Privacy-preserving iVector based Speaker Verification

Yogachandran Rahulamathavan, Kunaraj R Sutharsini, Indranil Ghosh Ray, Rongxing Lu, and Muttukrishnan Rajarajan

***Abstract*—This work introduces an efficient algorithm to develop a privacy-preserving (PP) voice verification based on iVector and linear discriminant analysis techniques. This research considers a scenario in which users enrol their voice biometric to access different services (i.e., banking). Once enrolment is completed, users can verify themselves using their voice-print instead of alphanumeric passwords. Since a voice-print is unique for everyone, storing it with a third-party server raises several privacy concerns. To address this challenge, this work proposes a novel technique based on randomisation to carry out voice authentication, which allows the user to enrol and verify their voice in the randomised domain. To achieve this, the iVector based voice verification technique has been redesigned to work on the randomised domain. The proposed algorithm is validated using a well known speech dataset. The proposed algorithm neither compromises the authentication accuracy nor adds additional complexity due to the randomisation operations.**

***Index Terms*—Privacy, security, speech, iVector, authentication, random domain.**

I. Introduction

TRADITIONAL methods of authentication including passwords, PINs, and memorable words can be easily forgotten, lost, guessed, stolen, or shared. However, authentication using anatomical traits such as fingerprint, face, palm print, iris and voice are very difficult to forge since they are physically linked to the user. Thus, biometric systems impart higher levels of security and have seen a rapid proliferation in a wide variety of government and commercial applications around the world in the last two decades [2]. However, several security and privacy challenges deter the public confidence in adopting biometric systems. Few of them are described below:

Non-revocability: Unlike the alphanumeric passwords, it is impossible to revoke the biometric data once its compromised; hence, once lost the same biometric cannot be reused.

Y. Rahulamathavan is with the Institute for Digital Technologies, Loughborough University London, London, U.K. (e-mail: y.rahulamathavan@lboro.ac.uk).

K R. Sutharsini is with TechInspire Ltd, London, U.K. (e-mail: sutharsini@techinspire.co.uk).

R. Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada. (e-mail: RLU1@unb.ca).

I G Ray and M. Rajarajan are with the Information Security Group, School of Engineering and Mathematical Sciences, City University London, EC1V 0HB, London, U.K. (e-mails: {i.g.ray, R.Muttukrishnan}@city.ac.uk).

The work was supported by the EU Horizon2020 programme under EU Grant H2020-EU.3.7 (Project ID: 653586), NSERC Discovery Grants (04009) and LMCRF-S-2018-03. The authors would like to thank the editor and the anonymous reviewers for their insightful and valuable comments that have helped to improve the clarity and the scope of this manuscript. The authors also like to thank Dr Lewis Herrington for his valuable comments to improve the scope and the presentation of the paper.

Privacy compromise: Inappropriate use of biometric data may breach the user's privacy directly and indirectly. These privacy breaches can be categorised into three types as below:

- *Biometric data privacy breach:* The raw biometric data of the user can be recovered from the stored templates if there are no protections [2]. For example, many fingerprint-based systems use minutiae features extracted from reference fingerprint images. It is possible to reconstruct the original fingerprint image from the stored minutiae.
- *Information privacy breach:* If someone enrols the same biometric data for different services then the biometric templates in all of these systems are identical. This will allow an adversary to use the templates from one system to gain access to another system.
- *Identity privacy breach:* Since the biometric templates used for different services are reasonably similar, there is a possibility for linkability based attacks.

Several cryptographic techniques have been proposed in literature to overcome the above security and privacy challenges (see Section II). The existing works modify various biometric algorithms designed for different types biometrics data. To the best of our knowledge, the work proposed in this paper is the first PP work that redesigns the state-of-the-art iVector based voice verification technique without compromising the accuracy for a negligible computational overhead [1]. The proposed scheme has been validated using the well known TIMIT speech corpus [6]. Theoretical proofs have been provided to validate the privacy and security of the proposed solution. Rigorous experiments show that the scheme mitigates the above issues without compromising the accuracy.

The rest of this paper is organised as follows: The related work is discussed in Section II. The speaker verification model without privacy restriction, and mathematical tools and notations necessary for the proposed algorithm are given in Section III. In Section IV, we redesign the iVector based speaker verification model using randomisation technique and the associated performance results are given in Section V. The security and privacy analysis is given in Section VI followed by conclusions are discussions.

II. Related Works

The use of voice verification over the Internet is becoming very popular. For example, several financial institutes are using speaker verification as a mean to verify its customers. At the same time, recent changes in privacy legislation i.e., GDPR in Europe, are enforcing organisations to provide sufficient privacy guarantee when they use, process and store customer data. Since voice data is unique, the privacy of the voice data should be guaranteed.

This requires a novel voice verification solution with high accuracy and privacy guarantee. PP research addresses this challenge by balancing privacy and usability of data. When it comes to a PP solution, it is all about transforming the existing algorithm to process the inputs when the inputs are either encrypted via homomorphic encryption [3], [5], [7]–[10] or transformed via salting [11], [17].

The aim of homomorphic encryption based PP solutions is protecting the privacy of the input data. However, the existing works redesign different machine learning algorithms i.e., face recognition based on the principal component analysis [7], facial expression recognition based on the linear discriminant analysis [10], multi-class problem based on support vector machine [5], [8], [9], are the few to mention here. These solutions achieve the same accuracy as their corresponding traditional algorithms subject to hefty computational overhead [3], [10].

On the other hand, salting based cancellable biometric solutions increase the computation speed significantly compared to the homomorphic encryption-based solutions [11], [17]. However, these solutions either decrease the accuracy or privacy.

In the domain of voice biometric, there are only a few notable PP voice verification works exist [3], [11], [14], [16], [17]. Smaragdis and Shashanka proposed the first application of secure multi-party computation (SMC) concepts for privacy-constrained speech technology [14]. In their work, they realised secure speech recognition using the hidden Markov model (HMM) and a generalised version of the Paillier public-key scheme, which allowed training and classification between multiple parties and achieved perfect accuracy.

Pathak et. al redesigned the Gaussian Mixture Model (GMM) based speaker recognition [3] to achieve a similar privacy goal. This work relies on homomorphic cryptosystems such as BGN and Paillier encryption. This work has shown a proof-of-concept of PP speaker recognition without compromising the accuracy. However, the shortcoming of the above cryptographic approaches [3], [14] is that far too much time is spent on the encryption, which makes it impractical for real applications i.e., [3] requires few minutes for authentication.

In order to overcome the heavy computation that is involved with the above homomorphic encryption schemes, string-matching frameworks were proposed in [11], [17]. These schemes convert the speech input represented by the super-vectors to bit strings using locality sensitive hashing (LSH) and counted the exact matches [11], [17]. Since it is easy to perform string comparison with privacy, the method proves to be more efficient; however, it lacks accuracy with EER=11.86%.

To the best of our knowledge, one and only work that proposes a PP solution for iVector based speaker verification is [16]. The work [16] presented a secure binary embedding (SBE) which is a hashing scheme in an attempt to enable privacy for iVector based speaker recognition. The work [16] uses a hashing technique where similar templates are placed in close proximity in the hash domain. Due to the inherent nature of hashing, the verification accuracy obtained in [16] is much lower than the true accuracy (when the traditional iVector solution provides EER = 1%, the solution [16] provides EER = 20% when the privacy is high).

In contrast to all the above works, the proposed work in this paper uses randomisation technique from information theory which is neither computationally inefficient nor compromises the privacy. This work not only provides the speed necessary for real-time computation but also provides information-theoretical privacy and highest possible accuracy. This solution is significantly advanced than the existing solution in terms of accuracy, privacy and speed. Note that the proposed solution can be applied to variants of ivector based speaker verification solutions that calculates scores using cosine distances. If a solution such as PLDA based ivector [19] uses different scoring method then the proposed solution may not be sufficient.

III. Speaker recognition based on iVector and Cosine Distance Scoring

Recently Dehak et. al proposed a pioneering work, namely *iVector*, for voice verification [1]. The iVector model generates a low-dimensional speaker-and-channel dependent space using factor analysis [1]. Serveral channel compensation techniques such as, within-class covariance normalisation (WCCN), linear discriminant analysis (LDA), and nuisance attribute projection (NAP), were applied on this low dimensional space to remove the channel dependent noise [1]. Through rigorous experiments, Dehak et. al concluded that iVector and LDA based speaker verification outperforms the other competitive techniques [1].

