Towards a Scalable and Privacy-Preserving Audio Surveillance System

Muhammad Mazhar Ullah Rathore, Elmahdi Bentafat, and Spiridon Bakiras, *Member, IEEE*

***Abstract*—The human voice is one of the passive biometrics that can be used in a surveillance system to uniquely identify individuals. It allows law enforcement agencies to detect and track suspects by deploying capturing devices (such as microphones) within a certain region. To address the clear privacy concerns of such an approach, we propose an efficient way of detecting suspects in public areas—through their voices—while preserving the privacy of innocent individuals. More precisely, our approach is quite suitable for large-scale surveillance systems, where millions of recordings are analyzed every day. Our privacy-preserving model is built on top of the most accurate speaker recognition systems, and we show that the accuracy loss due to the added privacy-preserving layer is negligible. The latter employs a highly efficient cryptosystem to securely compute the similarity scores between the captured utterances and the ones stored in the suspects' database. Specifically, the system computes, for each suspect, the encrypted Probabilistic Linear Discriminant Analysis (PLDA) score and obliviously matches it against a set threshold. More importantly, we show that our computation and communication overhead is significantly lower compared to the state-of-the-art techniques, which facilitates a real-time surveillance operation. Our protocol necessitates a single round of communication between the server and the capturing device and, for a database of 100 suspects, the online computation time is only 135 ms on the capturing device and 35 ms on the server, whereas the required communication is 12 KB.**

***Index Terms*—Surveillance systems, privacy, homomorphic encryption.**

I. Introduction

SURVEILLANCE systems are being deployed in numerous countries around the world. Surveillance relies on a variety of biometric data that uniquely identify individuals. One example is speech, which could be deployed for surveillance purposes, given the increasing popularity of portable devices equipped with microphones. In a naive surveillance implementation, the law enforcement agency (LEA) collects the recordings from public areas and extracts the feature vector of every unique utterance. Then, it checks whether any one of the captured voices is present in their suspects' database. However, this approach clearly violates the privacy rights of lawful citizens. Indeed, the recorded data can be linked to the location of the recording device, thus disclosing the precise location of the recorded individuals. If employed on a large-scale, this naive approach may disclose private information, such as health status, habits, etc. Moreover, international standards are becoming more stringent regarding the privacy of biometric information. For instance, the ISO/IEC 24745:2011 standard [1] emphasizes the need to preserve the privacy of biometric information, by implementing properties such as unlinkability, irreversibility, and renewability.

Speaker recognition has made very significant advances in the past few years, due to the application of machine learning algorithms. These techniques facilitate both speaker verification and speaker identification. In a speaker verification system (one-to-one matching), speech is used to authenticate the speaker in a given recording. Whereas in a speaker identification system (one-to-many matching), the goal is to identify the speaker within a dataset of speech samples, if present. Audio-based surveillance is one of the applications of speaker identification. Audio surveillance through speaker identification can be performed by LEAs through various means, e.g., listening to phone conversations, capturing recordings from microphones in public areas, or using captured audio from smartphone devices that have a dedicated application installed. This surveillance mode could be very effective in apprehending criminals, however, the underlying privacy concerns must be addressed.

One example of existing solutions for wide-scale surveillance is the European SEcuRity KEeps Threats away (SERKET) project [2], which captures audio and video feeds to perform surveillance tasks. (Without focusing too much on privacy concerns). At the same time, multiple privacy-related laws (including biometric data privacy) are being deployed around the world. These include the General Data Protection Regulation 2016/679 (GDPR) [3] and the Data Protection Law Enforcement Directive 2016/680 [4] by the European member states, the California Consumer Privacy Act (CCPA), the Illinois Biometric Protection Act (BIPA), the Stop Hacks and Improve Electronic Data Security (SHIELD) law in the US, the Protection of Personal Information Act (POPIA) in South Africa, the General Personal Data Protection Law (LGDP) in Brazil, to name a few. It is true that most data protection regulations exclude LEA activities (e.g., GDPR Article 2.2.d). However, our work is inspired by those principles, which treat biometric

Received 28 March 2022; revised 28 November 2024; accepted 2 December 2024. Date of publication 11 December 2024; date of current version 24 January 2025. The associate editor coordinating the review of this article and approving it for publication was Dr. Brian Kingsbury. *(Corresponding author: Muhammad Mazhar Ullah Rathore.)*

Muhammad Mazhar Ullah Rathore is with the Department of Computer Science, Lakehead University, Thunder Bay, ON P7B 5E1, Canada (e-mail: mazhar.rathore@lakeheadu.ca).

Elmahdi Bentafat is with the Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar (e-mail: elbentafat@hbku.edu.qa).

Spiridon Bakiras is with the Infocomm Technology Cluster, Singapore Institute of Technology, Singapore 138683 (e-mail: spiridon.bakiras@singaporetech.edu.sg).

Digital Object Identifier 10.1109/TASLP.2024.3516521

data as highly sensitive and emphasize limiting its exposure. Our system ensures that the biometrics of lawful citizens are not unnecessarily accessed or stored by LEAs. By aligning with the principle of data minimization (e.g., GDPR Article 5.1.c), it mitigates the risk of misuse or overreach. More importantly, by protecting individuals' biometric data, our system aims to build trust in surveillance technologies, which is essential for their acceptance and effectiveness.

In this article, we propose a framework that performs the task of audio surveillance in a privacy-preserving manner. To this end, the LEA only has access to the potential suspects' recordings, and nothing else. Furthermore, all recordings at the surveillance devices are deleted immediately after executing the protocol with the server. More importantly, the capturing devices do not have access to the server's cleartext database, and, therefore, the suspects' list remains secret. Our solution is based on one of the most accurate speaker recognition techniques, which is x-vector [5]. This method is based on neural networks that generate an embedding for each speaker, given a set of his/her audio recordings. The embeddings are generated in a way, such that (i) the embeddings from the same speaker are similar (according to a distance metric); and (ii) the embeddings from different speakers are dissimilar. On top of x-vector, our method employs Homomorphic Encryption (HE), in order to compute the similarity distance in a privacy-preserving manner. At the end of the protocol execution, the server will only learn the ID of a suspect that triggered a positive match (if any). As such, the server will remain oblivious with regards to the voices that did not produce a positive match.

It is worth mentioning that our goal is not to propose or improve any speaker recognition algorithm. Instead, our aim is to introduce an additional cryptographic layer that maintains the accuracy of existing speaker recognition techniques, while providing privacy to both parties of the surveillance system (the general public and the LEA). We applied our approach to two of the state-of-the-art speaker recognition methods (x-vector [5] and VGGVox [6]), but we opted for x-vector, because it offers a good trade-off between recognition accuracy and computational overhead.

The main contributions of this work can be summarized as follows:

- We propose an efficient protocol to securely compute the PLDA [7] score between two speakers' embeddings. The output reveals the binary result of the speaker identification operation, but makes it very difficult to deduce any information about the underlying embeddings.
- We leverage the computing capabilities of the distributed recording devices that participate in the surveillance network, to undertake the majority of the computational cost incurred by the added cryptographic layer. This improves the system's scalability significantly and allows the server to process audio surveillance tasks at a very high rate.
- We implemented a proof-of-concept voice surveillance system that employs one of the most accurate speaker recognition techniques. We also utilized a very efficient homomorphic encryption cryptosystem that allows us to identify a potential match in a matter of milliseconds. Our extensive experimental evaluation demonstrates the superior accuracy and efficiency of our system compared to the current state-of-the-art approaches.

The rest of this article is organized as follows. Section II presents a literature review on speaker recognition techniques and privacy-preserving speaker recognition protocols. Section III introduces the various tools that we have incorporated into our system, while Section IV presents the threat model in the context of the surveillance environment. Section V highlights the details of our protocol, and Section VI discusses its security. Sections VII and VIII present the system implementation details and the results of the experimental evaluation, respectively. Finally, Section IX concludes our work.

II. Related Work

A. Speaker Recognition

Speaker recognition aims at identifying a speaker, based on the features of his/her voice that are extracted from a recorded sample. The features may be both *text-dependent*, where the speaker is required to utter a specific phrase, or *text-independent*, where the system should be able to extract the characteristics of the speech regardless of the spoken text. In our specific application, a text-independent solution is necessary. The most widely used features in speaker recognition are Mel-frequency cepstral coefficients (MFCC) [8], linear predictive cepstral coefficients (LPCC), and perceptual linear prediction (PLP) [9]. Several studies (summarized in [10]) have compared the accuracy of different feature sets under diverse environments, and those based on MFCC and LPCC were the best choices overall.

Early speaker recognition methods relied on MFCC and Gaussian mixture models (GMM) [11], [12], [13]. More recently, several approaches have emerged that are based on embeddings. In 2011, Dehak et al. [14] introduced i-vector, a technique that utilizes probabilistic embeddings and was considered as the standard for speaker recognition. However, in the past few years, deep learning techniques have been applied in the field of speaker recognition, with very promising results. In particular, these techniques employ neural networks (NNs) to generate more representative and discriminative embeddings.

