Secure Biometric Authentication With Improved Accuracy

Manuel Barbosa[2], Thierry Brouard[1]
, Stéphane Cauchie[1,3], and Simão Melo De Sousa[3]

[1] Laboratoire Informatique de l'Université François Rabelais de Tours
`stephane.cauchie@univ-tours.fr,`
[2] Departamento de Informática, Universidade do Minho
`mbb@di.uminho.pt`
[3] Departamento de Informática, Universidade da Beira Interior
`desousa@ubi.pt`

Abstract. We propose a new hybrid protocol for cryptographically secure biometric authentication. The main advantages of the proposed protocol over previous solutions can be summarised as follows: (1) potential for much better accuracy using different types of biometric signals, including behavioural ones; and (2) improved user privacy, since user identities are not transmitted at any point in the protocol execution. The new protocol takes advantage of state-of-the-art identification classifiers, which provide not only better accuracy, but also the possibility to perform authentication without knowing who the user claims to be. Cryptographic security is based on the Paillier public key encryption scheme.
Keywords: Secure Biometric Authentication, Cryptography, Classifier.

1 Introduction

Biometric techniques endow a very appealing property to authentication mechanisms : *the user is the key*, meaning there is no need to securely store secret identification data. Presently, most applications of biometric authentication consist of closed self-contained systems, where all the stages in the authentication process and usually all static biometric profile information underlying it, are executed and stored in a controlled and trusted environment. This paper addresses the problem of implementing *distributed* biometric authentication systems, where data acquisition and feature recognition are performed by separate sub-systems, which communicate over an insecure channel. This type of scenario may occur, for instance, if one intends to use biometric authentication to access privileged resources over the Internet. Distributed biometric authentication requires hybrid protocols integrating cryptographic techniques and pattern recognition tools. Related work in this area has produced valid solutions from a cryptographic security point of view. However, these protocols can be seen as rudimentary from a pattern-recognition point of view. In fact, regardless of the security guarantees that so-called fuzzy cryptosystems provide, they present great limitations on the accuracy that can be achieved, when compared to purely biometric solutions resorting to more powerful pattern recognition techniques.

In this paper, we propose a solution which overcomes this accuracy limitation. Our contribution is a protocol offering the accuracy of state-of-the-art pattern recognition classifiers *and* strong cryptographic security. To achieve our goals we follow an approach to hybrid authentication protocols proposed by Bringer et al. [1]. In our solution we adapt and extend this approach to use a more accurate and stable set of models, or *classifiers*, which are widely used in the pattern recognition community in settings where cryptographic security aspects are not considered. Interestingly, the characteristics of these classifiers allow us, not only to achieve better accuracy, but also to improve the degree of privacy provided by the authentication system. This is possible because we move away from *authentication classifiers* and take advantage of an *identification classifier*. An identification classifier does not need to know who the user claims to be, in order to determine if she belongs to the set of valid users in the system and determine her user identifier. An additional contribution of this paper is to formalise the security models for the type of protocol introduced by Bringer et al. [1]. We show that the original protocol is actually insecure and under the original security model, although it can be easily fixed. We also extend the security model to account for eavesdroppers external to the system, and provide a security argument that our solution is secure in this extended security model.

The remaining of the paper is organized as follows. We first summarise related work in Section 2 and we introduce our notational framework for distributed biometric authentication systems in Section 3. We propose our secure biometric authentication protocol and security models in Section 4. In Section 5 we present a concrete implementation based on the *Support Vector Machine* classifier and the Paillier public key encryption scheme, including the corresponding security analysis. Finally, we discuss our contributions in Section 6.

2 Related Work

Fuzzy extractors are a solution to secure biometric authentication put forward by the cryptographic community [2]. Here, the pattern recognition component is based on error correction. A fuzzy extractor is defined by two algorithms. The generation algorithm takes a user's biometric data w and derives secret randomness r. To allow for robustness in reconstructing r, the generation algorithm also produces public data $\texttt{pub}$. On its own, $\texttt{pub}$ reveals no useful information about the biometric data or the secret randomness. The reconstruction algorithm permits recovering r given a *sufficiently close* measurement w' and $\texttt{pub}$. To use a fuzzy extractor for secure remote authentication, the server would store $(\texttt{pub}, r)$ during the enrolment stage. When the user wants to authenticate, the server provides the corresponding public information $\texttt{pub}$, so that it is possible reconstruct r from a fresh reading w'. The user is authenticated once the server confirms that r has been correctly reconstructed; for example, r can be used to derive a secret key.

A problem with this solution is that security is only guaranteed against eavesdroppers: the server must be authenticated and the public information transmitted reliably. Additionally, Boyen [3] later showed that, even in this scenario, it is not possible to guarantee that security is preserved if the same fuzzy extractor is used to authenticate a user with multiple servers. An adversary might put together public information and secrets leaked from some of the servers to impersonate the user in another server. The same author proposed improved security models and constructions to solve this problem. Boyen et al. [4] later addressed a different problem which arises when the channel to the server is not authenticated and an active adversary can change the value of `pub`. The original fuzzy extractor definition and security model does not ensure that such an adversary is unable to persuade the user that it is the legitimate server. The authors propose a robust fuzzy extractor that permits achieving mutual authentication over an insecure channel.

The protocol proposed by Bringer et al. [1] uses the Goldwasser-Micali encryption scheme, taking advantage of its homomorphic properties. The protocol performs biometric classification using the Hamming distance between fresh biometric readings and stored biometric profiles. User privacy protection is ensured by hiding the association between biometric data and user identities. For this to be possible one must distribute the server-side functionality: an *authentication service* knows the user's claimed identity and wants to verify it, a *database service* stores user biometric data in such a way that it cannot possibly determine to whom it belongs, and a *matching service* ensures that it is possible to authenticate users without making an association between their identity and their biometric profile. These servers are assumed to be honest-but-curious and, in particular, they are assumed to follow the protocol and not to collude to break its security.

Authentication Accuracy In this paper we propose a protocol which improves authentication accuracy while ensuring strong cryptographic security. It is important to support our claims from a pattern recognition accuracy perspective. In the following table we present experimental results found in literature, to compare the accuracy (Equal Error Rate[1]) of advanced pattern recognition classifiers (*Classifier Error*) with that of those adopted in existing hybrid authentication protocols, or so called fuzzy cryptosystems (*Fuzzy Error*).

Biometric Data	References	Bit Length	Fuzzy Error	Classifier Error
Key stroke	[5]/[6]	12	48%	1.8%
Voice	[7]/[8]	46	20%	5%
Tactim	[9]	16	15%	1%
Signature	[10]/[11]	40	28%	5%
Face	[12]/[13]	120	5%	0.6%
Fingerprint	[14]/[15]	128	17%	8%
Iris	[16]	140	5%	5%

[1] Percentage of recognition errors when the biometric system is adjusted in order to obtain the same false positive and false negative rates.

Results are presented for both physiological (iris, face and fingerprint) and behavioural (key stroke, voice, tactim, signature) biometric data. From the results in the table, one can conclude that advanced classifiers consistently outperform simple distance-based (fuzzy) classification techniques. However, this is most important for behavioural biometry, where fuzzy techniques present significantly worse accuracy rates. An empirical explanation for this shortcoming is that fuzzy pattern recognition components can deal with acquisition variability but not with the user variability, which plays a major role in behavioral biometry. From a pattern recognition point of view, advanced classifiers are built on the assumption that two users may produce close measurements. Classification focuses on the boundaries between users, and some of them like the Support Vector Machine (SVM) classifier [17], can optimally minimize the error risk.

