA security game. $\mathcal{B}$ receives back from the challenger the ciphertext $\overline{s^*}$, where s^* equals either s or s' (with equal probability).

$\mathcal{B}$ first computes $\overline{s}$ and $\overline{t}$ from s and t, and then runs the distinguisher $\mathcal{D}$ with inputs $\overline{s^*}$ and $\overline{s}$ as the encrypted samples and $\overline{t}$ as the encrypted template. If $\mathcal{D}$ returns $\overline{s^*}$ (which we call event e_X), then $\mathcal{B}$ outputs $(s, \overline{s^*})$ in the IND-CPA game. If $\mathcal{D}$ returns $\overline{s}$ (which we call event e_Y), then $\mathcal{B}$ outputs $(s', \overline{s^*})$ in the IND-CPA game.

To evaluate the probability that $\mathcal{B}$ wins the game, we consider two cases.

- Suppose $s^* = s$ (event e_A which has probability 0.5). Then the two encrypted samples $\overline{s^*}$ and $\overline{s}$ submitted to $\mathcal{D}$ both match the template. Hence the probability that $\mathcal{D}$ will return $\overline{s^*}$ (event e_X) = the probability it returns $\overline{s}$ (event e_Y) = 0.5.
- Suppose $s^* = s'$ (event e_B which also has probability 0.5). Then of the two encrypted samples $\overline{s^*}$ and $\overline{s}$ submitted to $\mathcal{D}$, only $\overline{s}$ will match the template. Hence the probability that $\mathcal{D}$ will return $\overline{s}$ (event e_Y) is $0.5 + p$, where $p > 0$ is non-negligible (this follows since $\mathcal{D}$ is a distinguisher).

Hence we have:

$$\Pr(e_A \wedge e_X) = \Pr(e_A)\Pr(e_X) = 0.5^2 = 0.25; \quad \text{and}$$
$$\Pr(e_B \wedge e_Y) = \Pr(e_B)\Pr(e_Y) = 0.5(0.5 + p) = 0.25 + 0.5p.$$

If e_X occurs then, by assumption, $\mathcal{B}$ outputs $(s, \overline{s^*})$ in the IND-CPA game. The probability this wins is simply $\Pr(e_A|e_X)$. Similarly, if e_Y occurs then the probability of $\mathcal{B}$ winning is $\Pr(e_B|e_Y)$. Hence, since events e_X and e_Y are mutually exclusive, the probability that $\mathcal{B}$ wins the game is:

$$\Pr(e_A|e_X)\Pr(e_X) + \Pr(e_B|e_Y)\Pr(e_Y)$$
$$= \Pr(e_A \wedge e_X) + \Pr(e_B \wedge e_Y) = 0.5(1 + p).$$

By definition the advantage for $\mathcal{B}$ is $2(0.5(1+p))-1 = 2p$, which is non-negligible since p is non-negligible. This contradicts the assumption that $\mathcal{E}$ is IND-CPA secure, and hence $\mathcal{P}$ is privacy-preserving. ∎

C. Entity authentication

We next show that Steps 2–4(a) of $\mathcal{P}$ constitute a secure authentication protocol. We follow the approach of Boyd et al. [35], based on the Bellare-Rogaway model [36], adapting a proof of Blake-Wilson and Menezes [37]. We first give an informal definition of entity authentication.

Definition 21 (Menezes et al. [34])**.** Entity authentication *is the process whereby one party is assured (through acquisition of corroborative evidence) of the identity of a second party involved in a protocol, and that the second has actually participated (i.e. is active at, or immediately prior to, the time the evidence is acquired).*

Steps 2–4(a) of $\mathcal{P}$ by design constitute a unilateral entity authentication protocol, i.e. only C authenticates to S. Before formally defining the authentication notion, we need the concept of matching conversations due to Bellare and Rogaway [36]. We suppose that an adversary $\mathcal{A}$ has access to an infinite family of oracles denoted by $\Omega^i_{a,b}$, where a and b are in the space of participants of a protocol, $i \in \mathbb{N}$ denotes the i-th instance of a protocol, and the oracle behaves as if entity a is performing protocol $\mathcal{P}$ in the belief it is communicating with the entity b for ith time.

Definition 22 (Conversation)**.** *For any oracle $\Omega^i_{a,b}$, its* conversation *for instance i is the following n-tuple*

$$K = (t_1, \alpha_1, \beta_1), (t_2, \alpha_2, \beta_2), ..., (t_n, \alpha_n, \beta_n)$$

where at time t_j, the oracle $\Omega^i_{a,b}$ received α_j and sent β_j $(1 \leq j \leq n)$.

We can now define matching conversations, again following Bellare and Rogaway [36, Definition 4.1]. We assume that the number of moves n in a protocol is odd (n even is investigated by Boyd et al. [35]).

