Privacy-Preserving Speaker Authentication

Manas Pathak[1], Jose Portelo[2], Bhiksha Raj[1], and Isabel Trancoso[2]

[1] Carnegie Mellon University, Pittsburgh, PA, USA
[2] INESC-ID/IST, Lisbon, Portugal

Abstract. Speaker authentication systems require access to the voice of the user. A person's voice carries information about their gender, nationality etc., all of which become accessible to the system, which could abuse this knowledge. The system also stores users' voice prints – these may be stolen and used to impersonate the users elsewhere. It is therefore important to develop *privacy preserving* voice authentication techniques that enable a system to authenticate users by their voice, while simultaneously obscuring the user's voice and voice patterns from the system. Prior work in this area has employed expensive cryptographic tools, or has cast authentication as a problem of exact match with compromised accuracy. In this paper we present a new technique that employs secure binary embeddings of feature vectors, to perform voice authentication in a privacy preserving manner with minimal computational overhead and little loss of classification accuracy.

1 Introduction

The number of services provided to users over the network grows daily. Many of these services require users to authenticate themselves, typically through passwords. Increasingly, these services are accessed through small, voice-enabled client devices such as smartphones and cellphones. In these cases, authenticating a user by voice is often more convenient than typing in passwords or passphrases. Even in cases where passwords may be typed conveniently, they may be combined with voice-based authentication as an added form of security. As a result, voice-based authentication systems, also often referred to as *speaker authentication systems* or *speaker verification systems* are becoming increasingly popular.

In a speaker authentication system, a user enrolls with the system by providing some "enrollment" recordings of themselves from which the system may form a voice print or model for the user. Subsequently, during operation the user announces their identity and provides an "authentication" recording to the system. The system compares this recording to the stored model for the announced identity and determines if the the user is indeed who they claim to be.

However, in the process of using speaker authentication systems, the user currently places a lot of faith in the system. A person's voice, in addition to being a biometric, carries information regarding the person's gender, their emotional state, their nationality, etc. The implicit trust is that the system will not extract this information from the signal and use it for other undesired purposes, such as to generate advertisements aimed at the user, or to sell it to third parties.

D. Gollmann and F.C. Freiling (Eds.): ISC 2012, LNCS 7483, pp. 1–22, 2012.

There are other risks as well. In the discussion that follows, we will not distinguish between the system and a malicious hacker who may have compromised the system and has access to all data and information it possesses. The system may manipulate and edit the recordings provided by the speaker to create fake recordings that the user never spoke. It could also use the recordings to break into other systems where the user has authenticated themselves by voice.

In addition to the privacy and security violations listed above which result from unintended use of the speaker's data, there is scope for direct privacy violation through usage of the authentication system in *intended* ways. The system could use the model it possesses for a user to uncover other recordings by the user, such as on services like YouTube, or even from recordings of audio in public places, where the user may have assumed anonymity.

Clearly then, in order to ensure the user's privacy the following conditions must ideally be satisfied: First, the system should not have clear access to the audio recorded by the user. Second, the system should also not possess a model of the user's speech that it could use to identify the speaker elsewhere. These are not entirely unreasonable requirements; similar criteria are in fact stated as being desirable for other forms of secure biometrics as well [1].

In this paper we describe two recently-developed *privacy-preserving* frameworks for speaker authentication that address these problems by enabling a user to perform authentication without revealing their actual voice patterns to the system, either during enrollment or during authentication phases.

The first framework [2] employs *secure function evaluation* (SFE) protocols [13] to implement a conventional state-of-art voice authentication algorithm [14] in a manner that enables the user and system to interact only on encrypted data which the system cannot decrypt and observe in plaintext. Both the audio transmitted by the user to the system and the models stored by it are encrypted in this manner. Thus the system (or an adversary who breaks into it) can neither manipulate the incoming audio for nefarious purposes, nor make any undesired inferences from the audio or the models it possesses.

The second approach [3] takes a different tack – it modifies the basic pattern matching algorithms employed for speaker authentication to enable them to be performed through exact and inexact match of hash strings derived from the audio obtained through stochastic locality-sensitive-hashing functions. Here too, the data transmitted by the user to the system, including both enrollment and authentication data, are transformed in a manner that does not permit the system to make undesired inferences from it, thereby protecting the user.

The SFE-based approach provides the same accuracy as the state-of-the-art in voice authentication systems, since it essentially only embeds the same computation into a privacy-preserving framework. It also generalizes to other problems in speech processing, such as the training of Gaussian mixture densities or HMMs from private speech data [15] and inference over HMMs from private data [16], both of which are component problems in many speech applications. On the other hand, it requires repeated encryption and decryption of the data, imposing high computational overhead on the process. Moreover, it remains

vulnerable to imposters who may take over the user's client device, and while these can also be protected against, such protection would impose additional computational expense. The hashing-based methods, on the other hand, are relatively lightweight and fast; however by modifying the fundamental classification algorithms themselves they may compromise classification accuracy. Also, they are specific to the problem at hand – speaker authentication, and do not generalize beyond it in their current format.

In all cases we envision a client-server model, where a user employs a computation-capable client device such as a smartphone or computer, which can perform the operations necessary to protect the data. For completeness, we must also consider the possibility of an imposter who gains access to the user's client device and can manipulate the data its transmits. The proposed frameworks must ideally ensure that such an imposter cannot break into the system and pretend to be the user. We do *not* address the issue of fraudulent voice input, such as by imposters who may attempt to mimic the user's voice or criminals who may force the user to speak; these do not fall under the scope of the presented work. We will also generally assume that all communication between the user and the system is over an appropriately secured channel and hence impervious to man-in-the-middle or replay attacks by eavesdroppers.

In closing this introductory section, we note that while there has been a substantial body of work on general techniques for data processing with privacy constraints, concerns about the privacy of voice-based biometric systems, and speech applications in general have largely been ignored until recent times [17], and literature addressing the topic is sparse. Smaragdis and Shashanka [15] propose SFE protocols for training and evaluating Gaussian mixtures and hidden Markov models on speech data, with privacy constraints. Pathak, et al. [16] develop and implement an efficient protocol for privacy preserving HMM inference applied to isolated word recognition. The work reported in this article and the related citations are the only extant publications that address the issue of privacy-preserving speech biometrics, to the best of our knowledge.

In the following sections we first describe the problem of speaker authentication and common state-of-art methods for the task (Section 2). Subsequently, we describe the SFE-based procedure in Section 3 and the hashing-based approach in Section 4, followed by conclusions and discussion in Section 5.

2 Speaker Authentication

We begin by briefly outlining the actual algorithms used for speaker authentication, as a prelude to describing our privacy-preserving frameworks. Recall that a user enrolls with a speaker authentication system by providing it enrollment voice samples. The system builds a "model" for the user from these samples. Later, to authenticate himself the user announces his identity to the system and provides it with a voice sample. The system compares the sample to the model it has for the user to authenticate him. In the outline we will discuss both how the model is learned, and how it is used for authentication.

Before we proceed, it must be noted that the system does not actually work on the digitized speech signal directly. Instead, speech signals are parameterized into a sequence of *feature* vectors, typically *mel-frequency cepstral coefficients* (MFCC) vectors [6] augmented by their temporal derivatives and double derivatives, which characterize the variation in the frequency content of the signal as a function of time. Thus, when we refer to the audio signal, we are in fact only referring to the sequence of feature vectors derived from it.

