Privacy-Preserving Speaker Verification and Identification Using Gaussian Mixture Models

Manas A. Pathak and Bhiksha Raj, *Member, IEEE*

***Abstract*—Speech being a unique characteristic of an individual is widely used in speaker verification and speaker identification tasks in applications such as authentication and surveillance respectively. In this article, we present frameworks for privacy-preserving speaker verification and speaker identification systems, where the system is able to perform the necessary operations without being able to observe the speech input provided by the user. In a speech-based authentication setting, this privacy constraint protect against an adversary who can break into the system and use the speech models to impersonate legitimate users. In surveillance applications, we require the system to first identify if the speech recording belongs to a suspect while preserving the privacy constraints. This prevents the system from listening in on conversations of innocent individuals. In this paper we formalize the privacy criteria for the speaker verification and speaker identification problems and construct Gaussian mixture model-based protocols. We also report experiments with a prototype implementation of the protocols on a standardized dataset for execution time and accuracy.**

***Index Terms*—Secure multiparty computation, speaker identification, speaker verification.**

I. Introduction

As speech is a unique characteristic of an individual, a person's voice and manner of speaking are his/her biometric signatures. This property allow us to classify speech samples by their speakers using probabilistic representation such as Gaussian mixture models (GMMs), and forms the underlying principle of speaker verification and speaker identification tasks. In speaker verification, we authenticate a person based on speech input. In speaker identification the objective is to identify which, if any, of a given set of speakers produced a given speech sample.

In order to perform authentication, a speaker verification system needs to store speech patterns of all enrolled users. This leads to the system being vulnerable to attacks that other types of authentication systems, such as password-based systems, are subjected to. For instance, the verification system may itself be compromised for *phishing*, i.e., made to act as a front to capture users' speech patterns when they enroll with the system. These voice patterns could then be used to impersonate users in other voice authentication services. Alternatively, a malicious agent may break into a system and gain access to stored voice patterns and later apply them to generate fake voice data to impersonate enrolled users. Each of these attacks lead to the disclosure of the speech data provided by the users and therefore form a breach of privacy.

However, no form of speech-based authentication is perfectly secure. An obvious way of circumventing the authentication process is by imitation. Imposters may attempt to imitate a subject's voice, or produce speech similar to the user's voice using methods such as playing out recordings of the user's voice, or morphing their own voice into the user's voice [2], [3]. Imitation of a person's speech, however, does not lead to the imposter or the system gaining any additional information and therefore does not result in a loss of privacy. We consider these form of attacks as only *security* threats and not privacy issues.

Speaker identification, on the other hand, finds application in audio surveillance applications. The audio-based surveillance can be in the form of wiretapping, where the a security agency, e.g., police, listens in on telephone conversations or the audio captured by hidden microphones in public areas. A basic characteristic of surveillance is that the agency needs to perform it obliviously, i.e., the subjects under surveillance should not know about it. Although listening in on personal conversations either over the telephone or from physical sources is an effective surveillance method to identify credible security threats, this directly infringes on the privacy of innocent individuals who may not be intended targets of the surveillance. To prevent this, the agency would first need to perform speaker identification to determine if the speech input belongs to a speaker who is supposed to be under surveillance. In order to perform speaker identification using a conventional setup, the agency would need complete access to the speech input, which itself would be a privacy violation, resulting in a circular problem.

In this article we develop frameworks for *privacy preserving* speaker verification and speaker identification tasks, where the system performs verification or identification without being able to observe the speech input. Our solution is based on secure multiparty computation (SMC) protocols [4] that enables the system to perform computation only on encrypted speech data without requiring to observe in plaintext. We envision a client-server model, where a user executes a client program on a network computation device, such as a computer or smartphone, coupled with a public key cryptosystem. The user retains its private key, while the public key is shared with the system. We, therefore, eliminate the possibility that the system could phish for a user's voice in speaker verification or listen in to the speech input in

Manuscript received April 16, 2012; revised June 22, 2012 and August 11, 2012; accepted August 12, 2012. Date of publication August 28, 2012; date of current version December 21, 2012. This work was supported by the National Science Foundation (NSF) under Grant 1017256 and an Amazon AWS in Education Research Grant. An earlier version of this article appeared as [1]. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Thomas Fang Zheng.

The authors are with Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: manasp@cs.cmu.edu; bhiksha@cs.cmu.edu).

Digital Object Identifier 10.1109/TASL.2012.2215602

speaker identification. Secondly, in speaker verification, we require the system to store only encrypted speech patterns provided by enrolling users, thereby protecting against an adversary who breaks into the system. We later present mechanisms where system is able to perform verification over encrypted speech input provided by the user against these encrypted speech patterns. We assume that one adversary does not gain access to both the user's client device and the system at the same time. We discuss the detailed privacy criteria in the later sections.

We should note that we do not aim to design superior speaker verification and speaker identification algorithms for achieving higher accuracy. We instead aim to create a mechanism to ensure the privacy of user's speech data and the models learned by the system while implementing existing verification and identification algorithms. Also, there is always a computational overhead in the privacy-preserving mechanisms due to the time required to perform the encryption and decryption operations. We also present experiments on a prototype implementation of the protocols to analyze the execution time.

Although there has been substantial work on general techniques for data processing with privacy constraints [5], [6], including protocols for privacy-preserving biometric authentication tasks such as face recognition [7] and fingerprint recognition [8], privacy-preserving speech processing is a nascent area of research. Smaragdis and Shashanka [9] propose protocols for training and evaluating Gaussian mixtures and hidden Markov models on speech data, under privacy constraints. Pathak, *et al.* [10] develop and implement an efficient protocol for privacy-preserving HMM inference applied to isolated word recognition. In this article, we extend some of these techniques to develop protocols for speaker verification and speaker identification.

II. Preliminaries

A. Speaker Verification Using GMMs

We briefly overview the technique for text-independent speaker verification. We outline the basic algorithm here; for a detailed tutorial, please refer to [11].