Definition 23 (Matching conversations)**.** *Suppose P is a n-move protocol, where $n = 2k-1$ for some integer k. Run P and suppose oracles $\Omega^i_{a,b}$ and $\Omega^j_{b,a}$ engage in conversations K_i and K_j, respectively. If there exist $t_0 < t_1 < ... < t_n$ and $\alpha_1, \beta_1, ..., \alpha_k, \beta_k$ such that K_i is prefixed by*

$$(t_0, \emptyset, \alpha_1), (t_2, \beta_1, \alpha_2), (t_4, \beta_2, \alpha_3), ..., (t_{2k-2}, \beta_{k-1}, \alpha_k)$$

and K_j is prefixed by

$$t_1, \alpha_1, \beta_1), (t_3, \alpha_2, \beta_2), (t_5, \alpha_3, \beta_3), ...,$$

$$(t_{2k-3}, \alpha_{k-1}, \beta_{k-1}), (t_{2k-1}, \alpha_k, *)$$

then K_j is a matching conversation *to K_i.*

$\emptyset$ means that the oracle has no input, because it initiates the protocol; we call it an initiator oracle*; otherwise, an oracle is a* responder oracle*. $*$ means that the oracle has no output, because the protocol ends with this last move.*

Informally, this means that conversation K_j of $\Omega^j_{b,a}$ (a responder oracle) matches conversation K_i of $\Omega^i_{a,b}$ (an initiator oracle). We also need the following definition, which has been modified for the unilateral (as opposed to mutual) authentication case.

Definition 24 (No match)**.** *Suppose P is a protocol and $\mathcal{A}$ is an adversary. Suppose also that when P is run against $\mathcal{A}$ there exists an initiator oracle $\Omega^i_{a,b}$ with a conversation K_i in the ACCEPT state but no oracle $\Omega^j_{b,a}$ has a conversation matching with K_i. We denote this event by $\texttt{No-Match}^{\mathcal{A}}_P$ and its probability by Pr($\texttt{No-Match}^{\mathcal{A}}_P$).*

These preliminaries enable us to state the following key definition. Note that this definition corresponds to the case where the protocol responder (entity b) is authenticated by the protocol initiator (entity a), i.e. in the case of protocol $\mathcal{P}$ where the server is entity a and the client is entity b.

Definition 25 (Secure unilateral authentication protocol)**.** *A protocol P is a secure unilateral entity authentication protocol if for every adversary $\mathcal{A}$:*

1) *If $\Omega^i_{a,b}$ and $\Omega^j_{b,a}$ have matching conversations, then the initiator oracle $\Omega^i_{a,b}$ accepts;*
2) *Pr($\texttt{No-Match}^{\mathcal{A}}_P$) is negligible.*

The first condition refers to completeness. The second condition says that the only way for an adversary to corrupt an honest responder oracle to the ACCEPT state is to relay the messages in the protocol without modification, i.e. an adversary can only observe and relay messages.

We can now state the main result.

Theorem 4. *If $\mathcal{E}$ is IND-CPA, then Steps 2–4(a) of $\mathcal{P}$ form a secure unilateral authentication protocol.*

Proof: Since for the purposes of the Theorem we are ignoring Steps 1, 4(b) and 4(c) of $\mathcal{P}$, the server is the protocol initiator and the client is the responder, although the reverse is true for $\mathcal{P}$ in its entirety. Suppose λ is the security parameter of the underlying homomorphic encryption scheme $\mathcal{E}$. Suppose also that Steps 2–4(a) of $\mathcal{P}$ do not form a secure authentication protocol. From Theorem 1, we know that $\mathcal{P}$ is complete, i.e. that the first condition of Definition 25 holds. Thus the second condition does not hold, i.e. there exists a PPT adversary $\mathcal{A}$ such that Pr($\texttt{No-Match}^{\mathcal{A}}_{\mathcal{P}}$) is non-negligible.

We say that $\mathcal{A}$ *succeeds* against $\Omega^i_{a,b}$ if, at the end of $\mathcal{A}$'s operation, there exists an initiator oracle $\Omega^i_{a,b}$ with a conversation K_i in the ACCEPT state but no oracle $\Omega^j_{b,a}$ has a conversation K_j matching with K_i. We denote the probability that $\mathcal{A}$ succeeds against the initiator oracle $\Omega^i_{a,b}$ by Pr($\mathcal{A}$ succeeds) $= p$. Then, by assumption, p is non-negligible. Suppose also $\mathcal{A}$ possesses the public key $pk_{\mathcal{A}}$ of a genuine client. We next construct an adversary $\mathcal{B}$ from $\mathcal{A}$ against the IND-CPA security of $\mathcal{E}$.

