PRIVACY-PRESERVING SPEAKER VERIFICATION USING GARBLED GMMS

José Portêlo[12]*, Bhiksha Raj*[3]*, Alberto Abad*[12]*, Isabel Trancoso*[12]

[1] INESC-ID Lisboa, Portugal; [2] Instituto Superior Técnico, Lisboa, Portugal
[3] Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA

ABSTRACT

In this paper we present a privacy-preserving speaker verification system using a UBM-GMM technique. Remote speaker verification services rely on the system having access to the user's recordings, or features derived from them, and a model representing the user's voice. Preserving privacy in our context means that neither the system observes voice samples or speech models from the user nor the user observes the universal model owned by the system. Our approach uses Garbled Circuits for obtaining an implementation that simultaneously is secure, has high accuracy and is efficient. To the best of our knowledge this is the first privacy-preserving speaker verification system that accomplishes all these three goals.

Index Terms— Speaker Verification, Gaussian Mixture Models, Garbled Circuits, Data Privacy

1. INTRODUCTION

Given the rapid increase of storage capability over the Internet, a widespread development of a variety of online services occurred, ranging from banking to social networks, video organizers, blogs, online games, etc. All these services have in common the fact that users must first register with them and receive a unique user ID and a written password with which they can authenticate themselves to the system. In many situations, an attractive alternative is to authenticate users using their voice, since it is a more natural way of communication.

Voice-based authentication systems, also referred to as speaker verification systems, however, have significant privacy concerns. Current systems require access to a person's voice, or at least parameterized versions of it. An individual's voice not only is a biometric identifier of a person but also carries additional information about a person's gender, nationality, etc. This means that a malicious system, or a hacker who has compromised the system, could both edit the recordings to impersonate the user and use the additional information for further potential abuse, e.g., sell it to third parties. Even if the user only transmits features extracted from the voice, such that a recording cannot be synthesized from them, some risks remain. Ideally, the service should not have access to any biometric information, though it seems nearly paradoxical to require the system not to have access to many key voice characteristics that themselves help establish the user's identity.

Consider the situation where a user wishes to register himself with an online service provider using his own voice. A typical approach to implementing a speaker verification system is to consider a Gaussian mixture model (GMM) technique [1]. Since the user wants to be able to use his voice instead of typing a password, he first needs to obtain a model for his own voice. In the enrollment phase, the online system can provide an application with a built-in universal background model (UBM), to which the user can supply voice samples in order to obtain an adapted model. The user then sends this adapted model to the system. Later, during the verification phase, the user tries to login to his account and sends new voice recordings (or features derived from them) to the system, who then decides on whether or not to authenticate the user by evaluating these recordings against the user-specific model and the UBM. As an additional security layer, during the enrollment phase the user can apply a cryptographic hash function (e.g. SHA-2 [2]) to his model and send the output to the system, allowing it to be used as a password during the verification phase. As mentioned before, in this situation the user's identity may be compromised, since he must supply the system with features and a model representing his own voice.

To counter this issue, in this paper we propose a novel approach for accomplishing speaker verification in a privacy-preserving manner in scenarios such as the one described above. We reformulate the GMM-based technique for speaker verification by performing all the required operations using a cryptographic primitive known as the Garbled Circuit (GC) [3]. In our approach no private information is transmitted, only ciphers corresponding to the bits representing data values. The ciphers are transmitted using Oblivious Transfer (OT) [4] and all the operations are performed in the encrypted domain, so none of the parties involved can learn anything about the other parties except for the ciphers themselves (but not what they represent). Using our technique, the privacy requisites are fulfilled, since the system obtains all the necessary data for performing an informed user authentication decision but is unable to use it in any other way, due to it being encrypted.

The remainder of this paper is structured as follows. In Section 2 we briefly describe the relevant concepts for this work, namely speaker verification using GMMs and Garbled Circuits. Section 3 contains some of the previous work on

the privacy-preserving speaker verification task, on which we build upon. In Sections 4 and 5 we describe our technique for privacy-preserving speaker verification using Garbled Circuits and illustrate its performance with some experimental results, respectively. Finally, we present some conclusions and directions for future work in Section 6.