Note that our discussion assumes *text-independent* speaker authentication, where the user is not required to say a specific passphrase. However the techniques are easily extended to text-dependent authentication also.

2.1 Speaker Authentication Using Likelihood Ratios from GMMs

The most common and successful technique for text-independent speaker authentication treats the problem as one of hypothesis testing [18], performed using a likelihood ratio test. To authenticate a recording $\mathbf{X}$ given by a speaker, the system computes the probability of $\mathbf{X}$ using a model λ_s for the speaker and compares it to the probability computed from a *Universal Background Model* (UBM) λ_U representing generic speech. Authentication uses the following rule:

$$\frac{P(\mathbf{X}|\lambda_s)}{P(\mathbf{X}|\lambda_U)} \begin{cases} \geq \theta & \text{accept speaker,} \\ < \theta & \text{reject speaker.} \end{cases} \tag{1}$$

Each recording $\mathbf{X}$ actually comprises a sequence of feature vectors, *i.e.* $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T]$. The individual vectors $\mathbf{x}_t$ are assumed to be independent and identically distributed. The probability distribution of the individual vectors is assumed to be a Gaussian mixture [19] with the form

$$P(\mathbf{x}_t|\lambda_C) = \sum_k w_k^C \mathcal{N}(\mathbf{x}_t; \mu_k^C, \Sigma_k^C) \tag{2}$$

where C is either s or U, w_k^C represents the mixture weight of the k^{th} Gaussian in the Gaussian mixture density for C, $\mathcal{N}()$ represents a Gaussian, and μ_k^C and Σ_k^C represent the mean and variance of the k^{th} Gaussian for C.

To effectively represent the characteristics of a generic speaker, the parameters of the UBM λ_U are usually learned from a collection of speech recordings from a large number of speakers, typically using the expectation-maximization (EM) algorithm. The parameters λ_s of the model for a speaker s are obtained by adapting the UBM to the enrollmentp data from the speaker as follows.

Model Adaptation

The UBM parameters are adapted to individual speakers using *maximum a posteriori* (MAP) estimation. It has been empirically established that speaker models obtained by MAP estimation signicantly outperform the models trained directly on the enrollment data [20][18].

The MAP estimation procedure comprises estimation of a sample estimate of the parameters of the distribution for the speaker, followed by interpolation

with the UBM. Given set of feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_T$ obtained from enrollment samples, we first compute the *a posteriori* probabilities of the individual Gaussians in the UBM. For the i^{th} mixture component of the UBM, the *a posteriori* probability conditioned on $\mathbf{x}_t$ is given by

$$P(i|\mathbf{x}_t) = \frac{w_i^U \mathcal{N}(\mathbf{x}_t; \mu_i^U, \Sigma_i^U)}{\sum_k w_k^U \mathcal{N}(\mathbf{x}_t; \mu_k^U, \Sigma_k^U)}. \tag{3}$$

Similarly to the M-step of EM, the *a posterior* probabilities are then used to compute the updated sample estimates of mixture weights, means, and variances.

$$\begin{aligned} w_i' &= \frac{1}{T}\sum_t P(i|\mathbf{x}_t), & \mu_i' &= \frac{\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t}{\sum_t P(i|\mathbf{x}_t)}, \\ \Sigma_i' &= \frac{\sum_t P(i|\mathbf{x}_t)\mathrm{diag}(\mathbf{x}_t\mathbf{x}_t^T)}{\sum_t P(i|\mathbf{x}_t)}. \end{aligned} \tag{4}$$

Finally, the parameters of the adapted model $\lambda_s = \{w_i^s, \mu_i^s, \Sigma_i^s\}$ are given by the convex combination of these new parameters and the UBM parameters:

$$\begin{aligned} w_i^s &= \alpha_i w_i' + (1-\alpha_i) w_i^U, & \mu_i^s &= \alpha_i \mu_i' + (1-\alpha_i)\mu_i^U, \\ \Sigma_i^s &= \alpha_i \Sigma_i' + (1-\alpha_i)\left[\Sigma_i^U + \mu_i^U {\mu_i^U}^T\right] - \mu_i^s {\mu_i^s}^T. \end{aligned} \tag{5}$$

The adaptation coefficients α_i for the mixture components control the contribution of the enrollment data to the estimate, relative to the UBM.

A number of variations on the above scheme have also been proposed, primarily aimed at accounting for the fact the amount of "registration" data may be limited, and that the recording conditions for the registration and authentication recordings may differ. These variants typically employ various flavors of factor analysis [8][9] to assign *a priori* probabilities to the parameters of λ_s. However, the fundamental structure remains what has been described above, and we shall remain with it for the presentation in this article.

2.2 Speaker Authentication Using GMM Supervectors

An alternate equally-successful approach to likelihood ratio tests obtains a separate Gaussian mixture for each of multiple enrollment recordings from the speaker through MAP adaptation of the UBM. Thus, instead of having a single Gaussian mixture repesenting the speaker, we now have multiple Gaussian mixtures, each representing one of the enrollment recordings. For each recording, the parameters of Gaussian mixture, namely the means, covariances (which are usually assumed to be diagonal matrices) and mixture weights, are concatenated into a "supervector" representing the recording. Similar supervectors are obtained for a number of recordings from each of a large number putative imposters. The collection of supervectors from the target speaker and the imposters are used as positive and negative exemplars to train a binary classifier.

For authentication, a supervector is similarly obtained from the test recording and classified by the binary classifier.

Note that once again *a priori* probability distributions may be assigned to the parameters of the Gaussian mixtures through factor analysis as in [21]. In this case the vectors of factors may be used instead of the actual supervectors to represent the recordings.

2.3 A SVM Based Authentication System

The most common binary classifiers used in this framework are kernel-based support vector machines [10]. A number of differnt kernels may be used; however we consider only RBF kernels. Here the kernel is given by $k(\mathbf{v}_i, \mathbf{v}_j) = e^{-\gamma \cdot d^2(\mathbf{v}_i, \mathbf{v}_j)}$, where $d(\mathbf{v}_i, \mathbf{v}_j)$ refers to the Euclidean distance between vectors $\mathbf{v}_i$ and $\mathbf{v}_j$, and γ is a scaling factor. RBF kernels are observed to result in good classification accuracies for the speaker authetication task. Moreover, they are easily adapted to the hashing-based privacy-preserving speaker authentication schemes we will describe later.

2.4 *k*-Nearest-Neighbor Based Classification

As an alternative, we also consider a k-NN based classifier. k-NN classifiers are not normally employed in speaker authentication systems; however they are easily adapted to the hashing-based privacy-preserving authentication system we will describe later. Here, for each speaker S, the system retains a set of "exemplar" vectors $\mathcal{V}_S$ derived from registration recordings by the speaker. In addition, for each of a large number of *imposters* I from an imposter set $\mathbb{I}$, it retains a set of exemplar vectors $\mathcal{V}_I$. The procedure for classification is given by Algorithm 1.

Note that although we have referred to the algorithm as k-NN, we do not restrict ourselves to the k closest neighbors, but consider *all* exemplars. Also, the weights assigned to exemplars are based on mean reciprocal rank (MRR), rather than the distance to $\mathbf{y}$.