We consider the details of the conversation of the oracle $\Omega^i_{S,C}$. Since we only consider Steps 2–4(a) of $\mathcal{P}$, we have $n = 3$. Suppose the conversation for $\Omega^i_{S,C}$ is

$$K = (t_0, \emptyset, \alpha_1), (t_2, \beta_1, \alpha_2)$$

where at time t_0, the oracle sent α_1 and at time t_2 the oracle received β_1 and sent α_2. Then it follows that we have $\alpha_1 = \overline{y} = \mathsf{Enc}(pk_C, r_w)$ (where w is 0 or 1), $\beta_1 = y$, and $\alpha_2 = *$), where we ignore the ID token τ since its construction is independent of the design of the protocol.

Since $\mathcal{A}$ is successful against $\Omega^i_{S,C}$ with probability p, it follows that $y \in \{r_0, r_1\}$ with probability p. Since r_0 and r_1 are chosen uniformly at random for each conversation instance, and since we are also assuming that there is no matching conversation, $\mathcal{A}$ must have a means for recovering r_w from $\mathsf{Enc}(pk_C, r_w)$ which works with probability at least p. Hence $\mathcal{A}$ must have access to an oracle $\mathcal{O}$ which, when given an input encrypted using the public key of C, with non-negligible probability returns its decrypted version. However, since $\mathcal{A}$ does not have access to the private key of C, this oracle can immediately be used to construct an adversary $\mathcal{B}$ against the IND-CPA security of $\mathcal{E}$. This gives the desired contradiction and hence it follows that $\mathcal{P}$ is a secure unilateral authentication protocol. ∎

V. Implementation

The protocol has been implemented using the C/C++ Fully Homomorphic Encryption over the Torus (TFHE) library due to Chillotti et al. [38]. One feature of TFHE is that it implements *gate bootstrapping*, i.e. at each evaluated gate the bootstrapping method is executed. This enables the evaluation of arbitrary circuits on encrypted data. In practice, TFHE offers the fastest gate bootstrapping in the literature, namely of the order of 13 milliseconds per gate on a single core; however, "bootstrapped bit operations are still about one billion times slower than their plaintext equivalents" [1].

In Section II, we described a homomorphic encryption scheme as a public key encryption system. The TFHE scheme is symmetric but can easily be used in the context of $\mathcal{P}$ because it provides a pair of keys: a secret key sk and a cloud key ck. In the context of $\mathcal{P}$ (see Section III), sk is kept secret and used by the client C to encrypt and decrypt data. C sends ck to the server S during the registration phase. S is then able to compute arbitrary circuits on data encrypted under sk using ck without being able to decrypt them. For further information on the design and security of TFHE see Chillotti et al. [1].

A. Biometric matching

We chose facial recognition as the biometric method for our proof-of-concept implementation for two main reasons: it is a mature technology (see, for example, the NIST report [39]) and one that suits the homomorphic setting. For our purposes, facial samples and templates are vectors $\mathbf{x} = \langle x_1, ..., x_n \rangle \in (\mathbb{Z}_m)^n$, where $\mathbb{Z}_m$ is the set of the integers modulo m (for some m). Samples and templates are compared using Euclidean distance, as defined below.

Definition 26 (Euclidean distance)**.** *Suppose* $\mathbf{x}, \mathbf{y} \in (\mathbb{Z}_m)^n$. *The* Euclidean distance *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined to be:*

$$\Delta_{\mathbf{x},\mathbf{y}} = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}.$$

To simplify calculations, we used the square of the distance as the metric. As in the following definition, a sample and a template are deemed to match if the (square of) the distance is at most B, for some B.

Definition 27 (Match)**.** *A pair of vectors* $\mathbf{x}, \mathbf{y} \in (\mathbb{Z}_m)^n$ *are said to* match *if and only if* $(\Delta_{\mathbf{x},\mathbf{y}})^2 \leq \mathcal{B}$.

The function f, defined in §III-B, is implemented in accordance with Definition 27 as follows: $f\colon (\mathbb{Z}_m)^n \times (\mathbb{Z}_m)^n \longrightarrow \mathbb{B}$, where $f(\mathbf{x}, \mathbf{y}) = 1$ if and only if $(\Delta_{\mathbf{x},\mathbf{y}})^2 \leq \mathcal{B}$, and we assume this implementation throughout Section V. The algorithm used to implement f is given in Appendix A (see Algorithm 1).

For comparison purposes, when verifying the correctness of the implementation, we also implemented the *Manhattan distance*, defined below.