In a speaker verification task, we attempt to authenticate an individual based on the characteristics of his/her speech. We parameterize the speech samples by the sequences of Mel-frequency cepstral coefficients augmented by differences and double differences, i.e., a recording x consists of a sequence of feature vectors. In the enrollment phase, we require each user to submit a set of speech samples $x_1, \ldots, x_n$. We represent a person's speech characteristics by a Gaussian mixture model (GMM) λ_s. The GMM has the following form:

$$P(x_t|\lambda_s) = \sum_j w_j \mathcal{N}(x_t; \mu_j, \Sigma_j),$$

where $\mathcal{N}(x; \mu_j, \Sigma_j)$ is the j^{th} multivariate Gaussian distribution with mean μ_j and covariance Σ_j. We learn these parameters from the enrollment data using the expectation-minimization (EM) algorithm.

Although we can potentially learn the speaker model from the enrollment samples for the speaker, learning the GMM for the imposter class is less obvious as an imposter could potentially be from a very large set of speakers. We represent the generic speaker by a *universal background model* (UBM) λ_U that is trained on a large and diverse set of speakers.

In the verification phase, given a test speech sample x, we aim to check if it is likely to be uttered by the enrolled speaker or by an imposter. Then we compute the probabilities of x using the speaker model λ_s and the (UBM) λ_U. We perform the verification using the following likelihood ratio test with respect to a pre-calibrated threshold θ.

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

Model Adaptation: We mentioned above that we train the speaker model λ_s, directly from the training data, but in practice the speaker models obtained from maximum *a posteriori* (MAP) adaptation with the UBM empirically outperform the models trained directly on the enrollment data [11], [12].

The MAP adaptation procedure comprises estimation of a sample estimate of the speaker's parameters, followed by interpolation with the UBM. Given set of enrollment speech samples $x_1, \ldots, x_n$, we first compute the *a posteriori* probabilities of the individual Gaussians in the UBM λ_U. For the i^{th} mixture component of the UBM,

$$P(i|x_t) = \frac{w_i^U \mathcal{N}(x_t; \mu_i^U, \Sigma_i^U)}{\sum_j w_j^U \mathcal{N}(x_t; \mu_j^U, \Sigma_j^U)}. \quad (2)$$

Similar to the M-step of EM, we use the *a posteriori* probabilities to compute new weights, mean, and second moment parameters.

$$w_i' = \frac{1}{T}\sum_t P(i|x_t), \quad \mu_i' = \frac{\sum_t P(i|x_t)x_t}{\sum_t P(i|x_t)},$$
$$\Sigma_i' = \frac{\sum_t P(i|x_t)x_t x_t^T}{\sum_t P(i|x_t)}. \quad (3)$$

Finally, we obtain the parameters of the adapted model $\lambda_s = \{\hat{w}_i^s, \hat{\mu}_i^s, \hat{\Sigma}_i^s\}$ from the convex combination of the above parameters and the UBM parameters as follows.

$$\hat{w}_i^s = \alpha_i w_i' + (1-\alpha_i) w_i^U, \quad \hat{\mu}_i^s = \alpha_i \mu_i' + (1-\alpha_i)\mu_i^U,$$
$$\hat{\Sigma}_i^s = \alpha_i \Sigma_i' + (1-\alpha_i)\left[\Sigma_i^U + \mu_i^U \mu_i^{UT}\right] - \hat{\mu}_i^s \hat{\mu}_i^{sT}. \quad (4)$$

The adaptation coefficients α_i control the amount of contribution of the enrollment data relative to the UBM.

B. Speaker Identification Using GMMs

Speaker identification can be considered to be the generalization of speaker verification task to the case of multiple speakers. We consider the setting where we already have a set of K speakers $\{S_1, \ldots, S_K\}$, and given a test speech sample, we are interested in assigning it to one of the K speakers. In open-set speaker identification, we consider a default or *none of*

the above case in which the speech sample does not correspond to any of the K speakers.

Similar to speaker verification, we represent the speakers by GMMs $\{\lambda_1, \ldots, \lambda_K\}$, and the default case by the UBM λ_U. We obtain the speaker models either by learning a GMM directly from the speech data for that speaker or by adapting the UBM to the training data for individual speakers.

In the identification step, we individually compute the probabilities of all the speaker models with respect to the given test speech sample x. We choose the speaker corresponding to the model having the highest probability.

C. Homomorphic Encryption

Homomorphic encryption schemes allow for operations to be performed directly on encrypted data (*ciphertext*) without requiring knowledge of their unencrypted data (*plaintext*). If the homomorphic encryption scheme is asymmetric, i.e., provides public and private keys, the party with private key "Alice" can encrypt its input and transfer it to the party with the public key "Bob," who can perform the necessary operations on the encrypted data alone. The operations on encrypted data protect privacy as Bob cannot decrypt and observe the input provided by Alice. This property is the foundation of our privacy preserving mechanisms.

A cryptosystem in which we can perform any operations on the plaintext by performing operations on corresponding ciphertext is called a *fully* homomorphic cryptosystem (FHE). The first such cryptosystem was proposed in a breakthrough work by Gentry [13], [14]. Although the construction satisfies the necessary properties for FHE, it is found to be computationally inefficient to be used in practice [15], and developing computationally practical FHE schemes is an active area of research [16].

There are well-established efficient partially homomorphic encryption schemes that allow a few operations to be performed on plaintext by performing operations on ciphertext, e.g., unpadded RSA is multiplicatively homomorphic, El Gamal [17] and Paillier [18] are additively homomorphic. The Paillier cryptosystem allows us to compute inner products of an encrypted vector with a plaintext vector. Given a ciphertext vector $E[x] = (E[x_1], \ldots, E[x_m])$ and a plaintext vector $y = (y_1, \ldots, y_m)$, we can homomorphically compute $(E[x_1]^{y_1}, \ldots, E[x_m]^{y_m})$ to obtain $(E[x_1 y_1], \ldots, E[x_m y_m])$. We can then homomorphically add these elements to obtain $E[x_1 y_1 + \cdots + x_m y_m]$ which is the encrypted inner product $E[x^T y]$.