3 Biometric systems

In this section we present a precise definition of a pattern recognition system for biometric authentication and identification, which we will later use in the definition of our hybrid authentication protocol. We take a particular type of biometric parameter $b \in \mathbb{B}$, where $\mathbb{B}$ denotes the complete set of biometric parameters. The basic tool associated with b is an adequate sensor, denoted by the application $\rho_b : \mathbb{U} \to \mathbb{V}$ where $\mathbb{U}$ is a set representing the universe of possible users and $\mathbb{V}$ represents a sensor-dependent space of biometric features (usually an n-tuple of real numbers). We will refer to the output of the sensor as a *feature.* [2]

Consider a set of users $U \subset \mathbb{U}$. The goal is to recover the pre-image of a feature $\rho_b(u)$, for $u \in U$, using prior knowledge of a users profile $w_U^* \in \mathbb{W}$, where $\mathbb{W}$ is a sensor-dependent set of possible users profiles, and an inversion function called a *classifier*. Usually a classifier is a two-stage procedure: (1) there is a pre-decision processing stage cl, which takes a feature and pre-established profile information and returns classification data such as confidence intervals, distances, etc.; and (2) a decision stage $\mathcal{D}$ which makes the final decision using an appropriate criterion, for example a pre-defined threshold, majority rules, etc. Ideally, one expects that classification satisfies

$$\forall u \in U, \mathcal{D}(cl(\rho_b(u), w_U^*)) = u$$
$$\forall u \in \mathbb{U}/U, \mathcal{D}(cl(\rho_b(u), w_U^*)) = \perp$$

At this stage a distinction must be made between *biometric authentication* and *biometric identification* systems. A system satisfying the previous predicate (or a close enough relaxation that is good enough for practical applications) for a set of users U such that $|U| > 1$ is called a biometric identification system.

[2] In practice raw sensor outputs must be pre-processed using *feature extraction* before classification can be performed. To be precise, we could denote the acquisition of the raw signal by a non deterministic application a_b, and feature extraction by a deterministic application f. We would then have $\rho_b = a_b \circ f$.

Systems satisfying these conditions for only a single user are called biometric authentication systems. Note that it is possible to use a biometric authentication system for identification, e.g. by trying all possible users in a database. However, depending on the biometric parameter and sensor technology, the accuracy of such a system may suffer from overlaps in user profiles. From the point of view of cryptographic protocols, this distinction is also important. In fact, all solutions we have encountered in literature assume that we are dealing with a biometric authentication system, which means that the user's claimed identity must be transmitted over the network. If we move to a biometric identification system, the authentication protocol can be implemented by transmitting only the user's biometric data. We will return to this issue in the next section.

Setting-up and operating a biometric authentication system involves two separate procedures: a set-up stage called *Enrolment*, and the actual operation stage called *Generalisation*. We now describe these in more detail.

Enrolment This is usually split into two steps: (1) the *acquisition and feature extraction* step, and (2) the *learning* step. The first step constructs a reference set of feature values $\rho_b(u)$ $(\forall u \in U)$, called a *training set*. The learning step uses the training set to construct the users' profile w^*_U.

Generalisation This is also split in two steps: (1) the *acquisition and feature extraction* step, and (2) the *decision* step. The former consists of collecting a feature $v = \rho_b(\texttt{unknown})$ for an unknown user. The decision step uses the classifier cl and profile data w^* to determine which user is `unknown`. More precisely the decision check is $\{u \in U, \perp\} \leftarrow \mathcal{D}(cl(v, w^*_U))$.

In this context, we define a pattern recognition system for biometric identification Γ as follows.

Definition 1. *A pattern recognition system for biometric identification Γ is a 5-tuple $< b, U, \rho_b, \mathcal{D} \circ cl, w^*_U >$, where the tuple elements are as described above.*

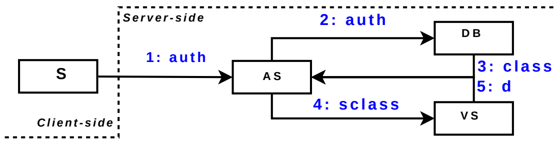
Remark. We stress that the concept of profile w^*_U usually adopted within the pattern recognition community constitutes, in the context of our work, a security-critical parameter. This is because it usually reveals private user information such as a user-specific region in a sensor-dependent parameter space $\mathbb{W}$. In particular, if this information is leaked, anyone can determine whether a feature belongs to a particular user. The vulnerability detected in the protocol proposed by Bringer et al. is based on the fact that an attacker may recover a user profile from a protocol trace. This means that it can perform classification itself, even thought it would never be able to break the encryption scheme protecting the user features used in an authentication run.

4 Proposed Authentication Protocol

In this section we propose a new authentication protocol based on the approach in [1]. We take advantage of a biometric identification scheme implemented using a more powerful pattern recognition technique in the form of a multi-class classifier to achieve improved accuracy and security properties.

4.1 Participants and their roles

The following diagram depicts the data flow between the different participants in our protocol.



The server-side functionality is partitioned in three components to ensure that no single entity can associate a user's identity with the biometric data being collected during authentication. The participants in the authentication protocol are the following:

1. The *Sensor* (S) is the only client-side component. Following the approach in [1], we assume that the sensor is capable of capturing the user's biometric data, extracting it into a binary string, and performing cryptographic operations such as public key encryption. We also assume a *liveness link* between the sensor and the server-side components, to provide confidence that the biometric data received on the server-side is from a present living person.
2. The *Authentication Service* (AS) is responsible for communicating with the user who wants to authenticate and organizing the entire server-side procedure. In a successful authentication the AS will obviously learn the user's identity, which means that it should learn nothing about the biometric data being submitted.
3. The *Database Server* (DB) securely stores the users' profile (w_U^*) and its job is to execute the pre-decision part of classification (cl). Since the DB is aware of privileged biometric data, it should learn nothing about the user's identity, or even be able to correlate or trace authentication runs from a given (unknown) user.
4. The *Verification Server* (VS) completes the authentication process by taking the output produced by the DB server and computing the final decision ($\mathcal{D}$) step. This implies that the VS possesses privileged information that allows it to make a final decision, and again that it should not be able to learn anything about the user's real identity, or even be able to correlate or trace authentication runs from a given (unknown) user.

4.2 Enrolment and system set-up

In this section we describe the procedures that must be carried out to prepare a system using the proposed authentication protocol for normal operation. These include the data collection procedures associated with enrolment, the construction of the static data sets assigned to each actor in the protocol, and the security assumptions/requirements we impose on these elements.

The output of the initialisation procedure are three sets of static data ($AS_{\mathtt{data}}$, $DB_{\mathtt{data}}$ and $VS_{\mathtt{data}}$) which allow the different servers to carry out their roles:

- $AS_{\texttt{data}}$ consists of a list $U = \{\texttt{ID}_1, \ldots, \texttt{ID}_{|U|}\}$ of user identities $\texttt{ID}_i \in \{0,1\}^*$. The index of the user in this list will be used as the application-specific user identifier $uid \in \{1 \ldots |U|\}$.
- $DB_{\texttt{data}}$ consists of biometric classification data (w_U^*) for the set of valid users. This should permit computing pre-decision classification information (cl) over authentication requests, but should be totally anonymous for the DB. In particular, we require that the DB obtains information which permits performing pre-classification for the $|U|$ system users consistently with the application-specific user identifiers assigned by the AS. However, it should not receive any information about the user identities themselves.
- $VS_{\texttt{data}}$ consists of information which will allow the VS to obtain a verdict from obfuscated pre-decision classification information. The need for obfuscation is justified by the apparently contradictory requirement that only the VS is capable of producing a decision verdict, but still should be unable to learn the user's real identity, or even trace requests by the same user.