Definition 28 (Manhattan distance)**.** *Suppose* $\mathbf{x}, \mathbf{y} \in (\mathbb{Z}_m)^n$, *and let* $|z|$ *denote the absolute value of* z. *The* Manhattan distance *between* $\mathbf{x}$ *and* $\mathbf{y}$ *is defined to be:*

$$\delta_{\mathbf{x},\mathbf{y}} = \sum_{i=1}^{n} |y_i - x_i|.$$

B. Results

To obtain performance results, the implementation was run on an Ubuntu 20.04.1 LTS 64-bit machine with 8 GB of RAM and a four-core Intel(R) Core(TM) i3-6100CPU @ 3.70GHz. TFHE was used with the default parameter, which achieves 110-bit cryptographic security [38]. We chose to use biometric vectors of length 128 (i.e. $n = 128$) because it is a likely real-world value.

To obtain timing figures, we first measured the 'homomorphic' (ciphertext domain) computation times for most of the arithmetic and bit comparison subroutines given in Appendix A. For comparison purposes, we also implemented and measured the performance of all the subroutines in the plaintext domain. Table I summarises the results.

It is clear that homomorphic computations have a substantial performance cost, with an order of magnitude of at least 10^6. This finding is in line with previous work [33], despite the optimisations included in the TFHE library [19].

Building on the implementations of fundamental operations, we implemented a naive version of $\mathcal{P}$. The performance results are shown in Table II, and confirm that the current proof-of-concept implementation is certainly not practical, and needs considerable optimisation in order to be usable in practice. For comparison we also show computation results in the plaintext domain. Note that none of the performance results given in Table I include the encryption and decryption time.

TABLE I
PERFORMANCE RESULTS FOR BASIC OPERATIONS

Subroutines	Execution time on plaintexts (in nanoseconds)	Execution time on ciphertexts (in seconds)
n-bit addition	335	9
Two's complement	422	10
Absolute value	396	10
n-bit subtraction	1108	30
n-bit multiplication	2094	206
Manhattan distance	210370	5049
Euclidean distance	425022	33536

TABLE II
PERFORMANCE RESULTS FOR THE PROTOCOL $\mathcal{P}$ AND ITS UNDERLYING FUNCTIONS

Subroutines	Execution time on plaintexts (in microseconds)	Execution time on ciphertexts (in seconds)
Function f	790	34308
Function g	5	456
Protocol $\mathcal{P}$	810	34765

These results demonstrate the importance of optimising the design of an algorithm and its implementation. The performance results are not only due to the homomorphic paradigm, but also because we implemented the most naive routines without any optimisations or parallelisations. We project from those results that with an optimised and targeted implementation $\mathcal{P}$ could be practical in the real world.

To conclude, we showed that, implemented naively, homomorphic encryption does not meet the performance criteria for practical use, since a user cannot wait for a few hours to be authenticated in most (if not all) authentication use cases. Indeed, Nah [40] showed that a typical user will not tolerate a wait of more than two seconds for a web page to appear. Nonetheless, there are considerable possibilities for optimisation, and the implementation and design of $\mathcal{P}$ can be enhanced in various ways, as we next briefly discuss.

C. Possible optimisations

The most obvious improvement would be from the *algorithmics* perspective. As explained above all the subroutines are implemented in a very naïve way.

There exist various public libraries that could be used to add *parallel computing* features. One example would be a C++ library such as OpenMP. Many of the subroutines have *for* loops in which all execution instances are independent.

Finally, perhaps the most effective optimisation would be to *mix* the FHE schemes, as proposed by Boura et al. [41], [42]. Existing libraries are optimised for certain targeted homomorphic computations; the main idea is to switch between libraries, choosing the most efficient for each homomorphic computation. In our case, the arithmetic subroutines would be faster on libraries other then TFHE; however, bit comparisons are much better handled by the TFHE library. This idea is practically effective, as shown by Lou et al. [43] who present Glyph, a tool which switches between TFHE [38] and BGV cryptosystems [14].

VI. CONCLUSIONS AND FUTURE WORK

We presented the design and a proof-of-concept implementation of a novel privacy preserving authentication protocol based on fully homomorphic encryption. Human authentication is based on biometric matching, implemented in the proof-of-concept using face matching. In the implementation, all underlying operations are executed using FHE, including biometric matching, Euclidean distance computation, and integer comparison. We showed that the protocol is privacy-preserving and a secure unilateral authentication protocol if the underlying homomorphic encryption scheme is IND-CPA.

The implementation results are for a naive and unoptimised version, i.e. the worst-case scenario. However, producing it involved developing a set of elementary routines in the ciphertext domain that can be used as low-level building blocks in other applications. The results confirm that FHE is not practical in a naive worst-case model, and real-world implementations would require optimisations. However, the results suggest that, with already identified improvements, the protocol can be made ready for real-world adoption.