The most prominent NN-based frameworks are x-vector [5] and VGGVox [6]. These methods have been shown to achieve high recognition accuracy on several public audio datasets, including Voxceleb1 [15], Voxceleb2 [16], and SITW [17]. Both VGGVox and x-vector use compact feature vector representations. More precisely, the dimensionality of a VGGVox vector is 512, whereas the dimensionality of an x-vector is 1024, but can be reduced to 200 (using the linear discriminant analysis transformation) with a very small accuracy loss. To compare two embeddings, multiple distance metrics have been used in the literature, but PLDA has been shown to be very accurate. In the case of recordings with multiple speakers, diarization techniques [18] are employed to separate and label the speech signal corresponding to the identity of speakers. Diarization uses various segmentation and clustering algorithms, in order

to answer the question "who spoke when?"[19] in an audio segment.

B. Privacy-Preserving Speaker Recognition

Privacy-preserving protocols transform the existing speaker recognition algorithms to process inputs that are obfuscated either through encryption or with the addition of noise. The first privacy-preserving recognition systems were designed for speech recognition [20], [21], where one party holds a private speech data and the other party holds a private speech recognition model. Smaragdis et al. [20] employed secure two-party computation (STPC) protocols to securely evaluate hidden Markov models (HMM) and Gaussian mixture models (GMMs). The problem with this approach is that the HMM of the subject is sent to the server in cleartext, which reveals some information about the underlying speaker. Other studies by Pathak et al. [21], [22], [23] and Aliasgari and Blanton [24] are based on homomorphic encryption (HE) and secure two-party computation protocols, such as garbled circuits (GCs). All these methods employ GMMs or HMMs, but their main limitation is the high computational overhead. In particular, their results report hundreds of seconds to securely perform a single speech recognition operation [21].

Likewise, Portêlo et al. [25] introduced a protocol that leverages Yao's garbled circuits [26] on universal background model GMM (UBM-GMM). Their approach improves the computation overhead compared to the previous methods, reaching 300 s of response time with a single compute core. The work by Jimenez et al. [27] proposed a novel method, which utilizes a secure hashing algorithm that preserves the Euclidean distance between the feature vectors, if their distance is smaller than a certain threshold. Their solution relies on a somewhat-trusted third party, i.e., a party that does not necessarily need to be trusted in the absence of collusion with any of the other two parties. In 2018, Nautsch et al. [28] presented a privacy-preserving speaker recognition protocol that evaluates the 2Cov distance over the encrypted domain (using homomorphic encryption). The authors leveraged the Paillier cryptosystem, but their experimental results demonstrated a large communication and computation overhead.

Rahulamathavan et al. [29] proposed a privacy-preserving speaker verification protocol based on i-vectors. Their method leverages secret sharing to securely store the feature vectors on the server during the enrolment phase. Then, to compute the similarity score, the server and the client jointly invoke a two-party protocol to evaluate certain scalar products. The security of the latter protocol relies on randomization techniques, i.e., multiplying the feature vector values and the elements of the projection matrix by some large numbers. The proposed implementation is very efficient in terms of computational overhead, because it does not rely on any public-key cryptography operations. However, this method is designed to compute a cosine distance, and it is not suitable for more complex similarity metrics, such as PLDA. Moreover, the solution requires multiple rounds of communication to privately compute each scalar product. More importantly, this system has been proven insecure by Schneider et al. [30], where the authors conclude that it is infeasible to build privacy-preserving scalar product protocols without relying on at least some cryptographic assumptions like Homomorphic Encryption (HE) or Oblivious Transfer (OT).

Another work by Nautsch et al. [31] introduced a privacy-preserving speaker recognition framework based on Cohort score normalization. The framework leverages Kaldi's [32] i-vectors and achieves good accuracy (an equal error rate (EER) of 4.1%). Nevertheless, the privacy-preserving layer employs secure multi-party computation protocols, which incur a few minutes to compute the as-norm with cohort pruning.

More recently, Treiber et al. [33] proposed a privacy-preserving PLDA computation protocol. Their solution achieves high accuracy, and can be applied to both i-vectors and x-vectors. However, the shortcoming of this approach is that it requires two non-colluding servers, which is a strong assumption to make in a real-world application. The protocol relies on secure two-party computations, and involves multiple rounds of communication. In particular, the proposed implementation employs garbled circuits, where a pairwise comparison between two embeddings with dimensionality of 200 takes around 77 ms over a LAN network.

Finally, Brasser et al. [34] and Bayerl et al. [35] proposed an architecture that leverages a trusted execution environment (TEE) to build an efficient privacy-preserving speech recognition system, demonstrating real-time processing capabilities. However, TEEs necessitate specialized hardware that is not always available. More importantly, numerous vulnerabilities have been found in different TEE implementations [36] that are not easy to mitigate. In our work, we did not want to rely on the security of a third party hardware component. Instead, we built our system on top of well-known cryptographic primitives that are provably secure.

III. Tools

A. Speaker Identification With PLDA

Consider a dataset of N recordings $\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$. Each vector $\mathbf{x}^i$ is a column vector of length d that belongs to one of the K speakers (classes). C_k represents the set of all recordings of speaker k, and let n_k denote the cardinality of C_k. We also use $\mathbf{m}^k$ to represent the mean of class k and $\mathbf{m}$ to denote the mean of the entire dataset. Linear discriminant analysis (LDA) is a technique that generates feature vectors that maximize the between-class separation of data, while minimizing the within-class scatter. The within-class and between-class matrices are given as:

$$S_w = \frac{\sum_k \sum_{i \in C_k} \left(\mathbf{x}^i - \mathbf{m}^k\right)\left(\mathbf{x}^i - \mathbf{m}^k\right)^\mathsf{T}}{N}$$

$$S_b = \frac{\sum_k n_k \left(\mathbf{m}^k - \mathbf{m}\right)\left(\mathbf{m}^k - \mathbf{m}\right)^\mathsf{T}}{N}$$

The goal of LDA is to find a linear transformation $\mathbf{x} \rightarrow W^\mathsf{T}\mathbf{x}$ that maximizes the between-class variance relative to the within-class variance, where W is a $d \times d'$ matrix (d' is the desired number of dimensions). To this end, GMM can be used to fit the LDA projections. Even though the derived mixture model can

classify samples into the limited number of classes present in the training data (K classes), it is unable to assign samples to new, unseen classes.

Probabilistic LDA (PLDA) [7] addresses this problem with the observation that, members of the same class share the same covariance matrix Φ_w. As such, the class-conditional distribution can be written as

$$P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\mathbf{y}, \Phi_w)$$

where $\mathbf{x}$ is the example and $\mathbf{y}$ is the latent variable. And to allow for the handling of unseen classes, the latent variable $\mathbf{y}$ is defined as

$$P(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{m}, \Phi_b)$$

where Φ_b denotes the between-class covariance matrix.

A nice property of Φ_w and Φ_b is that both matrices can be made diagonal by setting $\Psi = V^{\mathsf{T}}\Phi_b V$ and $\mathbf{I} = V^{\mathsf{T}}\Phi_w V$, where Ψ is a diagonal matrix, $\Psi > 0$, and V is obtained by solving the generalized Eigen problem. Therefore, by defining $A = V^{-\mathsf{T}}$, it follows that $\Phi_w = AA^{\mathsf{T}}$ and $\Phi_b = A\Psi A^{\mathsf{T}}$. As such, the model can be defined as:

$$\mathbf{x} = \mathbf{m} + A\mathbf{u} \text{ where } \mathbf{u} \sim \mathcal{N}(\cdot|\mathbf{v}, \mathbf{I}) \text{ and } \mathbf{v} \sim \mathcal{N}(\cdot|0, \Psi)$$

In the above equation, $\mathbf{v}$ identifies the class and $\mathbf{u}$ represents an example from class $\mathbf{v}$. Recall that $\mathbf{m}$ is the global mean of all speakers in the training data. The goal now is to estimate, from the training data, both Φ_w and Φ_b or, equivalently, Ψ and A. These variables can be learned using the expectation maximization (EM) technique. The details of the EM derivation are given in [7].

Once the model parameters are learned, the PLDA score s between two given examples is computed via the log likelihood ratio, which is defined as

$$s = \frac{P(\mathbf{u}^p|\mathbf{u}^g_{1..n})}{P(\mathbf{u}^p)}$$

Here, $\mathbf{u}^p$ is a probe sample, and $\mathbf{u}^g$ is a sample from the gallery. Moreover, $\mathbf{u}^g_{1..n}$ denotes n independent samples from the same class g, where $g \in [1, K]$. Therefore, $P(\mathbf{u}^p|\mathbf{u}^g_{1..n})$ is the probability that $\mathbf{u}^p$ belongs to class g, i.e.,

$$P(\mathbf{u}^p|\mathbf{u}^g_{1..n}) = \mathcal{N}\left(\mathbf{u}^p|\tfrac{n\Psi}{n\Psi+I}\bar{\mathbf{u}}^g, I + \frac{n\Psi}{n\Psi+I}\right)$$

On the other hand, $P(\mathbf{u}^p)$ is the probability of $\mathbf{u}^p$ without a class assumption. It can be computed from the previous equation, by setting $n = 0$. Finally, the likelihood ratio is computed as

$$s = \frac{\mathcal{N}(\mathbf{u}^p|\tfrac{n\Psi}{n\Psi+I}\bar{\mathbf{u}}^g, I + \tfrac{\Psi}{n\Psi+I})}{\mathcal{N}(\mathbf{u}^p|0, I+\Psi)} \tag{1}$$

where $\bar{\mathbf{u}}^g$ is the mean of the individual class samples $\mathbf{u}^g_{1..n}$. In the Kaldi implementation, the log likelihood ratio is expanded as follows, where $\mathbf{m} = \frac{n\Psi}{n\Psi+I}\bar{\mathbf{u}}^g$.

$$s = -0.5\left[(\mathbf{u}^p - \mathbf{m})^{\mathsf{T}}\left(I + \tfrac{\Psi}{n\Psi+I}\right)^{-1}(\mathbf{u}^p - \mathbf{m}) + \text{logdet}\left(I + \tfrac{\Psi}{n\Psi+I}\right)\right]$$

$$+ 0.5\left[\mathbf{u}^{p\mathsf{T}}(I+\Psi)^{-1}\mathbf{u}^p + \text{logdet}(I+\Psi)\right] \tag{2}$$

In our work, we adapted the similarity score s from Equation (2), and we designed a protocol that computes it securely in the ciphertext domain.

