A Secure Authentication Protocol for Cholesteric Spherical Reflectors Using Homomorphic Encryption

Mónica P. Arenas[1(✉)], Muhammed Ali Bingol[2(✉)], Hüseyin Demirci[1], Georgios Fotiadis[1], and Gabriele Lenzini[1(✉)]

[1] SnT, University of Luxembourg, Esch-sur-Alzette, Luxembourg
{monica.arenas,huseyin.demirci,georgios.fotiadis,
gabriele.lenzini}@uni.lu
[2] Cyber Technology Institute, De Montfort University, Leicester, UK
muhammed.bingol@dmu.ac.uk

Abstract. Sometimes fingerprint-like features are found in a material. The exciting discovery poses new challenges on how to use the features to build an object authentication protocol that could tell customers and retailers equipped with a mobile device whether a good is authentic or fake. We are exactly in this situation with Cholesteric Spherical Reflectors (CSRs), tiny spheres of liquid crystals with which we can tag or coat objects. They are being proposed as a potential game-changer material in anti-counterfeiting due to their unique optical properties. In addition to the problem of processing images and extracting the minutiæ embedded in a CSR, one major challenge is designing cryptographically secure authentication protocols. The authentication procedure has to handle unstable input data; it has to measure the distance between some reference data stored at enrollment and noisy input provided at authentication. We propose a cryptographic authentication protocol that solves the problem, and that is secure against semi-honest and malicious adversaries. We prove that our design ensures data privacy even if enrolled data are leaked and even if servers and provers are actively curious. We implement and benchmark the protocol in Python using the Microsoft `SEAL` library through its Python wrapper `PySEAL`.

Keywords: Anti-counterfeiting · Cholesteric Spherical Reflectors · Image processing · Authentication · Biometric hashing · Homomorphic encryption

1 Introduction

To verify that an object is authentic, one has to seek in the object for certain features that are hard to reproduce in a counterfeited copy. To verify that an object is exactly that object and not another of the same family, those features must be unique. The identifying features can be derived from the object itself as it happens for our fingerprints [25]; or, they can be borrowed because of a tag,

L. Batina and J. Daemen (Eds.): AFRICACRYPT 2022, LNCS 13503, pp. 425–447, 2022.
https://doi.org/10.1007/978-3-031-17433-9_18

a coating, or a watermarking applied onto the object. The features have to be extracted, recognized, and authenticated in both cases.

A common process is to extract the features from an image of the object or of its tag, coat, or watermark, and verify them (authentication phase) by measuring how similar they are from a reference version of the features previously extracted from the same object and securely stored (enrollment phase). A "match" means authentication, a "mismatch" a non-authentication[1]. The process sounds simple, but the devil is in the details. The nature of the identifying features is extremely variable and strongly object-dependent. The analysis of features from an image requires dedicated minutiæ extraction procedures [5,21]. The authentication, i.e., matching a freshly input and the securely stored version, must be robust to noise. Furthermore, if the database that holds the original set of reference features is stored remotely, e.g., if the authentication works as a service, the process needs a security protocol that foresees the different drawbacks caused by security vulnerabilities and authorization attacks [18].

Needless to say, there is no general authentication protocol that works across different objects. Here, we address the problem of a particular material with fingerprint-like features that can be used to create tags and, to a certain extent, to coat objects. The material is composed of *Cholesteric Liquid Crystals (CLCs)*, the liquid we know from our digital device's screens, made in a spherical shape. Called CSRs in [12,14], spherical CLCs reflect light creating patterns (see Fig. 1) that have been argued to be unique and physically unclonable [12,24]. Although the nature of CSR is different from biometric data, they converge in their noisy and unique behavior. A reliable and secure process needs to be implemented to reconstruct binary strings from noise inputs. Once reliable information extraction is granted, it must be protected. Homomorphic encryption can ensure the process of the data in the encrypted domain, which has proven effective in ensuring the privacy of sensitive data [28]. CSRs tags have been demonstrated as a promising material in applications such as anti-counterfeiting, track-and-tracing, authentication systems, and fiducial markers [32].

Contributions. We showed how to extract minutiæ from CSR images in previous works [2,3]. However, a procedure was still missing to obtain robust information from the extracted features and to develop a cryptographically secure and effective authentication protocol. In this paper, we design a cryptographic authentication protocol for CSRs, which securely authenticates objects with a coat or tag containing CSRs.

Specifically, the contribution of this paper can be viewed from both theoretical and practical angles. More concretely, we have developed the following procedures: *i) features embedding* and *robust feature identification* of the minutiæ to increase information stability in the presence of noise; *ii)* cryptographic protocol for remote authentication that relies on *biometric hashing* and on *homomorphic encryption*; which we proved to be robust to attacks from semi-honest and malicious adversaries; and *iii)* formulation of the Hamming distance in the encrypted

[1] Match and mismatch will be defined over the features metric space.

domain, based on which authentication is achieved by comparing two homomorphically encrypted bitstrings. Our proposed protocol is accompanied by a proof-of-concept implementation in Python, using the `PySEAL` library for homomorphic encryption[2] which implements the Brakerski/Fan-Vercauteren (BFV) (somewhat) homomorphic encryption scheme [9,11]. Based on our implementation, we provide extensive performance benchmarking results, focusing especially on the core homomorphic operations that are required for computing the Hamming distance on encrypted data. The source code of our implementation is available at https://gitlab.uni.lu/irisc-open-data/2021-nofakes.

To the best of our knowledge, there is no previous work that combines CSR's optical responses in a cryptographic authentication protocol, which is unsurprising since the use of CSRs in security is quite recent and largely unexplored. But, our research has a wider application than just in relation to CSRs. The minutiæ extraction process and the protocol that we design are applicable to several anti-counterfeiting technologies. As we will see in Sect. 2—where we briefly recall how a CSR response looks like—our information extraction strategy assumes that the identifying information contained in an image (i.e., the minutiæ) are *colored blobs* (i.e., connected pixels forming a circular shape) "randomly"[3] distributed in a bi-dimensional space. Our information extraction and authentication protocol are built assuming vectors of blobs, internally represented as a list of tuples whose elements model the blobs as circles (i.e., centers and radii) and colors. Thus, what we have developed, modeled, and implemented here works for any watermarking, steganography, or optical unclonable functions that embody colored blobs as minutiæ.

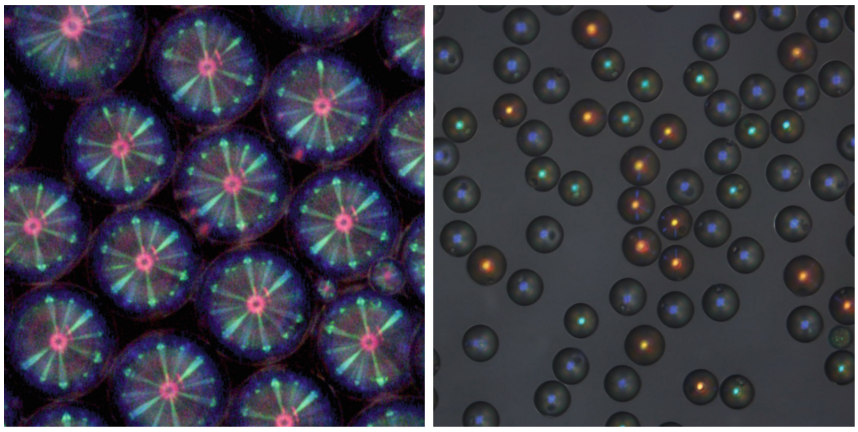
Outline. Section 2 recalls the necessary background for CSRs and discusses the related work. In Sect. 3, we introduce our design, showing a robust-to-noise scheme to extract binary information from CSR images (Sect. 3.1), and an innovative cryptographic protocol design for object authentication. It uses tools such as biometric hashing to create stable bitstrings of information and homomorphic encryption to securely compare them in the authentication phase (Sect. 3.2). Section 4 discusses the security of the design under malicious and semi-honest adversary models enabling security against outsider attackers and insider provers, respectively. In Sect. 5, we demonstrate our implementation details and results of the proposed authentication protocol. Finally, Sect. 6 concludes the work and discusses open questions and future research.

2 Background and Related Work

Cholesteric Liquid Crystals and Reflectors. CLCs have been intensively studied in chemistry and matter physics for their optical properties and versatility [12,13,31]. Here, we recall the very basics necessary to understand our contribution. What is important to know is that "CSR" is a name to indicate

[2] https://github.com/Lab41/PySEAL.

[3] "Random" is intended informally, meaning "in a way that we cannot anticipate".



(a) (b)

Fig. 1. CSRs images acquired with (a) a DinoLite microscope and (b) a professional microscope.