There are number of possible directions for future work in improving performance. First, as identified in §V-B, mixing FHE schemes to take advantage of the best of each scheme (see [42], [43]) would significantly benefit performance without compromising the IND-CPA security of the homomorphic encryption scheme. Better algorithmics and implementation design is also an obvious target for improvement. Another possibility would be to change the biometric matching paradigm. Deep Learning is known to be useful in this context, and the performance in particular for face matching has been much improved recently thanks to initiatives such as that of NIST[3]. However, when such deep learning techniques are used in combination with homomorphic encryption, only the inference phase is run homomorphically and the training phase is run on clear data (see e.g. [44], [45]). To achieve the level of security we showed in this paper with FHE, both phases need to be executed in the ciphertext domain. However, encrypting both phases may not be straightforward to achieve, as recent experience shows that it is costly [43], [46], despite improvements in making FHE practical.

REFERENCES

[1] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, pp. 34–91, 2020.

[2] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[3] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *TCC*, ser. Lecture Notes in Computer Science, vol. 3378. Springer, 2005, pp. 325–341.

[4] D. Naccache and J. Stern, "A new public key cryptosystem based on higher residues," in *CCS*. ACM, 1998, pp. 59–66.

[3]See https://www.nist.gov/speech-testimony/facial-recognition-technology-frt-0 for more details.

[5] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 1403. Springer, 1998, pp. 308–318.
[6] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 1592. Springer, 1999, pp. 223–238.
[7] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009, crypto.stanford.edu/craig.
[8] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *FOCS*. IEEE Computer Society, 2011, pp. 97–106.
[9] ——, "Lattice-based FHE as secure as PKE," in *ITCS*. ACM, 2014, pp. 1–12.
[10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 6110. Springer, 2010, pp. 24–43.
[11] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 7417. Springer, 2012, pp. 850–867.
[12] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, vol. 6056. Springer, 2010, pp. 420–443.
[13] D. Stehlé and R. Steinfeld, "Faster fully homomorphic encryption," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 6477. Springer, 2010, pp. 377–394.
[14] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *ITCS*. ACM, 2012, pp. 309–325.
[15] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *STOC*. ACM, 2005, pp. 84–93.
[16] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *CRYPTO (1)*, ser. Lecture Notes in Computer Science, vol. 8042. Springer, 2013, pp. 75–92.
[17] J. Alperin-Sheriff and C. Peikert, "Faster bootstrapping with polynomial error," in *CRYPTO (1)*, ser. Lecture Notes in Computer Science, vol. 8616. Springer, 2014, pp. 297–314.
[18] L. Ducas and D. Micciancio, "FHEW: bootstrapping homomorphic encryption in less than a second," in *EUROCRYPT (1)*, ser. Lecture Notes in Computer Science, vol. 9056. Springer, 2015, pp. 617–640.
[19] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *ASIACRYPT (1)*, ser. Lecture Notes in Computer Science, vol. 10031, 2016, pp. 3–33.
[20] C. Lu and X. Tang, "Surpassing human-level face verification performance on LFW with gaussianface," in *AAAI*. AAAI Press, 2015, pp. 3811–3819.
[21] E. Pagnin and A. Mitrokotsa, "Privacy-preserving biometric authentication: Challenges and directions," *Secur. Commun. Networks*, vol. 2017, pp. 7 129 505:1–7 129 505:9, 2017.
[22] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, "Scifi - A system for secure face identification," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2010, pp. 239–254.
[23] B. Schoenmakers and P. Tuyls, "Efficient binary conversion for paillier encrypted values," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 4004. Springer, 2006, pp. 522–537.
[24] M. Blanton and P. Gasti, "Secure and efficient protocols for iris and fingerprint identification," in *ESORICS*, ser. Lecture Notes in Computer Science, vol. 6879. Springer, 2011, pp. 190–209.
[25] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, "Packed homomorphic encryption based on ideal lattices and its application to biometrics," in *CD-ARES Workshops*, ser. Lecture Notes in Computer Science, vol. 8128. Springer, 2013, pp. 55–74.
[26] A. Abidin, E. Pagnin, and A. Mitrokotsa, "Attacks on privacy-preserving biometric authentication," in *NordSec 2014*, ser. Secure IT Systems, K. Bernsmed and S. Fischer-Hübner, Eds. Springer, 2014, pp. 293–294.
[27] J. Bringer and H. Chabanne, "An authentication protocol with encrypted biometric data," in *AFRICACRYPT*, ser. Lecture Notes in Computer Science, vol. 5023. Springer, 2008, pp. 109–124.
[28] J. R. Troncoso-Pastoriza, D. González-Jiménez, and F. Pérez-González, "Fully private noninteractive face verification," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 7, pp. 1101–1114, 2013.
[29] J. H. Cheon, H. Chung, M. Kim, and K. Lee, "Ghostshell: Secure biometric authentication using integrity-based homomorphic evaluations," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 484, 2016.
[30] V. N. Boddeti, "Secure face matching using fully homomorphic encryption," in *BTAS*. IEEE, 2018, pp. 1–10.
[31] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 144, 2012.
[32] O. Goldreich, *The Foundations of Cryptography — Volume 1: Basic Techniques*. Cambridge University Press, 2001.
[33] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and M. Strand, "A guide to fully homomorphic encryption," Cryptology ePrint Archive, Report 2015/1192, 2015, https://eprint.iacr.org/2015/1192.
[34] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.
[35] C. Boyd, A. Mathuria, and D. Stebila, *Protocols for Authentication and Key Establishment, Second Edition*, ser. Information Security and Cryptography. Springer, 2020.
[36] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 773. Springer, 1993, pp. 232–249.
[37] S. Blake-Wilson and A. Menezes, "Entity authentication and authenticated key transport protocols employing asymmetric techniques," in *Security Protocols Workshop*, ser. Lecture Notes in Computer Science, vol. 1361. Springer, 1997, pp. 137–158.
[38] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption library," August 2016, https://tfhe.github.io/tfhe/.
[39] *Facial Recognition Technology (FRT)*, National Institute of Standards and Technology (NIST), https://www.nist.gov/speech-testimony/facial-recognition-technology-frt-0. Accessed: 2020-02-06.
[40] F. F. Nah, "A study on tolerable waiting time: How long are web users willing to wait?" in *AMCIS*. Association for Information Systems, 2003, p. 285.
[41] C. Boura, N. Gama, M. Georgieva, and D. Jetchev, "Chimera: Combining ring-lwe-based fully homomorphic encryption schemes," Cryptology ePrint Archive, Report 2018/758, 2018, https://eprint.iacr.org/2018/758.
[42] ——, "CHIMERA: combining ring-lwe-based fully homomorphic encryption schemes," *J. Math. Cryptol.*, vol. 14, no. 1, pp. 316–338, 2020.
[43] Q. Lou, B. Feng, G. C. Fox, and L. Jiang, "Glyph: Fast and accurately training deep neural networks on encrypted data," in *NeurIPS*, 2020.
[44] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *CRYPTO (3)*, ser. Lecture Notes in Computer Science, vol. 10993. Springer, 2018, pp. 483–512.
[45] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. E. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 201–210.
[46] K. Nandakumar, N. K. Ratha, S. Pankanti, and S. Halevi, "Towards deep neural network training on encrypted data," in *CVPR Workshops*. Computer Vision Foundation / IEEE, 2019, pp. 40–48.
[47] H. Vollmer, *Introduction to Circuit Complexity — A Uniform Approach*, ser. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.
[48] J. L. H. Crawford, C. Gentry, S. Halevi, D. Platt, and V. Shoup, "Doing real work with FHE: the case of logistic regression," in *WAHC@CCS*. ACM, 2018, pp. 1–12.
[49] A. A. Karatsouba and Y. P. Ofman, "Multiplication of multidigit numbers on automata," *Soviet Physics — Doklady*, pp. 595—-596, 1963.
[50] F. Bourse, O. Sanders, and J. Traoré, "Improved secure integer comparison via homomorphic encryption," in *CT-RSA*, ser. Lecture Notes in Computer Science, vol. 12006. Springer, 2020, pp. 391–416.
[51] A. C. Yao, "Protocols for secure computations (extended abstract)," in *FOCS*. IEEE Computer Society, 1982, pp. 160–164.