Being able to perform homomorphic addition alone on the ciphertext has its limitations; given two encrypted vectors $E[x] = (E[x_1], \ldots, E[x_m])$ and $E[y] = (E[y_1], \ldots, E[y_m])$, we are not able to directly compute the inner product $E[x^T y]$. Boneh-Goh-Nissim (BGN) [19] is a homomorphic cryptosystem provides arbitrary number of additions along with one multiplication on ciphertexts. We can use it to directly compute the encrypted inner product $E[x^T y]$ from two encrypted vectors. The homomorphic operations provided by the BGN cryptosystem are a superset of those provided by the Paillier cryptosystem. There is, however, a significant difference in performance. We prefer to use the Paillier cryptosystem in constructing interactive protocols. When we require the homomorphic computation of inner products from ciphertexts in our privacy preserving mechanisms, we use the BGN cryptosystem.

Interactive and Non-Interactive Protocols: If we need to perform operations on ciphertexts beyond those provided by the partially homomorphic cryptosystem, we need to construct interactive protocols, where both the parties Alice and Bob perform part of the computation. In interactive protocols, Alice encrypts the data using her public key and transfers it to Bob. Using the operations provided by the homomorphic cryptosystem, Bob obtains the necessary intermediate results, and sends randomly perturbed ciphertexts back to Alice, typically perturbed by additive or multiplicative blinding. Alice decrypts the data using her private key, performs some intermediate operations, encrypts the results and sends it back to Bob. In this way, Alice and Bob each perform part of the computation till they obtain the required output. On the other hand, in non-interactive protocols, Bob requires no assistance from Alice beyond receiving the input ciphertexts and directly obtains the output by performing the necessary computation himself. As we shall see in Section III-C, we use the Paillier cryptosystem to constuct interactive protocols and the BGN cryptosystem to construct non-interactive protocols.

III. Privacy-Preserving Speaker Verification

In this section, we develop our framework for privacy-preserving speaker verification system. The system uses UBM and adapted GMMs (Section II-A) to represent the speakers. We discuss the privacy issues of our framework by considering the adversarial roles of the various parties. We then present the system architecture along with the enrollment and verification protocols.

A. Adversarial Model

We assume the user and the system to be independent parties that have access to separate computing devices operating in a client-server framework. We assume the parties to be computationally bounded, i.e., we consider that the parties cannot directly break the encryption to obtain plaintext from ciphertext without the decryption key.

In SMC, we consider two types of adversarial behavior of parties: *semi-honest* and *malicious*. We consider a semi-honest party to follow the steps of the protocol correctly, but keep a record of all intermediate results while trying to gain as much information as possible about the input data belonging to other parties. A malicious party, in addition to the semi-honest behavior, takes active steps in disrupting the protocol by using fraudulent input data to gain information about the input data belonging to other parties, and to obtain an unrealistic result.

Our main privacy constraint is that the system should not be able to observe the speech samples belonging to the user both during the enrollment and verification phases. In order to achieve this, we require that users submit encrypted adapted models in the enrollment phases and encrypted test speech data in the verification phase, where the encryption is performed with the private key belonging to the user. In the verification phase, the system needs to perform all the necessary operations

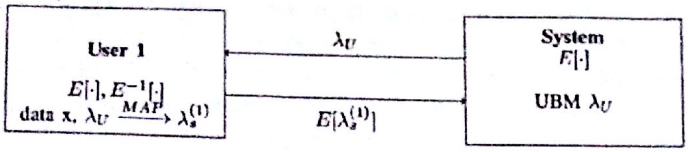

Fig. 1. Enrollment protocol: user has enrollment data x and system has the UBM λ_U. System obtains encrypted speaker model $E[\lambda_s^{(1)}]$.

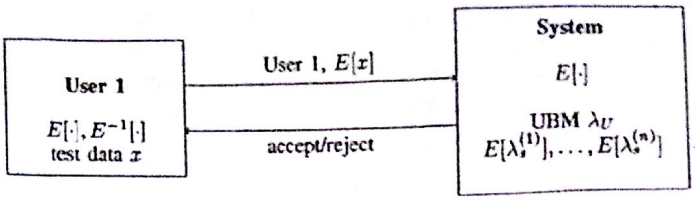

Fig. 2. Verification protocol: user has test data x and system has the UBM λ_U and encrypted speaker model $E[\lambda_s^{(1)}]$. The user submits encrypted data and the system outputs an accept/reject decision.

over this encrypted data using the homomorphic operations described in Section II-C, which we refer to as the verification protocol. The only malicious behavior the system can exhibit in the verification protocol is to modify the steps of the procedure to obtain incorrect output. As the system never observes the speech input in plaintext, this will not help it in anyway to obtain any information about the input. On the other hand, a system giving arbitrary accept/reject decisions will only antagonize the users and accepting false users would lead to a security problem, but not a loss in privacy. For the same reasons, we assume that the system provides the user with the correct copy of the algorithm which we assume to be public. We therefore assume that the system to be semi-honest.

We also assume that the user is semi-honest during the enrollment phase. By maliciously submitting false adapted models, the user will only help in creating a weak authentication system, and there is no incentive for the user to do so. In the verification phase, however, the user could possibly be an imposter who is an adversary using a compromised device belonging to the user. We therefore assume that the user is malicious. In this model, we cannot make assumptions about the correctness of the input data provided by the user. In order to counter such an adversary, we require that the system apply the same input data for both the UBM and the adapted models.

B. System Architecture

As an initialization step, the user generates a public/private key pair and sends the public key to the system. We assume that the system trains a UBM λ_U on publicly available data and stores it with itself as plaintext. In the enrollment protocol (Fig. 1), the system sends the UBM to the user in plaintext and the user performs the adaptation. The user then encrypts the adapted model with its key and sends it to the system. After executing the enrollment protocol with all users, the system has encrypted models for all users along with the UBM. At the end of the protocol, we require the user to delete the enrollment data from its computation device in order to protect it from an adversary who might gain unauthorized access to it. The user device stores the encryption and decryption keys. Similarly, as the server stores only the encrypted speaker models, it is also protected against an adversary who might compromise the system to gain the speaker models, in order to impersonate the user later. If an adversary compromises the user device as well as the system, we consider the system to be completely compromised as the adversary can use the decryption key to obtain the speaker model in plaintext.