B. Kaldi

Kaldi [32] is a C++ toolkit for speech recognition that has been widely used by the research community. It was launched in 2009 [37] as a subspace Gaussian mixture model (SGMM) tool. Since then, more scripts and algorithms have been added to the library. Its source code is open and easy to modify and extend. Currently, Kaldi supports multiple signal processing algorithms and acoustic models, including GMM, SGMM, and others. In terms of feature extraction, Kaldi implements the MFCC and PLP methods, with the ability to tune most of their parameters.

Furthermore, the library supports a large set of feature and model-space transformations and projections. Examples include LDA, frame splicing and delta feature computation, heteroscedastic linear discriminant analysis (HLDA), and others. More importantly, Kaldi employs deep neural networks (DNNs) for the most recent modeling techniques, such as the x-vector [5] embeddings. Most of Kaldi's code is designed to operate in a cluster, where multiple machines are leveraged during execution. Moreover, a CUDA library is available, which provides access to GPU-based operations. Finally, Kaldi includes a module to perform the diarization operation, which exhibits very good accuracy (the diarization error rate (DER) is equal to 8.39%). This is performed with x-vectors as embeddings and PLDA as the scoring metric. Note that, in a surveillance scenario where audio is acquired in public areas, the diarization operation would be more challenging and error-prone. Nevertheless, this is still an active area of research and is beyond the scope of this paper.

Most relevant to our work is Kaldi's x-vector extraction algorithm, which is based on DNN embeddings. The system starts by extracting the MFCCs of the available utterances. Then, a lower and upper cutoff frequencies are set close to zero and the Nyquist frequency, respectively (e.g., 20 Hz and 7800 Hz for 16 kHz sampled speech), in order to preserve the full information in the signal. After that, sliding-window cepstral mean normalization is applied to every utterance. The output then propagates through the x-vector neural network model to generate the corresponding embeddings. Once the x-vectors are generated, the global mean is subtracted, and the PLDA transformation is applied using the pre-trained PLDA model. This reduces the dimensionality of the x-vector from 1024 to 200. The reduced dimensionality is very important to our protocol, because it considerably decreases the overhead of the cryptographic operations.

Note that the x-vector neural network engine that we employed was trained with the combined VoxCeleb dataset (1 and 2) [6], which contains over 1 million utterances from over 6,000 celebrities. We should emphasize that we did not train our own x-vector model, but rather utilized the one that is available on the VoxCeleb website. The characteristics of this dataset are given in Table I. In addition to the VoxCeleb datasets, the developers

TABLE I
DATASET STATISTICS FOR THE VOXCELEB DATASETS [6]

Dataset	VoxCeleb1	VoxCeleb2
Number of persons	1,251	6,112
Number of male persons	690	3,761
Number of hours	352	2,442
Number of utterances	153,516	1,128,246
Average number of utterances per person	116	185
Average length of utterances (s)	8.2	7.8

of x-vector used both Musan [38] and RIR_noises [39] datasets for data augmentation.

C. Homomorphic Encryption

Homomorphic cryptosystems [40] allow for the evaluation of certain arithmetic operations directly on the ciphertext domain. A homomorphism is a structure-preserving map between two algebraic structures of the same group [41]. A function $f : G \rightarrow H$ is a homomorphism for groups $(G, \square)$, $(H, \diamond)$ with sets G, H and operators $\square$, $\diamond$, respectively, if:

$$f(g \square g') = f(g) \diamond f(g') \qquad \forall (g, g') \in G^2$$

A cryptosystem is said to be homomorphic, if $\mathsf{Enc}(m_1 \square m_2) = \mathsf{Enc}(m_1) \diamond \mathsf{Enc}(m_2)$, with messages m_1 and m_2 from the clear-texts domain, and an encryption function Enc.

Two types of homomorphic encryption exist. The first one is fully homomorphic encryption (FHE) [42], which supports an arbitrary number of addition and multiplication operations. As such, FHE allows one to evaluate any complex circuit, solely on the ciphertext domain. Nevertheless, the problem with FHE systems is their inefficiency in terms of computation and storage requirements, which renders them impractical in real-time applications (like surveillance). The second type of homomorphic encryption is partially homomorphic encryption (PHE). PHE cryptosystems allow only one of the basic operations, either addition or multiplication, and are, thus, more restrictive in the types of circuits that they can evaluate. Typical examples are Paillier's [43] cryptosystem, which is an additive-PHE scheme, and RSA [44], which is a multiplicative-PHE scheme. In our protocol, we opted for the implementation of ElGamal's cryptosystem [45] over elliptic curves, which is an additive-PHE system. Specifically, this scheme is very efficient in terms of computations, and the ciphertext representation necessitates only 128 bytes. The operations of the cryptosystem are as follows:

- *Key generation:* Instantiate an elliptic curve group of prime order q with generator P. Choose a *private* key x uniformly at random from $\mathbb{Z}_q^*$ and set the *public* key $Q = x \cdot P$.
- *Encrypt:* Let m be the secret message. Choose r uniformly at random from $\mathbb{Z}_q^*$ and compute ciphertext $\mathsf{Enc}(m) = \langle r \cdot P, (m + r) \cdot Q \rangle$.
- *Decrypt:* Compute $m \cdot Q = (m + r) \cdot Q - x \cdot r \cdot P$ and solve the discrete log to recover m.

ElGamal's encryption scheme produces indistinguishable ciphertexts, and its security is based on the decisional Diffie-Hellman assumption. However, a notable limitation is that the decryption operation requires solving an instance of a discrete logarithm problem, which is a computationally hard task (for large messages). Therefore, in our implementation, we utilized a lookup table of precomputed $m \cdot Q$ values (for all theoretically possible values of m, under our protocol specifications), in order to speed up the decryption operation at the server.

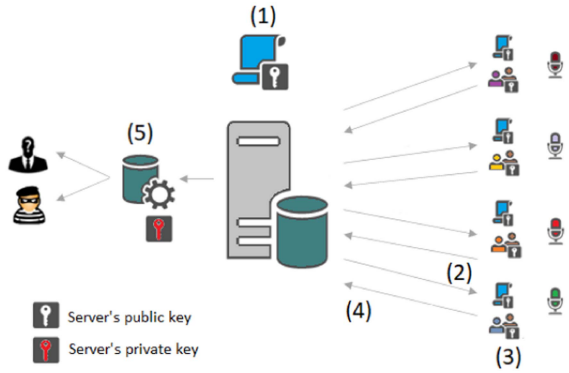


Fig. 1. System overview.

The homomorphic properties of ElGamal's cryptosystem over elliptic curves are shown below.

- *Homomorphic addition:* Given the encryption of two messages m_1 and m_2, $\mathsf{Enc}(m_1) + \mathsf{Enc}(m_2)$ is equal to

$$\begin{aligned} &\langle r_1 \cdot P, (m_1 + r_1) \cdot Q \rangle + \langle r_2 \cdot P, (m_2 + r_2) \cdot Q \rangle \\ &= \langle (r_1 + r_2) \cdot P, (m_1 + m_2 + r_1 + r_2) \cdot Q \rangle \\ &= \mathsf{Enc}(m_1 + m_2) \end{aligned}$$

- *Homomorphic scalar multiplication:* Given a plaintext scalar λ and the encryption of a message m, $\lambda \cdot \mathsf{Enc}(m)$ is equal to

$$\langle \lambda \cdot r \cdot P, (\lambda \cdot m + \lambda \cdot r) \cdot Q \rangle = \mathsf{Enc}\,(\lambda \cdot m)$$

IV. SYSTEM AND THREAT MODEL

We consider an audio surveillance system with M suspects, as shown in Fig. 1. Each suspect is represented by the mean of his or her x-vectors, $\mathbf{m}^i$, where $i \in \{1, 2, \ldots, M\}$. Our main assumptions and privacy goals are listed below:

1) The suspects' plaintext reference information (biometric data) is available only to the law enforcement agency (LEA) and is not shared with other entities. For example, the LEA may generate the reference information from the suspects' voice recordings that are collected through various means. This is different from speaker recognition systems, where users intentionally enroll their biometric data to the server (encrypted with their own public keys) for authentication purposes.
2) The suspects' reference information should remain secret at the listening devices, by encrypting them with the LEA's public key. Indeed, these devices are located in public areas and they are susceptible to physical attacks by hackers that could expose the suspects' database. Furthermore, the devices could be operated by a third party that is not

associated with the LEA. As such, the devices have no knowledge of the LEA's private decryption key.