APPENDIX

We next formally introduce notions related to circuits. For more complete versions of these definitions, see Vollmer [47].

Definition 29 (Boolean function)**.** *A* Boolean function *is a function* $f : \mathbb{B}^n \to \mathbb{B}$ *for some* $n \in \mathbb{N}$.

Definition 30 (Family of Boolean functions)**.** *A* family of Boolean functions *is a sequence* $f = (f_n)_{n\in\mathbb{N}}$*, where* f_n *is an* n*-ary Boolean function.*

Definition 31 (Basis)**.** *A* basis *is a finite set consisting of Boolean functions and families of Boolean functions.*

Informally, a *Boolean circuit* is a directed acyclic graph with internal nodes marked by elements of $\{\wedge, \vee, \neg\}$. Nodes with no in-going edges are called *input nodes*, and nodes with no outgoing edges are called *output nodes*. A node marked $\neg$ may have only one outgoing edge. Computation in the circuit begins with placing input bits on the input nodes (one bit per node) and proceeds as follows. If the outgoing edges of a node (of in-degree d) marked $\wedge$ (similarly for nodes marked $\vee$ and $\neg$) have values $v_1, v_2, ..., v_d$ then the node is assigned the value $\wedge_{i=1}^{d} v_i$. The output of the circuit is read from its output nodes. The *size* of a circuit is the number of its edges. A *polynomial-size circuit family* is an infinite sequence of Boolean circuits $\pi_1, \pi_2, ...$ such that, for every n, the circuit π_n has n input nodes and size $p(n)$, where p is a polynomial fixed for the entire family.