In the verification protocol (Fig. 2), the user produces a test speech sample x and encrypts it using its key and sends it to the system along with the claimed identity. The system evaluates the encrypted test sample with the UBM and the encrypted model for the claimed speaker it had obtained in the enrollment protocol using the homomorphic operations and obtains two encrypted scores. The system makes its decision by comparing the difference between the two encrypted scores with a threshold using the compare protocol.

C. Speaker Verification Protocols

We now describe the enrollment and verification protocols in detail. We use the following construction from [9]: the multivariate Gaussian $\mathcal{N}(x;\mu,\Sigma)$ computed on any d-dimensional vector x can be represented in terms of of a $(d+1)\times(d+1)$ matrix W.

$$\tilde{W} = \left[\begin{array}{c|c} -\frac{1}{2}\Sigma^{-1} & \Sigma^{-1}\mu \\ \hline 0 & w^* \end{array}\right],$$
$$\text{where } w^* = -\frac{1}{2}\mu^T\Sigma^{-1}\mu - \frac{1}{2}\log|\Sigma|. \quad (5)$$

This implies $\log\mathcal{N}(x;\mu,\Sigma) = \tilde{x}^T\tilde{W}\tilde{x}$, where $\tilde{x}$ is an extended vector obtained by concatenating 1 to x. As suggested by [10], we reduce this computation to a single inner product $\bar{x}^T W$, where the extended feature vector $\bar{x}$ consists of all pairwise product terms $\tilde{x}_i\tilde{x}_j \in \tilde{x}$ and W is obtained by unrolling $\tilde{W}$ into a vector. In this representation, we have

$$\log P(x|i) = \log\mathcal{N}(x;\mu,\Sigma) = x^T W. \quad (6)$$

We assume that the user computes MFCC features from the speech samples. In the following discussion, we refer to the MFCC features as the speech sample itself.

1) Private Enrollment Protocol: We assume that the system already has access to the UBM, λ_U trained on a collection of publicly available speech data. The speaker verification algorithm requires a speaker model obtained from adapting the UBM to the enrollment data provided by the speaker. We require that the speaker model is kept with the system only after it is encrypted by the user's key using an additively homomorphic encryption scheme like the Paillier cryptosystem. We outline this enrollment protocol below.

Private Enrollment Protocol.

Inputs:

(a) User has the enrollment samples $x_1, \ldots, x_n$ and both encryption key $E[\cdot]$ and decryption key $E^{-1}[\cdot]$.

(b) System has the UBM $\lambda_U = W_i^U$ for $i = 1, \ldots, N$, mixing weight α, and the encryption key $E[\cdot]$.

Output: System has the encrypted user model $E[\lambda_s] = E[\hat{W}_i^s]$, for $i = 1, \ldots, N$.

(i) The system sends the UBM λ_U to the user.

(ii) User performs the model adaptation of λ_U with the enrollment samples $x_1, \ldots, x_n$ (see Section II-A) to obtain the adapted model λ_s.
(iii) The user represents the mixture components of the adapted model using the $\hat{W}_i$ matrix representation described above.
(iv) The user encrypts $\hat{W}_i$ using its encryption key and sends it to the system.

Although this arrangement, where the user performs the adaptation, is adequate in most applications, we also construct a protocol for the system to perform the adaptation over encrypted enrollment data to obtain the encrypted speaker models.

2) Private Verification Protocols: In the verification protocol, the system needs to evaluate the probabilistic score of the given test sample using the UBM λ_U and the adapted model λ_s. This score is evaluated for all frames of the test sample; for a test sample $x = \{x_1, \ldots, x_T\}$ and the model λ, this score is given by

$$P(x|\lambda) = \prod_t \sum_j P(x_t|j) = \prod_t \sum_j w_j \mathcal{N}(x_t; \mu_j, \Sigma_j).$$

We compute this score in the log domain to prevent numerical underflow,

$$\log P(x|\lambda) = \sum_t \log \sum_j P(x_t|j) = \sum_t \log \sum_j e^{x_t^T W_j}, \quad (7)$$

using the W matrix representation from (6).

In our privacy model, we assume that the user has the speech sample x and the system has the encrypted matrices W_j. Private verification protocol proceeds as follows: the user sends the encrypted frame vectors $E[x_t]$ to the system and which is then used to compute the inner products $E[x_t^T W_j]$ using the homomorphic properties of the Paillier cryptosystem. In order to use the inner products to compute the log scores, we need to perform an exponentiation operation on ciphertext. As our cryptosystem only supports homomorphic additions and a single multiplication, it is not possible to do this directly, and we therefore use the *logsum protocol* which requires user participation in the intermediate steps. We outline this interactive verification protocol below.

Interactive Private Verification Protocol.

Inputs:
(a) User has the test sample x with frame vectors $\{x_1, \ldots, x_T\}$ and both encryption key $E[\cdot]$ and decryption key $E^{-1}[\cdot]$.
(b) System has $E[\lambda] = E[W_j]$, for $j = 1, \ldots, N$, and the encryption key $E[\cdot]$.

Output: System obtains the score $E[\log P(x|\lambda)]$.
(i) The user encrypts the frame vectors $E[x_t]$ and sends it to the system.
(ii) For each mixture matrices $E[W_j]$ and each frame vector $E[x_t]$, the system computes the inner product $E[x_t^T W_j]$.
(iii) The system and the user then participate in the logsum protocol to obtain $E[\log P(x_t|\lambda)] = E[\log \sum_j e^{x_t^T W_j}]$.
(iv) The system adds the logsums homomorphically to obtain the $E[\log P(x|\lambda)]$.