Hence, as discussed in Section I, a PP version of ivector model is developed in this paper. The aim of the proposed work is to achieve privacy within user-server settings without reducing the accuracy subject to negligible complexity overhead. The following section briefly describes the original speaker verification model [1].

The work [1] mainly constitutes of two parts: 1) iVector feature extraction and speaker model building and 2) speaker verification. The first part extracts features of voice using several techniques i.e., Mel frequency cepstral coefficients (MFCC), Gaussian mixture model (GMM), Universal back ground model (UBM), and maximum a posterior adaptation (MAP) [4]. The feature extraction step is followed by a speaker model building step i.e., obtaining matrix $\mathbf{R}$ in the equation (2) below. Once $\mathbf{R}$ is obtained, voice feature of a user, called ivector, $\mathbf{w}_{\text{target}}$, can be enrolled in the server. Refer [1] for additional technical details of the first part.

During the second part (i.e., speaker verification), the user is required to send a voice feature vector $\mathbf{w}_{\text{test}}$ to the server. The work [1] uses cosine distance scoring for speaker verification. The cosine distance scoring computes the value of the cosine kernel between the target speaker ivector $\mathbf{w}_{\text{target}} \in \mathbb{R}^{d\times 1}$ and the test ivector $\mathbf{w}_{\text{test}} \in \mathbb{R}^{d\times 1}$ as a decision score [1]

$$\text{score}(\mathbf{w}_{\text{target}}, \mathbf{w}_{\text{test}}) = \frac{< \mathbf{w}_{\text{target}}, \mathbf{w}_{\text{test}} >}{||\mathbf{w}_{\text{target}}||||\mathbf{w}_{\text{test}}||} \gtreqless \theta, \qquad (1)$$

where dimension d is the size of the iVector (i.e., $d = 200$ used for the experiments in Section V-D). To compensate the channel effect, as mentioned above, the work [1] considered three different techniques namely 1) WCCN, 2) LDA, and 3) NAP. These techniques compute projection matrices $\mathbf{P}_{\mathrm{WCCN}}$, $\mathbf{P}_{\mathrm{LDA}}$, and $\mathbf{P}_{\mathrm{NAP}}$ from training speech data. In the following, the projection matrices are denoted by $\mathbf{P}$ (i.e., $\mathbf{P} \equiv \mathbf{P}_{\mathrm{WCCN}} \equiv \mathbf{P}_{\mathrm{LDA}} \equiv \mathbf{P}_{\mathrm{NAP}}$). To preserve the inner-product in (1), and to apply these channel compensation techniques, Dehak et. al used the following approach [1]:

$$\begin{aligned} &\mathrm{score}(\mathbf{w}_{\mathrm{target}}, \mathbf{w}_{\mathrm{test}}) \\ =\ & \frac{(\mathbf{P}^T\mathbf{w}_{\mathrm{target}})^T(\mathbf{P}^T\mathbf{w}_{\mathrm{test}})}{\sqrt{(\mathbf{P}^T\mathbf{w}_{\mathrm{target}})^T(\mathbf{P}^T\mathbf{w}_{\mathrm{target}})}\sqrt{(\mathbf{P}^T\mathbf{w}_{\mathrm{test}})^T(\mathbf{P}^T\mathbf{w}_{\mathrm{test}})}}, \\ =\ & \frac{\mathbf{w}_{\mathrm{target}}^T\mathbf{R}\mathbf{w}_{\mathrm{test}}}{\sqrt{\mathbf{w}_{\mathrm{target}}^T\mathbf{R}\mathbf{w}_{\mathrm{target}}}\sqrt{\mathbf{w}_{\mathrm{test}}^T\mathbf{R}\mathbf{w}_{\mathrm{test}}}} \gtrless \theta \end{aligned} \tag{2}$$

where $\mathbf{R} = \mathbf{P}\mathbf{P}^T \in \mathbb{R}^{d\times d}$.

In a traditional system (i.e., without privacy constraints), the user device extracts ivector $\mathbf{w}_1$ from a speech utterance and sends it to the server during the enrolment phase. The server obtains $\mathbf{R}$ from all the users who use the system for speaker verification. During the recognition phase, the user device extracts another ivector $\mathbf{w}_2$ from a speech utterance and sends it to the server. Now the server computes the score using the ivectors $\mathbf{w}_1$ and $\mathbf{w}_2$, and matrix $\mathbf{R}$ as follows:

$$\mathrm{score}(\mathbf{w}_1, \mathbf{w}_2) = \frac{\mathbf{w}_1^T\mathbf{R}\mathbf{w}_2}{\sqrt{\mathbf{w}_1^T\mathbf{R}\mathbf{w}_1}\sqrt{\mathbf{w}_2^T\mathbf{R}\mathbf{w}_2}}. \tag{3}$$

If $\mathrm{score}(\mathbf{w}_1, \mathbf{w}_2) > \theta$ then the server decides that the ivectors $\mathbf{w}_1$ and $\mathbf{w}_2$ are generated by the same speaker.

IV. Model, Overview and Privacy-preserving approach

Consider a voice verification system with N users. Lets call the ivector $\mathbf{w}_1$ enrolled at the server as *speaker model*. During the speaker verification stage, the user needs to send another ivector $\mathbf{w}_2$ to the server. Lets call this ivector as *test feature*. The server verifies the test feature against the speaker model.

Lets introduce another variable called *secret key* to randomise the speaker model and test feature. Refer Section IV-B for more details. The randomisation operation converts the speaker model and test feature into *randomised speaker model* and *randomised test feature*. Now the secret key is split into two shared-secret-keys: one for user and one for server.

Since there are N users, the server holds N randomised speaker models and N shared secret keys. During the verification stage, the user device randomises the test feature vector using it's shared secret key and sends the randomised test feature to the server. Within this context, lets define the following privacy threats and goals of the proposed work:

- *Revocablity:* In the event of data breach, it should be possible to revoke the randomised speaker model and enrol a new randomised speaker model.
- *Template diversity:* It should be infeasible for an adversary to reveal whether the same user has been registered for different services.
- *Compromising the test feature:* If an adversary has access to the the test feature, then it should be infeasible for the adversary to impersonate the user in future.
- *Compromising the data from the user device:* If the user device is compromised then the stored shared secret key shouldn't be used to impersonate the user in future.

To satisfy the above privacy threats, the traditional speaker verification should be redesigned. Lets introduce a cryptographic primitive called *randomisation technique* in the following section which will be used to develop a PP speaker verification.

A. Randomisation technique

Denote an integer message $m \in \mathsf{M} = \{-2^M \text{ to } 2^M\}$ and an integer secret key $s \in \mathsf{S} = \{-2^R - 2^M \text{ to } 2^R + 2^M\}$ where M and R are integers satisfy $2^R >> 2^M$. The secret key $s \in \mathsf{S}$ is generated randomly from a uniform distribution in the range of $\mathsf{S} = \{-2^R - 2^M \text{ to } 2^R + 2^M\}$. Now we propose the following algorithm to randomise the message m into a randomised message $r \in \mathsf{R} = \{-2^R \text{ to } 2^R\}$ using s.

Algorithm 1 Randomisation Technique

```
1: procedure RANDOMISE($m$)
2:     Generate secret key $s$
3:     DO $r = m + s$
4:     IF $r \in \{-10^R \text{ to } 10^R\}$
5:         Return $r,\ s$
6:     ElseIF
7:         Go to Step 2
8:     EndIF
```

Table I
Randomisation: A toy example.