Definition 32 (Circuit)**.** *Let* B *be a basis. A* Boolean circuit *over* B *with* n *inputs and* m *outputs is a tuple*

$$\pi = (V, E, \alpha, \beta, \omega),$$

where (V, E) *is a finite directed acyclic graph,* $\alpha : E \to \mathbb{N}$ *is an injective function,* $\beta : V \to B \cup \{x_1, x_2, ..., x_n\}$*, and* $\omega : V \to \{y_1, y_2, ..., y_m\} \cup \{*\}$*, such that the following conditions hold:*

1) *If* $v \in V$ *has in-degree 0, then* $\beta(v) \in \{x_1, x_2, ..., x_n\}$ *or* $\beta(v)$ *is a 0-ary Boolean function (i.e. a Boolean constant) from* B*.*
2) *If* $v \in V$ *has in-degree* $k > 0$*, then* $\beta(v)$ *is a* k*-ary Boolean function from* B *or a family of Boolean functions from* B*.*
3) *For every* i*,* $1 \leq i \leq n$*, there is at most one node* $v \in V$ *such that* $\beta(v) = x_i$*.*
4) *For every* i*,* $1 \leq i \leq m$*, there is at most one node* $v \in V$ *such that* $\omega(v) = y_i$*.*

Remark 4. *A Boolean circuit* π *with* n *inputs and* m *outputs computes a Boolean function*

$$f : \mathbb{B}^n \to \mathbb{B}^m.$$

Definition 33 (Circuit family)**.** *Let* B *be a basis. A* circuit family *over* B *is a sequence* $\Pi = (\pi_0, \pi_1, \pi_2, ...)$*, where for every* $n \in \mathbb{N}$*,* π_n *is a circuit over* B *with* n *inputs. Let* f_n *be the function computed by* π_n*. Then we say that* Π *computes the function* $f : \mathbb{B}^* \to \mathbb{B}^*$*, defined for every* $w \in \mathbb{B}^*$ *by*

$$f(w) \stackrel{def}{=} f_{|w|}(w).$$

Remark 5. *For simplicity of presentation, we often abuse our notation slightly by considering circuit families* $(\pi_n)_{n\in\mathbb{N}}$*, where* π_n *has* $p(n)$ *rather than* n *input bits, for some fixed polynomial* p*.*

A. Biometric matching

Algorithm 1 implements the function f defined in §III-B.

Algorithm 1: Pseudo-code of the biometric matching f

Input : $\mathbf{x}, \mathbf{y} \in (\mathbb{Z}_m)^n$ and $\mathcal{B} \in \mathbb{Z}$
Output: $b \in \mathbb{B}$
$\Delta_{x,y} \leftarrow \sum_{i=1}^{n} (y_i - x_i)^2$;
if $\Delta_{x,y} \leq \mathcal{B}$ **then**
  $b = 1$
else
  $b = 0$
return b

B. Basic operations

As stated by Crawford et al. [48], a key step for practical homomorphic encryption is to implement basic routines and tools, e.g. binary arithmetic, and make them available for use and optimisation. We implemented the following basic arithmetic functions, needed to calculate Euclidean distance (see §V-A). In each case pseudo-code (using mainly logic) is provided below. Apart from the specified functions, we also used the bitwise routines implemented in the TFHE library[4]. All the functions are presented as they are executed in the plaintext domain, although the implementations of those routines are specific to the ciphertext domain.

1-bit addition

We denote naive binary addition by `1bit_add`. Two bits a and b are `XOR`-ed with carry; the carry is updated and returned for use in another 1-bit addition as part of n-bit addition. Algorithm 2 implements the 1-bit addition routine.

Algorithm 2: Pseudo-code of 1-bit addition

Input : $a, b, carry_{in} \in \mathbb{B}$
Output: $res, carry_{out} \in \mathbb{B}$
$res \leftarrow a$ `XOR` b `XOR` $carry_{in}$;
$carry_{out} \leftarrow (a$ `AND` $b)$ `XOR` $(a$ `AND` $carry_{in})$ `XOR` $(b$ `AND` $carry_{in})$;
return $(res, carry_{out})$

n-bit addition

We denote naive bitwise addition by `nbit_add`. This routine uses `1bit_add` and applies to all bits of two n-bit numbers. Algorithm 3 implements the n-bit addition routine.

Two's complement

We implemented subtraction as addition between a number and the two's complement of the other number. Thus, we require this subroutine. We denote by $\mathring{a}$ the two's complement of a and by `twos` the two's complement function. Algorithm 4 implements the two's complement routine.