As the system has access to the UBM in plaintext, the user and the system can execute the private mixture of Gaussians evaluation protocol (MOG) given by [9]. However, we observe that the above protocol is substantially faster than MOG using unencrypted models. This is because in the above protocol, the user computes part of the inner products $x_t^T W_i^s$ in plaintext. We therefore repeat the above protocol with the encrypted UBM $E[W_i^U]$ to obtain the encrypted probability $E[\log P(x|\lambda_U)]$. The system and the user can finally execute the millionaire protocol [4], to privately compute if $\log P(x|\lambda_U) > \log P(x|\lambda_s) + \theta$ and the system uses this as the decision to authenticate the user.

Throughout this protocol, the system never observes the frame vectors x_t in plaintext. The various supplementary protocols require the system to send encrypted partial results to the user. While doing so the system either adds or multiplies the values it sends to the user by a random number. This also prevents the user from learning anything about the partial results obtained by the system. Even after satisfying these privacy constraints, the system is able to make the decision on authenticating the user.

A drawback of the above protocol is that it requires participation from the user in the intermediate steps. Apart from the computational and data transfer overhead incurred, this also results in a privacy vulnerability. A malicious user can provide fake inputs in the logsum step to disrupt the protocol and try to authenticate itself without having a speech sample belonging to a genuine user. To prevent this, we construct a non-interactive protocol, where the user needs to submit the encrypted speech sample and the system directly computes the probability scores. As this is not possible due to the exponentiation involved in (7), we modify the score function itself by alternating the log and sum functions.

$$\text{score}(x, \lambda) = \sum_t \sum_j \log P(x_t|j) = \sum_t \sum_j x_t^T W_j. \quad (8)$$

This has the advantage that using homomorphic multiplication provided by the BGN cryptosystem, the system can by itself compute the inner products $E[x_t^T W_j]$ for both the speaker model and the UBM for the same encrypted input $E[x]$ provided by the user. Beyond that, the system only needs homomorphic addition to compute the score without requiring any user participation. We experimentally observe that the accuracy of this score is close to that of the original probabilistic score. We outline the non-interactive verification protocol below.

Non-Interactive Private Verification Protocol.

Inputs:
(a) User has the test sample x with frame vectors $\{x_1, \ldots, x_T\}$ and both encryption key $E[\cdot]$ and decryption key $E^{-1}[\cdot]$.
(b) System has $E[\lambda] = E[W_j]$, for $j = 1, \ldots, N$, and the encryption key $E[\cdot]$.

Output: System obtains the score $E[\log P(x|\lambda)]$.

(i) The user encrypts the frame vectors $E[x_t]$ and sends it to the system.
(ii) For each mixture matrices $E[W_j]$ and each frame vector $E[x_t]$, the system computes the inner product $E[x_t^T W_j]$ homomorphically.
(iii) The system adds the inner products homomorphically to obtain the $E[\text{score}(x, \lambda)]$.

The system executes the same protocol using the adapted model and the UBM using the same inputs it receives from the user in step (i). The system never observes the frame vectors in plaintext in this protocol as well. As there is no user participation after the initial encrypted frame vectors are obtained, there is no loss of privacy of the user data.

IV. Privacy-Preserving Speaker Identification

In this section, we develop our framework for privacy-preserving speaker identification. Similar to the speaker verification framework, we use the GMMs to represent the speakers (Section II-B). We discuss the privacy issues of our framework by considering the adversarial roles of the various parties. We then present the system architecture along with the enrollment and verification protocols.

A. Adversarial Model

We consider speaker identification with two parties: the client who has access to the test speech sample and the server who has access to the speaker models and is interested in identifying the most likely speaker corresponding to the test sample. For concreteness, we consider the setting of a surveillance operation as an application scenario for our framework. In case of surveillance, the server would be the security agency, and the client would be the telephone company that has access to the conversations of all subscribers. It is important to note that we do not consider the individual phone users as the client as these individuals should be unaware about being subjected to surveillance, as no user would willingly participate in such an activity.

The agency has access to the speaker models for the individuals it already has wiretapping warrants against. The agency directly deals with the telephone company in order to identify if any such individual is participating in a telephone conversation. If that is found to be the case, the agency would follow the necessary legal procedure to obtain the phone recording. The same analogy also holds for the case of physical surveillance; e.g., a supermarket installs hidden security cameras with microphones in their premises, and the police might want the audio recordings to gain evidence about criminal activity. In this case the supermarket would be the client and the police would be the server.

Although we aim to construct mechanisms to protect the privacy of the client data, we assume that the client cooperates with the server. In the speaker identification task, it is possible that the client can refuse the server by simply not providing the speech input or providing white noise or speech from some other source as input. In this case, it will not be possible for the server to perform surveillance. In order to prevent this, the server can legally require the client to use only the correct data as input. We also assume that the server already knows the K speakers it is looking to identify. The server uses the publicly available data for the K speakers to train the corresponding speaker models without requiring the participation of the client. The server trains a UBM as a speaker model corresponding to above case, that is the input speech not matching any of the K speakers. In the process of performing speaker identification with these models, our privacy criteria are:

1) The server should not observe the speech data belonging to the client.
2) The client should not observe the speaker models belonging to the server.

The first criterion follows directly from the discussion above, the server being able to observe the speech data causes the violation of client privacy. The client, i.e. phone company, can also include in its privacy policy that the user privacy will be protected because if an agency needs to perform surveillance, it will do so using a privacy-preserving speaker identification system. The second criterion is important because it is possible to identify the speaker by reverse-engineering the speaker models. The client might do this to gain information about the blacklisted individuals the server is performing surveillance on. This would cause problems in the investigation process as the client can convey this information to those individuals.

We consider the adversarial behaviors of the client and the server below. In the speaker identification task, the server tries to gain as much information as possible from the input provided by the client. The server cannot do much to disrupt the protocol, the server can use incorrect speaker models, but that would only result in incorrectly identified speaker. As that is not in the interest of the server, we assume that the server is *semi-honest*. Similarly, the client tries to gain information about the server models from the intermediate steps of the protocol. As discussed above, we assume that the client cooperates in the speaker identification task by submitting the correct input, and therefore we also require the client to be *semi-honest*.