Randomised Messages [-4 to 4] (i.e., $R = 2$)	Possible Messages [-2 to 2] (i.e., $M = 1$)	Possible Secret Key Values [-6 to 6] (i.e., $M = 1$ and $R = 2$)
-4	-2, -1, 0, 1, 2	-2, -3, -4, -5, -6
-3	-2, -1, 0, 1, 2	-1, -2, -3, -4, -5
-2	-2, -1, 0, 1, 2	0, -1, -2, -3, -4
-1	-2, -1, 0, 1, 2	1, 0, -1, -2, -3
0	-2, -1, 0, 1, 2	2, 1, 0, -1, -2
1	-2, -1, 0, 1, 2	3, 2, 1, 0, -1
2	-2, -1, 0, 1, 2	4, 3, 2, 1, 0
3	-2, -1, 0, 1, 2	5, 4, 3, 2, 1
4	-2, -1, 0, 1, 2	6, 5, 4, 3, 2

Lets consider a toy example with a message domain $\mathsf{M} = \{-2 \text{ to } 2\}$ (i.e., $M = 1$), randomised message domain $\mathsf{R} = \{-4 \text{ to } 4\}$ (i.e., $R = 2$) and secret key domain $\mathsf{S} = \{-6 \text{ to } 6\}$ ($M = 1$ and $R = 2$). The possible messages, secret keys and the corresponding randomised messages for the toy example are shown in Table I.

Let's suppose the randomised message is -4. This -4 could have been obtained from any messages in the message domain (i.e., -4 = -2 + -2 or = -1 + -3 or = 0 + -4 or = 1 + -5 or = 2 + -6). Similarly if the randomised message is -3, this randomised message could have been obtained from any possible messages (i.e., -3 = -2 + -1 or = -1 + -2 or = 0 + -3 or = 1 + -4 or = 2 + -5). It is obvious from Table I that any randomised message could be generated from any message from the message domain. Hence, if an attacker compromises a randomised message, then it is impossible for the adversary to recover message m from the randomised message r without knowing the secret key s i.e., *posterior probability* and *prior probability* of the messages are equal. Hence, this algorithm follows information-theoretic security [13] (refer Section VI for formal proof).

B. The Proposed Privacy-preserving Approach

This section proposes the following two algorithms: 1) Basic Approach and 2) Strong Approach. The basic approach protects the speaker model $\mathbf{w}_1$ residing at the server side. The strong approach protects both the speaker model and test feature vector, $\mathbf{w}_1$ and $\mathbf{w}_2$. The following sections explain both the approaches in detail.

C. The Basic Approach

The basic approach (BA) transforms the speaker model $\mathbf{w}_1$ into a different vector using one-way cryptographic function. Therefore the transformed version leak nothing about the original values of $\mathbf{w}_1$ albeit it can still be used for speaker verification. This approach protects against any unwanted privacy leakages if the server happens to be compromised. The randomisation technique proposed in Algorithm 1 in Table I can be used as a one-way cryptographic function.

The user device executes Algorithm 1 to randomise $\mathbf{w}_1$ using a random vector $\mathbf{r}_1$. Then user enrols $\mathbf{w}_1+\mathbf{r}_1$ at the server and keeps $\mathbf{r}_1$. During the verification phase, the user sends not only $\mathbf{w}_2$ but also $\mathbf{r}_1$ to the server. The server first obtains $\mathbf{w}_1$ by subtracting $\mathbf{r}_1$ from the stored randomised feature $\mathbf{w}_1+\mathbf{r}_1$ followed by executing (3). Once the verification process is completed, the server will keep only the randomised vector $\mathbf{w}_1+\mathbf{r}_1$ and delete all other parameters (i.e., $\mathbf{r}_1$, $\mathbf{w}_2$ and $\mathbf{w}_1$).

Since the speaker model is randomised in the BA approach, any attack on the server will not reveal $\mathbf{w}_1$ to the adversaries. In the event of an attack, the speaker model can be revoked and a new speaker model can be enrolled. Note that this approach cannot protect the privacy of user biometric if the server has already been infected by a malware which can monitor the speaker verification process. Hence, the BA approach trusts the server and assumes that the server follows the procedure and free from malware.

D. The Strong Approach

The strong approach (SA) does not require a trusted server for speaker verification. The aim of the SA approach is that even if the server is infected by a malware, it should be infeasible for the malware to obtain $\mathbf{w}_1$ and $\mathbf{w}_2$. To achieve this objective, during the enrolment phase, the user randomises the feature vector $\mathbf{w}_1$ using random vectors $\mathbf{r}_1$ and $\mathbf{r}_x$ using the Algorithm 1 and enrols $\mathbf{w}_1+\mathbf{r}_1$ and $\mathbf{w}_1^x=\mathbf{w}_1+\mathbf{r}_x$ at the server and keeps $\mathbf{r}_1$ and $\mathbf{w}_1^y=-\mathbf{r}_1$ where

$$\mathbf{w}_1=\mathbf{w}_1^x+\mathbf{w}_1^y. \tag{4}$$

Then the user deletes $\mathbf{w}_1$ from the user device (the intuition behind this split is explained in Section VI-B). During the speaker verification phase, the user device randomises the test feature vector $\mathbf{w}_2$ using a random vector $\mathbf{r}_2$ and sends $\mathbf{w}_2+\mathbf{r}_2$ to the server and keeps $\mathbf{r}_2$.

Then the server uses $\mathbf{w}_1+\mathbf{r}_1$, and $\mathbf{w}_2+\mathbf{r}_2$ to computes (3) as follows:

$$\begin{aligned}
&\text{score}(\mathbf{w}_1+\mathbf{r}_1,\mathbf{w}_2+\mathbf{r}_2)\\
=\;&\frac{(\mathbf{w}_1+\mathbf{r}_1)^T\mathbf{R}(\mathbf{w}_2+\mathbf{r}_2)}{\sqrt{(\mathbf{w}_1+\mathbf{r}_1)^T R(w_1+\mathbf{r}_1)}\sqrt{(\mathbf{w}_2+\mathbf{r}_2)^T\mathbf{R}(w_2+\mathbf{r}_2)}},\\
=\;&\frac{\mathbf{w}_1^T\mathbf{R}\mathbf{w}_2+n_1}{\sqrt{\mathbf{w}_1^T\mathbf{R}\mathbf{w}_1+n_2}\sqrt{\mathbf{w}_2^T\mathbf{R}\mathbf{w}_2+n_3}}, \qquad (5)
\end{aligned}$$

where

$$\begin{aligned}
n_1 &= \mathbf{w}_1^T\mathbf{R}\mathbf{r}_2+\mathbf{r}_1^T\mathbf{R}\mathbf{w}_2+\mathbf{r}_1^T\mathbf{R}\mathbf{r}_2,\\
&= \mathbf{w}_1^{xT}\mathbf{R}\mathbf{r}_2+\mathbf{w}_1^{yT}\mathbf{R}\mathbf{r}_2+\mathbf{r}_1^T\mathbf{R}\mathbf{w}_2+\mathbf{r}_1^T\mathbf{R}\mathbf{r}_2, \qquad (6)\\
n_2 &= \mathbf{w}_1^T(2\mathbf{R})\mathbf{r}_1+\mathbf{r}_1^T\mathbf{R}\mathbf{r}_1,\\
&= \mathbf{w}_1^{xT}(2\mathbf{R})\mathbf{r}_1+\mathbf{w}_1^{yT}(2\mathbf{R})\mathbf{r}_1+\mathbf{r}_1^T\mathbf{R}\mathbf{r}_1, \qquad (7)\\
n_3 &= \mathbf{w}_2^T(2\mathbf{R})\mathbf{r}_2+\mathbf{r}_2^T\mathbf{R}\mathbf{r}_2. \qquad (8)
\end{aligned}$$

In the numerator of (5), the true value $\mathbf{w}_1^T\mathbf{R}\mathbf{w}_2$ has been randomised by n_1. Similarly, in the denominator of (5), the true values $\mathbf{w}_1^T\mathbf{R}\mathbf{w}_1$ and $\mathbf{w}_2^T\mathbf{R}\mathbf{w}_2$ have been randomised by n_2, and n_3, respectively. In order to correctly verify the user, the server needs to calculate n_1, n_2, and n_3. However, the server does not have all the variables to correctly computes n_1, n_2, and n_3. The table in Figure 1 shows all the variables that are known only to the server and known only to the user.