CLCs in a spherical shape. The spheres —droplets, if full of liquid, or shells, if cave with the liquid only on the external surface— are tiny: they measure between 10 μm and 300 μm in diameter. CSRs can be produced in a large number, and they are delivered in a medium, e.g., a piece of plastic or a dried varnish, which we call a *CSR tag*. The absolute and the relative position of the CSRs in the tag is unpredictable because it is not possible to control that variable at the production phase. When illuminated, CSRs reflect the light, creating a peculiar colorful pattern. We can capture that image with a microscope, e.g., one of those we can plug into a computer with a USB cable. Such images (see Fig. 1-*a*) are what we assume to have as input. From the images, we can extract blobs, for instance, by following the procedure described in [2,3], using image processing.

Cryptographic Techniques in Object Authentication. Regarding object authentication and anti-counterfeiting, two topics closely related to one another and addressed by academic and industrial research alike, the literature is vast. What is relevant to position our contribution is the works that use cryptographic techniques such as those we also employ, namely biometric hashing and homomorphic encryption.

Before applying any cryptographic technique, we need to ensure the extraction of reliable information. For this purpose, we applied some techniques reported in the literature [35]. Tuyls *et al.* [35] proposed a helper data that guarantees the extraction of a unique and robust string from fingerprint biometrics data during the enrollment and authentication phases. As this helper data was stored in a database (public), the authors hashed it to keep the reference data sheltered from somebody that has access to the database. Once reliable information extraction is granted, it must be protected. Fully Homomorphic Encryption (FHE) can ensure to process the data in the encrypted domain. Again, research in biometrics is rich in inspiring results. Lattice-based FHE have been proved

effective in ensuring privacy in biometric iris authentication [34], and private face recognition [8]. Homomorphic probabilistic encryption, a version compatible with the ISO/IEC 24745 standard on biometric data protection, has been used in a general framework for multi-biometric template protection [16].

Pradel and Mitchell [28] have recently proposed a privacy-preserving biometrics authentication protocol based on FHE, where a user's biometric sample is matched against an encrypted biometric template held by a remote system. This remote authentication protocol protects the privacy of users' sensitive data. They also conducted a proof-of-concept implementation using the TFHE[4] library and analyzed it in terms of efficiency. The underlying basic operations needed to execute the biometric matching are well described. Still, the performance results from the implementation show how complex it is to make FHE practical in this context. Pradel and Mitchell's implementation, if improved and optimized, could be used for real-world applications.

3 The Proposed Protocol

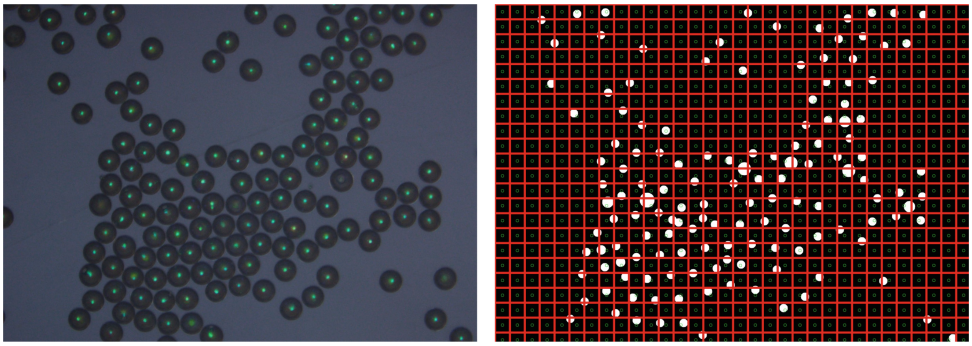
We propose an object authentication protocol that consists of two main parts where in each part, a set of various functionalities is deployed. The first part copes with data instability due to macro problems in image acquisition and processing. This is described in Sect. 3.1. The second part, which implements the authentication protocol and its enrollment and verification phases, is robust to noise at the level of bitstring values introduced when we extract and encode minutiæ from the inputs. It relies on biometric hashing and ensures the security of the protocol by storing encrypted data using homomorphic encryption. Authentication is achieved by comparing two homomorphically encrypted bitstrings obtained in the enrollment and authentication phases, in terms of their Hamming distance which is computed in the encrypted domain. When the Hamming distance is below a predefined threshold, the authentication is successful, otherwise it fails. Our protocol is introduced in Sect. 3.2, while the enrollment and authentication phases are described in Sects. 3.3 and 3.4, respectively.

3.1 Extraction of Robust Features

One necessary step must be done before we can describe any cryptographic protocol, which is to digitalize the information that is embedded into a CSR. The contact point with the reality of CSR is a picture of a CSR's optical response, an object that we indicate as $[\texttt{CSR}]$. Such pictures are noisy. Retaken pictures from the same CSR, i.e., $\{[\texttt{CSR}]'_1, \ldots, [\texttt{CSR}]'_m\}$, are slightly different one from the other due to uncontrollable factors, such as ambient light, read-out process, and variances in the sensor of the digital camera.

The reader can refer to the original work [3] for details. Still, here we recall that any $[\texttt{CSR}]$ contains identifying minutiæ (technically, blobs) that can be identified and extracted by processing the image. These minutiæ are modeled and

[4] https://tfhe.github.io/tfhe/.



(a) (b)

Fig. 2. (a) an example of [CSR] and (b) its extracted minutiæ embedded on a grid.

represented as a *list of colored circles*. We abstract the *feature extraction* process as a function Extract, whose domain is the set of all CSR images. An output $\mathsf{Extract}([\mathtt{CSR}]_i)$ is a list $\boldsymbol{\omega}_i = (\omega_i(1), \ldots, \omega_i(n))$. Each element ω in that list is a pair $(\omega.c, \omega.rgb)$, where the first item is a circle and the second its color. A circle c is described in terms of coordinates of its center, $c.x$ and $c.y$, and radius, $c.r$.

The features extraction algorithm operates on noisy inputs and returns noisy outputs: depending on i, $\mathsf{Extract}([\mathtt{CSR}]_i)$ varies not only in the numerical values but also in the length of the outputted list, that is, in the number of blobs detected. We have to correct and unify that high-level noise. Abstracting from any detail, we represent this process with two functions:

FeaturesEmbedding– using an $N \times M$ mesh grid (see Fig. 2-b), it returns a list of *exactly* $n = N \times M$ elements $\boldsymbol{\omega} = (\omega(1), \ldots, \omega(n))$. Each element is either a blob, the blob found in that position in the grid, or the undefined element $\perp$, if no blob is found there.

RobustPositions– taking a list of extracted embedded features, $(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m)$, from m sample images $\{[\mathtt{CSR}]'_i\}$ for $i \in \{1, \ldots, m\}$, each $\boldsymbol{\omega}_i$ returns a set $K \subseteq \{1, \ldots, n\}$ of *reliable positions* i.e., indexes where, in all the m extracted features $\{(\omega_i(1), \ldots, \omega_i(n))\}$, there is either all blobs or all $\perp$. It returns also a list $\boldsymbol{\omega} = \{\omega(j)\}_{j \in K}$ of *reliable features* which are obtained by "averaging"[5] the values across the samples for all the reliable positions.

We use the two functions to pre-process CSR images in both authentication and enrollment. We call ReliableFeatures the front-end function, so defined:

$$\begin{aligned} &\mathsf{ReliableFeatures}(\{[\mathtt{CSR}]_1, \ldots, [\mathtt{CSR}]_m\}) := \\ &\quad \mathsf{RobustPositions}(\\ &\qquad \mathsf{FeaturesEmbedding}(\\ &\qquad\quad (\mathsf{Extract}([\mathtt{CSR}]_1), \ldots, \mathsf{Extract}([\mathtt{CSR}]_m)))) \end{aligned} \tag{1}$$

[5] It is a means on the values of blobs, and it returns $\perp$ when one of the elements is $\perp$.