3 Privacy-Preserving Authentication as Secure Function Evaluation

Secure Function Evaluation is a formalism under which two parties with private inputs can jointly compute a function of both their inputs without exposing them to one another [13]. Under the SFE framework we treat speaker authentication as a problem where the system and user must compute the likelihood-ratio test described in Section 2.1 from their private inputs, namely the users audio and the models retained by the system, without exposing them to one another.

The general principle behind the SFE framework employed is illustrated in Figures 1 and 2. The key is to decompose the overall operation into a sequence

Algorithm 1. MRR algorithm for authentication

Training:

• Store the exemplar set of vectors $\mathcal{V}_S$ from the training instances for the target speaker S.

• For each imposter I in imposter set $\mathbb{I}$, store a set of exemplar vectors $\mathcal{V}_I$. Let $\mathcal{V} = \mathcal{V}_S \cup_{I \in \mathbb{I}} \mathcal{V}_I$ be the set of all exemplars for all speakers, including target and imposters.

Testing:

• Given "test" vector $\mathbf{y}$ from claimed speaker S

• Rank order all exemplars $\mathbf{v} \in \mathcal{V}$ as follows:

$$rank(\mathbf{v}) = |\mathcal{V}| - \sum_{\mathbf{u} \in \mathcal{V} \setminus \mathbf{v}} I(d(\mathbf{u}, \mathbf{y}) > d(\mathbf{v}, \mathbf{y}))$$

where $I()$ is an indicator function.

for each speaker $C \in S \bigcup \mathbb{I}$ **do**

$Score(C) = \sum_{\mathbf{v} \in \mathcal{V}_C} \frac{1}{rank(\mathbf{v})}$

end for

If $Score(S) > \max_{C \in \mathbb{I}} Score(C)$ accept S **else** reject S

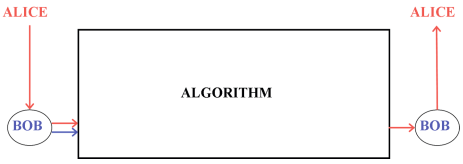

Fig. 1. In a regular computation, user "Alice" sends her data a to system "Bob", who combines it with his data b to compute the function. The output c is returned to Alice.

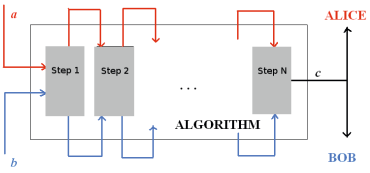

Fig. 2. In an SFE, the computation of the function is decomposed into several intermediate steps. Each step requires inputs from both parties. The output of each step is distributed to both parties as random additive shares.

of intermediate steps. Each step requires inputs from both the user and the system, and the output of each step is also distributed across both as random additive shares that individually do not reveal anything about the outcome of the step. The final outcome is received by the system. The manner in which the computation is broken down and performed in this manner is specified through detailed protocols.

We now briefly present a preliminary discussion of some of the tools we employ that enable us to decompose the computation in this manner, and subsequently outline the protocols we employ for various aspects of speaker verification, followed by the performance, security and computation caveats to consider.

3.1 Preliminaries

Homomorphic Encryption. The entire framework is predicated on the use of *homomorphic encryption* techniques that allow for arithmetic and logical operations to be performed directly on encrypted data to obtain encrypted results. If $\cdot$ and $+$ are two operators and x and y are two plaintexts, a homomorphic encryption function E satisfies $E[x] \cdot E[y] = E[x + y]$. Thus, a party who only possesses the encrypted values $E[x]$ and $E[y]$ can obtain $E[x+y]$, the encryption of the outcome of the operation $x + y$, without actually ever knowing the true values of x and y.

In this work, we use the Paillier cryptosystem [22] which is a *public-key* cryptosystem that satisfies additive homomorphism. Given two numbers x and y, the Paillier cryptosystem satisfies $E[x] \cdot E[y] = E[x + y]$, and as a consequence, $E[x]^y = E[x \cdot y]$. We refer the reader to [22] for more details on the crypto system.

Using the homomorphic encryption scheme, we can now also define a set of "primitive" protocols that enable the user and system to perform some operations. We will present these primitives as they are required by our operations. Given space constraints, we will not explain exactly how the primitives are implemented, and refer the reader to [2] [15] for details.

Gaussian Computation as a Dot Product. We make extensive use of the fact that the computation of the log of a Gaussian can be expressed as a dot product. We use the following construction from [15]. The logarithm of a multivariate Gaussian density $\mathcal{N}(x; \mu, \Sigma)$ computed on a d-dimensional vector $\mathbf{x}$ can be represented as the quadratic product $\log \mathcal{N}(\mathbf{x}; \mu, \Sigma) = \tilde{x}^T \tilde{W} \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}}$ is an extended vector obtained by concatenating 1 to $\mathbf{x}$, and $\tilde{W}$ is a $(d+1) \times (d+1)$ matrix given by.

$$\tilde{W} = \begin{bmatrix} -\frac{1}{2}\Sigma^{-1} & \Sigma^{-1}\mu \\ \hline 0 & w^* \end{bmatrix}, \tag{6}$$

where $w^* = -\frac{1}{2}\mu^T \Sigma^{-1} \mu - \frac{1}{2} \log |\Sigma|$.

We can reduce this computation further to a single inner product $\bar{\mathbf{x}}^T W$, where $\bar{\mathbf{x}}$ is a *quadratically extended* feature vector derived from $\tilde{\mathbf{x}}$ which consists of all pairwise product terms $\tilde{x}_i \tilde{x}_j$ of all components $\tilde{x}_i \in \tilde{\mathbf{x}}$, and W is obtained by unrolling $\tilde{W}$ into a vector. In this representation $\log \mathcal{N}(x; \mu, \Sigma) = \bar{\mathbf{x}}^T W$.

We can now describe the SFE protocols for the following operations: user enrollment including model adaptation to the user's data, and authentication.

In all of the description below, we assume only the user has the private key to the homomorphic encryption scheme which can be used to decrypt encrypted data, but both the user and the system possess the public encryption key and can encrypt data. Thus the user can perform both encryption $E[\cdot]$ and decryption $E^{-1}[\cdot]$, while the system can only preform $E[\cdot]$.

We assume that the user computes MFCC feature vectors from the audio on his client device. We also assume that the user has computed both the feature vectors $\mathbf{x}_t$ and their quadratic extensions $\bar{\mathbf{x}}_t$. All communication of data from the user to the system are encrypted with his private key. All references to Gaussians actually refer to the unrolled vector W derived from the $\tilde{W}$ matrices for the Gaussians.

3.2 Private Enrollment Protocol

We assume that the system has access to the UBM, λ_U trained on a collection of speech data. We propose the following protocol to adapt the system's UBM to the user's enrollment samples. We refer to individual Gaussian components in a mixture as $P(\mathbf{x}|i)$ for conciseness.

Inputs:

1. User has feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_T$ and their quadratic extensions $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_T$ derived from enrollment samples.
2. System has the UBM $\lambda_U = W_i^U$ for $i = 1, \ldots, N$, and mixing weight α.

Output: System obtains encrypted adapted model $E[\lambda_s] = \{E[w_i^s],\ E[\mu_i^s],\ E[\Sigma_i^s]\}$.