3) The matching result is revealed to the LEA. If the similarity score between a recording and a suspect is below a certain threshold, the LEA will learn the ID of the matched suspect.
4) The server (LEA) should not be able to reconstruct the feature vector of any speaker that was captured by the listening devices. That is, the server should remain oblivious to the identity of the recorded individuals.

As shown in Fig. 1, the M samples are generated by the server (Step 1), and distributed to the surveillance devices in encrypted form (Step 2), using the server's public key. This guarantees goals 1 and 2 listed above. The surveillance device intercepts the surrounding voices, and generates the corresponding x-vectors, $\mathbf{u}_j^p$, for each captured speaker j. Next, the device computes the similarity scores between $\mathbf{u}_j^p$ and every $\mathbf{m}^i$ in the suspects' list (Step 3), and sends the results back to the law enforcement server (Step 4). The similarity scores are both obfuscated and permuted, which satisfies goal 4, because the server is unable to solve the underlying system of equations that would reveal the speaker's reference information. The server finally decrypts the scores and reveals the result of the identification operation (Step 5), which meets goal 3.

Note that the server is the only party with knowledge of the secret decryption key, so the surveillance devices (or any eavesdropper) are unable to decrypt the exchanged messages. As such, it is not necessary to establish a secure channel between the devices and the server, however, TLS should be used to (i) authenticate all participating entities; and (ii) prevent messages from being altered (due to HE, an attacker might modify certain ciphertexts using the server's public key).

In this work, we only consider semi-honest adversaries. A semi-honest adversary is a passive attacker who follows the protocol specification, but is curious. For instance, a semi-honest server will analyze the replies from the recording devices, in order to identify speakers that are not present in the suspects' database (thus tracking innocent individuals). On the other hand, a semi-honest device tries to reveal the identity of one or more individuals in the server's database that did not trigger an alert locally. Finally, a semi-honest third party is simply an eavesdropper that sees all the messages exchanged between the server and the surveillance devices. In this case, the adversary may try to infer any non-trivial information about the underlying individuals. It is worth noting that, neither our system nor any other privacy-preserving speaker identification protocol, is immune to unlawful inputs. For example, a malicious server may add the feature vector of a lawful citizen in the database, in order to track his/her movements. Whereas, a malicious recording device may send the scores corresponding to a specific individual, thus verifying whether he/she is part of the server's database.

V. Privacy-Preserving Audio Surveillance

In our approach, we follow the general framework proposed recently by Bentafat et al. [46]. Specifically, our protocol consists of an offline phase (performed once, during the system's initialization) and an online phase. The online phase involves (i) the similarity score computation; (ii) the similarity score obfuscation; and (iii) the matching operation. Before presenting the details of the two phases, we will first introduce the quantization operation, which converts the floating-point representation of the feature vector to an integer representation. This is necessary because public-key homomorphic cryptosystems can only operate on integer numbers.

A. Floating-Point Quantization

Our goal is to convert the x-vector embeddings into integer values, while incurring a negligible accuracy loss. To do so, we first analyzed the x-vectors generated by existing large-scale speaker datasets. We noticed that the generated embeddings range between -10 and 9, so we applied the following transformation.

$$u_i' = \min\left(\lfloor \alpha u_i + \beta \rceil, 255\right) \tag{3}$$

In the above equation, u_i is a coefficient from the speaker's feature vector ($1 \leq i \leq d$, and d is the vector's dimensionality), u_i' the quantized coefficient, and $\alpha = 13.5$ and $\beta = 135$ are two constants. This transformation converts an x-vector floating-point coefficient $u_i \in \mathbb{Q}, u_i \in [-10, 9]$ into an integer coefficient $u_i' \in \mathbb{Z}, u_i' \in [0, 256)$. In the remainder of this paper, we use *primed* variables to denote quantized (integer) values, and normal variables to denote floating-point values.

In addition to quantizing the individual x-vectors ($\mathbf{u}^p$ and $\mathbf{m}$), the computation of the PLDA-based similarity score s, as depicted in (2), necessitates the quantization of some additional terms in that equation. First, it is worth noting that, to compare two given embeddings, we do not need to compute the exact value of the log likelihood ratio. Instead, we need to compare that ratio to a given threshold that is suggested by the adopted model (the value of the threshold may change over time, if needed). Thus, we can drop the constants and the scaling factors in (2), and simplify it as

$$s = -\Xi\left(\mathbf{u}^p - \mathbf{m}\right)^2 + \Theta(\mathbf{u}^p)^2 \tag{4}$$

where $\Xi = (I + \frac{\Psi}{n\Psi + I})^{-1}$, $\Theta = (I + \Psi)^{-1}$, and $\mathbf{m} = \frac{n\Psi}{n\Psi + I}\bar{\mathbf{u}}^g$. This is possible, because matrix Ψ is diagonal and could be interpreted as a row-vector, such that $\Psi = (\psi_1, \psi_2, \ldots, \psi_d)$ and, therefore, $\Xi = (\xi_1, \xi_2, \ldots, \xi_d)$ and $\Theta = (\theta_1, \theta_2, \ldots, \theta_d)$. Note that, Ψ is given by the training model and we can, thus, compute the floating-point row-vectors Ξ and Θ.

Since we are only interested in whether the similarity score s is below or above a certain threshold τ, we can use the following equation instead:

$$\begin{aligned} s \gtrless \tau \iff & -\Xi(\mathbf{u}^p - \mathbf{m})^2 + \Theta(\mathbf{u}^p)^2 \gtrless \tau \\ \iff & -\Xi(\alpha\mathbf{u}^p + \beta I - \alpha\mathbf{m} - \beta I)^2 \\ & + \alpha^2\Theta(\mathbf{u}^p)^2 - \alpha^2\tau \gtrless 0 \end{aligned}$$

TABLE II
ACCURACY COMPARISON (EER) BETWEEN DIFFERENT SPEAKER RECOGNITION METHODS AND THEIR QUANTIZED VERSIONS [6]

Method	Original	Quantized	Dim.
X-vector [5]	3.128%	3.600%	200
VGGVox [6]	2.87%	3.213%	512

After expanding and rounding the floating-point values of the left-hand side of the latter inequality, we get:

$$s' = \left\lfloor \sum_{i=1}^{d} \left(\alpha^2 \theta_i u_i^{p2} - \xi_i u_i^{p'2} \right) - \alpha^2 \tau \right\rceil + \sum_{i=1}^{d} 2 \lfloor \xi_i m_i' \rceil u_i^{p'} + \left\lfloor \sum_{i=1}^{d} -\xi_i m_i'^2 \right\rceil \quad (5)$$

where s' is the quantized threshold. As shown above, $s \gtrless \tau$ is almost equivalent to $s' \gtrless 0$, except for the errors that may be caused by the rounding operations, which we will quantify shortly. More importantly, we try to perform the rounding operations only on aggregated terms, in order to get an approximation as close as possible to the real score. Note that, s' is an integer number and can be computed over the ciphertext domain without intermediate decryption operations. Furthermore, a *positive* speaker identification result is signified when $s' < 0$.

1) Quantifying the Accuracy Loss: Most modern speaker verification and identification protocols utilize the equal error rate (EER) as a metric to assess the accuracy of their models [47]. The EER is derived from the PLDA score τ, by computing the following values:

$$P_{fp}(\tau) = \frac{\text{\# impostor trials with score} > \tau}{\text{\# total impostor trials}} \times 100\%$$

$$P_{fn}(\tau) = \frac{\text{\# target trials with score} \leq \tau}{\text{\# total target trials}} \times 100\%$$

where $P_{fp}(\tau)$ and $P_{fn}(\tau)$ represent the false positive and false negative rates, respectively. The EER corresponds to the best threshold value τ_{EER}, at which the two error rates are equal, i.e., $P_{fp}(\tau_{\text{EER}}) = P_{fn}(\tau_{\text{EER}})$. Table II illustrates the accuracy loss when using the original vs. the quantized version of the PLDA score, i.e., (4) vs. (5). The results were obtained from the publicly available dataset VoxCeleb,[1] using the VoxCeleb1 test split that contained 4,874 utterances from 40 different speakers.

In particular, we run the quantization step on the two state-of-the-art methods for speaker recognition, namely x-vector and VGGVox. For x-vector, we applied the quantization shown in (3), whereas for VGGVox, we used slightly different values for α and β, based on the maximum and minimum values of the underlying feature vectors coefficients. The goal was to keep the values of the quantized vector within the range [0,256), in order to represent them using a single byte. It is clear that the VGGVox model is more accurate than x-vector. Nevertheless, in our work, we opted for the implementation of the x-vector, because of its compact embedding vectors (200 vs. 512) that reduce the online computation cost of the cryptographic operations by a factor of $\times 2.5$. It is worth noting that the accuracy loss for both models is less than 0.4%, which is an acceptable trade-off for incorporating privacy-preserving mechanisms into a surveillance system.