2. BACKGROUND

2.1. Speaker Verification using GMMs

Conventional speaker verification systems require users to provide voice samples to the system during an enrollment phase. The system then uses these samples to build a model for the corresponding user. Finally, the user provides fresh samples to be compared with his model during the verification phase. These speaker verification systems normally perform a likelihood ratio test to confirm the user's identity. Each sample $\mathbf{x}$ is first parameterized into a sequence of feature vectors $x_1, \ldots, x_T$, usually mel-frequency cepstral coefficient (MFCC) vectors. A large collection of recordings from non-target speakers is then used to train a universal background model (UBM), λ_U, which is a Gaussian mixture model (GMM) representing the global distribution of speech. The UBM is adapted to each user's enrollment recording in order to obtain user-specific GMMs, λ_S, through maximum a posteriori (MAP) adaptation [1]. Performing MAP adaptation ensures a one-to-one correspondence between the Gaussians in the UBM and the ones in the user models. The likelihood of a feature vector x_t given by either a UBM or a user-adapted GMM composed of M Gaussians is computed by:

$$P(x_t|\lambda_{S/U}) = \sum_{i=1}^{M} w_i^{S/U} \mathcal{N}\left(x_t, \mu_i^{S/U}, \Sigma_i^{S/U}\right), \quad (1)$$

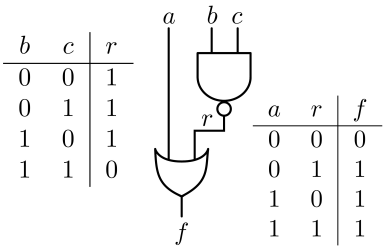
where $\mathcal{N}\left(x_t, \mu_i^{S/U}, \Sigma_i^{S/U}\right)$ is the i^{th} multivariate Gaussian with mean $\mu_i^{S/U}$ and covariance $\Sigma_i^{S/U}$ computed at the feature vector x_t and $w_i^{S/U}$ is the relative weight of that Gaussian in the mixture. This process is repeated for all feature vectors in $\mathbf{x}$. Finally, the likelihood ratio is computed and a verification decision is made using:

$$\frac{P(\mathbf{x}|\lambda_S)}{P(\mathbf{x}|\lambda_U)} \begin{cases} \geq \theta & \text{accept user,} \\ < \theta & \text{reject user,} \end{cases} \quad (2)$$

where $P(\mathbf{x}|\lambda_{S/U})$ is the product of all the $P(x_t|\lambda_{S/U})$ with $t = 1, \ldots, T$ and θ is a predefined threshold parameter. Since typical values for $P(x_t|\lambda_{S/U})$ are usually prone to underflow issues, their logarithms are normally considered instead.

2.2. Garbled Circuits

Secure Function Evaluation (SFE) [3] refers to the framework whereby two parties can compute a function on their combined inputs without revealing their individual inputs to one another. Several approaches can be considered for performing SFE. The most popular is to express the function to be evaluated as a *garbled* Boolean circuit, usually referred to as a Garbled Circuit (GC) [3]. The way GCs work is best explained using an example. Alice has Boolean inputs a and b, Bob has Boolean input c. They wish to compute $f(a, b, c) = a \vee \overline{b \wedge c}$. The logic circuit for $f(a, b, c)$ and the truth tables for the gates in it are shown in Figure 1. However, Alice does not want to disclose a or b to Bob, and Bob does not want to disclose c to Alice. In order to still be able to compute $f(a, b, c)$, they must therefore *garble* the circuit.



b	c	r
0	0	1
0	1	1
1	0	1
1	1	0

a	r	f
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 1. Example of a logic circuit and respective truth tables.