Computing the posterior probabilities:
For each t:

1. The user sends the encrypted vector $E[\bar{x}_t]$ to the system.
2. The system computes the encrypted log probabilities for each Gaussian in the UBM: $E\left[\log w_i^U P(\mathbf{x}_t|i)\right] = \sum_j E[\bar{\mathbf{x}}_{t,j}]^{W_{i,j}^U} + E[\log w_i^U]$.
3. The *logsum* protocol [15] enables a party holding $E[\log x]$ and $E[\log y]$ to collaborate with another party who holds the private encryption key to obtain $E[\log(x+y)]$ without revealing x or y. The system participates with the user's client in the logsum protocol to obtain $E\left[\log \sum_i w_i^U P(x_t|i)\right]$.
4. The system computes encrypted log posteriors: $E[P(i|\mathbf{x}_t)] = E\left[\log w_i^U P(\mathbf{x}_t|i)\right]$ - $E\left[\log \sum_i w_i^U P(\mathbf{x}_t|i)\right]$.
5. The *private exponentiation protocol* [2] enables a party holding $E[\log x]$ to collaborate with the party who holds the encryption key to obtain $E[x]$ without revealing x. The system then executes the private exponentiation protocol with the user to obtain $E\left[P(i|\mathbf{x}_t)\right]$.

Learning w_i^s.

6. The system $E[w_i']$ as $E\left[\sum_t P(i|\mathbf{x}_t)\right] = \prod_t E\left[P(i|\mathbf{x}_t)\right]$.
7. The system then computes the encrypted updated mixture weights as

$$E[w_i^s] = E[w_i']^{\alpha/T} E[w_i^U]^{1-\alpha}.$$

Learning μ_i^s.

8. The system generates subtracts a random number r homomorphically from $E[P(i|\mathbf{x}_t)]$ to obtain $E[P(i|\mathbf{x}_t) - r]$ and sends it to the user.
9. The user decrypts this value and multiplies it by feature vectors to obtain $P(i|\mathbf{x}_t)x_t - r\mathbf{x}_t$. He encyrpts it and sends $E[P(i|\mathbf{x}_t)x_t - r\mathbf{x}_t]$ and $E[\mathbf{x}_t]$ to the system.
10. The system computes $E[P(i|\mathbf{x}_t)\mathbf{x}_t] = E[P(i|\mathbf{x}_t)\mathbf{x}_t] = E[\mathbf{x}_t]^r + E[P(i|\mathbf{x}_t)\mathbf{x}_t - r\mathbf{x}_t]$. It then computes $E[\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t] = \prod_t E[P(i|\mathbf{x}_t)\mathbf{x}_t]$.
11. The *private division protocol* [2] enables a party holding $E[x]$ and $E[y]$ to collaborate with the party holding the encryption key to obtain $E[x/y]$ without revealing x or y.The system engages the user in a private division protocol with $E[\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t]$ and $E[w']$ as inputs to obtain $E[\mu']$.
12. The system then computes the encrypted adapted mean as.

$$E[\hat{\mu}_i^s] = E[\mu_i']^{\alpha} E[\mu_i^U]^{1-\alpha}.$$

Learning $\hat{\Sigma}_i^s$.

This is similar to learning $\hat{\mu}_i^s$ and continues from Step 9 of the protocol.

13. The user multiplies $P(i|\mathbf{x}_t) - r$ by $\mathbf{x}_t\mathbf{x}_t^T$ to obtain[1]. $P(i|\mathbf{x}_t)x_t\mathbf{x}_t^T - r\mathbf{x}_t\mathbf{x}_t^T$. He sends $E[P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T - r\mathbf{x}_t\mathbf{x}_t^T]$ and $E[\mathbf{x}_t\mathbf{x}_t^T]$ to the system.
14. The system, which knows r, computes $E[P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T - r\mathbf{x}_t\mathbf{x}_t^T]E[\mathbf{x}_t\mathbf{x}_t^T]^r$ to obtain $E[P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T]$ for each t. The system then computes $\prod_t E[P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T]$ to obtain $E[\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T]$.
15. The system engages the user in the private division protocol with $E[w']$ and $E[\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T]$ as inputs to obtain $E[\Sigma_i']$.
16. The *private vector product protocol* [2] permits a user with encrypted vectors $E[\mathbf{x}]$ and $E[\mathbf{y}]$ to engage with a user who has the private decryption key to compute $E[\mathbf{x} \odot \mathbf{y}]$, where $\odot$ represents a Schur (component-wise) product. The system and user participate in the private vector product protocol with input $E[\mu_i^s]$ to obtain $E[\mu_i^s\mu_i^{sT}]$. The system then computes the encrypted updated covariance as

$$E[\Sigma_i^s] = E[\Sigma_i']^{\alpha} E[\Sigma_i^U + \mu_i^U\mu_i^{UT}]^{1-\alpha} - E[\mu_i^s\mu_i^{sT}].$$

The above protocol only represents the first iteration of the iterative MAP adaptation algorithm. In practice many iterations are required. The primary variation in subsequent iterations is that the user now only possesses the encrypted model learned in the previous iteration, and Step 1. of the above protocol cannot be performed. Instead this must be replaced by the protocol followed for authentication, which computes encrypted Gaussian log likelihoods from encrypted models and data.

At each iteraion (and finally) the encrypted model parameters obtained by the system cannot be directly used, since it only obtains the encrypted values of

[1] For efficiency purposes, only the diagonal terms of the product $x_t x_t^T$ can be included without having a significant impact on accuracy.

the means, mixture weights and variances; the product and logarithmic terms in W_i^s must be computed from it. This can be done through a rather simple protocol in which the system adds (or multiplies) random terms to the individual parameters of the model homomorphically to mask their true value and ships the masked values to the user. The user decrypts the data, performs the necessary operations, re-encrypts the values and returns them to the system. The system thereafter proceeds to eliminate the random maskers from the encrypted result, also homomorphically. The details of the protocol can be found in [2].

3.3 Private Verification Protocol

Once enrolment is complete, the system possesses encrypted models for the user, as well as the UBM. These are used to authenticate the user as follows:

Inputs:

1. User has extended feature vectors $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_T$ for a test sample.
2. System has $E[\lambda_s] = E[W_i^s]$, for all Gaussians $i = 1, \ldots, N$.

Output: System obtains the score $E[\log P(x_1, \ldots, x_T|\lambda_s)]$.

1. The system generates a random vector p and computes $E[W_i^s] \cdot E[-p]$ to obtain $E[\hat{W}_i^s - p]$. This is sent to the user.
2. The user decrypts it and obtains $W_i^s - p$. He computes the inner product $\bar{\mathbf{x}}_t^T(W_i^s - p)\ \forall t$, and sends $E[\bar{\mathbf{x}}_t^T(W_i^s - p)]$ and $E[\bar{\mathbf{x}}_t]\ \forall t$ to the system.
3. The system computes the inner product $E[\bar{\mathbf{x}}_t^T p] = \prod_k E[\mathbf{x}_{t,k}]^{p_k}\ \forall t$. It then computes $E[\bar{\mathbf{x}}_t^T W_i^s] = E[\bar{\mathbf{x}}_t^T(W_i^s - p)] \cdot E[\bar{x}_t^T p]\ \forall i$. The system and the user then participate in the *logsum* protocol to obtain $E[\log P(\mathbf{x}_t|\lambda_s)]$. The system finally computes $E[\log P(\mathbf{x}_1, \ldots, \mathbf{x}_T|\lambda_s)] = \prod_t E[\log P(\mathbf{x}_t|\lambda_s)]$.