[1] [Online]. Available: http://www.robots.ox.ac.uk/\;vgg/data/voxceleb/

TABLE III
PERFORMANCE COMPARISON BETWEEN THE ORIGINAL AND QUANTIZED X-VECTOR SYSTEMS

X-vector	min_{DCF} $_{P_{Target}=0.01}$	min_{DCF} $_{P_{Target}=0.001}$	C_{llr}	C_{llr}^{min}
Original	0.3492	0.5452	0.1200	0.1139
Quantized	0.4055	0.5859	0.1380	0.1297

To further analyze the accuracy loss of the quantized x-vector method, we measured two other metrics, as suggested in the *NIST 2016 Speaker Recognition Evaluation Plan*.[2] These metrics are (i) the minimum detection cost function (min_{DCF}); and (ii) the log-likelihood ratio cost (C_{llr}). Specifically, min_{DCF} is the value that minimizes the detection cost, which is given as:

$$C_{Norm} = \frac{C_{Det}}{C_{Default}} \quad (6)$$

with:

$$C_{Det} = C_{Miss} \times P_{Target} \times P_{Miss|Target} + C_{FA} \times (1 - P_{Target}) \times P_{FA|NonTarget}$$

and:

$$C_{Default} = \min \begin{cases} C_{Miss} \times P_{Target}, \\ C_{FA} \times (1 - P_{Target}) \end{cases}$$

where C_{Det} is the detection cost, $C_{Default}$ is the best cost without processing the data, and C_{Norm} is the normalized detection cost. C_{Miss}, C_{FA}, and P_{Target} are the parameters of the cost function, i.e., the cost of missed detection, the cost of a spurious detection, and the a priori probability of the specified target speaker, respectively.

On the other hand, C_{llr} [48] is a representation of how well the scores reflect the likelihood ratio and penalizes for errors in score calibration. It is defined as:

$$C_{llr} = \frac{1}{2 \times \log(2)} \times \left(\frac{\sum \log(1 + 1/s)}{N_{TT}} + \frac{\sum \log(1 + s)}{N_{NT}} \right) \quad (7)$$

where N_{TT} are the target trials, N_{NT} are the non-target trials, and s represents a trial's likelihood ratio. Moreover, the minimum log-likelihood ratio cost (C_{llr}^{min}) can be interpreted as a measure of discriminating power, which can be used as a discriminating metric.

Table III summarizes our findings. It suggests that the accuracy loss due to the quantization operations is quite low.

B. Offline Phase

In a nutshell, the offline phase allows the server to enroll the suspects' encrypted reference information into the system. The server first extracts the feature vectors from the available

[2] [Online]. Available: https://www.nist.gov/itl/iad/mig/speaker-recognition-evaluation-2016

suspects' utterances, encrypts them, and distributes them to all the surveillance devices. After that, all parties precompute a series of cryptographic values that allow the system to perform the online surveillance phase in a real-time fashion.

1) Encrypted Database: On the server side, the offline phase begins by extracting the feature vectors from all the suspects' recordings. The identification accuracy of the system depends on the quality of the available data, in terms of utterance duration and number of utterances per suspect. For example, as depicted in Table I, the VoxCeleb dataset includes, on average, over 100 utterances per person, each lasting an average of 8 seconds. Given the set of feature vectors, the server builds the corresponding encrypted database that is distributed to the surveillance devices. The database consists of the server's public key and all the ciphertexts that are necessary for computing the encrypted similarity score. More precisely, the encrypted similarity score s' is computed from (5) as follows:

$$\mathsf{Enc}(s') = \mathsf{Enc}\left(\left\lfloor \sum_{i=1}^{d}\left(\alpha^2\theta_i u_i^{p2} - \xi_i u_i^{p'2}\right) - \alpha^2\tau \right\rfloor\right) + \sum_{i=1}^{d} \mathsf{Enc}\left(2\lfloor \xi_i m_i' \rceil\right) u_i^{p'} + \mathsf{Enc}\left(\left\lfloor \sum_{i=1}^{d} -\xi_i {m_i'}^2 \right\rfloor\right) \tag{8}$$

Therefore, for every suspect j, the server computes ciphertexts $\mathsf{Enc}(\lfloor \sum_{i=1}^{d} -\xi_i {m_i'}^2 \rfloor)$ and $\mathsf{Enc}(2\lfloor \xi_i m_i' \rceil), \forall i \in [1, d]$, and adds them to the encrypted database.

2) Device Precomputations: Once a surveillance device receives the encrypted database, it engages in a series of precomputations that aim to reduce the online cost of the cryptographic operations. Looking back at (8), the end device has to compute—for every detected speaker (and suspect)—the first and second terms of the encrypted score. Also note that, the only unknown variables in those terms are the original and quantized versions of the speaker's embeddings, i.e., $\mathbf{u}^p$ and $\mathbf{u}^{p'}$, respectively.

Starting from the first term, the device has to encrypt the value $\lfloor \sum_{i=1}^{d}(\alpha^2\theta_i u_i^{p2} - \xi_i u_i^{p'2}) - \alpha^2\tau \rfloor$, after it derives the new speaker's embedding $\mathbf{u}^p$. Notice that, all the remaining values are constants and, therefore, the device can precompute the ciphertexts of all possible integer outcomes, since it is known that $u_i^p \in [-10, 9], \forall i \in [1, d]$. Substituting the min and max values for all involved variables, we computed that the largest integer outcome of this term is $< 10.5 \times 10^6$, i.e., the required storage space for the entire range is approximately 1.27 GB. Nevertheless, for a device with very low storage capabilities, we can avoid this cost by encrypting only a logarithmic-size set of the range, at the expense of some additional computations. More specifically, we can simply store the ciphertexts of all powers of 2 within that range, i.e., $\mathsf{Enc}(2^i), \forall i \in [0, 30]$. Therefore, during the online phase, the surveillance device can compute the ciphertext of any possible outcome with, at most, 30 point addition operations (which are significantly cheaper than encryption operations).

Regarding the second term of (8), the end device precomputes, for every suspect's x-vector $\mathbf{m}'$, the values $\mathsf{Enc}(2\lfloor \xi_i m_i' \rceil) u_i^{p'}, \forall i \in [1, d]$. Note that, $u_i^{p'} \in [0, 256)$, so the total number of ciphertexts computed in this step is $256 \cdot d \cdot M$. Again, if the end device lacks the capability to store the entire set of ciphertexts, it can store a smaller subset with a minimal loss of efficiency. This is due to the fact that, some x-vector coefficients tend to occur more frequently than others. This is noticeable in Fig. 2 where we plot the distribution of the x-vector coefficients of more than 1000 random utterances. In this case, the surveillance device can precompute, for example, 50% of the values that fall in the range [76,204), or 30% that fall in the range [100,185). During the online phase, if the derived $u_i^{p'}$ is not part of the precomputed subset, the device performs the point-scalar multiplication operation on the spot.

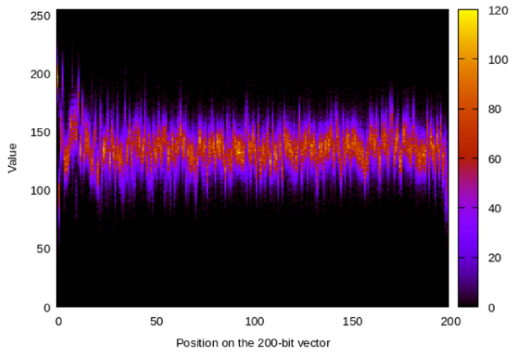


Fig. 2. Heat map of feature vector coefficients for 1,002 random utterances.

3) Server Precomputations: In a large-scale surveillance network, the server is clearly the performance bottleneck, because it has to interact with possibly tens of thousands of surveillance devices. As such, we need to optimize the server's computational tasks to the extent possible. To this end, the server's role during the online phase is to decrypt M ciphertexts for every voice sample captured by the end devices. But, as explained in Section III-C, the decryption operation involves solving an instance of a discrete log problem, which is computationally expensive. In particular, given a point on the elliptic curve $s' \cdot Q$, where Q is the server's public key and s' is the quantized similarity score, solve for s'. To speed up the decryption operation, the server precomputes all the possible outcomes of $s' \cdot Q$. More precisely, the server stores all the points on the elliptic curve that correspond to integers $s' \in [0, 2^{30})$. Note that we can obtain the maximum value of s' by choosing two vectors with maximal dissimilarity in (5), e.g., $\mathbf{u}^{p'} = (0, \ldots, 0)^\mathsf{T}$ and $\mathbf{m}' = (255, \ldots, 255)^\mathsf{T}$. This gives us an upper bound of $s' < 2^{22}$. However, in practice, having the maximal dissimilarity is very improbable, as evident from the distribution of the x-vector coefficients (Fig. 2). Instead, we analyzed the reported similarity scores empirically, and realized that the maximum reported score is $s' < 2^{17}$. This allows us to utilize the remaining 13 bits to *obfuscate* the real score s', as explained in Section V-C2.