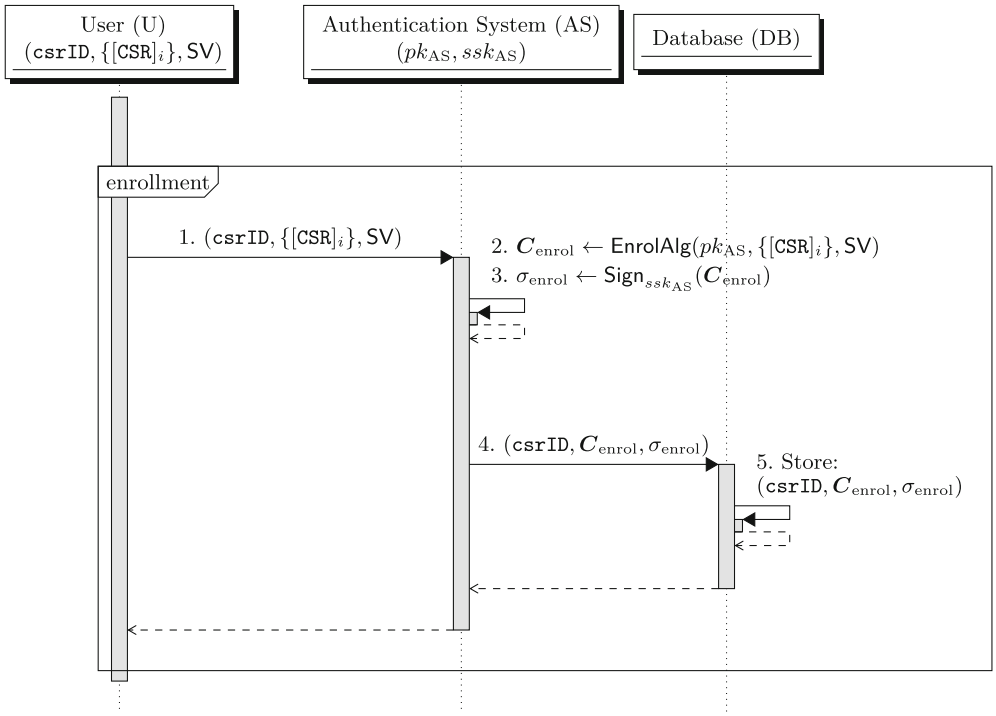


Fig. 3. Message sequence chart of the enrollment phase.

The output of the function ReliableFeatures is a vector $\boldsymbol{\omega} = (\omega(1), \ldots, \omega(k))$, of reliable features.

3.2 Protocol Description

We present an authentication protocol that uses CSR images and a biometric hashing (in short, *biohashing*) mechanism. Biohashing schemes are simple yet powerful biometric template protection methods [4,20,27,33].

Our protocol is divided into two main phases: *enrollment*, where the reference data from each CSR image are collected, and *authentication*, where it is verified whether a specific CSR image matches the one that has been enrolled.

We distinguish three entities that participate in the protocol: namely the *User*, the *Authentication System*, and an outsourced *Database*, with the following roles and responsibilities:

User (U): holds the CSR and provides the CSR images, $[\texttt{CSR}]_i$[6]. CSR comes with a numerical identifier, csrID, which U is able to read, and with a SV (i.e., a PIN/password, QR code, RFID data, or item image), a token that we intend to use for second-factor authentication.

[6] Here, the number of pictures that a User takes in the enrollment and authentication can be defined by the process. We used five images in our implementation, see Sect. 5.

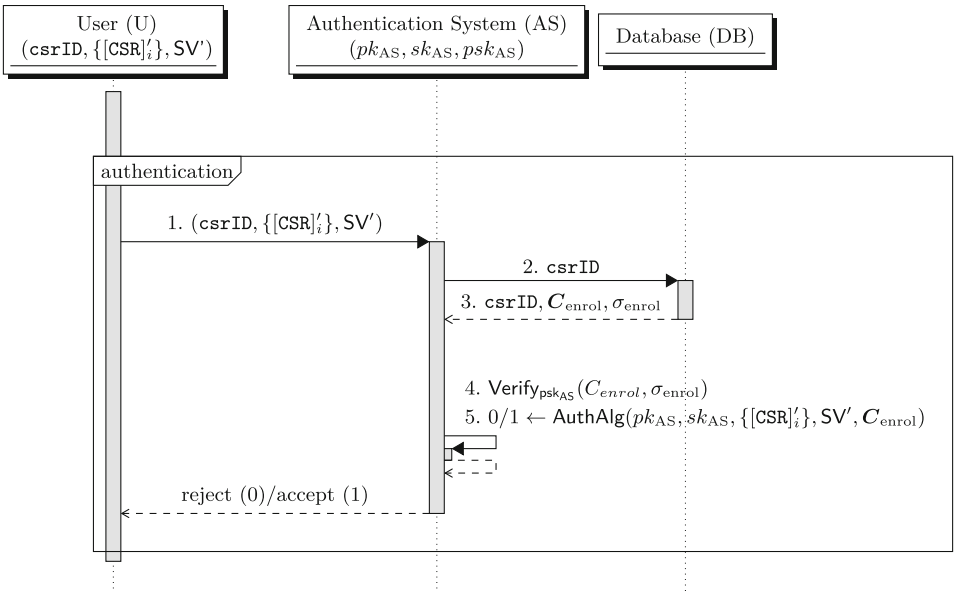


Fig. 4. Message sequence chart of the authentication phase.

Authentication System (AS): it is an entity that is responsible for extracting reliable features and carrying out the enrollment and authentication procedures. It can be a trusted hardware, for instance, in the device operated by the User. In the enrollment phase, it receives from the User CSR images and other data that it needs to calculate a binary vector $\boldsymbol{V}_{\text{enrol}}$. The vector is encrypted homomorphically and stored in an outsourced database for later uses (see Fig. 3). In the *authentication phase*, it gets from the User an authentication request, with fresh CSR images, the object identifier, and the second-factor value, which AS processes to produce a new binary vector $\boldsymbol{V}_{\text{auth}}$. The encryption of this value is compared against the one stored, which is retrieved thanks to the object identifier (see Fig. 4). The comparison is accomplished by computing the Hamming distance of the plaintext vectors $\boldsymbol{V}_{\text{enrol}}$ and $\boldsymbol{V}_{\text{auth}}$ in the encrypted domain, using the homomorphic operations. A successful authentication implies that the Hamming distance is below an acceptable threshold, while a failure indicates that the Hamming distance exceeds that threshold. The enrollment and authentication phases are discussed in more detail in Sects. 3.3 and 3.4, respectively. We further assume that the AS is equipped with a public/private key pair $(pk_{\text{AS}}, sk_{\text{AS}})$, to be used in the homomorphic cryptosystem, and with a public/private signing key pair $(psk_{\text{AS}}, ssk_{\text{AS}})$, to be used in a digital signature scheme.

Outsourced database (DB): it is the database of encrypted features maintained by an outsourced entity. For instance, it can be "in the Cloud". It is responsible for securely storing the data received from the AS in the enrollment phase. In contrast, in the authentication phase, it is responsible for retrieving the necessary information requested from the AS and transmitting this information to the AS.

3.3 Enrollment Phase

The first step in the CSR authentication protocol is the enrollment phase, in which the User sends data necessary for the enrollment for the first time to the AS. Without loss of generality, we assume that the User performs the enrollment. In reality, it can be another role, for instance, the producer of the object that carries the CSR or the producer the CSR. Then the AS executes a series of actions which are described in Algorithm 1. In this phase CSR images (i.e., a set of images $\{[\texttt{CSR}]_i\}$) are taken for the first time and converted by the AS into a CSR template $\boldsymbol{\omega} = (\omega(1), \ldots, \omega(k))$, according to the procedure ReliableFeatures, described in Sect. 3.1. Then AS executes the biohashing scheme of Ngo *et al.* [27] based on random projection. It consists of two steps:

Random Projection (RP)– In this step, an identically distributed pseudo RP matrix, $\boldsymbol{R} \in \mathbb{R}^{n \times k}$, is generated from a Gaussian distribution with zero mean and unit variance to transform the reliable features vector built from the CSR images provided by the User. The elements of RP can be generated from a Pseudorandom Number Generator (PRNG), with a seed derived from the User's SV value, and Gram-Schmidt (GS) procedure [29] is applied to obtain an orthonormal projection matrix. After that, the RP matrix is multiplied with the CSR template $\boldsymbol{\omega}$ to project this vector onto an n-dimensional intermediate vector $\boldsymbol{I} = \boldsymbol{R}\boldsymbol{\omega}$ where $\boldsymbol{I} \in \mathbb{R}^{n \times 1}$.

Quantization– This step involves the binarization of the elements of the intermediate vector $\boldsymbol{I}$ with respect to a quantization threshold β. The User's SV is mapped to an n-bit string $\boldsymbol{h} = H(\mathsf{SV})$, where $\boldsymbol{h} = (h(1), \cdots, h(n)) \in \{0,1\}^n$ and H is a cryptographic hash function. Then the enrollment biohash vector $\boldsymbol{V}_{\text{enrol}} = (V_{\text{enrol}}(1), \cdots, V_{\text{enrol}}(n))$ of the User is constructed, via the relation:

$$V_{\text{enrol}}(i) = \begin{cases} 1 \oplus h(i), & \boldsymbol{I}(i) \geq \beta \\ h(i), & \text{otherwise} \end{cases},$$

where $V_{\text{enrol}}(i) \in \{0,1\}$. Without loss of generality, β can be selected as the mean value of the intermediate vector $\boldsymbol{I}$ or the sign operator (i.e., 0), depending on the design of the system.