The above protocol can be used to compute scores for both the speaker model and the UBM (although the system has access to the UBM in plain text, it is actually more efficient to simply encrypt it and compute the encrypted UBM score in this manner).

Once both the encrypted "user" score $E[\log P(\mathbf{x}_1, \ldots, \mathbf{x}_T|\lambda_s)]$ and the UBM score $E[\log P(\mathbf{x}_1, \ldots, \mathbf{x}_T|\lambda_U)]$ are available, the system can engage a secure comparison protocol [23] to determine which is greater.

3.4 The Efficacy of the SFE Approach

Since the SFE protocols described above essentially enable a direct privacy-preserving implementation of the likelihood-ratio test, actual authentication performance is not significantly different from that obtained with the original "unsecured" method. The only degradation comes from the fact that the encryption works over finite integer fields, and to utilize it all numbers must be converted to finite-precision integers. However, even with only 16-bit precision the performance of the algorithm is largely comparable to that obtained with floating-point valued features and models.

The larger issue arises from the computational overhead. The computations require repeated exchange of information between the user and the system, as well as repeated encryption and decryption. Table 1 shows execution times for the verification phase of the protocol using different encryption schemes – enrollment times would be several factors higher. All Gaussian mixtures employed in this experiment were mixtures of 32Gaussians. The table only evaluates the computation time, most of which is dominated by the time required for encryption and decryption. Communication overhead is negligible compared to these times. For comparison, the conventional "insecure" version of the algorithm took 14 seconds to run on the same computer, on the same utterance. Clearly the feasibility of this approach in practical terms is questionable within current computational infrastructures.

Table 1. Execution time of the verification protocol for a single 4.4second utterance

Steps	**Time (256-bit)**	**Time (1024-bit)**
Encrypting $\bar{x}_t$ $\forall t$	576.81 s	35747.82 s
Evaluating Adapted	407.21 s	7597.57 s
Evaluating UBM	same as adapted	same as adapted
Comparison	0.3 s	16.88 s
Total $=E[\bar{\mathbf{x}}_t]$ + adapted + UBM + compare	1391.53 s $\sim$ 23 min	50959.84 s $\sim$ 14 hr, 9 min

A second factor to be considered is security. The SMC protocols described above ensure that if they are correctly followed, no information is leaked whatsoever. The system learns nothing about the user's data, and the user in turn does not learn anything about the models stored by the system. The original objectives of privacy we set forth are satisfied.

However, the framework described above assumes "honest but curious" users and system. The user and the system are assumed to correctly perform the operations required of them, although they may inspect the output. Such an assumption is clearly naive. For instance, an imposter who has broken into the user's client device could modify the numbers sent out by the user's client device in only the last step of the protocol, *i.e.* the secure comparison protocol; this would result in incorrect results being obtained by the system, which result in the imposter obtaining a much-higher-than-random probability of breaking into the system. These too can be protected against, by including a stage of verification after each operation, *e.g.* using zero-knowledge proofs [24]; however such verification can increase the communication and computation overhead manifold over the current values.

On the other hand, the SMC framework not only has the benefit of being exact; it can also be generalized to incorporate other models, and perform additional tasks such as actual recognition of speech. Moreover, with appropriate verification, the procedures can be made arbitrarily secure.

4 Privacy Preserving Speaker Authentication through Hashing

The computational overhead from the SFE framework arises from two reasons. First, homomorphic encryption is expensive, as are homomorphic operations on data. Secondly, the processing is *interactive*, requiring repeated exchange of large amounts of data between the user and system.

By comparison, text-password based systems are blindingly fast, and yet highly secure. The reasons for this are not difficult to identify. Firstly, authentication requires an exact match between the password stored by the system and the string typed in by the user. This means that fast one-way hashing functions (*e.g.* DES) may be employed to encrypt the password. Since the stored password and the password transmitted by the user encrypt to the same cryptostring, comparison can be performed in the encrypted domain: a perfect match results in authentication; a mismatch provides no information about the original (unencrypted) text strings. A consequence of this is that the exchange need not be interactive: the user transmits a password, and the system determines whether the user must be authenticated in a single comparison operation; in the entire process the user's password remains secure.

On the other hand, the conventional password-based authentication scheme cannot be applied to speaker authentication for an obvious reason: speech recordings are not deterministic, and features derived from enrollment and test recordings are highly unlikely to be identical. Moreover, speech recordings tend to vary in length, making such direct exact comparisons impossible in most cases.

In the Hashing-based framework we attempt to map the authentication problem to a process similar to password matching. To do so, we employ the supervector-based mechanisms for authentication described in Section 2.2. Recall that in this formalism each recording, both from the enrollment data and the test data, is converted to a single "supervector" by adapting a UBM to the recording, and concatenating the parameters of the resultant distribution into a single vector. The length of the supervector only depends on the number of Gaussians in the UBM and is independent of the length of the recording, thereby resulting in a fixed-length parameterization of the audio. Nevertheless the supervectors are not determinsitic and will vary from recording to recording, and cannot be matched to one another.

In [3] a technique was proposed to convert these supervectors to password-like strings by hashing them using a *locality sensitive hashing* (LSH) scheme [25]. Once hashed in this manner, the supervectors of different recordings from a speaker could map onto the same hash with high probability. The problem here was that recordings from imposters too could map onto the same hashes. To reduce the incidence of such collisions, enrollment recordings were hashed multiple times using several hashes, and the collection of hash values was used as a password set for the speaker. Test recordings were hashed in the same manner, and the set of test hashes were compared to the hashes from the enrollment set. If a sufficient number of exact matches were found between the two sets, the user was authenticated.

By converting the problem of authentication to a password matching problem, privacy issues were effectively eliminated. Computation of supervectors and hashes was performed on the user's client device. The server only obtained hashes that had been encrypted by a one-way function such as SHA-256, both from the enrollment data and during authentication. Speaker authentication in this manner was as secure as text-based password matching. The problem however, was one of accuracy – by requiring only exact matches between hashes of test and enrollment data, the accuracy of the system was compromised.

The hashing-based mechanism we employ approaches the problem of conversion of audio into hashes differently – we employ a hashing mechanism based on the principle of *secure binary embeddings* (SBE) [5][4]. SBEs have an interesting property – they convert the audio into hashes that can be compared to one another in a manner that permits a small degree of mismatch between the hashes. At the same time, they provide information theoretic guarantees that the distance between hashes of vectors provides no information about the true relationship between the vectors, if the vectors are sufficiently far apart. Furthermore, they cannot be inverted to recover supervectors from their hashes, if the key to the hashing function is kept private.

In the following sections we will describe the SBE and its properties, and how we apply it to the problem of speaker authentication. Since the actual mechanism employed to authenticate users is no longer identical to existing techniques, which are based on the ability to estimate the actual distances between vectors, we also include a section with experiments demonstrating the effectiveness of the approach.