It is important to note that, unlike most other privacy-preserving systems, the offline phase is not invoked for each utterance captured by the devices. Instead, it is performed only once before the system goes live.

C. Online Phase

The online phase constitutes the actual surveillance operation, where the listening devices record the utterances from the surrounding area. For every captured utterance, the device computes an obfuscated similarity score for each suspect in the encrypted database. The computations are done entirely in the ciphertext domain, using the public key of the law enforcement agency. The obfuscated scores are then randomly permuted and forwarded to the server. Finally, the server simply decrypts the received scores, using its private key, and detects any possible matches. Note that, the suspect identification accuracy depends on the quality of the recorded utterance, e.g., its duration, the level of background noise, etc.

1) Score Computation: Once the server and all surveillance devices have precomputed the necessary ciphertexts, the system is ready to perform voice-based surveillance in real time. To this end, each surveillance device continuously analyzes all the captured utterances and classifies them into different speaker classes. Whenever a new speaker is detected, the device computes the similarity scores $s'_j, \forall j \in \{1, 2, \ldots, M\}$, between the x-vector of the new speaker and each of the M x-vectors in the server's database. The scores are computed as shown in (8). Note that the end device has already precomputed or received all the terms of the equation, so it simply has to perform $M \times (d + 2)$ point addition operations.

2) Score Obfuscation: To hide the real similarity score from the database server, the end device selects two random numbers, r_1 and r_2, and obfuscates the score as $\delta_j = r_1 \cdot s'_j + r_2$. In the ciphertext domain, this is computed as

$$\mathsf{Enc}(\delta_j) = r_1 \cdot \mathsf{Enc}(s'_j) + \mathsf{Enc}(r_2) \tag{9}$$

As mentioned previously, r_1 and r_2 are 13-bit integers and, to maintain the signs of s'_j and δ_j identical, we always choose $r_1 > r_2$. By doing so, the obfuscation step does not have any impact on the accuracy of the model. (Recall that a positive match has occurred when $s'_j < 0$.) It is worth noting that the surveillance devices can also precompute $\mathsf{Enc}(r_2)$ during the offline phase. As such, the online cost of the obfuscation step is reduced to M point multiplications and M point additions. When the device completes the obfuscation of all M similarity scores, it applies a random *permutation*, and sends the permuted set of ciphertexts to the database server.

3) Matching: The final step is for the server to decrypt the M ciphertexts and determine the suspect identification result. In particular, if all M scores are positive, the captured voice does not match any suspect in the server's database. In this case, the protocol terminates. On the other hand, if one of the scores is negative, the protocol continues with an additional verification step. Specifically, to discourage a malicious server from claiming incorrectly that a certain recording produced a positive identification match, the server has to prove that a specific ciphertext produced a negative score. To this end, we utilize Schnorr's identification protocol [49] for proving (in zero knowledge) knowledge of a discrete log computation. We also employ the Fiat-Shamir heuristic to make the protocol non-interactive. Our protocol works as follows:

1) The server sends to the end device the position j of the negative score in the permutation and its exact value s'_j.
2) The device produces a *new* ciphertext for the claimed s'_j and homomorphically subtracts it from the stored ciphertext. If the server is honest, the resulting ciphertext is an encryption of zero, i.e., it is equal to $\langle r \cdot P, r \cdot Q \rangle$, for some unknown r. The device sends $G = r \cdot P$ to the server.
3) The server selects a uniformly random $t \in \mathbb{Z}_q^*$ and sends back to the device the triplet
$$\langle Y, c, e \rangle = \langle t \cdot G, H(Y), t + c \cdot x \bmod q \rangle$$
where $H(\cdot)$ is a secure hash function, x is the server's private key, and q is the order of the elliptic curve group.
4) The device verifies that $e \cdot G = Y + c \cdot r \cdot Q$, where Q is the server's public key.

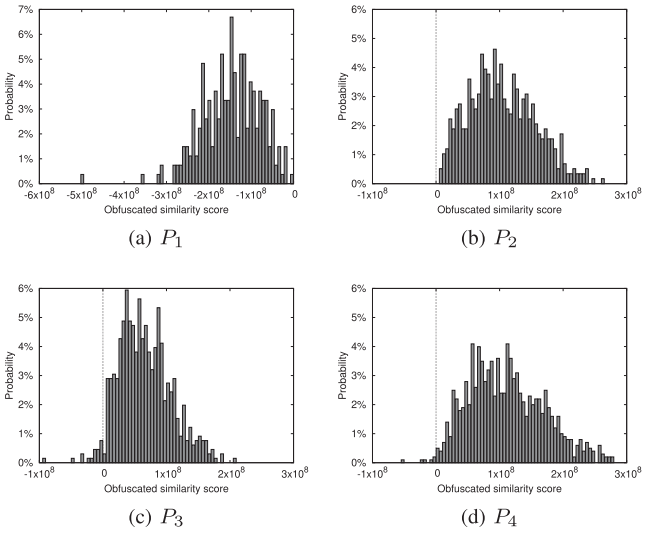
If the last equation is true, it means that x is the solution of the discrete log of $r \cdot Q$ base $r \cdot P$, i.e., the ciphertext at the device is indeed an encryption of zero. When the protocol execution terminates, the capturing device deletes immediately the recording as well as the computed scores. Finally, we should note that the matching operation (without the verification protocol that is rarely invoked) involves M point additions, M point multiplications, and M lookup operations per captured recording.

VI. Security

The security of our system stems mainly from the random permutation that is applied to the computed scores prior to them being sent to the server. As such, it is infeasible for the server to solve the correct set of M linear equations, in order to retrieve the coefficients of the speaker's x-vector. (There are a total of $M!$ possible outcomes for the permutation, making the problem exponentially hard.) Nevertheless, there are two additional features in our system that make the server's task even more difficult. The first one is the score obfuscation, which hides the real similarity scores that were computed at the end device. As such, even with the correct permutation, there is still a lot of uncertainty in computing the speaker's x-vector coefficients.

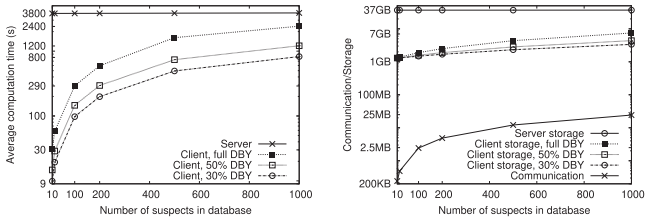
More importantly, a great source of randomness that significantly strengthens the security of our system, is the distribution of the x-vector coefficients under Kaldi's neural network engine. To illustrate this point, we plotted the distribution of the obfuscated similarity scores for a large number of utterances that belong to 4 different speakers (Fig. 3). The data was collected from the VoxCeleb1 [15] dataset. In particular, we chose the speakers with the largest number of recordings, which correspond to speaker IDs 10018, 10020, 10096, and 10986 (P_1, P_2, P_3, and P_4, respectively). The main observation is that there is a large overlap between the generated scores from different speakers. As a result, the server's task of inverting the permutation becomes significantly harder, because the individual scores can not be tied to specific speakers in the server's database.

Regarding the ISO/IEC 24745:2011 standard requirements, from the perspective of ordinary citizens, the collected biometric data (x-vectors) are never shared with the law enforcement servers. In addition, as mentioned above, it is infeasible to



(a) P_1 (b) P_2 (c) P_3 (d) P_4

Fig. 3. Distribution of obfuscated similarity scores for speaker P_1 against a set of utterances from speakers P_1, P_2, P_3, P_4.



(a) Computation time (b) Communication/Storage

Fig. 4. Offline cost.

reconstruct the x-vectors from the reported similarity scores, which guarantees the irreversibility property. This also guarantees unlinkability because, given two sets of permuted and obfuscated scores, it is infeasible to tell whether they belong to the same individual. Finally, the renewability property is not required for the x-vectors extracted from individuals, since they are not enrolled into the system. Instead, these x-vectors are deleted from the devices immediately after the computation of the similarity scores.

From the perspective of the suspects' biometric data, these properties are guaranteed due to the underlying encryption with ElGamal's cryptosystem. Specifically, the surveillance devices lack knowledge of the server's private key, which makes it infeasible to reconstruct the suspects' x-vectors (irreversibility). Likewise, encryption prevents the listening devices from distinguishing between two given encrypted feature vectors (unlinkability). Finally, if needed, the server can generate and redistribute a new version of the encrypted database to the surveillance devices, which guarantees renewability.

It is worth mentioning that, under our system, the plaintext PLDA parameters must be shared between the various entities. While this does not violate the privacy goals of our system (note that we only protect the privacy of the law enforcement database and the recorded individuals), it merits some further investigation. For example, the PLDA parameters could be based on proprietary speaker recognition models that the owners do not wish to share with the public. To this end, in our future work, we will investigate whether our methods can be extended to the case where the PLDA parameters are kept secret from the surveillance devices.

VII. System Implementation

We implemented our system on two machines, one to emulate the law enforcement server and the other to emulate one surveillance device. The server is a Ubuntu desktop machine with Intel Xeon CPU E5-2620 2.10 GHz$\times$ 16, 64 GB of RAM, and a 512 GB SSD. The other machine is a Ubuntu laptop with Intel Core i7-6500U CPU 2.50 GHz$\times$4 and 8 GB of RAM (it is also equipped with a microphone). The two machines are connected via a TCP/IP4 LAN over Gigabit Ethernet.