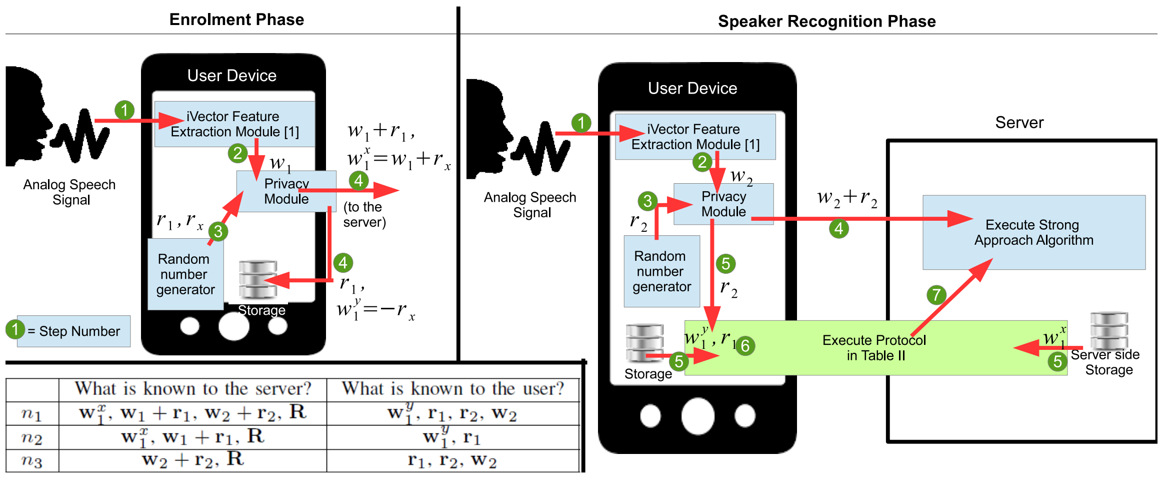
Therefore, the user and server need to compute n_1, n_2, and n_3 jointly without leaking any sensible information to each other (i.e., secure two-party computation [15]). Lets define six vectors $\mathbf{u}_1$, $\mathbf{s}_1$, $\mathbf{u}_2$, $\mathbf{s}_2$, $\mathbf{u}_3$, and $\mathbf{s}_3$ as follows:

$$\begin{aligned}
\mathbf{u}_1 &= \left[\mathbf{r}_2^T\ \text{vec}(\mathbf{r}_2\mathbf{w}_1^{yT}+\mathbf{w}_2\mathbf{r}_1^T+\mathbf{r}\mathbf{r}_1^T)\right]^T \in \mathbb{R}^{(d+d^2)\times 1},\\
\mathbf{s}_1 &= \left[\mathbf{w}_1^{xT}\mathbf{R}\ \text{vec}(\mathbf{R})^T\right]^T \in \mathbb{R}^{(d+d^2)\times 1},\\
\mathbf{u}_2 &= \left[\mathbf{r}_1^T\ \text{vec}(\mathbf{r}_1\mathbf{w}_1^y)^T\ \text{vec}(\mathbf{r}_1\mathbf{r}_1^T)^T\right]^T \in \mathbb{R}^{(d+2d^2)\times 1},\\
\mathbf{s}_2 &= \left[2\mathbf{w}_1^{xT}\mathbf{R}\ \text{vec}(2\mathbf{R})^T\ \text{vec}(\mathbf{R})^T\right]^T \in \mathbb{R}^{(d+2d^2)\times 1},\\
\mathbf{u}_3 &= \left[\text{vec}(\mathbf{r}_2\mathbf{w}_2^T)^T\ \text{vec}(\mathbf{r}_2\mathbf{r}_2^T)^T\right]^T \in \mathbb{R}^{2d^2\times 1},\\
\mathbf{s}_3 &= \left[\text{vec}(2\mathbf{R})^T\ \text{vec}(\mathbf{R})^T\right]^T \in \mathbb{R}^{2d^2\times 1},
\end{aligned}$$

where vec(.) denotes the vectorisation operation. From the table in Figure 1, the vectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ can be obtained by the user without interacting with the server. Similarly, the vectors $\mathbf{s}_1$, $\mathbf{s}_2$, and $\mathbf{s}_3$ can be obtained by the server without interacting with the user. Hence, the equations (6) - (8) can be modified into

$$n_1=\mathbf{u}_1^T\mathbf{s}_1,\ n_2=\mathbf{u}_2^T\mathbf{s}_2,\ \&\ n_3=\mathbf{u}_3^T\mathbf{s}_3. \tag{9}$$

To calculate n_1, n_2, and n_3, the user and server need to interact with each other. The following subsection explains this procedure.



	What is known to the server?	What is known to the user?
n_1	$\mathbf{w}_1^x$, $\mathbf{w}_1+\mathbf{r}_1$, $\mathbf{w}_2+\mathbf{r}_2$, $\mathbf{R}$	$\mathbf{w}_1^y$, $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{w}_2$
n_2	$\mathbf{w}_1^x$, $\mathbf{w}_1+\mathbf{r}_1$, $\mathbf{R}$	$\mathbf{w}_1^y$, $\mathbf{r}_1$
n_3	$\mathbf{w}_2+\mathbf{r}_2$, $\mathbf{R}$	$\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{w}_2$

Figure 1. Message flow diagram of the proposed SA algorithm. The left figure shows the enrolment steps, the right figure shows the verification steps, and the table shows the parameters that are known only to the user and the parameters known only to the server.

E. The privacy-preserving scalar product algorithm

To compute the required scalar products in (9), the user and server follow a PP protocol where no party can learn the other party's input. At the end of the protocol server should be able to obtain n_1, n_2, and n_3. This can be achieved by the PP scalar product algorithm in Table II [18].

The user and server jointly execute the protocol in Table II three times to compute (9). Initially the user generates a number of random values to randomises its input vector $\mathbf{a}$ ($= \mathbf{u}_1$ for first execution) and obtains randomised vector $[C_1, C_2, \ldots, C_n]$. As these randomisation operations use modulo reduction, the server cannot reverse engineer and reveal the user's input data from $[C_1, C_2, \ldots, C_n]$ (refer [18] for the formal security proof and correctness).

Upon receiving $[C_1, C_2, \ldots, C_n]$, the server now performs multiplication operations to get $[D_1, D_2, \ldots, D_n]$. These values are then added through modulo addition followed by randomisation operation using γ. The final value is sent to the user. The user obtains $\mathbf{a}^T\mathbf{b}+\gamma$ using the secret key s^{-1} *mod* p. Finally the server receives $\mathbf{a}^T\mathbf{b}+\gamma$ from user and subtracts γ to get $\mathbf{a}^T\mathbf{b}$ (i.e., $= \mathbf{u}_1^T\mathbf{s}_1$). Message flow diagram of the proposed SA algorithm is shown in Figure 1.

Table II
PP SCALAR PRODUCT ALGORITHM [18].

```
Input by User: $\mathbf{a} = [a_1, \ldots, a_n]^T \in F_q^n$
    and Server: $\mathbf{b} = [b_1, \ldots, b_n]^T \in F_q^n$
Output to Server: $\mathbf{a}^T\mathbf{b}$
--------------------------------------------------------------
User performs the following operations:
  Given security parameters $k_1$, $k_2$, $k_3$, $k_4$,
      choose two large primes $\alpha$, $p$
      such that $|p| = k_1$, $|\alpha| = k_2$, set $a_{n+1} = a_{n+2} = 0$
  Choose a large random number $s \in Z_p$, and $n+2$ random
       numbers $c_i$, $i = 1, 2, ..., n+2$, with $|c_i| = k_3$
      FOR EACH $a_i$, $i = 1, 2, ..., n+2$
          Compute
              $C_i = s(a_i.\alpha + c_i) \bmod p, \ a_i \neq 0$
              $C_i = sc_i \bmod p, \ a_i = 0$
      END FOR
  keeps $s^{-1}$mod $p$ secret, and sends $(\alpha, p, C_1 \ldots C_{n+2})$ to Server
--------------------------------------------------------------
Now Server executes the following operations
      set $b_{n+1} = b_{n+2} = 0$
      FOR EACH $b_i$, $i = 1, 2, ..., n+2$
          Compute
              $D_i = b_i.\alpha.C_i \bmod p, \ b_i \neq 0$
              $D_i = r_i.C_i \bmod p, \ b_i = 0,$
              where $r_i$ is a random number with $|r_i| = k_4$
      END FOR
    compute $D = \sum_{i=1}^{n+2} D_i + \gamma \bmod p$
    and send $D$ back to User where $|\gamma| > |\sum_{i=1}^{n+2} a_i.b_i|$
--------------------------------------------------------------
Now User computes
  $E = s^{-1}.D \bmod p$ and get $\mathbf{a}^T\mathbf{b} + \gamma$
    $= \sum_{i=1}^{n} a_i.b_i + \gamma = \frac{E-(E \bmod \alpha^2)}{\alpha^2}$ and sends $\mathbf{a}^T\mathbf{b} + \gamma$ to Server
--------------------------------------------------------------
Now Server obtains
      $\mathbf{a}^T\mathbf{b}$ by subtracting $\gamma$ from $\mathbf{a}^T\mathbf{b} + \gamma$
end procedure
```

V. PERFORMANCE ANALYSIS

This section describes the dataset used for the experiments, results and the complexity, security, and privacy analysis of the proposed algorithm.