After the quantization, the User's biohash vector $\boldsymbol{V}_{\text{enrol}}$ is encrypted bit-by-bit with the public key pk_{AS} of the AS, using a secure homomorphic encryption scheme. The result is a vector $\boldsymbol{C}_{\text{enrol}} = (C_{\text{enrol}}(1), \cdots, C_{\text{enrol}}(n))$ which is composed of n ciphertexts. Finally, the AS signs the ciphertext $\boldsymbol{C}_{\text{enrol}}$ with its secret signing key ssk_{AS}, using a standard secure digital signature algorithm (such as ECDSA [19]), and stores the triple $(\texttt{csrID}, \boldsymbol{C}_{\text{enrol}}, \sigma_{\text{enrol}})$ in an outsourced DB. We note here that once the AS transmits the data to the DB and the enrollment is completed, it clears all the residual data from its memory. In other words, the AS does not store any processed information related to the User, such as CSR images, the SV or $\boldsymbol{V}_{\text{enrol}}$ which are considered sensitive information. On the contrary, we assume that the private parts of the encryption and signing key pairs are securely stored, for example, in a Hardware Security Module (HSM).

Algorithm 1: Enrollment algorithm (EnrolAlg)

Input: pk_{AS}, $\{[\texttt{CSR}]_i\}$, SV.
Output: $\boldsymbol{C}_{\text{enrol}}$.
System Parameters: The RP matrix $\boldsymbol{R} \in \mathbb{R}^{n\times k}$, a threshold β.

Feature extraction: $\boldsymbol{\omega} \leftarrow \mathsf{ReliableFeatures}(\{[\texttt{CSR}]_i\})$
Compute the intermediate vector: $\boldsymbol{I} \leftarrow \boldsymbol{R\omega}$, where $\boldsymbol{I} \in \mathbb{R}^{n\times 1}$
Compute: $\boldsymbol{h} \leftarrow H(\mathsf{SV})$ where $\boldsymbol{h} = (h(1), \ldots, h(n))$ and $h(i) \in \{0, 1\}$
$\boldsymbol{C}_{\text{enrol}} \leftarrow \{\}$
for $i \in \{1, \ldots, n\}$ **do**
  if $\boldsymbol{I}(i) \geq \beta$ **then**
    $V_{\text{enrol}}(i) \leftarrow 1 \oplus h(i)$
  else
    $V_{\text{enrol}}(i) \leftarrow h(i)$
  Compute: $C_{\text{enrol}}(i) \leftarrow \mathsf{HEnc}_{pk_{\text{AS}}}(V_{\text{enrol}}(i))$
  Append $C_{\text{enrol}}(i)$ to $\boldsymbol{C}_{\text{enrol}}$
return $\boldsymbol{C}_{\text{enrol}}$

It is worth saying that, the User's SV also provides privacy-friendly revocation ability of the CSR's biohash in case it is needed (due to customization or membership termination) without revealing CSR data. In addition, the same CSR template of an item/subject can be utilized in various recognition systems without violating the privacy, as two biohash vectors $\boldsymbol{V}_{\text{enrol}}$ of the CSR with different User SV will be unlinkable.

3.4 Authentication Phase

The authentication phase is initiated by the User who sends the freshly taken set of images $\{[\texttt{CSR}]'_i\}$ from the object holding the CSR, the potentially different secret value SV′ and object identifier `csrID` to the AS. The AS requests from the DB to search for an entry with CSR identifier `csrID`. If such an entry exists, the DB transmits to the AS the $\boldsymbol{C}_{\text{enrol}}$ and σ_{enrol} that correspond to the identifier `csrID`. Before proceeding to the authentication, the AS verifies the signature with the public signing key psk_{AS}. If the signature is not verified, the AS returns a failure message to the User. Once the signature is verified, the AS executes the authentication Algorithm 2, on input its public and private keys $pk_{\text{AS}}, sk_{\text{AS}}$, the images $\{[\texttt{CSR}]'_i\}$, the secret value SV′, and the ciphertext $\boldsymbol{C}_{\text{enrol}}$.

The first step in the authentication algorithm is to extract a reliable CSR template $\boldsymbol{\omega}' \leftarrow \mathsf{ReliableFeatures}(\{[\texttt{CSR}]'_i\})$ following the procedure described in Sect. 3.1. Using $\boldsymbol{\omega}'$ and SV′ the authentication algorithm creates a ciphertext vector $\boldsymbol{C}_{\text{auth}}$ that corresponds to the CSR′ and SV′, in the same way as in the enrollment phase. More concretely, $\boldsymbol{C}_{\text{auth}} = (C_{\text{auth}}(1), \cdots, C_{\text{auth}}(n))$ is the result of the bit-by-bit homomorphic encryption of the biohash vector $\boldsymbol{V}_{\text{auth}} = (V_{\text{auth}}(1), \cdots, V_{\text{auth}}(n))$, which is obtained through the quantization process, where $V_{\text{auth}}(i) \in \{0, 1\}$ for each $i = 1, \ldots, n$.

Algorithm 2: Authentication algorithm (AuthAlg)

Input: pk_{AS}, sk_{AS}, $\{[\texttt{CSR}]'_i\}$, SV', $\boldsymbol{C}_{\text{enrol}}$.
Output: 0 (reject) or 1 (accept).
System Parameters: The RP matrix $\boldsymbol{R} \in \mathbb{R}^{n \times k}$, thresholds β and ε.

Feature extraction: $\boldsymbol{\omega}' \leftarrow \mathsf{ReliableFeatures}(\{[\texttt{CSR}]'_i\})$
Compute the intermediate vector: $\boldsymbol{I}' \leftarrow \boldsymbol{R}\boldsymbol{\omega}'$, where $\boldsymbol{I}' \in \mathbb{R}^{n \times 1}$
Compute: $\boldsymbol{h}' \leftarrow H(\mathsf{SV}')$ where $\boldsymbol{h}' = (h'(1), \ldots, h'(n))$ and $h'(i) \in \{0, 1\}$
$\boldsymbol{C}_{\text{auth}} \leftarrow \{\}$
for $i \in \{1, \ldots, n\}$ **do**
    if $\boldsymbol{I}'(i) \geq \beta$ **then**
        $V_{\text{auth}}(i) \leftarrow 1 \oplus h'(i)$
    else
        $V_{\text{auth}}(i) \leftarrow h'(i)$
    Compute: $C_{\text{auth}}(i) \leftarrow \mathsf{HEnc}_{pk_{\text{AS}}}(V_{\text{auth}}(i))$
    Append $C_{\text{auth}}(i)$ to $\boldsymbol{C}_{\text{auth}}$
Compute the encrypted Hamming distance $\text{HD}_E(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})$ using Eq. (2)
Compute: $\text{HD}(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}}) \leftarrow \mathsf{HDec}_{sk_{\text{AS}}}(\text{HD}_E(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}}))$
if $\text{HD}(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}}) \leq \varepsilon$ **then**
    return 1
else
    return 0

For user authentication, the AS computes the Hamming distance of the two plaintext vectors $\boldsymbol{V}_{\text{enrol}}$ and $\boldsymbol{V}_{\text{auth}}$, using only the corresponding ciphertexts $\boldsymbol{C}_{\text{enrol}}$ and $\boldsymbol{C}_{\text{auth}}$, i.e., in the encrypted domain. This is denoted as:

$$\begin{aligned} \text{HD}_E(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}}) &= \mathsf{HEnc}_{pk_{\text{AS}}}(\text{HD}(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})) \\ &= \mathsf{HEnc}_{pk_{\text{AS}}}\left(\sum_{i=1}^{n} (V_{\text{enrol}}(i) \oplus V_{\text{auth}}(i))\right). \end{aligned}$$

In the above relation, $\text{HD}(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})$ is the Hamming distance of the two bitstrings $\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}}$, while $\text{HD}_E(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})$ is the Hamming distance of the two bitstrings in the encrypted domain and $\mathsf{HEnc}_{pk_{\text{AS}}}$ is the homomorphic encryption with the public key pk_{AS}. In general, the Hamming distance computation in the encrypted domain is not straightforward, and dedicated formulas need to be considered, which largely depend on the underlying homomorphic cryptosystem used. Further, doing this computation efficiently is challenging since such formulas typically require various homomorphic operations, including homomorphic multiplications, which are usually very expensive.

Once the encrypted Hamming distance is computed, the AS decrypts it to obtain the Hamming distance $\text{HD}(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})$, using its secret key sk_{AS} and checks whether the resulting Hamming distance falls within an accepted margin that is predefined by the threshold ε. If this is the case, then the authentication of the User is successful; otherwise, it fails.