B. System Architecture

We assume that the server knows the set of speakers $\{S_1, \ldots, S_K\}$ that it is interested in identifying and has access to data for each speaker. This data could be publicly available or extracted by the server in its previous interactions with the speakers. The server uses a GMM to represent each speaker and also a UBM λ_U. We consider the UBM to represent none of the above case, where the test speaker is outside the set $\{S_1, \ldots, S_K\}$. The server obtains GMMs for individual speakers $\{\lambda_1, \ldots, \lambda_K\}$ by either training directly over the data for that speaker or by performing MAP adaptation with the UBM. The client has access to the speech sample, that it represents using MFCC features. To perform identification, the server needs to evaluate the $K+1$ GMMs $\{\lambda_1, \ldots, \lambda_K, \lambda_U\}$ over the speech sample. The server assigns the speaker to the GMM that has the highest probability score, $\arg\max_i P(x|\lambda_i)$.

Our design of the first variant of the speaker identification framework is shown in Fig. 3 where the client sends speech samples to the server. Initially, the client generates a public/private key pair $E[\cdot], E^{-1}[\cdot]$ for the homomorphic cryptosystem and sends the public key to the server. The client will then encrypt the speech sample and send it to the server. The server

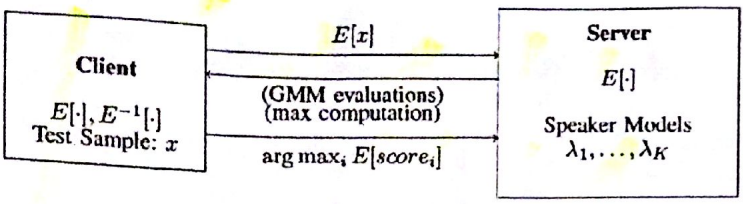

Fig. 3. GMM-based speaker identification: client sends encrypted speech sample to the server.

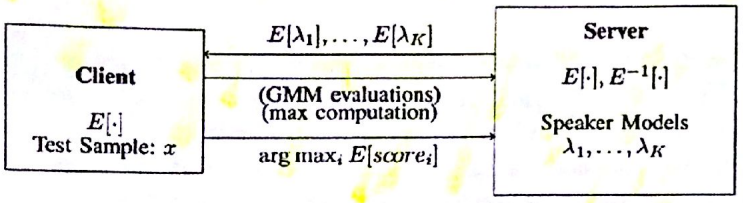

Fig. 4. GMM-based speaker identification: server sends encrypted models to the client.

uses the GMM evaluation protocol for each speaker model to obtain the $K+1$ encrypted probability scores. The server and client then engage in the private maximum computation protocol where only the server knows the model having the highest score at the end.

In the second variant of the speaker identification framework denoted in Fig. 4, the server sends models to the client. To do this privately, the server as opposed to the client creates a public/private key pair $E[\cdot], E^{-1}[\cdot]$ for the homomorphic cryptosystem and sends the public key to the client. The server encrypts all GMMs using this key and sends it to the client. The client evaluates all the GMMs over the speech sample it has and obtains $K+1$ encrypted scores. The client and the server then participate in the private maximum computation protocol where only the server will know the model having maximum score at the end.

C. Speaker Identification Protocols

We reuse the W construction for representing a Gaussian as given by (6):

$$\log P(x|i) = \log \mathcal{N}(x; \mu, \Sigma) = x^T W.$$

We construct the following GMM evaluation protocols using the additively homomorphic Paillier cryptosystem for the two cases: evaluating over private speech data and evaluating over private speaker models and later use the protocols in the speaker identification protocols.

1) Case 1: Client Sends Encrypted Speech Sample to the Server:

GMM Evaluation Protocol with Private Speech Sample.

Inputs:

(a) Client has the test sample x with frame vectors $\{x_1, \ldots, x_T\}$ and both encryption key $E[\cdot]$ and decryption key $E^{-1}[\cdot]$.

(b) Server has the GMM λ with N mixture components $W_1, \ldots, W_N$ and the encryption key $E[\cdot]$.

Output: Server obtains the encrypted score $E[\log P(x|\lambda)]$.

(i) The client encrypts the frame vectors $E[x_t]$ and sends it to the server.

(ii) For each Gaussian matrix W_j and each frame vector $E[x_t] = (E[x_{t1}], \ldots, E[x_{tn}])$, the server computes the inner product $E[x_t^T W_j]$ homomorphically as

$$\prod E[x_{ti}]^{W_{ji}} = E\left[\sum x_{ti} W_{ji}\right] = E\left[x_t^T W_j\right].$$

(iii) The server and the client then participate in the logsum protocol to obtain $E[\log P(x_t|\lambda)] = E[\log \sum_j e^{x_t^T W_j}]$.

(iv) The server adds the logsums homomorphically to obtain the $E[\log P(x|\lambda)]$.

By using this protocol, the server is able to privately evaluate a GMM it has in plaintext over speech data belonging to the client. As the server does not have the private key, it is not able to observe the encrypted speech sample provided by the client and the final encrypted probability score. The server executes this protocol for all the $K+1$ GMMs $\{\lambda_1, \ldots, \lambda_K\}$, including the UBM and obtains the encrypted scores $\{E[\log P(x|\lambda_1)], \ldots, E[\log P(x|\lambda_K)]\}$ and the server needs to find the GMM having the maximum probability score.

The server and the client participate in the private maximum computation protocol for this purpose. Our construction is based on the SMAX protocol of [9] and the blind and permute protocol of [20].

2) Case 2: Server Sends Encrypted Speaker Models to the Client:

GMM Evaluation Protocol with Private Speaker Model.

Inputs:

(a) Client has the test sample x with frame vectors $\{x_1, \ldots, x_T\}$ and the encryption key $E[\cdot]$.

(b) Server has the GMM λ with N mixture components $W_1, \ldots, W_N$ and both encryption key $E[\cdot]$ and decryption key $E^{-1}[\cdot]$.

Output: Client obtains the encrypted score $E[\log P(x|\lambda)]$.

(i) The server encrypts the Gaussian matrices $\{E[W_1], \ldots, E[W_N]\}$ and sends it to the client.