We assume that some trusted authority is available to control the enrolment procedure, and ensure that the static data is assigned to the servers in a secure way: no server obtains any information concerning another server's static data, and no information is leaked to eavesdroppers external to the system.

4.3 Authentication Protocol Definition

The proposed authentication protocol is a five-tuple of probabilistic polynomial time algorithms that the different participants will execute. Each server-side participant stores corresponding static information $AS_{\texttt{data}}$, $DS_{\texttt{data}}$ and $VS_{\texttt{data}}$. The algorithms are:

Participant	Algorithm
VS	$(\texttt{params}, k_d) \leftarrow \textbf{Gen}(1^\kappa)$
S	$\texttt{auth} \leftarrow \textbf{S}(v_{\texttt{ID}}, \texttt{params})$
DB	$\texttt{class} \leftarrow \textbf{Classify}(\texttt{params}, \texttt{auth}, DB_{\texttt{data}})$
AS	$(\texttt{sclass}, \pi) \leftarrow \textbf{Shuffle}(\texttt{params}, \texttt{class}, AS_{\texttt{data}})$
VS	$d \leftarrow \textbf{Decide}(\texttt{sclass}, \texttt{params}, k_d, VS_{\texttt{data}})$
AS	$\texttt{ID}/\perp \leftarrow \textbf{Identify}(d, \pi, AS_{\texttt{data}})$

1. The key generation algorithm **Gen** is executed by the VS, which stores the secret key k_d securely, and publishes a set of public parameters `params`.
2. On each authentication run, the sensor encrypts fresh biometric data $v_{\texttt{ID}}$ from a user with identity `ID` using algorithm **S** and the public parameters, and produces the authentication request `auth`.
3. The AS receives the authentication request and passes it on to the DB for pre-decision classification. This operation is represented by algorithm **Classify** which takes also public parameters and profile information $DB_{\texttt{data}}$ and returns encrypted classification information `class`.
4. The AS takes `class` and scrambles it in order to disassociate the decision result from previous authentication runs. This operation is represented by algorithm **Shuffle** which outputs scrambled data `sclass` and a de-scrambling key π which the AS keeps to itself.

5. The VS uses the secret key k_d and `sclass` to perform the final decision step and produces a verdict d. This operation is represented by algorithm **Decide**.
6. Finally, the AS can recover the user's real identity, or a failure symbol, from the verdict d and the de-scrambling key π using algorithm **Identify**.

The soundness condition for our protocol is that the server-side system as a whole, and the AS in particular, produces a correct decision on the user's authenticity, i.e. recognises whether a new feature belongs to a valid user, and determines the correct identity. Formally, for soundness we require that the following probability yields a value sufficiently close to one for practical use as an authentication protocol, for valid static data $AS_{\texttt{data}}$, $DB_{\texttt{data}}$ and $VS_{\texttt{data}}$ resulting from a successful enrolment procedure, and for all fresh features $v_{\texttt{ID}}$:

$$\texttt{Pr}\left[\mathbf{Identify}(d,\pi,AS_{\texttt{data}})=r\;\middle|\;\begin{array}{l}(\texttt{params},k_d)\leftarrow\mathbf{Gen}(1^\kappa)\\ \texttt{auth}\leftarrow\mathbf{S}(v_{\texttt{ID}},\texttt{params})\\ \texttt{class}\leftarrow\mathbf{Classify}(\texttt{params},\texttt{auth},DB_{\texttt{data}})\\ (\texttt{sclass},\pi)\leftarrow\mathbf{Shuffle}(\texttt{params},\texttt{class},AS_{\texttt{data}})\\ d\leftarrow\mathbf{Decide}(\texttt{sclass},\texttt{params},k_d,VS_{\texttt{data}})\end{array}\right].$$

where $r = \texttt{ID}$ when `ID` is in the valid set of users, and $r = \perp$ otherwise.

4.4 Security Model

Intuitively, the security requirements we want to impose are the following:

– **Privacy** None of the services (and no passive attacker observing communications) gets enough information to reconstruct an identity/feature pair. More precisely, none of the services can distinguish whether a particular measurement belongs to a particular person.
– **Untraceability** Except for the authentication service, none of the other services (and no passive attacker observing communications) gets enough information to recognize a previously authenticated user. More precisely, the database service and the matching service cannot distinguish whether two authentication requests belong to the same person.

We assume that the servers are honest-but-curious, namely that they do not collude and follow the protocol rules, but may try to use the information they obtain to subvert the previous requirements. Formally, this translates into two security models.

Privacy: Feature Indistinguishability The three server-side components, as well as any eavesdropper which is able to observe the message exchanges corresponding to a protocol execution, must be unable to distinguish between which of two features belongs to a particular system user. We call this requirement feature indistinguishability (`fIND`). We define it using the following experiment, which takes as input a parameter $\texttt{adv} \in \{AS, DB, VS, Eve\}$, and fresh readings v_0, from valid user $\texttt{ID} \in U$, and v_1 from any user.

```
$\mathtt{Exp}^{\mathtt{fIND}}_{\beta}(\mathtt{adv}, v_0, v_1)$
    $(\mathtt{params}, k_d) \leftarrow \mathbf{Gen}(1^{\kappa})$
    $\mathtt{auth} \leftarrow \mathbf{S}(v_0, \mathtt{params})$
    $\mathtt{class} \leftarrow \mathbf{Classify}(\mathtt{params}, \mathtt{auth}, DB_{\mathtt{data}})$
    $(\mathtt{sclass}, \pi) \leftarrow \mathbf{Shuffle}(\mathtt{params}, \mathtt{class}, AS_{\mathtt{data}})$
    $d \leftarrow \mathbf{Decide}(\mathtt{sclass}, k_d, SV_{\mathtt{data}})$
    $r \leftarrow \mathbf{Identify}(d, \pi, AS_{\mathtt{data}})$
    Return $(v_{\beta}, \mathtt{view}_{\mathtt{adv}})$
```

$$\begin{aligned}
\mathtt{view}_{AS} &:= (\mathtt{auth}, \mathtt{class}, \mathtt{sclass}, \pi, d, r, AS_{\mathtt{data}}, \mathtt{params})\\
\mathtt{view}_{DB} &:= (\mathtt{auth}, \mathtt{class}, DB_{\mathtt{data}}, \mathtt{params})\\
\mathtt{view}_{VS} &:= (\mathtt{sclass}, d, VS_{\mathtt{data}}, k_d, \mathtt{params})\\
\mathtt{view}_{Eve} &:= (\mathtt{auth}, \mathtt{class}, \mathtt{sclass}, d, \mathtt{params})
\end{aligned}$$

We require that, for all $\mathtt{ID} \in U$ and all $\mathtt{adv} \in \{AS, DB, VS, Eve\}$, the following distributions be computationally indistinguishable ($\equiv$):

$$\{(\mathtt{ID}, \mathtt{Exp}^{\mathtt{fIND}}_{\beta=1}(\mathtt{adv}, v_0, v_1))\} \equiv \{(\mathtt{ID}, \mathtt{Exp}^{\mathtt{fIND}}_{\beta=0}(\mathtt{adv}, v_0, v_1))\}$$

We define advantage $\mathtt{Adv}^{\mathtt{fIND}}(\mathtt{adv})$ as (the absolute value of) the deviation from $1/2$ in the probability that the adversary guesses β.