Computation of the Encrypted Hamming Distance. Our protocol follows the approach described in [36] for computing the encrypted Hamming distance. Given two ciphertext vectors $\boldsymbol{C}_{\text{enrol}} = (C_{\text{enrol}}(1), \ldots, C_{\text{enrol}}(n))$ and $\boldsymbol{C}_{\text{auth}} = (C_{\text{auth}}(1), \ldots, C_{\text{auth}}(n))$, corresponding to the plaintext vectors $\boldsymbol{V}_{\text{enrol}} = (V_{\text{enrol}}(1), \ldots, V_{\text{enrol}}(n))$ and $\boldsymbol{V}_{\text{auth}} = (V_{\text{auth}}(1), \ldots, V_{\text{auth}}(n))$, the encrypted Hamming distance can be computed via the formula:

$$\text{HD}_E(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}}) = \boxplus_{i=1}^{n} \left[C_{\text{enrol}}(i) \boxplus C_{\text{auth}}(i) \boxminus (C_{\text{enrol}}(i) \boxtimes C_{\text{auth}}(i)) \boxminus (C_{\text{enrol}}(i) \boxtimes C_{\text{auth}}(i)) \right], \quad (2)$$

where $\boxplus$, $\boxminus$, and $\boxtimes$ denote homomorphic addition, subtraction, and multiplication respectively. The homomorphic subtraction can be alternatively described as negating the ciphertext to be subtracted and then adding the result homomorphically to the first ciphertext. Besides negation, we define the homomorphic addition and multiplication in the usual way:

$$C_{\text{enrol}}(i) \boxplus C_{\text{auth}}(i) = \mathsf{HEnc}_{pk_{\text{AS}}}(V_{\text{enrol}}(i) + V_{\text{auth}}(i))$$
$$C_{\text{enrol}}(i) \boxtimes C_{\text{auth}}(i) = \mathsf{HEnc}_{pk_{\text{AS}}}(V_{\text{enrol}}(i) \cdot V_{\text{auth}}(i))$$

These suggest that Eq. (2) can be equivalently written as:

$$\begin{aligned} \text{HD}_E(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}}) &= \boxplus_{i=1}^{n} \mathsf{HEnc}_{pk_{\text{AS}}} \left(V_{\text{enrol}}(i) + V_{\text{auth}}(i) - 2V_{\text{enrol}}(i) \cdot V_{\text{auth}}(i) \right) \\ &= \boxplus_{i=1}^{n} \mathsf{HEnc}_{pk_{\text{AS}}} \left(V_{\text{enrol}}(i) \oplus V_{\text{auth}}(i) \right) \\ &= \mathsf{HEnc}_{pk_{\text{AS}}} \left(\sum_{i=1}^{n} \left(V_{\text{enrol}}(i) \oplus V_{\text{auth}}(i) \right) \right), \end{aligned}$$

which corresponds to the encryption of the Hamming distance $\text{HD}(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})$. The second equality follows since $V_{\text{enrol}}(i), V_{\text{auth}}(i) \in \{0, 1\}$, for each $i = 1, \ldots, n$. We also note here that $+$ and $\cdot$ denote usual addition and multiplication, while $\oplus$ denotes the XOR addition.

Based on the above notation, Eq. (2) implies that for each pair of ciphertexts $(C_{\text{enrol}}(i), C_{\text{auth}}(i))$, the computation of

$$C_{\text{enrol}}(i) \boxplus C_{\text{auth}}(i) \boxminus (C_{\text{enrol}}(i) \boxtimes C_{\text{auth}}(i)) \boxminus (C_{\text{enrol}}(i) \boxtimes C_{\text{auth}}(i))$$

requires one homomorphic multiplication (HM), one homomorphic negation (HN), and three homomorphic additions (HA). For each $i = 1, \ldots, n$, the resulting value is a ciphertext. There are n such ciphertexts that need to be computed and then added together homomorphically to obtain the encrypted Hamming distance. Consequently, the total cost for computing $\text{HD}_E(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})$ in terms of homomorphic operations is: $n(3\text{ HA} + 1\text{ HN} + 1\text{ HM}) + (n-1)\text{ HA}$, where the homomorphic multiplications constitute the main bottleneck in the computation of the encrypted Hamming distance.

4 Security Analysis

The proposed protocol is assumed to have three sources that are subject to attack: 1. the AS, 2. the communication channel between AS and DB, and 3. the DB. We start with the definitions of the adversary types.

Definition 1 (Semi-honest Adversary). *A semi-honest adversary (a.k.a. an* honest-but-curious adversary*) [6,7,15,30] is an attacker that can do any passive attack (including recording all the intermediate transactions, making analysis, and retrieving any knowledge about the other parties' private data, etc.) without changing the prescribed definition of the protocol.*

Definition 2 (Malicious Adversary). *A malicious adversary is the strongest type of adversary [6,7,15,30] that can arbitrarily deviate from the definition of the protocol and utilizes any effective strategy to retrieve some additional knowledge about other parties' private data and/or manipulate the outcome of the computation.*

We consider both adversary types in our security analysis as follows. We first assume the "semi-honest" security model on the User and AS sides, where the parties honestly follow the protocol and learn nothing beyond their own outputs. The security of AS is extremely important as the system takes the raw CSR image, the User's secret value SV and stores the private keys used in the homomorphic encryption and digital signature schemes. Under this security model, the communication channel between the User and AS is considered secure from any malicious attack. We assume that AS and DB are different entities, and collusion is not allowed between the AS and the DB. On the other hand, we assume the "malicious" security model on the DB side, which is the strongest adversary type. Therefore, depending on the application, DB can be outsourced to a third party (such as the cloud) for flexibility and cost-efficiency reasons. We also assume that the underlying cryptographic primitives (hash function, homomorphic encryption scheme, and digital signature algorithm) of the proposed authentication protocol in Sect. 3 are securely employed and implemented. Note that the algorithmic choices in Sect. 5 are for proof-of-concept and experimental purposes; without loss of generality, our protocol can be instantiated with different state-of-the-art secure algorithms.

In what follows, we analyze our scheme against attackers targeting to corrupt the DB to obtain or modify any sensitive data as DB is considered an outsourced and untrusted entity. The adversary can aim at obtaining the CSR template vector ($\boldsymbol{\omega}$) and the corresponding user secret value (SV) of a prover to attack the system for an unauthorized authentication. We also analyze our scheme against a legitimate but dishonest user trying to impersonate another user.

In addition, the adversary can also aim to clone or create an image of the CSR modality that can be used both to attack the system and to compromise the user's privacy. However, at this point, we follow the general assumption that CSRs reflect light-creating patterns that have been considered to be unique and physically unclonable [12,24]. Therefore, in this section, we only focus on

analyzing the security of the proposed authentication protocol given in Sect. 3 under the adversary types defined above.

Theorem 1 (Security against database corruption). *The proposed authentication protocol is secure in case of corruption of the database.*

Proof. Let $\mathcal{A}$ be the malicious attacker that corrupted the DB. As $\mathcal{A}$ has access to the database, she has all the encrypted $C_{\text{enrol}}(i) = \mathsf{HEnc}_{pk_{\text{AS}}}(V_{\text{enrol}}(i))$, for $i = 1, \ldots, n$ where n is the size of the bitstring. The proposed authentication protocol provides computational indistinguishability against $\mathcal{A}$ due to the semantic (IND-CPA) security property of the underlying homomorphic encryption system as $C_{\text{auth}}(i) = \mathsf{HEnc}_{pk_{\text{AS}}}(V_{\text{auth}}(i))$ is encrypted each time with freshly generated random values (which is also different from the one used in the enrollment stage). Furthermore, even if the $\boldsymbol{\omega}$ vectors are matched in two different sessions (similarly to the enrollment session), the adversary cannot distinguish between V_{auth} and V_{enrol} values due to the semantic security property. The attacker cannot obtain any information about a user's biohash vector $\boldsymbol{V}_{\text{enrol}}$, since it is stored in the DB in encrypted form $\boldsymbol{C}_{\text{enrol}}$, under a homomorphic encryption cryptosystem. Therefore, the attacker can retrieve neither the CSR template ($\boldsymbol{\omega}$) nor the user SV by corrupting the database. In addition, $\boldsymbol{C}_{\text{enrol}}$ is stored with the signature of the data, so $\mathcal{A}$ cannot modify the enrollment data as the private keys are stored in AS using a secure environment (such as HSM [17]). We also masked the template data with the User secret value SV that provides revocation ability of the CSR template in case it is needed (due to customization or membership termination) without revealing the CSR data. In addition, the same CSR of a subject can be utilized in different recognition systems without constituting a privacy threat, as two templates of the CSR with different user secret values will be unlinkable to different databases. □

Theorem 2 (Security against malicious outsider attacker). *The proposed authentication protocol is secure against an outsider malicious adversary.*

Proof. The biohash values are not revealed at the authentication phase as they are processed in a semantically secure encrypted domain. Namely, the authentication procedure (see Algorithm 2) is determined by executing

$$\text{HD}(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}}) \leftarrow \mathsf{HDec}_{sk_{\text{AS}}}(\text{HD}_E(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})).$$

Considering that we (homomorphically) decrypt a single value after the computation of Eq. (2), the only outcome is the Hamming distance of the authentication and the enrollment biohash vectors, namely $\boldsymbol{V}_{\text{auth}}$ and $\boldsymbol{V}_{\text{enrol}}$. We then conclude that the security of the system against an outsider malicious adversary can be reduced to the security of the underlying homomorphic encryption cryptosystem. □

For example, in our implementation, we have considered the lattice-based homomorphic encryption scheme of Fan and Vercauteren [11], whose security relies on the Ring Learning With Errors (RLWE) assumptions. Hence, in order

for a malicious adversary to gain partial information about the biohash vectors, she needs to be able to identify vulnerabilities either in the implementation of the homomorphic encryption scheme (e.g., side-channel attacks), or in the underlying RLWE hard mathematical problems.