(ii) For each frame vector x_t and each encrypted Gaussian matrix, the client computes the inner product $E[x_t^T W_j]$ homomorphically as

$$\prod E[W_{ti}]^{x_{ji}} = E\left[\sum W_{ti} x_{ji}\right] = E\left[x_t^T W_j\right].$$

(iii) The client and the server then participate in the logsum protocol to obtain $E[\log P(x_t|\lambda)] = E[\log \sum_j e^{x_t^T W_j}]$.

(iv) The client adds the logsums homomorphically to obtain the $E[\log P(x|\lambda)]$.

The client uses the above protocol to evaluate $K+1$ GMMs provided by the server in ciphertext on its speech data. The client obtains probability scores encrypted by the server at the end of the protocol. To perform speaker identification, only the server needs to find the GMM having the maximum score. The client cannot transfer the scores to the server, as that would lead to the loss of privacy of the client speech data. We instead construct the following protocol that uses another set of keys generated

by the client and then reuse the private maximum computation protocol we discussed above. We cannot directly use the private maximum computation protocol with the parties reversed, i.e., the client as the server and the server as the client. This is because our privacy criteria require that only the server should know about the identified speaker.

Comparison of the Two System Configurations: In the first variant of the system architecture described in Case 1, the main computation and communication overhead is due to the client encrypting its speech sample. This overhead is directly proportional to the length of the sample T, with 100 frames per second. In the remainder of the GMM evaluation and private maximum computation protocols, the client and server exchange small vectors that are independent of the sample length.

In the second variant described in Case 2, the main overhead is due to the server encrypting its speaker models. As discussed above, we represent the speaker model using N matrices $W_1, \ldots, W_N$ representing a single mixture component. Each W_j matrix is of size $(d+1) \times (d+1)$, where d is the dimensionality of the frame vector. In our implementation, we use $d = 39$ with MFCC features with $N = 32$. This size is independent of the sample length. The client evaluates these models on its own unencrypted speech data. Similar to the first variant, the overhead from the remainder of the private computation is relatively small.

In this way, the cost of using the two configurations is dependent on the length of the speech sample. If the length T is typically smaller than the matrix size $N \times (d+1) \times (d+1)$, it is advantageous to use the first variant, where the client encrypts the speech sample. If T is larger than the product, it is advantageous to use the second variant. As compared to speaker verification, speaker identification is performed on relatively large amount of speech input, often multiple minutes long in practical scenarios such as surveillance. In these situations, it is significantly more efficient to use the second variant. On the other hand the speech input can be only a few seconds long in problems where speaker identification is used as an initial step for other speech processing tasks. In these situations, it is efficient to use the first variant.

V. Experiments

We present the results of experiments with the privacy preserving speaker verification and speaker identification protocols described above. We created prototype implementations of the interactive and non-interactive verification protocols in C++ using the pairing-based cryptography (PBC) library [21] to implement the BGN cryptosystem and OpenSSL library [22] to implement the Paillier cryptosystem. We performed the experiments on a 2 GHz Intel Core 2 Duo machine with 3 GB RAM running 64-bit Ubuntu.

A. Speaker Verification Experiments

Both interactive and non-interactive protocols constructed using homomorphic encryption achieved the same final probability scores as the non-private verification algorithm up to 5 digits of precision.

1) Accuracy: We used the YOHO dataset [23] to measure the accuracy of the two speaker verification protocols. We trained a UBM with 32 Gaussian mixture components on a random subset of the enrollment data and performed MAP adaptation with the enrollment data for individual speakers to obtain the speaker models. We evaluate the UBM and the speaker models on the verification data for the speaker as the true samples and the verification data for all other speakers as the imposter samples. We use EER[1] as the evaluation metric. We observed an EER of 3.1% for the interactive protocol and 3.8% for the non-interactive protocol. This implies that there is only a marginal reduction in performance by modifying the scoring function.

Although the above EERs are acceptable to be used in most verification applications, they should be considered to be preliminary and indicative of the feasibility of the two protocols. The interactive protocol essentially follows the UBM-GMM approach [11], and by using 1024 mixture components and more discriminative features, the EER can be reduced to less than 2% [23], [24]. We hypothesize that the EER for the non-interactive protocol can also be reduced similarly.

2) Execution Time: We measured the execution times for the verification protocols using BGN encryption keys of sizes 256 and 512-bits.[2] In practice, 512-bit keys are used for strong security [25]. We use 256 and 1024 bit keys for Paillier cryptosystem. We perform the verification of a 1 second speech sample containing 100 frame vectors using the UBM and the speaker models each containing 32 mixture components. The non-private verification algorithm required 13.79 s on the same input.

TABLE I
Execution Time for the Interactive Protocol With Paillier Cryptosystem

Steps	256-bit	1024-bit
Encrypting $x_1, \ldots, x_T$	136.64 s	7348.32 s
Evaluating Speaker Model	101.80 s	1899.39 s
Evaluating UBM	95.24 s	1189.98 s
Total + adapted	333.68 s	10437.69 s
	$\sim$ 5 min. 33 s	$\sim$ 2 hr. 53 min

We use the Paillier cryptosystem for the interactive protocol and the BGN cryptosystem for the non-interactive protocol, as the private inner product is needed in the latter. We summarize the results in Tables I and II for the interactive and non-interactive protocols respectively. Also, the execution time varies linearly with the utterance length. We observe that the interactive protocol is faster than the non-interactive protocol. This is due to the execution of the *private inner product* for each frame vector needed for the non-interactive protocol. The system requires to perform multiplicative homomorphic operations to obtain the inner product. These operations in turn require the computation of a bilinear pairing which is much slower than homomorphically multiplying plaintexts with ciphertexts as we do in the interactive protocol.

In both protocols, we observe that the UBM evaluation is significantly faster than the speaker model evaluation: this is because the UBM is available in plaintext with the system and the inner product requires only additive homomorphic operations.

[1]Equal error rate of $x\%$ implies that when the false accept rate is $x\%$, the false reject rate is also $x\%$.