A. The dataset

TIMIT speech corpus has been used to evaluate the accuracy and reliability of the proposed algorithm [6]. The TIMIT speech corpus contains broadband recordings (each recording lasts for around 3 seconds) of 630 speakers of eight major dialects of American English. Each speaker has 10 speech samples. Out of 10 samples, 8 were used to build the speaker model.

For experiment, the TIMIT data corpus has been split into two: 1) the first two dialect regions with 151 speakers are used for training and testing and 2) the last four dialect region with 277 speakers were used to build background model. Table III shows the statistics of the TIMIT dataset.

B. Experiments on the TIMIT Database without Privacy

To validate the proposed method, we first obtain the verification accuracy of the iVector algorithm on TIMIT dataset using the pre-divided speech samples shown in Table III.

Table III
TIMIT DATABASE.

Dialect Region (DR)	#Male	#Female	**Total**
DR1	31	18	**49**
DR2	71	31	**102**
DR3	79	23	**102**
DR4	69	31	**100**
DR5	62	36	**98**
DR6	30	16	**46**
DR7	74	26	**100**
DR8	22	11	**33**
Total	**438**	**192**	**630**

C. *Definitions*

Next subsections present various tests to validate the proposed model. To facilitate the description of tests, lets introduce a few definitions used in speaker verification to measure the performance: False Acceptance Rate (FAR), False Rejection Rate (FRR), Equal Error Rate (EER), and Detection Error Tradeoff (DET) curve[1].

FAR and FRR are the two types of errors defined as follows:

- FAR= $\frac{\text{Number of False Acceptance}}{\text{Total Number of Imposter Attempts}} \times 100\%$,
- FRR = $\frac{\text{Number of False Rejection}}{\text{Total Number of Geinune Attempts}} \times 100\%$,

where False Acceptance means the system grants access to an impostor, and False Rejection means the system denies access to an enrolled speaker. EER represents the operating point where the FAR is equal to the FRR. DET curve has been used in speaker verification to view FAR, FRR, and EER on the same curve. The DET curve comprises FRR in the y-axis and FAR on the x-axis. The EER represents the point on the DET curve where both FRR and FAR are equal.

D. *Baseline Test*

As described above, there are 151 users enrolled at the server. There are two speech samples available for each user for verification. To test the performance of the traditional (i.e., without privacy constraints) speaker verification, the following two tests are conducted:

1) Genuine Attempts:- Client-Client: In this test, for each speaker, the score is calculated using the speaker's speaker model against the speaker's two test utterances. Hence, the scores for $151 \times 2 = 302$ tests are obtained using (3).

2) Imposter Attempts:- Imposter-Client: In this test, each speaker's test utterances are tested against other 150 users' speaker models. This leads to $151 \times (151 - 1) \times 2 = 45300$ tests and the score for each test has been obtained using (3). Figure 2 shows the distribution between genuine attempts and imposter attempts tests. When the threshold $\theta = 1.34$ the EER is 6.5%. We will use this result as a benchmark to compare the performance of the proposed algorithm.

E. *Testing the Proposed Algorithm*

Same experimental protocols described in Section V-D has been repeated to test the proposed algorithm. Since the PP protocol in Table II is suitable for integers, the decimal values in speaker models and test feature vectors must be scaled to integers via scaling and quantisation operations followed by randomisation.

Table IV shows few examples for scaling, quantisation and randomisation using the values of ivectors (i.e., $\mathbf{w}_1$ or $\mathbf{w}_2$) and projection matrix (i.e., $\mathbf{R}$). When the scaling factor increases, the effect of quantisation is decreasing e.g., the sample value in the first row in Table IV (0.010924) is almost half of the value in the second row (0.017854). However, when the scaling factor is equal to 100 (2nd column), both the values became equal. When the scaling factor is 1000 (fourth column), the ratio between both the values is getting closer to the correct ratio. As shown in Figures 5 and 6, the elements of iVectors follow normal distribution with a mean 0 (approximately) and standard deviation 0.01.

The last three columns of Table IV shows how the scaled values are randomised by different sizes of random numbers. The fifth and last columns in Table IV show how the scaled elements in fourth column are randomised using random numbers between -10 to 10, and -10^6 to 10^6, respectively. The experiments demonstrated that the output elements of the randomisation operation always follow uniform distribution e.g., when the iVectors are scaled by 1000 followed by randomisation operation in the range of -10^6 to 10^6 according to Algorithm 1, then the output distribution is uniform with standard deviation 5.7825×10^5 (where the theoretical standard deviation for the uniform distribution in the range -10^6 to 10^6 is 5.7735×10^5) - refer Figure 7.

To evaluate the impact of scaling and rounding operations, we repeated the two tests conducted in Section V-D but using the proposed algorithm for scaling factors $s = 100$, $s = 200$, $s = 400$, $s = 600$, $s = 800$, and $s = 1000$ and randomisation with random numbers in the range of -10^6 to 10^6. Figure 3 shows the DET curves for the above scaling factors. When the scaling factor increases from 100 to 1000, the accuracy of the proposed scheme approaches the benchmark accuracy. For a scaling factor $s = 1000$, the proposed algorithm illustrates identical recognition as the benchmark. This validates that the proposed model does not compromise the accuracy.

In order to test the effect of randomisation (or to answer why the scaled and quantised input elements are randomised using large random numbers (10^6) instead of 10), we repeated the baseline test (experiment conducted in Section V-D) but with a randomised test feature vector and projection matrix. We used different size of random values ranging from 1 to 10^5 to randomise the elements of test features and projection matrix. We also tested the baseline model with pure random vectors (i.e., generated independently of speech) as test features. As shown in Figure 4, when the size of the random values decreases, the accuracy increases. When the size of the random values are in the range of 10^5, there is no significant difference in accuracy between random testing (Pure Random in the Figure 4) and randomised testing. It means if the input elements are not masked by large random numbers then it is possible for the adversary to infer the identity of the input samples. However, when larger random numbers (i.e., 10^5 in this experiment) are used to randomise the test features, the accuracy of the system

[1]DET curves are plotted using NIST DET-ware-v2.1 tool: Available On line: https://www.nist.gov/file/65996, Accessed on 5th of June, 2018.

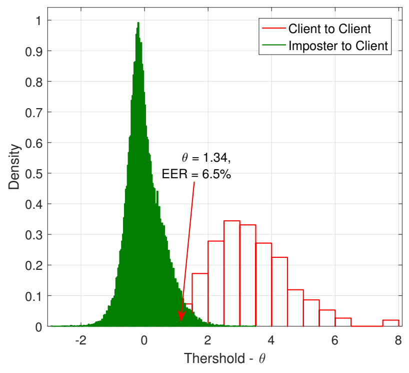


Figure 2. Score distribution of genuine and imposter tests.

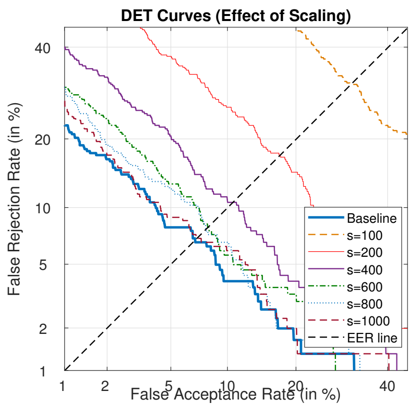


Figure 3. The accuracy of the proposed scheme for various scaling factors.