Theorem 3 (Security against semi-honest prover). *A dishonest insider semi-honest prover (a user) can impersonate a legitimate prover $\mathcal{P}_i$ with only a negligible probability.*

Proof. Let $\mathcal{P}_c$ be a semi-honest insider user that aims to impersonate a different legitimate user $\mathcal{P}_i$. In the enrollment phase, the prover needs to provide a signature of the encryption of enrollment vector i.e., $\boldsymbol{C}_{\text{enrol}}$. So, based on the assumption of secure key generation/distribution $\mathcal{P}_c$ cannot impersonate $\mathcal{P}_i$ during the enrollment phase. Considering the authentication phase, $\mathcal{P}_c$ needs to masquerade both a legal CSR reading and the user SV. As CSRs have been argued to behave as physical unclonable functions [12,24], it is very unlikely to make a physical counterfeit of CSRs. Assuming that a verifier always requires a CSR reading and entering SV, $\mathcal{P}_c$ can only impersonate a legitimate prover with a negligible probability. □

5 Implementation

In this section, we present the details of our proof-of-concept implementation and the experimental results regarding the execution of the CSR authentication protocol that is presented in Sect. 3.2. Specifically, we have implemented both the enrollment and authentication phases, described in Algorithms 1 and 2, where the implementation is conducted in *Python 3.6.9*. The homomorphic encryption scheme that we have chosen is the one presented by Fan and Vercauteren (FV) [11], and for the implementation of this scheme, as well as the required homomorphic operations, we use the `PySEAL` library[7]. The selected library for the hash functions is the `hashlib`[8], where for our purpose, we used SHA-256 and SHA-224, which are currently considered acceptable for hash function applications in [26], to obtain the digest of the User's secret value in the enrollment and authentication phases. Further, as a digital signature scheme, we have chosen ECDSA using the NIST Curve P-256; we used the Python `ecdsa 0.17.0` library[9]. In contrast, the feature extraction process described in Sect. 3.1 and specifically the function ReliableFeatures that is required in both the enrollment and authentication algorithms for extracting the reliable features was implemented from scratch.

[7] https://github.com/Lab41/PySEAL.
[8] https://docs.python.org/3/library/hashlib.html.
[9] https://pypi.org/project/ecdsa/.

5.1 Dataset

We analyzed a set of 17 CSR IDs from which we generated a set of CSR images. The CSR images were acquired by using two different optical microscopes. 7 CSR images were taken by using a USB `Dino-Lite` digital microscope with perpendicular illumination to the sample, and flexible LED control, as shown in Fig. 1-*a*. These images may be closer to the images acquired by an end-user without the expertise and/or professional microscopes. In Fig. 1-*a*, one can observe some green spots due to the photonic cross-communication, which consists of the coupling effect of the optical signal between neighbor spheres [13]. This effect can be advantageous due to the fact of having more information, but it can also be a disadvantage as it may introduce more noise to the system. The other 10 CSR images were taken with a professional polarized microscope equipped with a digital camera and illumination perpendicular to the sample. Those images present a well-defined and colored CSRs, see Figs. 1-*b* and 2-*a*. The images acquired with different microscopes clearly show different responses, meaning that a reliable process needs to be implemented for the minutiæ detection from CSR images when acquired with distinct readout devices.

Table 1. Operations applied to CSR image to simulate input noise.

Sequence	Operation	Range
1	Rotation	$1 - 5°$ (anticlockwise)
2	Blurring	$(2 \times 2) - (4 \times 4)$
3	Gaussian noise	0.2–0.4

We inject two types of noise: similarity noise and Gaussian noise [10]. The first one simulates noise coming from external conditions such as sudden illumination changes, lack of focus, rotation, etc. The Gaussian noise simulates the photonic and electronic noise inherent to each device, and it occurs during the image acquisition under low-light conditions, which makes it difficult for the visible light sensors to capture details of the object efficiently [10]. Thus, we generated a set of CSR images by applying the operations listed in Table 1 to the reference images, whose ranges are based on the idea of keeping the noise within realistic external conditions. The procedure of generating noisy responses ensures the extraction of reliable blobs.

Table 2. Dataset for the enrollment and authentication phases.

Protocol phase	CSR responses	Number of attempts	Acquired images per attempt
Enrollment	17	1	7
Authentication		10	5

For the enrollment phase, from each CSR image, we generated seven noise images to extract the truly reliable blobs. For the authentication phase, we simulated ten attempts to authenticate CSR images taken at different time intervals and, at each time, five images were acquired (also intending to extract the reliable blobs), as described in Table 2.

5.2 Homomorphic Encryption Implementation

Fan and Vercauteren proposed the FV scheme in 2012 [11] and essentially, it is based on the homomorphic encryption scheme of Brakerski [9]. The FV scheme is instantiated over the RLWE setting instead of the LWE setting, which is the case in the Brakerski scheme, and it is generally a more efficient version of Brakerski's scheme. However, because of the similarities between the two schemes, we often refer to the FV scheme as Brakerski/Fan–Vercauteren (BFV). We present a high-level description of the BFV scheme here and refer to [11,23] for a more detailed analysis.

Since the BFV scheme works in the RLWE setting, its core operations are performed over a polynomial ring $R_m = \mathbb{Z}_m[x]/(x^d + 1)$, containing polynomials modulo $x^d + 1$, with integer coefficients in $\{\lceil -m/2 \rceil, \ldots, \lfloor (m-1)/2 \rfloor\}$. For efficiency reasons, the degree d of the *polynomial modulus* is usually chosen as a power of 2. Based on this notation, the plaintext space in the BFV scheme is described by the polynomial ring R_t, for some positive integer *plaintext modulus* t and the ciphertext space is described as R_q^2, for some positive integer *coefficient modulus* q. In practice, the coefficient modulus is chosen to be larger than the plaintext modulus so that each plaintext can be mapped to multiple valid ciphertexts. Hence, a plaintext message is represented as a polynomial in R_t, and its encryption is transformed into a pair of polynomials in R_q. Integer plaintext messages can be encrypted using the BFV scheme after applying an encoding to convert them to polynomials in R_t. For key generation, the secret key sk is sampled from R_2, i.e., it is a polynomial of degree at most $d-1$, with coefficients in $\{-1, 0, 1\}$, while the public key pk is composed of two polynomials in R_q.

BFV Parameter Sets in `PySEAL`. The most crucial part of implementing the BFV scheme is selecting a suitable parameter set that maintains a reasonable balance between performance and security. The BFV parameter set consists of the polynomial, plaintext, and coefficient moduli, and these parameters are usually chosen following the homomorphic encryption standard of Albrecht *et al.* [1]. `PySEAL` offers different flavors of such parameter sets targeting 128-bit security, while it also allows implementers to initialize the scheme with their own parameters in order to achieve higher security levels.

Table 3. Chosen parameter set for BFV scheme in `PySEAL`.

Sec. level. (bits)	Parameter set		
	Poly modulus (d)	Plaintext modulus (t)	Coefficient modulus (q)
128	$x^{2048} + 1$	256	72057594036879361

The parameter set chosen in our implementation is given in Table 3. This is one of the instantiations that is used in the PySEAL library for 128-bit security. The coefficient modulus q in PySEAL is defined as a product of distinct primes of size up to 60-bits, where each prime is congruent to 1 modulo $2d$. In our instance, q is composed of only one prime, as shown in Table 3. The performance of the homomorphic encryption scheme is largely affected by the polynomial modulus, as well as by the number of prime factors in the coefficient modulus. Hence, one needs to keep the degree d and the number of prime factors in q small.