Untraceability – User Indistinguishability The back-end server-side components, DB and VS, as well as any eavesdropper which is able to observe the message exchanges corresponding to a protocol execution, must be unable to distinguish if two independent authentication runs correspond to the same system user. We call this requirement user indistinguishability (uIND). We define it using the following experiment, which takes as input a parameter $\mathtt{adv} \in \{DB, VS, Eve\}$, and two fresh readings v_0 and v_1 corresponding to valid users uid and uid' respectively.

```
$\mathtt{Exp}^{\mathtt{uIND}}_{\beta}(\mathtt{adv}, v_0, v_1)$
    $(\mathtt{params}, k_d) \leftarrow \mathbf{Gen}(1^{\kappa})$
    $\mathtt{auth} \leftarrow \mathbf{S}(v_{\beta}, \mathtt{params})$
    $\mathtt{class} \leftarrow \mathbf{Classify}(\mathtt{params}, \mathtt{auth}, DB_{\mathtt{data}})$
    $(\mathtt{sclass}, \pi) \leftarrow \mathbf{Shuffle}(\mathtt{params}, \mathtt{class}, AS_{\mathtt{data}})$
    $d \leftarrow \mathbf{Decide}(\mathtt{sclass}, k_d, SV_{\mathtt{data}})$
    $r \leftarrow \mathbf{Identify}(d, \pi, AS_{\mathtt{data}})$
    Return $\mathtt{view}_{\mathtt{adv}}$
```

where the different views are defined as above.

We require that, for all valid users with user identifiers uid and uid', and all $\mathtt{adv} \in \{DB, VS, Eve\}$, the following distributions be computationally indistinguishable ($\equiv$):

$$\{(uid, uid', \mathtt{Exp}^{\mathtt{uIND}}_{\beta=1}(\mathtt{adv}, v_0, v_1))\} \equiv \{(uid, uid', \mathtt{Exp}^{\mathtt{uIND}}_{\beta=0}(\mathtt{adv}, v_0, v_1))\}$$

Again, we define advantage $\mathtt{Adv}^{\mathtt{uIND}}(\mathtt{adv})$ as (the absolute value of) the deviation from 1/2 in the probability that the adversary guesses β.

5 A Concrete Implementation

5.1 The SVM Classifier

We consider a $|U|$-class identification classifier called the Support Vector Machine (SVM) [17] and provide a short description of its operation. The basic SVM is a mono class authentication classifier[3]. Extension to U classes follows the *one-against-all* strategy: for each user $u \in U$, a mono classifier is trained using the remaining users (U/u) as the rejected class. For each user, the learning stage of the SVM determines both an outer and an inner hyperplane in a k-dimensional features space. Said hyperplanes are expressed as a linear combination of S known samples (so called support vectors $SV_{i,j} \in \mathbb{V}_{\mathtt{SVM}}, i = 1...S, j = 1...|U|$) weighted with $\alpha_{i,j} \in \mathbb{N}$ coefficients. Formally, we have

$$\mathbb{V}_{\mathtt{SVM}} = \mathbb{N}^k \text{ and } \mathbb{W}_{\mathtt{SVM}} = (\mathbb{N} \times \mathbb{V})^{S \times |U|}$$

During authentication, the SVM classifier evaluates the distance of the fresh feature v to these hyperplanes using a scalar product. To account for the fact that the user profile regions may not be linearly separable, the SVM may compute the scalar product in a higher dimension space. For this, the SVM classifier uses a kernel function $\mathbf{K}$ to project the data into the higher dimension space and compute the scalar product in this space in a single step. The advantage is that the computational cost is reduced when compared to a basic projection followed by the scalar product. The classifier function is therefore

$$cl_{\mathtt{SVM}} : \mathbb{V}_{\mathtt{SVM}} \times \mathbb{W}_{\mathtt{SVM}} \longrightarrow \mathbb{N}^{|U|}$$
$$cl_{\mathtt{SVM}}(v, w^*_{|U|}) := (cl^{(1)}_{\mathtt{SVM}}(v, w^*_{|U|}), \ldots, cl^{(|U|)}_{\mathtt{SVM}}(v, w^*_{|U|}))$$

where $w^*_{|U|}$ contains $(\alpha_{i,j}, SV_{i,j})]$ for $1 \leq i \leq S$ and $1 \leq j \leq |U|$ and

$$cl^{(j)}_{\mathtt{SVM}}(v, w^*_{|U|}) := \sum_{i=1}^{S} \alpha_{i,j}\, \mathbf{K}(v, SV_{i,j}).$$

In this paper, and to simplify the presentation, we will use the particular case where $\mathbf{K}(a, b)$ refers to the scalar product between a and b in the initial space: $\mathbf{K}(a, b) = \sum_{l=1}^{k} a_l b_l$.

The decision is calculated by finding the index of the maximum positive scalar contained in the vector $cl_{\mathtt{SVM}}(v, w^*)$. If no positive scalar exists, then the reject symbol is returned $(\perp)$:

$$\mathcal{D}_{\mathtt{SVM}}(cl_{\mathtt{SVM}}(v, w^*)) := \begin{cases} d \leftarrow \operatorname{argmax}_{j=1}^{|U|}(cl^{(j)}_{\mathtt{SVM}}(v, w^*)) \\ \mathtt{If}\ cl^{(d)}_{\mathtt{SVM}}(v, w^*) > 0 \\ \mathtt{Then\ return}\ d \\ \mathtt{Else\ return}\ \perp \end{cases}$$

[3] A classifier used in an authentication context "Am I who I claimed to be ?"

5.2 Algorithm Implementations

We refer the reader to Appendix A for a description of the Paillier cryptosystem. The concrete implementations we propose for the algorithms composing our authentication protocol are the following:

- **Gen**$(1^\kappa) \rightarrow (\mathtt{params}, k_d)$**.** The generation primitive simply uses the key generation algorithm for the Paillier cryptosystem to obtain (k_e, k_d), sets $\mathtt{params} \leftarrow k_e$ and returns $(\mathtt{params}, k_d)$.
- **S**$(v) \rightarrow \mathtt{auth}$**.** This algorithm takes as input a fresh feature for an unknown user. Recall that the feature space for the SVM is $\mathbb{V}_{\mathtt{SVM}} = \mathbb{N}^k$, but we can look at the feature as $v := (v_1, \ldots, v_k) \in \mathbb{Z}_n^k$. Encryption is carried out one component at a time and the algorithm returns:

$$\mathtt{auth} \leftarrow (\mathcal{E}_{\mathtt{Paillier}}(v_1, k_e), \ldots, \mathcal{E}_{\mathtt{Paillier}}(v_k, k_e))$$