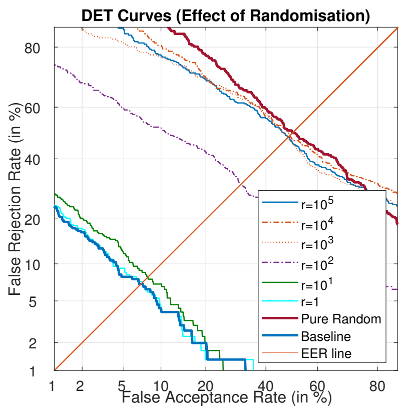


Figure 4. The accuracy of the proposed scheme for various scaling factors.

Table IV
SAMPLE DATA SHOWING THE EFFECTS OF SCALING, QUANTISATION AND RANDOMISATION.

Effect of Scaling (Supporting Data for Figure 3)				Effect of Randomisation (Supporting Data for Figure 4)		
Sample iVector Values	Scaling (x100) and Integer Quantisation	Scaling (x600) and Integer Quantisation	Scaling (x1000) and Integer Quantisation	Randomising with random number in the range of [-10 to 10]	Randomising with random number in the range of [-100 to 100]	Randomising with random number in the range of $[-10^6$ to $10^6]$
0.010924	1	6	11	7	23	447961
0.017854	1	10	17	8	-6	359424
-0.027501	-2	-16	-27	-33	25	310168
Sample elements of Projection Matrix $\mathbf{R}$	Scaling (x100) and Integer Quantisation	Scaling (x600) and Integer Quantisation	Scaling (x1000) and Integer Quantisation	Randomising with random number in the range of [-10 to 10]	Randomising with random number in the range of [-100 to 100]	Randomising with random number in the range of $[-10^6$ to $10^6]$
2.080734	208	1248	2080	2088	2079	-1191
1.714698	171	1028	1714	1719	1704	921203
1.098638	109	659	1098	1095	1128	-318130

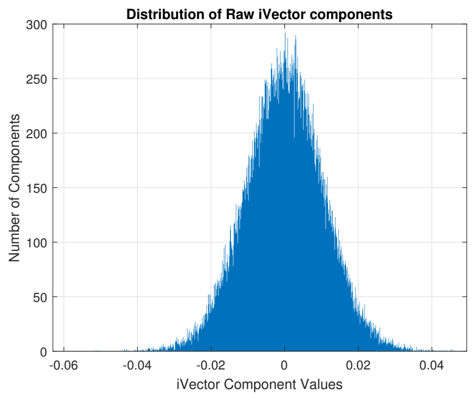


Figure 5. Distribution of raw iVector components (mean is 1.0245×10^{-5} and standard deviation 0.0106).

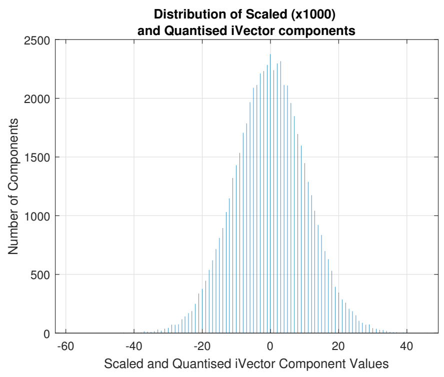


Figure 6. Distribution of raw iVector components after scaling operation.

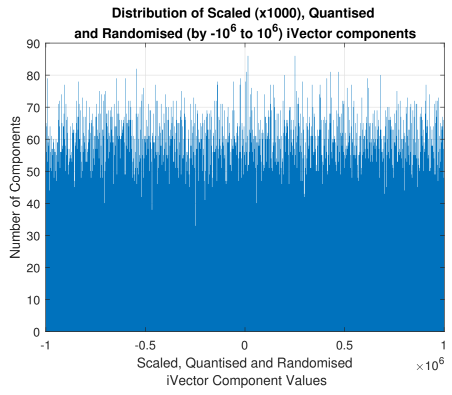


Figure 7. Distribution of raw iVector components after randomisation operation (standard distribution is 5.7825×10^5)

is closer to the accuracy of pure random inputs. It means when the input elements are randomised by large random numbers, there is no different in statistical properties between pure random values and randomised input values.

F. Complexity Analysis

The proposed PP algorithms require additional mathematical operations to protect the parameters from the untrusted server. The BA algorithm does not require any additional mathematical operations except addition and subtraction, hence we assume the complexity of BA is same as the traditional (i.e., without privacy constraint) algorithm. Lets denote the time complexity for one multiplication as t_{mul} and for square-root as $\sqrt{}$. Since the ivector feature extraction is common for both the traditional and the proposed SA algorithm, lets compare the complexity of both the algorithm after the feature extraction.

In the traditional algorithm, once the ivector has been extracted, the user device does not require to perform any operations. However, the server needs to compute (3) which requires $3(d^2+d)t_{\text{mul}} + 2\sqrt{\cdot}$ operations if the ivectors are $d-$dimensional. In the proposed SA algorithm, the user device and server need to perform some additional computations to obtain (9) via scalar product algorithm in Table II. To execute the algorithm in Table II, the server incurs $(2n+4)t_{\text{mul}}$ complexity and the user device incurs $(2n+5)t_{\text{mul}}$ complexity if the dimension of the input vectors is n.

In order to compute the scalar products in (3), the user and server need to invoke the protocol illustrated in Table II twice (to obtain n_1 and n_3). It should be noted that n_2 in (3) can be calculated offline and pre-stored at the server side as it does not require speaker recognition parameters. Hence the total computational cost for the user and server would be $2(2n+5)t_{\text{mul}}$ and $2(2n+4)t_{\text{mul}} + 3(d^2+d)t_{\text{mul}} + 2\sqrt{\cdot}$, respectively. Hence the computational *overheads* for the user and server are $2(2n+5)t_{\text{mul}}$ and $2(2n+4)t_{\text{mul}}$, respectively (i.e., subtract the traditional algorithm's complexity from proposed algorithm's complexity).

In order to evaluate the complexity, we implemented the proposed scheme on a computer - Intel(R) Core(TM) i5-4210U CPU @1.70GHz with 8GM RAM - using Matlab 2016a. We modified the iVector library from GitHub (github.com/pedrocolon93/ivectormatlabmsrit) to implement the proposed scheme. Using this implementation, we tested the complexity of the proposed scheme for different values of n. We performed 50 iterations of the proposed scheme by varying the input size n from 10^3 to 10^6. The average time taken is illustrated in Figure 8. The computational time increases linearly up to $n = 10^5$. From $n = 10^5$, the time increases exponentially due to processing large amount of data in a sequential order. This problem can be solved by parallel processing by executing the scalar product computation in multiple threads. For example, if $n = 6$, instead of calculating $[a_1\ a_2\ a_3\ a_4\ a_5\ a_6].[b_1\ b_2\ b_3\ b_4\ b_5\ b_6]^T$ sequentially, the problem can be split into two: $[a_1\ a_2\ a_3].[b_1\ b_2\ b_3]^T$ and $[a_4\ a_5\ a_6].[b_4\ b_5\ b_6]^T$. The results can be added in the end.

For the experiment in Section V, the dimension of iVector has been set to $d = 200$ [1]. Therefore the sizes of the input vectors in (9) for Table II, are in the range of $n = 40000$ to $n = 80000$. This is within the linear time complexity range i.e., the incurred computational overhead is less than 0.05 seconds for both the user and server.

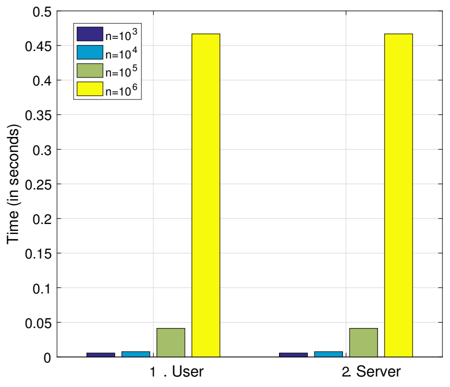


Figure 8. Computational overheads of the proposed scheme for user and server.