[2]For 512-bit keys, we choose the two prime numbers q_1 and q_2 each of 256-bits, such that that their product $n = q_1 q_2$ is a 512-bit number.

TABLE II
EXECUTION TIME FOR THE NON-INTERACTIVE PROTOCOL WITH BGN CRYPTOSYSTEM

Steps	256-bit	512-bit
Encrypting $x_1, \ldots, x_T$	40.0 s	96.4 s
Evaluating Speaker Model	17450.3 s	77061.4 s
Evaluating UBM	19.5 s	77.8 s
Total	17509.8 s $\sim$ 4 hr, 52 min	77235.6 s $\sim$ 21 hr, 27 min

TABLE III
GMM-BASED SPEAKER IDENTIFICATION: EXECUTION TIME. CASE 1: CLIENT SENDS ENCRYPTED SPEECH SAMPLE TO THE SERVER

Steps	256-bit	1024-bit
Encrypting $x_1, \ldots, x_T$	136.64 s	7348.32 s
Evaluating Speaker Model	95.24 s	1189.98 s
Total for 10 speakers	1089.04 s $\sim$ 18 min, 9 s	19248.12 s $\sim$ 5 hr, 20 min

TABLE IV
GMM-BASED SPEAKER IDENTIFICATION: EXECUTION TIME. CASE 2: SERVER SENDS ENCRYPTED SPEAKER MODELS TO THE CLIENT

Steps	256-bit	1024-bit
Encrypting $W_1, \ldots, W_N$	32.17 s	1854.07 s
Evaluating Speaker Model	460.07 s	4796.26 s
Total for 10 speakers	4922.40 s $\sim$ 1 hr, 22 min	66503.30 s $\sim$ 18 hr, 28 min

This is in contrast to evaluating the speaker model that is only available in ciphertext.

B. Speaker Identification Experiments

The precision for the speaker identification protocols was the same as the speaker verification protocols.

1) Accuracy: We also used the YOHO dataset [23] to measure the accuracy of the speaker identification task. We used the experimental setup similar to [26]. We observed the same accuracy for the two variants of the speaker identification protocol as they both resulted in the same speaker scores. We trained a UBM on a random subset of the enrollment data and performed MAP adaptation with the enrollment data for K speakers to obtain the speaker models. We evaluated the UBM and the speaker models on the test data for the K speakers, in addition to the speakers outside the set representing the none of the above case. We used identification accuracy, i.e., the fraction of the number of times a test speaker was identified correctly. For a 10-speaker classification task, we observed 87.4% accuracy.

2) Execution Time: We measured the execution times for the verification protocols using Paillier encryption keys of sizes 256 and 1024-bits. We identify a 1 second speech sample containing 100 frame vectors using $K + 1$ speaker models each containing 32 mixture components using the two variants of the speaker identification protocol. We report time for evaluating 10 speaker models. As the time required for evaluating each model is approximately the same, these numbers can be appropriately scaled to obtain estimated execution time for other number of speakers. We summarize the results in Tables III and IV.

VI. CONCLUSION AND FUTURE WORK

In this article we developed the privacy-preserving protocol for GMM-based algorithm for speaker verification using homomorphic cryptosystems such as BGN and Paillier encryption. The system observes only encrypted speech data, and hence, cannot obtain information about the user's speech. We constructed both interactive and non-interactive variants of the protocol. The interactive variant is relevant in the case of semi-honest adversary and the non-interactive variant is necessary in the case of malicious adversary. During the exchanges required by the protocols, the user only observes additively or multiplicatively masked data, and does not gain any information about the user's speech from it. The proposed protocols are also found to give results which are same up to a high degree of precision compared to a non-private GMM adaptation based scheme. The interactive protocol is more efficient than the non-interactive protocol as the latter requires homomorphic multiplication using BGN cryptosystem.

We also developed a framework for privacy-preserving speaker identification using GMM and Paillier cryptosystem. In this model, the server is able to identify which of the K speakers best correspond to the speech input provided by the client without being able to observe the input. We present two variants of the framework, where either the client submits encrypted speech to the server or the server submits encrypted speaker models to the client. The first variant of the protocol is faster than the second variant, but the bandwidth requirement of the latter is independent of the sample length.

In the timing experiments for both speaker verification and speaker identification, we observe that the parties spend a large amount of time performing encryption operations. This could be substantially reduced by using a parallel computation framework such as graphics processing units (GPUs). We can also leverage the tools and techniques used in this paper to create privacy-preserving protocols for other speaker recognition algorithms, e.g., supervectors with NAP [27] and GMM-UBM with JFA [28]. We leave these directions for future work.

REFERENCES

[1] M. Pathak and B. Raj, "Privacy preserving speaker verification using adapted GMMs," in *Proc. Interspeech*, 2011.
[2] B. Pellom and J. Hansen, "An experimental study of speaker verification sensitivity to computer voice-altered imposters," in *Proc. ICASSP*, 1999, pp. 837–840.
[3] D. Sundermann, H. Hoge, A. Bonaforte, H. Ney, A. Black, and S. Narayanan, "Text-independent voice conversion based on unit selection," in *Proc. ICASSP*, 2006, pp. 81–84.
[4] A. Yao, "Protocols for secure computations," in *Proc. FOCS*, 1982.
[5] J. Vaidya, C. W. Clifton, and Y. M. Zhu, *Privacy Preserving Data Mining*, ser. Advances in Information Security. New York: Springer, 2006, vol. 19.
[6] , C. C. Aggarwal and P. S. Yu, Eds., *Privacy Preserving Data Mining: Models and Algorithms*, ser. Advances in Database Systems. New York: Springer, 2008, vol. 34.
[7] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Privacy Enhanc. Technol.*, 2009, pp. 235–253.
[8] A. Nagar, S. Rane, and A. Vetro, "Alignment and bit extraction for secure fingerprint biometrics," in *SPIE Electron. Imaging, Media Forensics, Security*, 2010.
[9] P. Smaragdis and M. Shashanka, "A framework for secure speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 4, pp. 1404–1413, May 2007.