- **Classify**$(\mathtt{auth}, DB_{\mathtt{data}}, \mathtt{params}) \rightarrow \mathtt{class}$. This algorithm uses the homomorphic properties of the Paillier encryption scheme to compute pre-decision SVM classification values without ever decrypting the features in $\mathtt{auth}$. More precisely, the algorithm takes the profile data $w^*_{|U|}$ in $DB_{\mathtt{data}}$ and calculates for $1 \leq j \leq |U|$

$$\mathtt{c}_j = \prod_{i=1}^{S} \overset{*}{\mathbf{K}}(\mathtt{auth}, SV_{i,j})^{\alpha_{i,j}} = \mathcal{E}_{\mathtt{Paillier}}(\sum_{i=1}^{S} \alpha_{i,j}\, \mathbf{K}(v, SV_{i,j}), \mathtt{params})$$

where, using $[\cdot]_l$ to denote the l^{th} component in a tuple, $\mathbf{K}^*$ is defined by

$$\overset{*}{\mathbf{K}}(\mathtt{auth}, SV_{i,j}) := \prod_{l=1}^{k} [\mathtt{auth}_j]_l^{[SV_{i,j}]_l}$$

To prevent the AS from performing an exhaustive search of the profile space, the DB also re-randomizes the encryptions by calculating:

$$\mathtt{class}_j = (\mathtt{c}_j r_j^n) \mod n^2$$

The algorithm returns $\mathtt{class} = (\mathtt{class}_1, \ldots, \mathtt{class}_{|U|})$.
- **Shuffle**$(\mathtt{class}) \rightarrow (\mathtt{sclass}, \pi)$. This algorithm generates a fresh permutation $\pi : \{1, \ldots, |U|\} \rightarrow \{1, \ldots, |U|\}$, re-randomizes all the ciphertext components in $\mathtt{class}$ and returns the permutated re-randomized vector as $\mathtt{sclass}$. More precisely, we have $\mathtt{sclass} = (\mathtt{sclass}_1, \ldots, \mathtt{sclass}_{|U|})$ where

$$\mathtt{sclass}_j = (\mathtt{class}_{\pi(j)} r_j^n) \mod n^2$$

- **Decide**$(\mathtt{sclass}, k_d, VS_{\mathtt{data}}) \rightarrow d$. This algorithm decrypts the components in $\mathtt{sclass}$ and performs classification as described for the SVM classifier. The result d is the index in the input vector corresponding to the largest positive scaler, or $\perp$ if no positive scalar exists.
- **Identify**$(d, \pi, AS_{\mathtt{data}}) \rightarrow \mathtt{ID}$. For authentication runs where $d \neq \perp$, this algorithm simply finds uid such that

$$uid = \pi^{-1}(d)$$

and returns the associated identity $\mathtt{ID}$. Otherwise it returns $\perp$.

5.3 Security Analysis

In Appendices B and C we prove two theorems, which capture the security properties of the proposed protocol.

Theorem 1. *The proposed protocol ensures feature privacy. More precisely, any PPT adversary has negligible advantage in distinguishing the distributions associated with* $\mathtt{Exp}^{\mathtt{fIND}}$.

Theorem 2. *The proposed protocol ensures user untraceability. More precisely, any PPT adversary has negligible advantage in distinguishing the distributions associated with* $\mathtt{Exp}^{\mathtt{uIND}}$.

Remark: On the (in)security of the Bringer et al. protocol The fIND model we propose is a more formal version of *Security Requirement 2* proposed by Bringer et al. [1] for their authentication protocol. The security argument presented for this protocol describes a reduction to the semantic security of the Goldwasser-Micali cryptosystem. However, the argument fails to cover a simple attack by the AS. The attack is possible because the interaction between the AS server and the DB server does not include a re-randomization of the resulting ciphertexts. This means that it may be possible for the AS to recover the user profile data that the DB server has used in the calculations. After recovering a biometric profile, the AS server is able to determine on its own which features belong to a user, without even executing the protocol. More precisely, and referring to the notation in [1], the AS calculates $(\mathcal{E}(t_1, pk), \dots, \mathcal{E}(t_N, pk))$, where N is the number of users, $t_j = 0$ for all indexes except $j = i$ for which $t_j = 1$, and i is the index of the user to be authenticated. The DB server receives these ciphertexts and calculates $\mathcal{E}(b_{i,k}, pk) = \prod_{j=1}^{N} \mathcal{E}(t_j, pk)^{b_{j,k}} \mod n$, for $1 \leq k \leq M$, where $(b_{i,1}, \dots, b_{i,M})$ is the biometric profile corresponding to user i. On receiving $\mathcal{E}(b_{i,k}, pk)$, the AS can try to work out whether $b_{i,k}$ is 1 or 0. To do this, it tries to calculate $\mathcal{E}(b_{i,k}, pk)/\prod_{j \in J} \mathcal{E}(t_j, pk) \mod n$, for all subsets $J \subset \{1 \dots N\} \setminus i$, where $\mathcal{E}(t_j, pk)$ are exactly the same as those passed originally to the DB. If in these calculations the AS obtains 1, then it knows $b_{i,k} = 0$; if it obtains $\mathcal{E}(t_i, pk)$, then it knows $b_{i,k} = 1$. The feasibility of this attack depends on the number of users N: in fact its complexity is exponential in N, which means it may be infeasible for a very large N. However, a simple patch to the protocol, preventing the attack altogether even for small N, is to ensure that the DB server re-randomises ciphertexts after applying the homomorphic transformations. We emphasise that the security reduction presented in this paper for the proposed protocol explicitly precludes this type of attack.

6 Discussion and Conclusion

We have presented a hybrid protocol for secure biometric authentication which permits adopting state-of-the art pattern recognition classifiers to improve over the authentication accuracy of existing solutions. Our protocol follows the approach of Bringer et al. [1], adopting the point of view that biometric information may be stored in public servers, as long as it is guaranteed that it remains anonymous if security is breached. To allow for the use of more powerful classification techniques, namely the SVM classifier, we use the Pailler public key encryption scheme, taking advantage of its homomorphic properties.

The main advantages of the proposed protocol over previous solutions can be summarised as follows:

– Potential for much better accuracy using different types of biometric signals, including behavioural ones.
– Improved user privacy, since user identities are not transmitted at any point in the protocol execution. This is possible because the classifiers we adopt are *identification classifiers* which do not need to know who the user claims to be in order to perform authentication and recover the user identity.

Security of the proposed protocol has been formalised in two security models: feature indistinguishability and user indistinguishability. These are extended versions of the models proposed in [1], where we also account for eavesdroppers external to the system. We provide a reduction relating the security of our authentication protocol with the security of the Paillier encryption scheme. We also describe a simple attack against the Bringer et al. protocol, and show how it can be easily repaired.

Acknowledgements The authors would like to thank Michel Abdalla for reading and commenting on an earlier version of this paper.