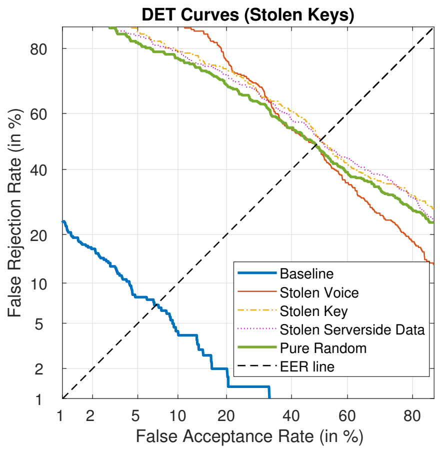


Figure 9. Stolen key attack scenarios

VI. Security and Privacy Analysis

Since the proposed algorithm relies on randomisation, the following subsections provide a formal security proof for the proposed randomisation algorithm in Section VI-A and a privacy analysis for the proposed SA algorithm in Section VI-B.

A. Security model and proof for the proposed randomisation algorithm in Section VI-A

This section proves that the proposed randomisation algorithm in Section IV-A satisfies the information-theoretic security. Denote a mapping $f : \mathcal{M} \times \mathcal{S} \longmapsto \mathcal{R}$. We call such a mapping f over a message space $\mathcal{M}$ to be perfectly random if and only if for an uniform probability distribution s over $\mathcal{S}$, every message $m \in \mathcal{M}$ and every randomised message $r \in \mathcal{R}$, probability $P[M = m|R = r]$ is constant greater than zero, i.e., looking at the randomised message no one can guess the message. Theorem 1 shows that Algorithm 1 in Section IV-A attains perfect randomisation.

Theorem 1: Let $\mathcal{M} = [-a, a] \cap \mathbb{Z}$ and $\mathcal{S} = [v_1, v_2] \cap \mathbb{Z}$ be two sets such that $a < v_1 < v_1 + a < v_2$. Also let
$\mathcal{R} = \{m + s; m \in \mathcal{M}, s \in \mathcal{S}, v_1 + a \le m + r \le v_2 - a\}$.
Then for any $r \in \mathcal{R}$, $P[M = m|R = r] = \frac{1}{|\mathcal{S}|}$.

Proof: Let $r \in \mathcal{R}$ and $m \in \mathcal{M}$. Then, it is easy to check that $v_1 \le r - m \le v_2$. So,

$$P[M = m|R = r] = P[r - s = m|R = r]$$
$$= P[r - s = m] = P[s = r - m] = \frac{1}{|\mathcal{S}|}$$

■

The perfect randomness leads to adaptive indistinguishability. But before giving the proof, we first consider the definition of adaptive indistinguishability game.

Definition 1: $[\text{Gm\_Ad}_{\mathcal{A},\pi}(1^s)]$

1. The adversary $\mathcal{A}$ is given oracle access to $\mathrm{Enc}_s(.)$ and outputs a pair of messages $m_0, m_1 \in \mathcal{M}$ of the same length.
2. Random bit $b \leftarrow \{0,1\}$ is chosen, and $s \leftarrow \mathcal{S}$ is also chosen randomly. Then a ciphertext $r = s + m_b$ is computed and given to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ continues to have oracle access to $\mathrm{Enc}_s(.)$ and outputs a bit b'.
4. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. In case $\mathrm{Gm\_Ad}_{\mathcal{A},\pi}(1^s) = 1$, we say that $\mathcal{A}$ succeeded.

Definition 2: An encryption scheme, denoted by $\pi = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$, is said to be adaptively secure under chosen plain text attack if for all probabilistic polynomial time adversaries $\mathcal{A}$, there exists a negligible function negl such that $P[\mathrm{Gm\_Ad}_{\mathcal{A},\pi}(1^s) = 1] \leq \frac{1}{2} + \mathrm{negl}(s)$, where the probability is taken over the random coins used by $\mathcal{A}$, as well as the random coins used in the game.

Let us consider the encryption scheme $\pi' = (\mathrm{Gen}, \mathrm{Enc}_s, \mathrm{Dec}_s)$ such that $\mathrm{Gen}()$ samples uniformly at random a key s from the set $\mathcal{S}$, i.e., $P[S = s] = \frac{1}{|\mathcal{S}|}$. For any $m \in \mathcal{M}$, $\mathrm{Enc}_s(m) = m + s$ and for any $r \in \mathcal{R}$, $\mathrm{Dec}_s(r) = r - s$. Following theorem shows that this scheme is adaptively secure.

Theorem 2: π' is adaptively secure under chosen plain text attack.

Proof: Let $|\mathcal{S}| \approx 2^\lambda$. Let us consider the game $\mathrm{Gm\_Ad}_{\mathcal{A},\pi'}(1^\lambda)$. Note that $\mathcal{A}$ being a polynomial adversary, may call the encryption oracle polynomial (in λ) number of times before receiving challenge cipher. Let this polynomial be $p(\lambda)$. Let Repeat be the event that the key used in challenge phase is used in any of the previous calls.

Note that $P[(\mathrm{Gm\_Ad}_{\mathcal{A},\pi'}(1^\lambda) = 1) \wedge \mathrm{Repeat}] \leq P[\mathrm{Repeat}] = \frac{p(\lambda)}{2^\lambda}$.

Also when Repeat does not occur, adversary has absolutely a random view and thus,
$P[(\mathrm{Gm\_Ad}_{\mathcal{A},\pi'}(1^\lambda) = 1) \wedge \overline{\mathrm{Repeat}}] = P[\overline{\mathrm{Repeat}}] \times P[(Gm\_Ad_{\mathcal{A},\pi'}(1^\lambda) = 1)|\overline{\mathrm{Repeat}}] \leq P[(Gm\_Ad_{\mathcal{A},\pi'}(1^\lambda) = 1)|\overline{\mathrm{Repeat}}] = \frac{1}{2}$.
So,

$$P[\mathrm{Gm\_Ad}_{\mathcal{A},\pi'}(1^\lambda) = 1] = P[(\mathrm{Gm\_Ad}_{\mathcal{A},\pi'}(1^\lambda) = 1) \wedge \mathrm{Repeat}]$$
$$+ P[(\mathrm{Gm\_Ad}_{\mathcal{A},\pi'}(1^\lambda) = 1) \wedge \overline{\mathrm{Repeat}}] \leq \frac{p(\lambda)}{2^\lambda} + \frac{1}{2}$$

We note that $\frac{p(\lambda)}{2^\lambda}$ is negligible in λ which completes the proof. ■

B. Privacy Analysis for the proposed SA algorithm in Section VI-B

The ultimate aim of the proposed algorithms is to protect user voice biometrics stored at and transmitted to the server. Both the proposed BA and SA algorithms randomise the voice feature vectors using random vectors and invoke two-party computation. During the two-party computations, if the user and server exploit the proposed randomisation algorithm to mask the data, as shown in the previous section, then the randomised data is information theoretically secure. Hence lets prove that the proposed SA algorithm does not leak any unintended data during the two-party computation.

1) Privacy proof for SA algorithm: During the enrolment process, the user device randomises the ivector $\mathbf{w}_1$ using the proposed randomisation algorithm and sends only the randomised ivector $\mathbf{w}_1 + \mathbf{r}_1$ and $\mathbf{w}_1^x = \mathbf{w}_1 + \mathbf{r}_x$ to the server and stores the random vectors $\mathbf{r}_1$ and $-\mathbf{r}_x$ in the user device. Then the user device deletes $\mathbf{w}_1$ and $\mathbf{r}_x$. After this enrolment process, the server holds $\mathbf{w}_1 + \mathbf{r}_1$ and $\mathbf{w}_1^x = \mathbf{w}_1 + \mathbf{r}_x$ while the user holds $\mathbf{r}_1$ and $\mathbf{w}_1^y = -\mathbf{r}_x$. Hence, even if the server has been compromised by an adversary after the enrolment, it is information-theoretically infeasible for the adversary to retrieve $\mathbf{w}_1$ from $\mathbf{w}_1 + \mathbf{r}_1$ and $\mathbf{w}_1^x = \mathbf{w}_1 + \mathbf{r}_x$ without $\mathbf{r}_1$ and $\mathbf{r}_x$. Similarly, if the user device which holds $\mathbf{r}_1$ and $\mathbf{w}_1^y = -\mathbf{r}_x$ is being compromised by an adversary, it is information-theoretically infeasible for the adversary to retrieve $\mathbf{w}_1$. To launch a successful attack, the adversary needs to compromise both the server and user device, which is an extreme condition and out of the scope of this paper.