Performance of BFV Homomorphic Operations in PySEAL. Besides homomorphic encryption and decryption, PySEAL supports all the required homomorphic operations for computing the encrypted Hamming distance using Eq. (2), i.e., homomorphic addition, negation, and multiplication. In Table 4, we list the average time required for performing the homomorphic operations that are needed in the authentication, Algorithm 2 (the homomorphic encryption in the enrollment takes similar times). The reason for focusing on the authentication phase in our experiments is that it contains the main bulk of homomorphic operations, including the computation of the Hamming distance in the encrypted domain. The timings in Table 4 refer to two different sizes of the biohash vector $\boldsymbol{V}_{\text{auth}}$, namely 224- and 256-bits, where for this purpose, we used SHA-224 and SHA-256 for hashing the User's secret value.

Table 4. Average execution times (in seconds) of homomorphic operations, with respect to the BFV homomorphic encryption scheme and the parameter set of Table 3, that are performed in the authentication phase (Algorithm 2), using the PySEAL library.

Size of biohash	Homomorphic operations					
	Encryption	Addition	Negation	Multiplication	HD_E	Decryption
224-bits	1.423	0.0044	0.0010	0.456	0.462	0.0002
256-bits	1.635	0.0052	0.0012	0.523	0.529	0.0002

The encryption column refers to the average time required to encrypt the biohash vector $\boldsymbol{V}_{\text{auth}}$ bit-by-bit. In other words, it refers to the average time needed to perform n homomorphic encryption operations for $n \in \{224, 256\}$. The rest of the columns concern the homomorphic operations required for computing the encrypted Hamming distance based on Eq. (2). Specifically, our implementation requires approximately 4.4 ms to perform $4n - 1$ homomorphic additions for $n = 224$ and approximately 5.2 ms for $n = 256$. Performing n homomorphic negations takes roughly 1 ms in both cases. On the other hand, our implementation requires 0.456 s to perform n homomorphic multiplications for $n = 224$ and 0.523 s for $n = 256$.

The average time for calculating the encrypted Hamming distance is 0.462 s for $n = 224$ and 0.529 s for $n = 256$, suggesting that the homomorphic multiplications consume approximately 98.78% of the computation time for the encrypted

Hamming distance in both cases. The last column concerns the average time taken to decrypt $\mathrm{HD}_E(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})$. This is approximately 0.2 milliseconds in both cases, which is expected since the encrypted Hamming distance represents a single ciphertext corresponding to the encryption of the actual Hamming distance $\mathrm{HD}(\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}})$ and hence to the encryption of an integer in $\{0, \ldots, n\}$. Finally, the experimental results presented in Table 4 indicate that switching to biohash vectors of size 224-bits offers around 12.82% speed-up compared to the 256-bit case, both in terms of bit-by-bit encryption as well as in the computation of the encrypted Hamming distance.

5.3 Performance of Enrollment and Authentication Phases

In Table 5, we summarize our experimental results regarding the performance of the enrollment (Algorithm 1) and authentication (Algorithm 2), considering biohash vectors of size $n \in \{224, 256\}$ to be encrypted.

Table 5. Average execution times (in seconds) of enrollment and authentication phases, with respect to the homomorphic operations performance of Table 4. In these timings, we exclude the performance of the feature extraction.

Size of biohash	Enrollment (Algorithm 1)		Authentication (Algorithm 2)		
	Encryption: $\boldsymbol{C}_{\text{enrol}}$	Enrollment	Encryption: $\boldsymbol{C}_{\text{auth}}$	HD_E	Authentication
224-bits	1.407	1.409	1.423	0.462	1.892
256-bits	1.648	1.651	1.635	0.529	2.172

It is clear from Table 5 that almost all of the execution time of the enrollment is dedicated to the encryption of $\boldsymbol{V}_{\text{enrol}}$, for both $n \in \{224, 256\}$. This is expected since all operations in Algorithm 1, except for the homomorphic encryption, are simple operations that can be efficiently performed, i.e., matrix multiplication, hash computation, and XOR. Specifically, the execution of the enrollment algorithm for 224-bits requires around 1.407 s, offering approximately a 14.66% performance advantage over the 256 case, which requires 1.648 s.

The authentication phase has the extra computational burden of computing the Hamming distance of the two bitstrings $\boldsymbol{V}_{\text{enrol}}, \boldsymbol{V}_{\text{auth}}$ using the encrypted vectors $\boldsymbol{C}_{\text{enrol}}, \boldsymbol{C}_{\text{auth}}$. For both $n \in \{224, 256\}$, the bit-by-bit encryption of $\boldsymbol{V}_{\text{auth}}$ consumes approximately 75.25% of the total execution time, while the computation of the encrypted Hamming distance requires approximately 24.39% of the execution time of Algorithm 2. As expected, the computation time for homomorphic encryption and the Hamming distance together corresponds to around 99.63% of the total execution time of the authentication algorithm. We also noted that the 224 case offers a performance gain of around 13.77% over the 256-bit case.

6 Conclusions and Future Work

We started to work with a challenge: proving that it is possible to securely authenticate objects enhanced with CSRs, a material that demonstrates fingerprint-like features and shows optical responses acclaimed to be unpredictable (at the production phase), physically non-reproducible, and identifying. Images of such responses —as often happen in biometrics— contain noise. Extracting information and minutiæ is a process with data instability, increasing the stake for those who intend to develop a cryptographically secure authentication protocol taking advantage of the materials' optical features.

We designed, proved it secure, and implemented a solution for authenticating CSRs tags. After illustrating a methodology to derive stable and reliable bitstrings from the minutiæ of CSRs images using techniques borrowed by the biometric traditions, the main contribution of the paper is the design of an authentication protocol. It uses biohashing and homomorphic encryption, and despite being thought to work only with CSRs, we argue that the proposed protocol can be applied to a wider range of use-cases. In fact, since CSRs minutiæ are abstractly represented as "randomly" distributed colored spots in a bi-dimensional plane, our solution works for any technology for authentication that relies on the same data structure, for example, dense cloud of ink dots used in watermarking, or certain allegedly Physical Unclonable Functions emerging from the optical analysis of fabric like silk [22].

We demonstrated that an encrypted bitstring can be stored securely in an outsourced database. When instantiated with a practical homomorphic encryption scheme, an authentication system can be designed to perform secure computations in the encrypted domain. One such operation is the computation of the Hamming distance of two bitstrings, using as input only their corresponding ciphertexts. This operation is the most significant in the authentication phase, as it determines whether the authentication is successfully performed and in a privacy-preserving manner, i.e., without revealing any sensitive information about the User's private data.

We presented a proof-of-concept implementation of the proposed CSR authentication protocol, which we instantiated with the BFV homomorphic encryption scheme. The implementation was conducted in Python, using the `PySEAL` library for performing the required homomorphic operations with the BFV scheme. Based on our implementation, we derived useful conclusions regarding the computational time that is consumed by heavy homomorphic operations such as bit-by-bit homomorphic encryption and homomorphic multiplication, which has a major performance impact in the computation of the Hamming distance of two bitstrings in the encrypted domain.

Future Work. We envision several extensions for this work, mainly targeting implementations with improved performance offering at the same time high-security guarantees. The first direction aims at achieving an optimized implementation. We believe that better timings can be obtained when switching to other programming languages and more up-to-date libraries, such as Microsoft's

SEAL library[10] which implements the BFV scheme in C++. In addition, we do not claim that our formula for computing the encrypted Hamming distance is the optimal one. It is likely that more efficient methods tailored to the BFV scheme are already available or can be designed. For example, in [36], the authors presented an optimized way for computing the encrypted Hamming distance with BFV homomorphic operations by applying different encodings ("packed" representations) for the polynomials representing ciphertexts and plaintexts. The second direction looks at the security of our implementation. We plan to emphasize more on instantiating the BFV scheme with optimal parameter sets, achieving at least 128-bits security. Still, from a security perspective, we also plan to look at implementations offering side-channel protection, focusing specifically on timing attacks.

Acknowledgements. We thank the reviewers for their valuable comments and suggestions. We would also like to acknowledge Prof. Dr. J. Lagerwall for providing the CSRs images. In addition, Mónica Arenas and Georgios Fotiadis would like to thank Dr. Kim Laine, from Microsoft Research, for his responsiveness and valuable comments regarding the BFV homomorphic encryption scheme and its implementation. The authors acknowledge the financial support from the Luxembourg National Research Fund (FNR) on the projects Security in the Shell "SSh" (C17/MS/11688643), No more Fakes "NoFakes" (PoC20/15299666/NOFAKES-PoC) and the CORE project Secure, Quantum-Safe, Practical Voting Technologies "EquiVox" (C19/IS/13643617/EquiVox/Ryan).