References

1. Bringer, J., Chabanne, H., Izabachène, M., Pointcheval, D., Tang, Q., Zimmer, S.: An application of the goldwasser-micali cryptosystem to biometric authentication. In Pieprzyk, J., Ghodosi, H., Dawson, E., eds.: ACISP. Volume 4586 of Lecture Notes in Computer Science., Springer (2007) 96–106
2. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Cryptology ePrint Archive, Report 2003/235 (2003) `http://eprint.iacr.org/`.
3. Boyen, X.: Reusable cryptographic fuzzy extractors. In: CCS '04: Proceedings of the 11th ACM conference on Computer and communications security, New York, NY, USA, ACM (2004) 82–91
4. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure remote authentication using biometric data. In: Advances in Cryptology—EUROCRYPT 2005. Volume 3494 of Lecture Notes in Computer Science., Berlin: Springer-Verlag (2005) 147–163 Available at `http://www.cs.stanford.edu/~xb/eurocrypt05b/`.
5. Monrose, F., Reiter, M.K., Wetzel, S.: Password hardening based on keystroke dynamics. In: CCS '99: Proceedings of the 6th ACM conference on Computer and communications security, New York, NY, USA, ACM (1999) 73–82
6. Hocquet, S., Ramel, J.Y., Cardot, H.: Fusion of methods for keystroke dynamic authentication. Automatic Identification Advanced Technologies, 2005. Fourth IEEE Workshop on (17-18 Oct. 2005) 224–229

7. Monrose, F., Reiter, M., Li, Q., Wetzel, S.: Cryptographic key generation from voice. Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on (2001) 202–213
8. Yegnanarayana, B., Prasanna, S., Zachariah, J., Gupta, C.: Combining evidence from source, suprasegmental and spectral features for a fixed-text speaker verification system. Speech and Audio Processing, IEEE Transactions on **13** (July 2005) 575–582
9. Cauchie, S., Brouard, T., Cardot, H.: From features extraction to strong security in mobile environment: A new hybrid system. In Meersman, R., Tari, Z., Herrero, P., eds.: OTM Workshops (1). Volume 4277 of Lecture Notes in Computer Science., Springer (2006) 489–498
10. Feng, H., Choong, W.C.: Private key generation from on-line handwritten signatures. Inf. Manag. Comput. Security **10** (2002) 159–164
11. Fuentes, M., Garcia-Salicetti, S., Dorizzi, B.: On-line signature verification: Fusion of a hidden markov model and a neural network via a support vector machine. iwfhr **00** (2002) 253
12. Goh, A., Ling, D.N.C.: Computation of cryptographic keys from face biometrics. In Lioy, A., Mazzocchi, D., eds.: Communications and Multimedia Security. Volume 2828 of Lecture Notes in Computer Science., Springer (2003) 1–13
13. Yan, T.T.H.: Object recognition using fractal neighbor distance: eventual convergence and recognition rates. Pattern Recognition, 2000. Proceedings. 15th International Conference on **2** (2000) 781–784 vol.2
14. Uludag, U.A.J.: Securing fingerprint template: Fuzzy vault with helper data. Computer Vision and Pattern Recognition Workshop, 2006 Conference on (17-22 June 2006) 163–163
15. Guo, H.: A hidden markov model fingerprint matching approach. Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on **8** (18-21 Aug. 2005) 5055–5059 Vol. 8
16. Hao, F., Anderson, R., Daugman, J.: Combining crypto with biometrics effectively. IEEE Transactions on Computers **55** (2006) 1081–1088
17. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research **2** (2001) 265–292
18. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT. (1999) 223–238
19. Paillier, P., Pointcheval, D.: Efficient public-key cryptosystems provably secure against active adversaries. In: ASIACRYPT. (1999) 165–179
20. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: EUROCRYPT. (2000) 259–274

Appendix A: Paillier Public Key Encryption Scheme

The Paillier public key encryption scheme [18, 19] can be described as follows:

– **Key generation:** $\mathcal{G}_{\texttt{Paillier}}(1^\kappa) = (k_d, k_e)$. The PPT key generation algorithm takes a security parameter 1^κ as input, and randomly generates two large prime numbers p and q, setting $n = pq$ and $\lambda = lcm(p-1, q-1)$. The algorithm then randomly selects $g \in \mathbb{Z}^*_{n^2}$, such that n divides the order of g. This can be ensured by checking that

$$\gcd(L(g^\lambda \mod n^2), n) = 1, \text{ where } L(u) = \frac{u-1}{n}$$

which in turn implies that the following multiplicative inverse exists:

$$\mu = (L(g^{\lambda} \mod n^2))^{-1} \mod n$$

The public key is then $k_e = (n, g)$ and the secret key is $k_d = (\mu, \lambda)$.

- **Encryption:** $\mathcal{E}_{\texttt{Paillier}}(m, k_e)$. The PPT encryption algorithm takes a message $m \in \mathbb{Z}_n$ and the public key $k_e = (n, g)$, generates r uniformly at random from $\mathbb{Z}_n^*$ and outputs a ciphertext $c \in \mathbb{Z}_{n^2}$, where $c = g^m \cdot r^n \mod n^2$.
- **Decryption:** $\mathcal{D}_{\texttt{Paillier}}(c, k_d)$. The deterministic decryption algorithm takes a ciphertext c and the secret key and outputs the plaintext m, which is recovered as $m = L(c^{\lambda} \mod n^2) \cdot \mu \mod n$.

It has been shown [19] that, under the composite residuosity assumption, the Paillier cryptosystem provides semantic security against chosen-plaintext attacks (IND-CPA). In other words, any PPT adversary $\mathcal{A}$ has only a negligible advantage in the following game against the Paillier cryptosystem:

```
$\texttt{Exp}^{\texttt{IND-CPA}}_{\texttt{Paillier}}(\mathcal{A})$
    $(k_d, k_e) \leftarrow \mathcal{G}_{\texttt{Paillier}}(1^{\kappa})$
    $(m_0, m_1, s) \leftarrow \mathcal{A}_1(k_e)$
    $\beta \leftarrow \{0, 1\}$
    $c \leftarrow \mathcal{E}_{\texttt{Paillier}}(m_{\beta})$
    $\beta' \leftarrow \mathcal{A}_2(c, s)$
    return $\beta'$
```

where the attacker's advantage $\mathbf{Adv}^{\texttt{IND-CPA}}_{\texttt{Paillier}}$ is defined as:

$$\mathbf{Adv}^{\texttt{IND-CPA}}_{\texttt{Paillier}} = |\Pr[\texttt{Exp}^{\texttt{IND-CPA}}_{\texttt{Paillier}} = 1 | \beta = 1] - \Pr[\texttt{Exp}^{\texttt{IND-CPA}}_{\texttt{Paillier}} = 1 | \beta = 0]|$$

In our scheme we will be using the Paillier cryptosystem to encrypt biometric features represented as short sequences of integer numbers. Encryption will be component-wise, where we assume that each integer component in the feature is in a range suitable for direct encoding into the message space[4]. For this reason we require a generalisation of the IND-CPA property allowing the adversary to make a polynomial number n of queries to a Left-or-Right challenge oracle. We call this notion n-IND-CPA and emphasize that the security of the Paillier encryption scheme in this setting is implied by its semantic security [20].

We will also take advantage of the following homomorphic properties of the Paillier encryption scheme:

$$\mathcal{E}_{\texttt{Paillier}}(a, k_e)\mathcal{E}_{\texttt{Paillier}}(b, k_e) = \mathcal{E}_{\texttt{Paillier}}(a + b, k_e)$$

$$\mathcal{E}_{\texttt{Paillier}}(a, k_e)^b = \mathcal{E}_{\texttt{Paillier}}(ab, k_e)$$

The aditive property also provides a method to re-randomize a given Paillier cryptosystem which we will use:

$$(\mathcal{E}_{\texttt{Paillier}}(a, k_e; r') \cdot r^n) \mod n^2 = \mathcal{E}_{\texttt{Paillier}}(a, k_e; r'r).$$

[4] In practice, SVM features can be represented using integers in the range -100 to 100, which can be easily encoded into $\mathbb{Z}_n$.

Appendix B: Proof of Theorem 1

The proof is divided in four claims, corresponding to the different values that $\texttt{adv}$ can take.