The speaker recognition subsystem was implemented with the Kaldi library. Both the client and the server extract the x-vectors of each speaker using the pre-trained model that is available online.[3] In addition to the trained neural network, the PLDA backend for scoring x-vectors is also available online. The server's input is a directory containing a set of folders, each named after the speaker's unique ID. Each speaker folder contains a number of utterances in . *wav* format. We run our experiments using a different number of speakers, which represents the size of the server's database, ranging from $M = 10$ to 1000. On the client-side, the input is a single recording in . *wav* format, but typically it can be a live stream from the device's microphone.

The cryptographic layer (ElGamal over elliptic curves) of our system was implemented in C/C++, using the BIGNUM library of OpenSSL (version 1.1.0g). We also used SWIG to connect C with Python (version 4.0.1). We set the order of the elliptic curve to be a 256-bit prime number, as per NIST's recommendations [50], which is secure against advances in computing power until 2030 and beyond. Under this environment, a ciphertext is represented by two points on the curve, which corresponds to 128 bytes of storage/communication. The average time for encryption, decryption, and point multiplication (with 256-bit scalars) is about 0.23 ms. On the other hand, point addition takes only about 0.02 ms. Finally, our implementation leverages the parallel computing abilities of the two multi-core machines, since all our algorithms are easily parallelizable. The source code of our implementation is available on GitHub.[4]

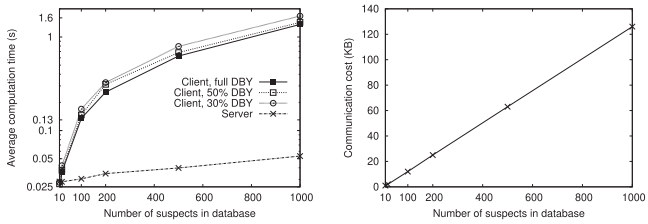
VIII. Experimental Results

In this section, we experimentally evaluate the overhead of our surveillance system in terms of computation and communication/storage costs. When measuring the computation overhead, we performed the experiment 4 times and plotted the average time. Fig. 4 depicts the offline cost for both the recording device and the server, as a function of the database size M. First, the server's CPU time is dominated by the generation of the

[3][Online]. Available: https://kaldi-asr.org/models/m7
[4][Online]. Available: https://github.com/mahdihbku/PPSpeakerRecognition

TABLE IV
CHARACTERISTICS OF VARIOUS PRIVACY-PRESERVING SPEAKER RECOGNITION PROTOCOLS

Method	Designed for	Speaker reco. method	Distance metric	Crypto primitives	Computation time	Reference protection
Pathak et al. [22]	Veri/Identi	UBM-GMM	Log likelihood	HE	82 min	Server
Portêlo et al. [25]	Verification	UBM-GMM	Log likelihood	GC	300 s	Server
Nautsch et al. [28]	Verification	i-vector	PLDA/2Cov	HE	241.9 s	Server
Treiber et al. [33]	Verification	i-/x-vector	PLDA/2Cov	STPC	77 ms	Server
Ours	Identification	x-vector	PLDA	HE	165 ms (100 suspects)	Client



(a) Computation time (b) Communication

Fig. 5. Online cost.

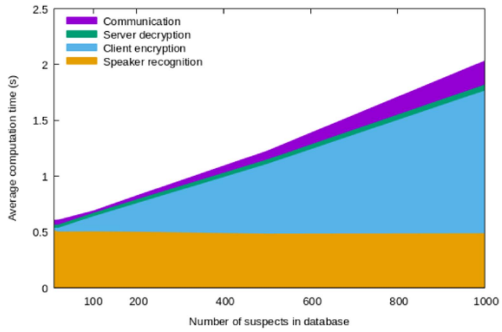


Fig. 6. Round-Trip Time for a positive match.

discrete log lookup table. (The cost of database encryption is, by comparison, negligible.) However, we should emphasize that this is a one-time cost that is incurred before the system goes online. For the surveillance device, the offline cost depends on the fraction `dby_portion` of the precomputations that are computed and stored on the device. Recall that, the required precomputations involve the first two terms of (8). Similar to the case of the server, the offline computations are performed only once.

Fig. 4(b) shows the corresponding storage requirements at both parties. The server necessitates approximately 37 GB of storage for the decryption lookup table, which is quite trivial for off-the-shelf servers. For $M = 1000$, the recording device requires around 6.3 GB and 2.1 GB when storing the full `dby` (`dby _ portion` $= 1.0$) and 30% of the `dby` (`dby_portion` $= 0.3$), respectively. Note that, `dby` corresponds to the second term of (8), although, there is an additional (fixed) cost of 1.3 GB for the first term, which is independent of M. The latter could be avoided at the expense of a slight increase in online computations, as explained in Section V-B. Finally, the offline communication cost entails the transfer of the encrypted database to the surveillance device.

Fig. 5 depicts the online cost of our protocol, i.e., the cost incurred during the surveillance operation for matching a captured speaker against the suspects' database. The most important observation is that the computation time is near real-time. For a database of 100 suspects, the cost is just 135 ms and 30 ms on the device and the server, respectively, while, for 1000 suspects, the corresponding times are 1.3 s and 53 ms. For the recording device, the aforementioned costs assume that the device stores 100% of the necessary precomputations. However, even when storing just 30% of the ciphertexts, the online cost is increased by only 23%. This is due to the fact that the device stores the ciphertexts of x-vector coefficients that are more likely to occur, as explained in Section V-B. In terms of online communication overhead, our protocol is extremely efficient. As evident in Fig. 5(b), for a database of 1000 suspects, the device has to transmit only 126 KB of data per captured speaker.

Fig. 6 illustrates the round-trip time (RTT) to detect a suspect (for a positive identification), which includes the speaker recognition step, the cryptographic operations at the server and the recording device (inclusive of the verification step), and the communication between the two parties. For 100 suspects, the RTT is just 705 ms, which is very short compared to the suggested recording time per speaker sample. (The recommendation is at least 6 s per utterance, in order to get the best embedding representation of the speaker [51].) Also observe that, approximately 500 ms are devoted to the speaker recognition module that is responsible for detecting a unique speaker and generating the corresponding x-vector. More importantly, this cost is absorbed by the recording device and not the server, which is the bottleneck in a wide-scale surveillance system.

Table IV summarizes the characteristics of various privacy-preserving speaker verification and identification protocols in the literature. For each approach, we show its design purpose, the underlying speaker recognition method, the distance metric used to compute the similarity score, the cryptographic primitives employed, the reported online computation time, and the reference protection requirement. We should emphasize that a direct comparison of our approach against privacy-preserving speaker verification protocols is not possible, because the underlying assumptions are quite different. Under speaker verification, the user's reference information is stored at the server in encrypted form (i.e., it is protected by the user's secret key). On the other hand, our system assumes that the server has access to the plaintext reference information (the suspects' database), but this information is encrypted at the distributed clients (the listening devices) with the server's public key.

In terms of speaker recognition techniques, earlier work relied on UBM-GMMs, however, they were subsequently replaced

by probabilistic embeddings, such as i-vectors. Nevertheless, the latest research leans towards neural-network-based embeddings (such as x-vectors), due to their superior performance. The typical dimensionality of i-/x-vector embeddings is around 200, whereas UBM-GMM-based methods, [22] and [25], utilize 32 and 64 Gaussian mixture components, respectively. When selecting an appropriate distance metric, there are two major issues to consider: (i) the accuracy of the model under the chosen metric; and (ii) the feasibility of computing the distance value in the ciphertext domain. As illustrated in Table IV, PLDA is being widely used because of its higher accuracy. Finally, in terms of computation time, our approach is comparable to the most efficient speaker verification protocol by Treiber et al. Specifically, for 100 suspects, the combined online computation time at the client and the server is just 165 ms. This is due to (i) the extensive use of precomputations; and (ii) the use of ElGamal's cryptosystem over elliptic curves, which is very efficient.

IX. Conclusion

Privacy-preserving surveillance aims to balance the need for large-scale surveillance with the requirement of protecting people's privacy. To this end, our work introduced the first system in the literature capable of performing near real-time voice surveillance in a privacy-preserving manner. Our solution leverages the distributed computing capabilities of the surveillance devices to reduce the computational burden of the centralized server. In addition, it relies heavily on precomputed ciphertexts that minimize the overhead at both the client and the server. We implemented a proof-of-concept for our surveillance system, and performed extensive experimentation to demonstrate its applicability in a wide-scale surveillance environment. In particular, we showed that our system is orders of magnitude faster than existing methods, and requires less than 200 ms to match a captured utterance against a database of 100 speakers. Although our protocol computes the PLDA score between two x-vectors, the underlying framework is generic enough to accommodate different embeddings (such as VGGVox) and distance metrics, which may lead to better speaker recognition accuracy.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive feedback that allowed us to improve significantly the quality of this work.