References

1. Albrecht, M.R., et al.: Homomorphic Encryption Standard. IACR, p. 939 (2019). Cryptol. ePrint Arch. https://eprint.iacr.org/2019/939
2. Arenas, M., Demirci, H., Lenzini, G.: Cholesteric spherical reflectors as physical unclonable identifiers in anti-counterfeiting. In: The 16th International Conference on Availability, Reliability and Security, pp. 1–11. ACM (2021). https://doi.org/10.1145/3465481.3465766
3. Arenas, M., Demirci, H., Lenzini, G.: An analysis of cholesteric spherical reflector identifiers for object authenticity verification. Mach. Learn. Knowl. Extr. **4**(1), 222–239 (2022). https://doi.org/10.3390/make4010010
4. Bai, Z., Hatzinakos, D.: LBP-based biometric hashing scheme for human authentication. In: 11th International Conference on Control Automation Robotics Vision (ICARCV), pp. 1842–1847 (2010). https://doi.org/10.1109/ICARCV.2010.5707216
5. Bicego M., Lagorio, A., Grosso, E., Tistarelli, M.: On the use of SIFT features for face authentication. In: Conference On Computer Vision And Pattern Recognition Workshop. CVPRW 2006, pp. 35–35 (2006)
6. Bicer, O., Bingol, M.A., Kiraz, M., Levi, A.: Highly efficient and re-executable private function evaluation with linear complexity. IEEE Trans. Dependable Secure Comput. **19**(02), 835–847 (2022). https://doi.org/10.1109/TDSC.2020.3009496

[10] https://github.com/Microsoft/SEAL.

7. Bingol, M.A.: Efficient and secure schemes for private function evaluation. Ph.D. thesis, Sabanci University, Istanbul (2019). https://research.sabanciuniv.edu/id/eprint/36861/
8. Boddeti, V.N.: Secure face matching using fully homomorphic encryption. In: 2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS), pp. 1–10. IEEE (2018)
9. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012–32nd Annual Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2012. Proceedings. Lecture Notes in Computer Science, vol. 7417, pp. 868–886. Springer (2012). https://doi.org/10.1007/978-3-642-32009-5_50
10. Deledalle, C.A., Denis, L., Tupin, F.: How to compare noisy patches? Patch similarity beyond gaussian noise. Int. J. Comput. Vision, **99**(1), 86–102 (2012). https://doi.org/10.1007/s11263-012-0519-6, https://hal-imt.archives-ouvertes.fr/hal-00672357, http://link.springer.com/10.1007/s11263-012-0519-6
11. Fan, J., Vercauteren, F.: Somewhat Practical Fully Homomorphic Encryption (2012). https://ia.cr/2012/144. Cryptology ePrint Archive, Report 2012/144
12. Geng, Y., Noh, J., Drevensek-Olenik, I., Rupp, R., Lenzini, G., Lagerwall, J.P.: High-fidelity spherical cholesteric liquid crystal Bragg reflectors generating unclonable patterns for secure authentication. Sci. Rep. **6**, 1–9 (2016). https://doi.org/10.1038/srep26840
13. Geng, Y., Noh, J., Drevensek-Olenik, I., Rupp, R., Lagerwall, J.: Elucidating the fine details of cholesteric liquid crystal shell reflection patterns. Liq. Cryst. **44**(12–13), 1948–1959 (2017)
14. Geng, Y., Kizhakidathazhath, R., Lagerwall, J.P.F.: Encoding hidden information onto surfaces using polymerized cholesteric spherical reflectors. Adv. Funct. Mater. **31** (2021). https://doi.org/10.1002/adfm.202100399
15. Goldreich, O.: Foundations of Cryptography, vol. 1. Cambridge University Press, New York, NY, USA (2006)
16. Gomez-Barrero, M., Maiorana, E., Galbally, J., Campisi, P., Fierrez, J.: Multi-biometric template protection based on homomorphic encryption. Pattern Recogn. **67**, 149–163 (2017)
17. Ivarsson, J., Nilsson, A.: A Review of Hardware Security Modules (2010). https://www.opendnssec.org/wp-content/uploads/2011/01/A-Review-of-Hardware-Security-Modules-Fall-2010.pdf. Accessed Mar 2022
18. Joshi, M., Mazumdar, B., Dey, S.: Security Vulnerabilities Against Fingerprint Biometric System (2018). arXiv1805.07116, http://arxiv.org/abs/1805.07116
19. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ECDSA). Int. J. Inf. Secur. **1**(1), 36–63 (2001). https://doi.org/10.1007/s102070100002
20. Karabat, C., Kiraz, M.S., Erdogan, H., Savas, E.: THRIVE: threshold homomorphic encryption based secure and privacy preserving biometric verification system. EURASIP J. Adv. Sig. Process. **2015**(1), 1–18 (2015). https://doi.org/10.1186/s13634-015-0255-5
21. Khan, M.A.: Fingerprint image enhancement and minutiae extraction (2011)
22. Kim, M.S., Lee, G.J., Leem, J.W., Choi, S., Kim, Y.L., Song, Y.M.: Revisiting silk a lens-free optical physical unclonable function. Nature Commun. **13**(1), 1–12 (2022). https://doi.org/10.1038/s41467-021-27278-5, https://www.nature.com/articles/s41467-021-27278-5

23. Laine, K.: Simple encrypted arithmetic library 2.3. 1. Microsoft Research (2017). https://www.microsoft.com/en-us/research/uploads/prod/2017/11/sealmanual-2-3-1.pdf
24. Lenzini, G., et al.: Security in the shell an optical physical unclonable function made of shells of cholesteric liquid crystals. In: 2017 IEEE Workshop on Information Forensics and Security, WIFS 2017 2018-Janua, pp. 1–6 (2017). https://doi.org/10.1109/WIFS.2017.8267644
25. Ratha, N., Bolle, R., Pandit, V., Vaish, V.: Robust fingerprint authentication using local structural similarity. In: Proceedings Fifth IEEE Workshop On Applications Of Computer Vision, pp. 29–34 (2000)
26. National Institute of Standards and Technology SP 800–131A Rev. 2: Transitioning the Use of Cryptographic Algorithms and Key Lengths (2018). https://csrc.nist.gov/publications/detail/sp/800-131a/rev-2/final
27. Ngo, D., Teoh, A., Goh, A.: Biometric hash high-confidence face recognition. IEEE Trans. Circuits Syst. Video Technol. **16**(6), 771–775 (2006). https://doi.org/10.1109/TCSVT.2006.873780
28. Pradel, G., Mitchell, C.: Privacy-Preserving Biometric Matching Using Homomorphic Encryption. arXiv preprint arXiv:2111.12372 (2021)
29. Pursell, L., Trimble, S.Y.: Gram-schmidt orthogonalization by Gauss elimination. Am. Math. Mon. **98**(6), 544–549 (1991). https://doi.org/10.1080/00029890.1991.11995755
30. Schneider, T.: Engineering Secure Two-Party Computation Protocols - Advances in Design, Optimization, and Applications of Efficient Secure Function Evaluation. Ph.D. thesis, Ruhr-University Bochum, Germany, Information Sciences (2011). http://thomaschneider.de/papers/S11Thesis.pdf
31. Schwartz, M., Lenzini, G., Geng, Y., Rønne, P.B., Ryan, P.Y., Lagerwall, J.P.: Cholesteric liquid crystal shells as enabling material for information-rich design and architecture. Adv. Mater. **30**(30), 1–19 (2018). https://doi.org/10.1002/adma.201707382
32. Schwartz, M., et al.: Linking physical objects to their digital twins via fiducial markers designed for invisibility to humans. Multifunctional Mater. **2**(4), 1–19 (2021). https://doi.org/10.1088/2399-7532/ac0060
33. Topcu, B., Karabat, C., Azadmanesh, M., Erdogan, H.: Practical security and privacy attacks against biometric hashing using sparse recovery. EURASIP J. Adv. Sig. Process. **2016**(1), 1–20 (2016). https://doi.org/10.1186/s13634-016-0396-1
34. Torres, W.A.A., Bhattacharjee, N., Srinivasan, B.: Effectiveness of fully homomorphic encryption to preserve the privacy of biometric data. In: Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services, pp. 152–158 (2014)
35. Tuyls, P., Akkermans, A.H., Kevenaar, T.A., Schrijen, G.J., Bazen, A.M., Veldhuis, R.N.: Practical biometric authentication with template protection. Lect. Notes Comput. Sci. **3546**, 436–446 (2005). https://doi.org/10.1007/11527923_45
36. Yu, H., Yin, L., Zhang, H., Zhan, D., Qu, J., Zhang, G.: Road distance computation using homomorphic encryption in road networks. CMC-Comput. Mater. Continua **69**(3), 3445–3458 (2021)