Claim 1: $\{(\texttt{ID}, \texttt{Exp}^{\texttt{fIND}}_{\beta=1}(AS, v_0, v_1))\} \equiv \{(\texttt{ID}, \texttt{Exp}^{\texttt{fIND}}_{\beta=0}(AS, v_0, v_1))\}$. To prove this claim we argue that any distinguisher with non-negligible advantage can be used to break the security of the Paillier cryptosystem. For this we construct a sequence of three games, where the first corresponds to distinguishing the distributions associated with $\texttt{Exp}^{\texttt{fIND}}$. The second game is identical to the original one, with the caveat that instead of encrypting v_0, the experiment now encrypts a random value in the feature space v_0'. We claim that the advantage of any adversary in distinguishing the distributions associated with this new experiment must be negligibly different from that in the original game. To show this we build a distinguisher D_1 which attacks the k-IND-CPA security of the Paillier cryptosystem, where k is the length of the feature vector v, given any adversary contradicting the previous claim. D_1 works as follows:

- D_1 receives the Paillier challenge public key and uses it as $\texttt{params}$.
- D_1 sets up a make-believe authentication system with a set of legitimate users U, generates one feature v_0 for a particular user $\texttt{ID}$, plus an additional feature v_1 for an arbitrary user, and a random value in the feature space v_0'.
- D_1 passes features v_0 and v_0' to the k-IND-CPA challenge oracle, obtaining a component-wise encryption of one of these features, and takes this encryption as $\texttt{auth}$.
- D_1 then simulates the protocol trace for AS by running the **Classify** and **Shuffle** algorithms. Since D_1 does not know the secret key associated with the challenge public key, it simply doesn't run **Decide** and **Identify**, taking d and r corresponding to $\texttt{ID}$ as the obvious result. Note that this is consistent with the feature indistinguishability security game. The protocol trace generated for the AS is therefore

$$\texttt{view}_{AS} = (\texttt{auth}, \texttt{class}, \texttt{sclass}, \pi, d, r, AS_{\texttt{data}}, \texttt{params})$$

- D_1 tosses a coin β and passes $\{(\texttt{ID}, (v_\beta, \texttt{view}_{AS}))\}$ to AS.
- Eventually, AS will return its guess β', and D_1 returns $b = 1$ if A's guess is correct and $b = 0$ otherwise.

Note that if the k-IND-CPA challenge encrypts v_0 (call this event E), then AS is run according to the correct rules of $\texttt{Exp}^{\texttt{fIND}}$ and therefore game 1. Conversely, if it encrypts v_0' then the adversary is run under the rules of game 2. Denoting by $Pr[S_i]$the probability of success in game i, we have:

$$|Pr[S_1] - Pr[S_2]| = |Pr[\beta' = \beta|E] - Pr[\beta' = \beta|\neg E]| = \mathbf{Adv}^{k-\texttt{IND-CPA}}_{\texttt{Paillier}}(D_1)$$

To bound the probability that the AS can distinguishing the distributions associated with game 2 we observe that the protocol trace itself contains no information about v_0 or v_1. Hence, any advantage in distinguishing the features can only be obtained by the AS by recovering biometric profile information from the protocol trace i.e. attacking $DB_{\texttt{data}}$.

To ensure that this is not possible, we introduce game 3, where $DB_{\texttt{data}}$ is replaced by a random value in the profile space. It is clear that under the rules of game 3, and since no information is provided to the AS regarding the biometric system at all, it can have no advantage in guessing β, i.e. $Pr[S_3] = 1/2$.

To complete the proof, we show that any adversary whose behaviour changes non-negligibly from game 2 to game 3 can be used to attack the $|U|$-IND-CPA security of the Paillier encryption scheme. For this, we build a distinguisher D_2 which works as follows:

- D_2 receives the Paillier challenge public key and uses it as `params`.
- D_2 sets up a make-believe authentication system with a set of legitimate users U, generates one feature v_0 for a particular user `ID`, plus an additional feature v_1 for an arbitrary user, and a random value in the feature space v_0'.
- D_2 (component-wise) encrypts v_0' of appropriate size with the challenge public key and calls this `auth`.
- D_2 then uses $DB_{\texttt{data}}$ to calculate the cleartext versions of pre-classification results corresponding to v_0' (call these scores $s = (s_1, \ldots, s_{|U|})$).
- D_2 then generates an alternative version of $DB_{\texttt{data}}$ by selecting a random value in the profile space, and calculates the cleartext versions of pre-classification results corresponding to v_0' (call these scores $r = (r_1, \ldots, r_{|U|})$) under this arbitrary pre-classification system .
- D_2 then uses the $|U|$-IND-CPA challenge oracle to construct `class` by taking $\texttt{class}_j$ as the answer to a query (s_j, r_j).
- D_2 then executes **Shuffle** to obtain `sclass` and sets d and r to the values corresponding to `ID`. The protocol trace generated for the AS is therefore

$$\texttt{view}_{AS} = (\texttt{auth}, \texttt{class}, \texttt{sclass}, \pi, d, r, AS_{\texttt{data}}, \texttt{params})$$

- D_2 tosses a coin β and passes $\{(\texttt{ID}, (v_\beta, \texttt{view}_{AS}))\}$ to AS.
- Eventually, AS will return its guess β', and D_2 returns $b = 1$ if A's guess is correct and $b = 0$ otherwise.

Clearly, D_2 interpolates between games 2 and 3 depending on the hidden bit in the Left-or-Right challenge oracle, and we have:

$$|Pr[S_2] - Pr[S_3]| = \mathbf{Adv}^{|U|-\texttt{IND-CPA}}_{\texttt{Paillier}}(D_2)$$

Finally, putting the previous results together, we have

$$\texttt{Adv}^{\texttt{fIND}}(AS) \leq \mathbf{Adv}^{k-\texttt{IND-CPA}}_{\texttt{Paillier}}(D_1) + \mathbf{Adv}^{|U|-\texttt{IND-CPA}}_{\texttt{Paillier}}(D_2) \qquad \square$$

Similarly to the arguments in [1], the remaining claims follow directly from the fact that the adversary, in each case, has no information about user identities.

Claim 2: $\{(\texttt{ID}, \texttt{Exp}^{\texttt{fIND}}_{\beta=1}(DB, v_0, v_1))\} \equiv \{(\texttt{ID}, \texttt{Exp}^{\texttt{fIND}}_{\beta=0}(DB, v_0, v_1))\}$.
Claim 3: $\{(\texttt{ID}, \texttt{Exp}^{\texttt{fIND}}_{\beta=1}(VS, v_0, v_1))\} \equiv \{(\texttt{ID}, \texttt{Exp}^{\texttt{fIND}}_{\beta=0}(VS, v_0, v_1))\}$.
Claim 4: $\{(\texttt{ID}, \texttt{Exp}^{\texttt{fIND}}_{\beta=1}(Eve, v_0, v_1))\} \equiv \{(\texttt{ID}, \texttt{Exp}^{\texttt{fIND}}_{\beta=0}(Eve, v_0, v_1))\}$. $\square$

Appendix C: Proof of Theorem 2

The proof is divided in three claims, corresponding to the different values that $\mathtt{adv}$ can take.