4.1 Secure Binary Embeddings

Secure binary embeddings (SBE) [5] convert vectors to bit sequences through random projections followed by band quantization. The resulting bit strings, which we will refer to as "hashes", have the following property: if the Euclidean distance between two vectors is below a threshold, the Hamming distance between their hashes is proportional to the Euclidean distance; above the threshold the Hamming distance provides no information regarding the true distance between the two vectors. Given an L-dimensional vector $\mathbf{x} \in \mathbb{R}^L$, the M-bit SBE of $\mathbf{x}$ is defined as

$$\mathbf{q}(\mathbf{x}) = Q\left(\Delta^{-1}(\mathbf{A}\mathbf{x} + \mathbf{w})\right) \tag{7}$$

where Δ is a precision parameter, $\mathbf{A} \in \mathbb{R}^{M \times L}$ is a random matrix whose elements are drawn from a normal distribution with zero mean and unit variance, and $\mathbf{w} \in \mathbb{R}^M$ is a vector composed of random numbers drawn uniformly from $[0, \Delta]$. $Q()$ is a quantization function given by $Q(\mathbf{y}) = \lfloor \mathbf{y} \% 2 \rfloor$, where the floor and modulus operations are performed component wise.

The binary hash $\mathbf{q}(\mathbf{x})$ generated by Equation 7 has the following properties [4]: given two vectors $\mathbf{x}$ and $\mathbf{x}'$ that have a Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$, the probability that the i^{th} bits, $q_i(\mathbf{x})$ and $q_i(\mathbf{x}')$ respectively of the hashes $\mathbf{q}(\mathbf{x})$ and $\mathbf{q}(\mathbf{x}')$ are identical depends only on the distance $d(\mathbf{x}, \mathbf{x}')$ between the vectors and

not on $\mathbf{x}$ and $\mathbf{x}'$ themselves. The following relationship results as a consequence [4]: with probability at most e^{-2t^2M} the normalized (per-bit) Hamming distance $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ between $\mathbf{q}(\mathbf{x})$ and $\mathbf{q}(\mathbf{x}')$ is bounded by:

$$\frac{1}{2} - \frac{1}{2}e^{-\left(\frac{\pi\sigma d(\mathbf{q}(\mathbf{x}))}{\sqrt{2}\Delta}\right)^2} - t \leq d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}')) \leq \frac{1}{2} - \frac{4}{\pi^2}e^{-\left(\frac{\pi\sigma d(\mathbf{q}(\mathbf{x}))}{\sqrt{2}\Delta}\right)^2} + t$$

where t is a control factor. The import of the above bound is that the Hamming distance $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ is correlated to the Euclidean distance $d(\mathbf{x}, \mathbf{x}')$ if $d(\mathbf{x}, \mathbf{x}')$ is less than a threshold (which depends on Δ). Specifically, for small $d(\mathbf{x}, \mathbf{x}')$, $E[d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))]$, the expected Hamming distance, can be shown to be bounded from above by $\sqrt{2\pi^{-1}}\sigma\Delta^{-1}d(\mathbf{x}, \mathbf{x}')$, which is linear in $d(\mathbf{x}, \mathbf{x}')$. However, if the distance between $\mathbf{x}$ and $\mathbf{x}'$ is greater than this threshold, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ is bounded by $0.5 - 4\pi^{-2}exp\left(-0.5\pi^2\sigma^2\Delta^{-2}d(\mathbf{x}, \mathbf{x}')^2\right)$, which rapidly converges to 0.5 and effectively gives us no information whatsoever about the true distance between $\mathbf{x}$ and $\mathbf{x}'$.

Figure 3 illustrates this relation through a simulation. For the simulation we randomly generated samples in a high-dimensional space ($L = 1024$) and computed their hashes. Figure 3 shows the scatter of the normalized per-dimension Euclidean distance between pairs of these vectors, and the normalized per-bit Hamming distance between their hashes. The number of bits in the hashes is also shown in the figures. We note that in all cases, once the normalized Eu-

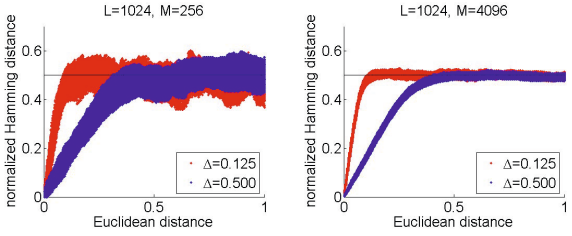

Fig. 3. Embedding behaviour for different values of Δ and different amounts of measurements M

clidean distance between two vectors exceeds Δ, the Hamming distance between their hashes no longer provides any information about the Euclidean distance. We also see that changing the value of the precision parameter Δ allows us to adjust the distance threshold until which the Hamming distance is informative. Increasing the number of bits M in the hashes also leads to a reduction of the variance of the Hamming distance.

A converse property of the embeddings is that for all $\mathbf{x}'$ except those that lie within a small radius of any $\mathbf{x}$, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ provides little information about how close $\mathbf{x}'$ is to $\mathbf{x}$ as evident from Figure 3. In fact, it can be shown that beyond this radius the embedding provides information theoretic security if the embedding parameters $\mathbf{A}$ and $\mathbf{w}$ remain unknown to a potential adversary

who may try to deduce the data from their hashes. Any algorithm attempting to recover a signal $\mathbf{x}$ from its embedding $\mathbf{q}(\mathbf{x})$ or to infer anything about the relationship between two signals sufficiently far apart using only their embeddings will fail to do so. Furthermore, even in the case where $\mathbf{A}$ and $\mathbf{w}$ are known, it seems computationally intractable to derive $\mathbf{x}$ from $\mathbf{q}(\mathbf{x})$ unless one can guess a starting point very close to $\mathbf{x}$. In effect, it is infeasible to invert the SBE without strong *a priori* assumptions about $\mathbf{x}$.

4.2 Speaker Verification with Secure Embeddings

SBE hashes thus enable us to compare data that are close together, without actually being able to observe the data. This immediately enables us to apply them to the problem of privacy-preserving speaker authentication, through a rather direct modification of the algorithms described in Section 2.2. The manner in which we modify the algorithms is predicated on the fact that $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ is more-or-less linearly related to $d(\mathbf{x}, \mathbf{x}')$ for nearby vectors. We explain the actual modifications below.

In all cases we assume that the speaker possesses the embedding parameters $\mathbf{A}$ and $\mathbf{w}$. The supervectors for the speaker's recordings are computed on the user's client device. The SBE hashes of the supervectors are also computed on the user's client device using $\mathbf{A}$ and $\mathbf{w}$. For both the enrollment data and for subsequent authentication data only the hashes are transmitted to the system over a secure channel.

During the enrolment phase the system also requires hashes derived from a large collection of recordings from imposters.

We and defer the discussion of the provenance of $\mathbf{A}$ and $\mathbf{w}$ and the consequent implications on user privacy, as well as the source of the imposter recordings, to Section 4.6

SVM-Based Classifier: The SVM based classifier used for speaker authentication systems utilizes an RBF kernel defined by $k(\mathbf{x}, \mathbf{x}') = e^{-\gamma \cdot d^2(\mathbf{x}, \mathbf{x}')}$. We modify this by replacing the Euclidean distance $d(\mathbf{x}, \mathbf{x}')$ between vectors by the hamming distance between their hashes $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$. The resulting SVM utilizes the modified pseudo-kernel: $k_q(\mathbf{x}, \mathbf{x}') = e^{-\gamma \cdot d_H^2(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))}$. Note that for a given $\mathbf{A}$ and $\mathbf{w}$, the modified kernel $k_q(\mathbf{x}, \mathbf{x}')$ closely approximates the conventional RBF for small $d(\mathbf{x}, \mathbf{x}')$, but varies significantly from it at larger $d(\mathbf{x}, \mathbf{x}')$. Although it does not satisfy Mercer's conditions and cannot be considered a true kernel, in practice it is effective as we shall see in the experiments described in Section 4.3.