During the verification stage, the ivector $\mathbf{w}_2$ is again randomised into $\mathbf{w}_2 + \mathbf{r}_2$ where the user device keeps $\mathbf{r}_2$ and the server gets $\mathbf{w}_2 + \mathbf{r}_2$. Similar to the above discussion, $\mathbf{w}_2$ cannot be obtained from $\mathbf{w}_2 + \mathbf{r}_2$. However, in order to get the true score, the user device and server need to perform the two-party computation using the PP scalar product algorithm in Table II. As shown in [18], the security of the algorithm in Table II relies on randomisation (User's inputs $a_1, a_2, \ldots$ are randomised by large random numbers $c_1, c_2, \ldots$) and achieves information-theoretic security.

C. Privacy Leakage Analysis

The previous section provided a theoretical proof showing that the proposed algorithm is information theoretically secure. To visualise this and analyse whether the randomised features still preserve the statistical properties of speech feature, a numerous experiments are conducted in this section. We can broadly split the parameters required for a successful verification into four: 1) voice 2) randomised iVector ($\mathbf{w}_1+\mathbf{r}_1$) 3) parameters stored on the user device and 4) server-side parameters. In order to evaluate the strength of the proposed algorithm, the following four attacks are considered:

1. Compromised user device attack
2. Compromised server attack
3. Compromised user voice attack
3. Pure random attack

1) Compromised user device attack: In this attack, the adversary has access to the user device and the parameters stored during the enrolment. But do not have access to the user voice to generate legitimate test ivector. Hence, the adversary tries to combine the parameters from the compromised user device with the test features of *other users*. Then the adversary tries to verify against the compromised user's speaker model residing at the server. To evaluate this, $2 \times 150 \times 151$ tests [300 test utterances from other users are combined with the parameters of the compromised user device and this is repeated for all the users] are conducted and the corresponding decision scores are obtained.

2) Compromised server attack: In this attack, the adversary has access to the randomised ivectors $\mathbf{w}_1 + \mathbf{r}_1$ of all the users stored at the server. Let's also assume that the adversary holds the feature vectors of all users but neither know the corresponding ivectors stored at the server or keys stored at the user device. Now the attacker uses these randomised ivectors to simulate a speaker verification system and tries to find out the corresponding users for each stolen randomised ivector. Hence, the adversary tries to measure the decision score by applying those features against each and every randomised ivector. Again $2 \times 151 \times 151$ tests are conducted and the corresponding decision scores are obtained.

3) Compromised user voice attack: In this attack, the attacker has access to the user's voice recording but does not have access to the parameters stored at the user device. Now the attacker generates random numbers and randomises the voice feature and tries to impersonate. Hence, this experiment generates 300 random vectors same size as the feature to obtain 300 randomised test features. To analyse the performance, 300×151 tests are conducted and the corresponding decision scores are obtained.

4) Pure random attack: In this final test, the traditional solution has been considered but instead of using the legitimate test features, purely random vectors in the same domain and same size as the legitimate test ivector used. Hence, we generate 300 random vectors for each user and conducted 300×151 tests.

Figure 9 displays the DET curves for the above attacks. In the same figure, we displayed the baseline model. Interestingly, from Figure 9, the equal error rate for all four attacks are around 50% and there is no significant difference between the first three attacks against the pure random attacks (the fourth attacks). This clearly shows that there are no advantages for an adversary who compromises the parameters of the proposed systems than just launching random attacks. This concludes that the proposed algorithm is information theoretically secure and all four parameters must be combined to reveal the statistical properties.

VII. Conclusion

In this paper, an efficient privacy preserving speaker verification protocol is proposed. To achieve better efficiency and privacy, the proposed solution algorithmically redesigned the iVector and linear discriminant analysis based speaker verification techniques to incorporate randomness without affecting the final outcome. The proposed scheme is based on randomisation technique and it only relies on multiplication and addition. In this scheme, two parties involved, the user and the server, need to perform verification interactively. In addition, it is proved using information-theoretic security that the algorithm is secure. It is also shown empirically that the proposed scheme provides good overall accuracy without increasing the computational overhead.

References

[1] Dehak, N., Kenny, P.J., Dehak, R., Dumouchel, P. and Ouellet, P., 2011. Front-end factor analysis for speaker verification. IEEE Transactions on Audio, Speech, and Language Processing, 19(4), pp.788-798.

[2] Jain, A. K., and Nandakumar, K. "Biometric Authentication: System Security and User Privacy," *Computer*, vol. 45, no. 11, 87-92, 2012.

[3] Pathak, M.A.; Raj, B., "Privacy-Preserving Speaker Verification and Identification Using Gaussian Mixture Models," *IEEE Trans. Audio, Speech, and Language Processing* , vol.21, no.2, pp.397-406, Feb. 2013

[4] Reynolds, D.A.; Rose, R.C., "Robust text-independent speaker identification using Gaussian mixture speaker models,"*IEEE Trans. Speech and Audio Processing* , vol.3, no.1, pp.72,83, Jan 1995.

[5] Rahulamathavan, Y., Rajarajan, M. "Efficient Privacy-preserving Facial Expression Classification," *IEEE Trans. Dependable and Secure Computing* , in press.

[6] Garofolo, John, et al. "TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1," Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[7] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proc. 9th International Symposium on Privacy Enhancing Technologies*, PETS '09, pp. 235–253, 2009.

[8] Y. Rahulamathavan, R. Phan, S. Veluru, K. Cumanan, and M. Rajarajan, "Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud," *IEEE Trans. Dependable Secure Computing*, vol. 11, no. 5, pp. 467–479, Sept. 2014.

[9] Y. Rahulamathavan, S. Veluru, R. Phan, J. Chambers, and M. Rajarajan, "Privacy-preserving clinical decision support system using gaussian kernel based classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 1, pp. 56–66, Jan. 2014.

[10] Y. Rahulamathavan, R. Phan, J. Chambers, and D. Parish, "Facial expression recognition in the encrypted domain based on local fisher discriminant analysis," *IEEE Trans. Affective Computing*, vol. 4, no. 1, pp. 83–92, Jan.-Mar. 2012.

[11] M. A. Pathak, B. Raj, S. D. Rane and P. Smaragdis, "Privacy-preserving speech processing: cryptographic and string-matching frameworks show promise," in *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 62-74, March 2013.

[12] Dehak, N, et al. "Front-end factor analysis for speaker verification." IEEE Transactions on Audio, Speech, and Language Processing, vol. 19, no. 4 pp. 788-798, 2011.

[13] C. E. Shannon, "Communication theory of secrecy systems*," *Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[14] Smaragdis, P.; Shashanka, M.V.S., "A Framework for Secure Speech Recognition," *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* , vol.4, no., pp.IV-969,IV-972, 15-20 April 2007

[15] O. Goldreich, "Secure multiparty computation", (working draft), available: http://www.wisdom.wei zmann.ac.il/ oded/pp.html. (Sep. 1998)

[16] J. Portêlo, B. Raj, A. Abad and I. Trancoso, "Privacy-preserving speaker verification using secure binary embeddings," *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*, Opatija, 2014, pp. 1268-1272.

[17] M. A. Pathak and B. Raj, "Privacy-preserving speaker verification as password matching," *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, Kyoto, 2012, pp. 1849-1852.

[18] R. Lu, H. Zhu, X. Liu, J. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," Network, IEEE, vol. 28, no. 4, pp. 46–50, July 2014.

[19] L. Burget, O. Plchot, S. Cumani, O. Glembek, P. Matějka, and N. Brümmer, "Discriminatively trained probabilistic linear discriminant analysis for speaker verification," In Int'l Conf. IEEE Acoustics, Speech and Signal Processing (ICASSP), pp. 4832-4835, May 2011.