Absolute value

The absolute value was required when calculating the Manhattan distance (see §V-A). We denote this function by

[4]A list is given at: https://tfhe.github.io/tfhe/gate-bootstrapping-api.html

Algorithm 3: Pseudo-code of n-bit addition

Input : $a, b \in \mathbb{B}^n$
Output: $res \in \mathbb{B}^{n+1}$
$carry \in \mathbb{B}$;
$carry \leftarrow 0$;
for $i \leftarrow 1$ **to** n **do**
  $(res, carry) \leftarrow \texttt{1bit\_add}(a_i, b_i, carry)$
return res

Algorithm 4: Pseudo-code of two's complement

Input : $a \in \mathbb{B}^n$
Output: $\mathring{a} \in \mathbb{B}^{n+1}$
for $i \leftarrow 1$ **to** n **do**
  $\mathring{a}_i \leftarrow a_i$ XOR 1
$\mathring{a} \leftarrow \texttt{nbit\_add}(\mathring{a}, 1)$;
return $\mathring{a}$

abs. MSB(a) below outputs the Most Significant Bit of a. Algorithm 5 implements the absolute value routine.

Algorithm 5: Pseudo-code of absolute value

Input : $a \in \mathbb{B}^n$
Output: $|a| \in \mathbb{B}^n$
$mask, tmp \in \mathbb{B}^n$;
for $i \leftarrow 1$ **to** n **do**
  $|mask|_i \leftarrow \text{MSB}(a)$
$tmp \leftarrow \texttt{nbit\_add}(a, mask)$;
for $i \leftarrow 1$ **to** n **do**
  $|a|_i \leftarrow tmp_i$ XOR $mask_i$
return $|a|$

n-bit subtraction

As explained above, when subtracting b from a, the routine adds a to $\mathring{b}$. We denote this routine by sub. After the addition, if the final carry denoted $carry_f$ over the sum is 1, the result is positive and remains unchanged. If not (i.e. $carry_f$ is 0), the result is negative thus its two's complement is returned. The cost of a branching condition being too great, a sequence of instructions is used instead, leading to the genuine result. Algorithm 6 implements the subtraction routine.

Algorithm 6: Pseudo-code of n-bit subtraction

Input : $a, b \in \mathbb{B}^n$
Output: $res, tmp, var \in \mathbb{B}^{n+1}$
$\mathring{b} \leftarrow \texttt{twos}(b)$;
$tmp \leftarrow \texttt{nbit\_add}(a, \mathring{b})$;
for $i \leftarrow 1$ **to** $n+1$ **do**
  $var_i \leftarrow carry_f$
$res \leftarrow \texttt{twos}(tmp$ AND $\overline{var})$ OR $(tmp$ AND $var)$;
return res

Multiplication

We implemented a naive multiplication algorithm; however, other algorithms have smaller complexity, e.g. Karatsuba multiplication [49]. Implementing this is left for future work. Algorithm 7 implements this routine denoted by mult.

Algorithm 7: Pseudo-code of n-bit multiplication

Input : $a, b \in \mathbb{B}^n$
Output: $res \in \mathbb{B}^{2n}$
$res \leftarrow 0^{2n}$;
$tmp \leftarrow 0^{2n}$;
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** n **do**
    $tmp_{i+j} \leftarrow a_j$ AND b_i
  $res \leftarrow \texttt{nbit\_add}(res, tmp)$
return res

1-bit comparison

Secure integer comparison has been studied for a long time [50]. The first solution was probably that of Yao [51] through the Millionaires' problem. Integer comparison is very costly in terms of computation when using FHE; this is why it is usually better to avoid computing such an operation. Moreover it can also be difficult to articulate in ciphertext spaces. In TFHE, this operation is done using logic gates, and a proposal for implementation is published in the tutorial section in [38]. The authors use a MUX gate in their function, which is exhaustively explained in [1, Section 3.4]. The authors provide two functions, one to compare bitwise and the other to compare two binary numbers, denoted by 1bit_comp and nbit_comp, respectively. We adapted their function in our implementation. Algorithm 8 implements the 1-bit comparison routine.

Algorithm 8: Pseudo-code of 1-bit comparison

Input : $a, b, carry \in \mathbb{B}$
Output: $res \in \mathbb{B}$
$res \leftarrow$ MUX $(a$ XNOR $b, carry, a)$;
return res

n-bit comparison

This routine performs a comparison of two n-bit numbers using the previous routine. Algorithm 9 implements the n-bit comparison routine.

Algorithm 9: Pseudo-code of n-bit comparison

Input : $a, b \in \mathbb{B}^n$
Output: $res \leftarrow a?b : carry$
$carry, tmp \in \mathbb{B}$;
$carry \leftarrow 0$;
for $i \leftarrow 1$ **to** n **do**
  $tmp \leftarrow \texttt{1bit\_comp}(a_i, b_i, carry)$
for $i \leftarrow 1$ **to** n **do**
  $res \leftarrow \texttt{MUX}(carry, b_i, a_i)$
return res