During enrollment, the SVM classifier is now trained using hashes of (supervectors derived from) enrollment recordings provided by the user, in conjunction with the hashes of a collection of imposter recordings. During operation, hashes of supervectors derived from test recordings are classified by the SVM.

***k*-NN Classifier:** The k-NN classifier of Section 2.4 can also be similarly modified. Here we replace the Euclidean distance $d(\mathbf{x}, \mathbf{x}')$ used to rank order the neighbors of a vector by the Hamming distance between their SBE hashes, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$. The motivation behind the design of the algorithm in Section 2.4, which was designed with the hashing-based classifiers in mind, can now be explained. Since the expected Hamming distance $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ between any the hashes of any two vectors $\mathbf{x}$ and $\mathbf{x}'$ with Euclidean distance $d(\mathbf{x}, \mathbf{x}') > M\Delta$ is $0.5M$, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ cannot reliably be used to choose the k closest neighbors to any $\mathbf{x}$ correctly except for small k. Hence we simply consider all exemplars to compute the score for any vector. Secondly, since the actual Hamming distance between the hashes of two vectors is a random variable and does not fall significantly above $0.5M$, particularly for larger M, weights are assigned to exemplars based on reciprocal rank, rather than the distance itself.

The actual implementation of the authentication is straight-forward. In the enrollment phase the system obtains the hashes of a set of enrolment recordings from the user, as well as those for a collection of recordings from several imposters. The k-NN algorithm is used thereafter to classify hashes derived from test recordings of the user.

4.3 Experiments in Speaker Authentication

Unlike the SFE-based framework which merely embedded a conventional algorithm within a privacy-preserving framework, the SBE-based mechanisms described above modify the actual classifier by changing the distance functions they use. Therefore, before we proceed, we first demonstrate the accuracy of the SBE-based speaker authentication.

Experiments were run on the YOHO Speaker Verification corpus [11]. It comprises of a collection of short utterances produced by 138 different speakers, 108 male and 30 female. Each utterance contains a sequence of three two-digit numbers (e.g. "26-81-57"). The recordings were sampled at 8kHz and digitized as 16-bit words. The corpus is divided into an enrollment set and a verification set. The enrollment set contains 96 utterances from each speaker, totaling 14.54 hours of audio. The verification set, which is the "test" set, contains 40 utterances from each speaker, totaling 6.24 hours of audio. We did not explicitly record imposters; instead for each of the 138 speakers in the corpus, the remaining 137 were used as imposters. All speech signals were parameterized into a sequence of MFCC feature vectors, each computed from an analysis frame of 25ms, at the rate of 100 vectors per second. The 13-dimensional MFCC vectors were augmented with the temporal differences and double-differences to result in a total of 39 features per frame.

A UBM (which, we recall is a Gaussian mixture) was trained from the data for all the speakers. The UBM was adapted to each recording from each speaker to obtain a single Gaussian supervector for each recording. The length of the supervectors depends on the number of Gaussians in the UBM: a UBM with

N Gaussians results in supervectors with $L = 39N$ dimensions. We evaluated UBMs of different sizes, with the number of Gaussian components ranging from 4 to 128 Gaussians, to find the optimal settings.

4.4 Experiments Using Supervectors

In the first experiment we directly employed the supervectors in the classifiers described in Section 2.2, to obtain reference speaker authentication results against which we could compare the performance of SBE-based authentication. We report results obtained from supervectors obtained with GMMs of different sizes. Results are reported using both the SVM and k-NN classifiers. In a binary classification task such as speaker authentication, the two key performance metrics are "precision", which records the percentage of all recordings that were "accepted" by the system that actually belonged to the target speaker, and "recall", which records the fraction of all recordings by a speaker that were correctly accepted. Ideally, both precision and recall must be close to 100%. We report performances in terms of "F-measure", which is the harmonic mean of precision and recall and encapsulates both. If either of them is low, the F-measure will be low. Note that for the k-NN classifier which actually obtains separate scores for each speaker in the corpus, the F-measure is also identical to the accuracy of a multi-class classifier that attempts to identify the speaker in each recording.

The obtained results averaged over all the speakers and are presented in Tables 2 and 3 respectively. In both cases the classifier achieves a close to optimal

Table 2. Speaker authentication F-measure (%age) using SVMs

#Gaussians	4	8	16	32	64	128
F-measure	76.8	89.2	92.8	94.0	94.1	94.4

Table 3. Speaker authentication F-measure (%age) using k-NN

#Gaussians	4	8	16	32	64	128
F-measure	71.8	86.7	92.4	94.8	91.5	85.4

performance with supervectors derived from mixtures of 32 Gaussians – for the k-NN classifier the performance falls off with increasing Gaussians; for the SVMs, although further increase in suprevector size results in improved performance, the improvements are marginal. Consequently, for experiments with the SBE we will consider only SBE hashes of supervectors obtained from mixtures of 32 Gaussians.

4.5 Experiments Using SBE Hashes

The secure binary embeddings have two parameters that can be varied: the quantization step size Δ and the number of bits M. The value of M by itself is

not informative, since increasing L (dimensionality of the supervector) requires increasing values of M to retain the same resolution. Hence we report our results as a function of *bits per coefficient* (*bpc*), computed as M/L. The results obtained for the SVM and k-NN classifiers are presented in Tables 5 and 4 respectively.

Table 4. Speaker authentication F-measure (%age) using k-NN + SBE

Δ	13.5	14.0	14.5	15.0	15.5
bpc=4	72.4	78.9	82.8	85.6	87.3
bpc=8	85.0	88.6	89.9	90.8	91.6
bpc=16	90.2	91.8	92.7	93.1	93.2

Table 5. Speaker authentication F-measure (%age) using SVMs + SBE

Δ	13.5	14.0	14.5	15.0	15.5
bpc=4	85.4	87.9	89.0	90.9	91.6
bpc=8	90.7	92.1	93.0	93.7	93.9
bpc=16	94.0	94.0	94.6	94.6	94.9

From Figure 3 and the discussion in Section 4.1 we can infer that smaller Δ values result in smaller neighborhoods over which the distances between adjacent vectors can be deduced from their hashes. Thus, smaller Δ values result in hashes $\mathbf{q}(\mathbf{x})$ that better obscure the value of the vectors $\mathbf{x}$. By the same token, however, they are also likely to result in poorer classification accuracy since $\mathbf{x}$ cannot reliably be compared to a model.

Tables 5 and 4 report performance as a function of Δ and *bpc*. As expected, increasing Δ generally improves classification performance, but then so does increasing *bpc*. For small *bpc* small Δ values result in unacceptably poor classification. However increasing *bpc* improves the classification performance in all cases, and at *bpc*=16, the performance stabilizes quickly and is thereafter largely independent of Δ. At these settings we notice little, if any degradation when we work from SBE hashes instead of the original supervectors.

4.6 Privacy and Other Practical Issues

Let us now analyze the actual privacy afforded by the SBE-based authentication.

When the User Possesses A and w as Private Keys: In the most secure case, the *user* generates $(\mathbf{A}, \mathbf{w})$ and retains it as his private key. Since the system only obtains the hashes and does not know $\mathbf{A}$ or $\mathbf{w}$, it provably has no means of recovering vectors from their hashes. Moreover, it also has no way of *computing* hashes given a vector. As a result, the process is totally secure from abuse by the system. Moreover, this also protects the user from imposters: the user's private key is tied to her client device(s), and if these are secured, *e.g.* by a password, the system cannot be broken into by an imposter who poses as the user.