References

[1] *Information technology — Security techniques — Biometric information protection*, ISO/IEC 24745:2011, International Organization for Standardization, Geneva, Switzerland, Jun. 2011.
[2] F.-X. Josset and J. Mattioli, "Serket: An open software platform for preventive security in public crowded places and for large events," *WIT Trans. Built Environ.*, vol. 94, pp. 451–460, 2007.
[3] GDPR, "Regulation EU 2016/679 of the European parliament and of the council of 27 Apr. 2016," *Official J. Eur. Union*, vol. OJ L 119, pp. 1–88, May 2016.
[4] J. Sajfert and T. Quintel, "Removing obstacles for cooperation between EU financial intelligence units: The data protection directive for police and criminal justice authorities as fitting data protection instrument," May 2019. [Online]. Available: https://ssrn.com/abstract=3855507
[5] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2018, pp. 5329–5333.
[6] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Comput. Speech Lang.*, vol. 60, 2020, Art. no. 101027.
[7] S. Ioffe, "Probabilistic linear discriminant analysis," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 531–542.
[8] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. TASSP-28, no. 4, pp. 357–366, Aug. 1980.
[9] H. Li et al., "The I4U system in NIST 2008 speaker recognition evaluation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2009, pp. 4201–4204.
[10] A. Lawson, P. Vabishchevich, M. Huggins, P. Ardis, B. Battles, and A. Stauffer, "Survey and evaluation of acoustic features for speaker recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2011, pp. 5444–5447.
[11] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digit. Signal Process.*, vol. 10, no. 1–3, pp. 19–41, 2000.
[12] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE Signal Process. Lett.*, vol. 13, no. 5, pp. 308–311, May 2006.
[13] L. Burget, P. Matejka, P. Schwarz, O. Glembek, and J. H. Cernocky, "Analysis of feature extraction and channel compensation in a GMM speaker recognition system," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 7, pp. 1979–1986, Sep. 2007.
[14] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 4, pp. 788–798, May 2011.
[15] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Proc. Interspeech*, 2017, pp. 2616–2620.
[16] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Proc. Interspeech*, 2018, pp. 1086–1090.
[17] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (SITW) speaker recognition database," in *Proc. Interspeech*, 2016, pp. 818–822.
[18] M. H. Moattar and M. M. Homayounpour, "A review on speaker diarization systems and approaches," *Speech Commun.*, vol. 54, no. 10, pp. 1065–1103, 2012.
[19] H. Beigi, "Speaker recognition," in *Proc. Fundamentals Speaker Recognit.*, 2011, pp. 543–559.
[20] 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.
[21] M. Pathak, S. Rane, W. Sun, and B. Raj, "Privacy preserving probabilistic inference with hidden Markov models," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2011, pp. 5868–5871.
[22] M. A. Pathak and B. Raj, "Privacy-preserving speaker verification and identification using Gaussian mixture models," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 2, pp. 397–406, Feb. 2013.
[23] M. A. Pathak, B. Raj, S. D. Rane, and P. Smaragdis, "Privacy-preserving speech processing: Cryptographic and string-matching frameworks show promise," *IEEE Signal Process. Mag.*, vol. 30, no. 2, pp. 62–74, Mar. 2013.
[24] M. Aliasgari and M. Blanton, "Secure computation of hidden Markov models," in *Proc. Int. Conf. Secur. Cryptogr.*, 2013, pp. 1–12.
[25] J. Portêlo, B. Raj, A. Abad, and I. Trancoso, "Privacy-preserving speaker verification using garbled GMMs," in *Proc. Eur. Signal Process. Conf.*, 2014, pp. 2070–2074.
[26] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. Symp. Foundations Comput. Sci.*, 1986, pp. 162–167.
[27] A. Jimenez and B. Raj, "Privacy preserving distance computation using somewhat-trusted third parties," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2017, pp. 6399–6403.
[28] A. Nautsch, S. Isadskiy, J. Kolberg, M. Gomez-Barrero, and C. Busch, "Homomorphic encryption for speaker recognition: Protection of biometric templates and vendor model parameters," in *Proc. Odyssey 2018 Speaker Lang. Recognit. Workshop*, 2018, pp. 16–23.
[29] Y. Rahulamathavan, K. R. Sutharsini, I. G. Ray, R. Lu, and M. Rajarajan, "Privacy-preserving iVector-based speaker verification," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 3, pp. 496–506, Mar. 2019.

[30] T. Schneider and A. Treiber, "A comment on privacy-preserving scalar product protocols as proposed in 'SPOC'," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 543–546, Mar. 2020.
[31] A. Nautsch et al., "Privacy-preserving speaker recognition with cohort score normalisation," in *Proc. Interspeech* 2019, pp. 2868–2872.
[32] D. Povey et al., "The Kaldi speech recognition toolkit," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2011.
[33] A. Treiber, A. Nautsch, J. Kolberg, T. Schneider, and C. Busch, "Privacy-preserving PLDA speaker verification using outsourced secure computation," *Speech Commun.*, vol. 114, pp. 60–71, 2019.
[34] F. Brasser, T. Frassetto, K. Riedhammer, A.-R. Sadeghi, T. Schneider, and C. Weinert, "Voiceguard: Secure and private speech processing," in *Proc. Interspeech*, 2018, vol. 18, pp. 1303–1307.
[35] S. P. Bayerl et al., "Offline model guard: Secure and private ML on mobile devices," in *Proc. 2020 Des. Autom. Test Eur. Conf. Exhib.*, 2020, pp. 460–465.
[36] D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto, "SoK: Understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 1416–1432.
[37] D. Poveya et al., "Low development cost, high quality speech recognition for new languages and domains: Report from 2009 Johns Hopkins/CLSP summer workshop," Johns Hopkins Univ., Baltimore, MD, USA, Tech. Rep., 2009.
[38] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," 2015 *arXiv:1510.08484*.
[39] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. 2017 IEEE Int. Conf. Acoust. Speech Signal Process.*, 2017, pp. 5220–5224.
[40] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, no. 4, 2018, Art. no. 79.
[41] T. W. Hungerford, "Algebra," in *Graduate Texts in Mathematics*. Berlin, Germany: Springer, 1974.
[42] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. ACM Symp. Theory Comput.*, 2009, pp. 169–178.
[43] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.
[44] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[45] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. TIT-31, no. 4, pp. 469–472, Jul. 1985.
[46] E. Bentafat, M. M. Rathore, and S. Bakiras, "A practical system for privacy-preserving video surveillance," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2020, pp. 21–39.
[47] N. Tomashenko et al., "The voiceprivacy 2020 challenge evaluation plan," 2020, *arXiv:2205.07123*.
[48] N. Brümmer and J. Du Preez, "Application-independent evaluation of speaker detection," *Comput. Speech Lang.*, vol. 20, no. 2–3, pp. 230–275, 2006.
[49] C. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991.
[50] E. Barker and Q. Dang, "NIST special publication 800-57 part 1, revision 4," National Institute of Standards and Technology, Gaithersburg, MD, USA, Tech. Rep., 2016.
[51] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2019, pp. 5791–5795.

Muhammad Mazhar Ullah Rathore received the master's degree in computer and communication security from the National University of Sciences and Technology, Islamabad, Pakistan, in 2012, and the Ph.D. degree in computer science and engineering from Kyungpook National University, Daegu, South Korea, in 2018. He was a Postdoctorate Researcher with the University of New Brunswick, Fredericton, NB, Canada, and Hamad Bin Khalifa University, Ar-Rayyan, Qatar. He is currently an Assistant Professor with Lakehead University, Thunder Bay, ON, Canada. His research interests include cybersecurity and privacy, Big Data and machine learning, Internet of Things, smart systems, and network traffic analysis. He is an ACM professional member. He was the recipient of the Best Project/Paper Award in the 2016 Qualcomm Innovation Award at Kyungpook National University, for his paper IoT-Based Smart City Development Using Big Data Analytical Approach. He was also a nominee for the Best Project Award in the 2015 IEEE Communications Society Student Competition, for his Project "IoT-Based Smart City." He is frequently as a reviewer for various IEEE, ACM, Springer, and Elsevier journals.

Elmahdi Bentafat received the B.Sc. and M.Sc. degrees in computer science from the Ecole Nationale Supérieure d'Informatique, Algeria, in 2012 and 2016, respectively, and the Ph.D. degree in computer science and engineering from Hamad Bin Khalifa University, Ar-Rayyan, Qatar, in 2021. He is currently a Postdoctoral Researcher with the College of Science and Engineering, Hamad Bin Khalifa University. His research interests include applied cryptography, privacy, information security, and network security.

Spiridon Bakiras (Member, IEEE) received the diploma in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, the M.S. degree in telematics from the University of Surrey, Guildford, U.K., and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA. He is currently an Associate Professor with the Infocomm Technology Cluster, Singapore Institute of Technology, Singapore. Before that, he held teaching and research positions with Hamad Bin Khalifa University, Ar-Rayyan, Qatar, Michigan Technological University, Houghton, MI, USA, the City University of New York, New York, NY, USA, the University of Hong Kong, Hong Kong, and the Hong Kong University of Science and Technology, Hong Kong. His research interests include security and privacy, applied cryptography, mobile computing, and spatiotemporal databases. He is a member of the ACM. He was the recipient of the U.S. National Science Foundation CAREER Award.