Claim 1: $\{(uid, uid', \mathtt{Exp}^{\mathtt{uIND}}_{\beta=1}(DB, v_0, v_1))\} \equiv \{(uid, uid', \mathtt{Exp}^{\mathtt{uIND}}_{\beta=0}(DB, v_0, v_1))\}$. The DB server shares with the AS server the notion of user identifier. However, it has no access to user features or decision results at any point, so the only means it would have to achieve user traceability would be to break the security of the underlying encryption scheme. More formally, we can construct a reduction to the k-IND-CPA security of the Paillier encryption scheme, where k is as before, by describing an algorithm B that attacks the k-IND-CPA security of the Paillier cryptosystem given an adversary which contradicts the previous claim:

- B receives the Paillier challenge public key and uses it as $\mathtt{params}$.
- B sets up a make-believe authentication system with a set of legitimate users U and generates valid feature/user identifier pairs (v_0, uid) and (v_1, uid').
- B passes (v_0, v_1) to the k-IND-CPA challenge oracle, obtaining a component-wise encryption of one of these features, and takes this encryption as $\mathtt{auth}$.
- B then simulates the protocol trace for DB by running the **Classify** algorithm. The protocol trace generated for the DB is therefore

$$\mathtt{view}_{DB} = (\mathtt{auth}, \mathtt{class}, DB_{\mathtt{data}}, \mathtt{params})$$

- B passes $(uid, uid', \mathtt{view}_{DB})$ to DB.
- Eventually, DB will return its guess β', and B simply returns this as its own guess of which feature is encrypted in the k-IND-CPA challenge.

Note that the way in which B is constructed directly transforms any advantage in A guessing β into an advantage in guessing the k-IND-CPA challenge bit. More precisely, and taking into account our definitions of advantage:

$$\mathtt{Adv}^{\mathtt{uIND}}(DB) = 2\mathbf{Adv}^{k-\mathtt{IND-CPA}}_{\mathtt{Paillier}}(B)$$

□

Claim 2: $\{(uid, uid', \mathtt{Exp}^{\mathtt{uIND}}_{\beta=1}(VS, v_0, v_1))\} \equiv \{(uid, uid', \mathtt{Exp}^{\mathtt{uIND}}_{\beta=0}(VS, v_0, v_1))\}$. The VS is unable to trace user authentication runs due to the fact that a fresh independent permutation π is generated each time the service is called. In fact, in the information-theoretical sense VS's view leaks nothing about user identifiers: the VS receives no information about user identifiers in its static data, and successive decision results produce indexes d are independent and uniformly independently distributed, due to the action of the random permutation in **Shuffle**. □

Claim 3: $\{(uid, uid', \texttt{Exp}^{\texttt{uIND}}_{\beta=1}(Eve, v_0, v_1))\} \equiv \{(uid, uid', \texttt{Exp}^{\texttt{uIND}}_{\beta=0}(Eve, v_0, v_1))\}$. Eavesdroppers cannot trace user requests because they cannot correlate the ephemeral index d associated with `sclass` with the static user identifier indexes associated with `class`. This is ensured by re-randomizing the ciphertexts contained in these protocol messages. Hence, without breaking the security of the Paillier encryption scheme, eavesdroppers can have no advantage in tracing user requests.

More formally, we argue that any distinguisher which contradicts the claim can be used to break the security of the Paillier cryptosystem. For this we construct a sequence of two games, where the first corresponds to distinguishing the distributions associated with $\texttt{Exp}^{\texttt{uIND}}$. The second game is identical to the original one, with the exception that the value of d, the result of **Decide**, is selected uniformly at random. We argue that the advantage of any adversary under the rules of this slightly altered security game must be negligibly different from its advantage in the original game. We support this argument by presenting a distinguisher D which is able to translate A's advantage in detecting this slight change of rules into an advantage in attacking the $|U|$-IND-CPA security of the Paillier cryptosystem:

- D receives the Paillier challenge public key and uses it as `params`.
- D sets up a make-believe authentication system with a set of legitimate users U and generates valid feature/user identifier pairs (v_0, uid) and (v_1, uid').
- D flips a bit β and (component-wise) encrypts v_β with the challenge public key and calls this `auth`.
- D then uses the $DB_{\texttt{data}}$ to calculate the cleartext versions of the pre-classification results corresponding to v_β (we call these scores $(s_1, \ldots, s_{|U|})$) and encrypts them with the challenge public key to obtain a simulated `class`.
- D generates two random permutations π and π' compatible with possible runs of the authentication system,
- D then constructs the simulated `sclass` by calling the external Left-or-Right oracle with $(\pi(s_j), \pi'(s_j))$ for each component $\texttt{sclass}_j$.
- D then finalises the protocol trace for Eve by taking $d = \pi(uid)$ if $\beta = 0$ or $d = \pi(uid')$ if $\beta = 1$. The protocol trace generated for the Eve is therefore

$$\texttt{view}_{Eve} = (\texttt{auth}, \texttt{class}, \texttt{sclass}, d, \texttt{params})$$

- D passes $(uid, uid', \texttt{view}_{Eve})$ to Eve.
- Eventually, Eve will return its guess β', and D returns 1 if Eve's guess is correct, and 0 otherwise.

Note that D perfectly simulates a protocol run under the rules of geme 1 using π, if the Left-or-Right oracle is encrypting the left-hand messages (call this event E). Conversely, if the oracle is encrypting the right-hand messages, then the protocol run is using π'. However, since the value of d is calculated using π, it will be independent and uniformly distributed under Eve's view, which means that D is running the adversary under the rules of game 2. Hence, any difference in the adversary's behaviour when run in games 1 or 2 is translated by D into an advantage in attacking the $|U|$-IND-CPA security of the Paillier cryptosystem. Denoting by $Pr[S_i]$the probability of success in game i, we have:

$$|Pr[S_1] - Pr[S_2]| = |Pr[\beta' = \beta|E] - Pr[\beta' = \beta|\neg E]| = \mathbf{Adv}^{|U|-\texttt{IND-CPA}}_{\texttt{Paillier}}(D)$$

To bound the probability of success of an adversary in game 2, we present an algorithm B which uses any attacker with non-negligible advantage in game 2 to break the k-IND-CPA security of the Paillier cryptosystem:

- B receives the Paillier challenge public key and uses it as `params`.
- B sets up a make-believe authentication system with a set of legitimate users U and generates valid feature/user identifier pairs (v_0, uid) and (v_1, uid').
- B passes (v_0, v_1) to the k-IND-CPA challenge oracle, obtaining a component-wise encryption of one of these features. We take this encryption as `auth`.
- B then simulates the protocol trace for Eve by running the **Classify** and **Shuffle** algorithms. Since B does not know the secret key associated with the challenge public key, it simply doesn't run **Decide** taking a random d as the result. The protocol trace generated for the Eve is therefore

$$\texttt{view}_{Eve} = (\texttt{auth}, \texttt{class}, \texttt{sclass}, d, \texttt{params})$$

- B passes $(uid, uid', \texttt{view}_{Eve})$ to Eve.
- Eventually, Eve will return its guess β', and B simply returns this as its own guess of which feature is encrypted in the k-IND-CPA challenge.

Putting together the result relating games 1 and 2 with the fact that B perfectly simulates the second game, we have:

$$\texttt{Adv}^{\texttt{uIND}}(Eve) \leq \mathbf{Adv}^{|U|-\texttt{IND-CPA}}_{\texttt{Paillier}}(D) + 1/2\mathbf{Adv}^{k-\texttt{IND-CPA}}_{\texttt{Paillier}}(B) \qquad \square$$