The problem here is that since the system has no means of computing hashes, the *user* is in charge of downloading imposter data and generating hashes from them for the system to train its classifiers from, an onerous responsibility. One alternative is to use a "single-class" classifier, which only requires positive instances from the user; however the performance of such classifiers tends to be poor [3]. A more practical solution is for the user to generate his set of negative samples from the positive samples. This can be achieved by using the method proposed in [12], in which the user iteratively estimates SVM classifiers and generates negative instances that lie on the wrong side of the decision boundary. The authors of this technique state that after a few iterations the classification results are comparable to the ones obtained using sets of genuine positive and negative samples; however whether this is true for supervectors of speech recordings remains to be seen.

The *System* Provides A and w: Recall that the SBE is essentially not invertible without strong *a priori* assumptions about the data. Under the assumption of non-invertibility, we can propose the following: the system generates $(\mathbf{A}, \mathbf{w})$ once, and transmits it to the speaker. The system is now in charge of generating imposter hashes. Users only send the SBE hashes of their enrollment data to the system. Even if the hashes are not invertible, this process is not secure: since the system can generate hashes on its own, it does not prevent the system from searching public sources such as YouTube for instances of the speaker's voice. This may not be an unacceptable compromise in many situations; however, it is still unknown whether the system can recover a vector from its hash given *a priori* information that it can obtain.

Finally, we note that one of the key advantages of the SFE approach was its generalizability – the same framework could also be employed for a variety of other tasks such as speaker *identification*, speech recognition, etc. Since the hashing-based approach can only be guaranteed to be secure if the *user* retains the hashing parameters, it does not lend itself to such generalization.

5 Discussion and Conclusions

With increasing use of speech-based services, the issue of the privacy of the users and their speech data is only just beginning to be considered. These issues go beyond the mere problem of biometrics. Even for speech-based biometrics, the proposed frameworks are not complete – while they enable some operations, specifically speaker authentication, to be performed securely, they cannot provide the same security to closely related tasks such as speaker identification. In the larger scenario, users who use speech recognition and speech mining services, social media sites etc. remain exposed.

The presented work can thus only be considered a first step towards establishing a truly secure framework for speech processing applications in general. We continue to investigate these issues and hope researchers around the world recognize the importance of the problem and take it up as well.

Acknowledgements. The authors would like to thank Petros Boufounos for many helpful suggestions. José Portêlo and Isabel Trancoso were supported by Portuguese funds through FCT – Fundação para a Ciência e a Tecnologia, under PhD grant SFRH/BD/71349/2010 and project PTDC/EIA-CCO/122542/2010. Bhiksha Raj was partially supported by NSF Grant 1017256.

References

1. Adler, A.: Biometric System Security. In: Jain, A.K., Flynn, P., Ross, A. (eds.) Handbook of Biometrics. Springer (2007)
2. Pathak, M., Raj, B.: Privacy Preserving Speaker Verification using adapted GMMs. In: Proc. Interspeech (2011)
3. Pathak, M., Raj, B.: Privacy-Preserving Speaker Verification as Password Matching. In: Proc. ICASSP (2012)
4. Boufounos, P., Rane, S.: Secure Binary Embeddings for Privacy Preserving Nearest Neighbors. In: Proc. Workshop on Information Forensics and Security, WIFS (2011)
5. Boufounos, P.: Universal Rate-Efficient Scalar Quantization. IEEE Trans. on Information Theory 58(3), 1861–1872 (2012)
6. Davis, S.B., Mermelstein, P.: Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. IEEE Transactions on Acoustics, Speech, and Signal Processing 28(4), 357–366 (1980)
7. Reynolds, D.A., Quatieri, T.F., Dunn, R.B.: Speaker Verification using Adapted Gaussian Mixture Models. Digital Signal Processing 10(1-3), 19–41 (2000)
8. Kenny, P., Boulianne, G., Ouellet, P., Dumouchel, P.: Joint Factor Analysis Versus Eigenchannels in Speaker Recognition. IEEE Trans. Audio, Speech and Language Processing 15(4), 1435–1447 (2007)
9. Sennoussaoui, M., Kenny, P., Brummer, N., de Villiers, E., Dumouchel, P.: Mixture of PLDA Models in I-Vector Space for Gender-Independent Speaker Recognition. In: Proc. Interspeech 2011, Florence, Italy (August 2011)
10. Campbell, W.M., Campbell, J.R., Reynolds, D.A., Singer, E., Torres-Carrasquillo, P.A.: Support Vector Machines for Speaker and Language Recognition. Computer Speech and Language 20, 210–229 (2006)
11. Campbell, J.P.: Testing with the YOHO CD-ROM voice verification corpus. In: Proc. ICASSP (1995)
12. Yu, H., Han, J., Chang, K.C.C.: PEBL: Positive Example based Learning for Web Page Classification using SVM. In: Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 239–248. ACM (2002)
13. Lindell, Y., Pinkas, B.: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
14. Kinnunena, T., Li, H.: An overview of text-independent speaker recognition: From features to supervectors. Speech Communication 52(1), 12–40 (2010)
15. Smaragdis, P., Shashanka, M.: A Framework for Secure Speech Recognition. IEEE Transactions on Audio, Speech, and Language Processing 15(4), 1404–1413
16. Pathak, M., Rane, S., Sun, W., Raj, B.: Privacy Preserving Probabilistic Inference with Hidden Markov Models. In: Proc. ICASSP, Prague, Czech Republic (May 2011)
17. Prabhakar, S., Pankanti, S., Jain, A.K.: Biometric recognition: security and privacy concerns. IEEE Security & Privacy 1(2), 33–42

18. Reynolds, D.A., Quatieri, T.F., Dunn, R.B.: Speaker Verification Using Adapted Gaussian Mixture Models. Digital Signal Processing 10(1-3), 19–41 (2000)
19. Bimbot, F., Bonastre, J.-F., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., Merlin, T., Ortega-García, J., Petrovska-Delacrétaz, D., Reynolds, D.A.: A Tutorial on Text-Independent Speaker Verification. EURASIP Journal on Advances in Signal Processing 4, 430–451 (2004)
20. Reynolds, D.A.: Comparison of Background Normalization Methods for Text-Independent Speaker Verification. In: Proceedings of the European Conference on Speech Communication and Technology (September 1997)
21. Shou-Chun, Rose, R., Kenny, P.: Adaptive score normalization for progressive model adaptation in text independent speaker verification. In: Proc. ICASSP, Las Vegas, Nevada, USA (April 2008)
22. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
23. Kershbaum, F., Biswas, D., de Hoogh, S.: Performance Comparison of Secure Comparison Protocols. In: Proceedings of the 2009 20th International Workshop on Database and Expert Systems Application, pp. 133–136 (2009)
24. Quisquater, J.-J., Guillou, L.C., Berson, T.A.: How to Explain Zero-Knowledge Protocols to Your Children. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 628–631. Springer, Heidelberg (1990)
25. Gionis, A., Indyk, P., Motwani, R.: Similarity Search in High Dimensions via Hashing. In: Proceedings of the 25th Very Large Database (VLDB) Conference (1999)