b	c	r
K_b^0	K_c^0	$\mathcal{E}_{K_b^0}(\mathcal{E}_{K_c^0}(K_r^1))$
K_b^0	K_c^1	$\mathcal{E}_{K_b^0}(\mathcal{E}_{K_c^1}(K_r^1))$
K_b^1	K_c^0	$\mathcal{E}_{K_b^1}(\mathcal{E}_{K_c^0}(K_r^1))$
K_b^1	K_c^1	$\mathcal{E}_{K_b^1}(\mathcal{E}_{K_c^1}(K_r^0))$

a	r	f
K_a^0	K_r^0	$\mathcal{E}_{K_a^0}(\mathcal{E}_{K_r^0}(0))$
K_a^0	K_r^1	$\mathcal{E}_{K_a^0}(\mathcal{E}_{K_r^1}(1))$
K_a^1	K_r^0	$\mathcal{E}_{K_a^1}(\mathcal{E}_{K_r^0}(1))$
K_a^1	K_r^1	$\mathcal{E}_{K_a^1}(\mathcal{E}_{K_r^1}(1))$

Table 1. Example of garbled truth tables.

Bob starts by generating the logic circuit implementing $f(\cdot)$, garbling it, and sending it to Alice. To do so he generates two private keys, one for each bit value, for each input and intermediate value (a, b, c, r in the example), totaling eight keys: $K_a^{0/1}$, $K_b^{0/1}$, $K_c^{0/1}$, $K_r^{0/1}$. For gates generating intermediate values ($r = \overline{b \wedge c}$ in our example), he replaces each output of the truth table with the encryption of the key corresponding to that output, performed with the keys corresponding to the inputs. For example, for $b = 1$, $c = 0$, the output is $r = 1$; the corresponding input keys are K_b^1, K_c^0 and the encrypted output is $\mathcal{E}_{K_b^1}(\mathcal{E}_{K_c^0}(K_r^1))$. For the output gates ($f = a \vee r$ in our example), he encrypts the output bit itself. The garbled values for our example are presented in Table 1.

To compute the function, Bob transmits the keys corresponding to his bit choice for his inputs, $K_c^?$, to Alice. Alice recovers the keys corresponding to the bit choice for her inputs, $K_a^?$ and $K_b^?$, as well as the truth tables for the circuit gates from Bob using oblivious transfer (OT) [4]. Because of the properties of OT, Bob does not learn either the value of Alice's bits or the entries of the truth table she actually requires to perform the computations. To evaluate the circuit, Alice successively evaluates each of the gates in the circuit. The nature of the computation is such that for each gate, Alice will possess two keys, one for each input. She decrypts

all four encrypted outputs in the truth table for the gate, using these two keys. The keys will be inappropriate for three of the four outputs; hence Alice can only correctly decipher one of the four encrypted values, which in turn will be one of the keys for the next gate. After repeating this process for all gates, she finally obtains the output value f. Alice and Bob never learn each others' inputs.

For a long time it was believed that GCs were of purely theoretical interest, but recent advances have made GC much more efficient and practical to use. These advances include offline execution of OT, fast decryption of the garbled tables, efficient evaluation of XOR gates and consequent custom design of circuits, etc., and have decisively contributed to the wide-spread popularity of GCs. As a consequence, a huge variety of software implementations of all the required protocols and encryption systems are available.

3. PREVIOUS WORK

In [5] a privacy-preserving method for speaker verification using GMM is presented. The personal information from each party is kept hidden from the other party by performing all operations using a partially homomorphic encryption system [6]. In short, the main idea behind these systems is to allow operations to be performed on encrypted data (ciphertext) without any knowledge regarding the corresponding unencrypted data (plaintext). The method presented follows the construction described in [7] and specifies private protocols for the enrollment and verification phases.

In this method, each multivariate weighted Gaussian from Equation 1 is represented as a $(N+1) \times (N+1)$ matrix $\overline{G_i}$:

$$\overline{G_i} = \begin{bmatrix} -\frac{1}{2}\Sigma_i^{-1} & \Sigma_i^{-1}\mu_i \\ \hline 0 & \overline{g_i} - \frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i \end{bmatrix},$$

$$\overline{g_i} = \log w_i - (N/2)\log(2\pi) - (1/2)\log|\Sigma_i|, \quad (3)$$

where N is the size of the feature vector x_t, leading to $\log\left(w_i\,\mathcal{N}\left(x_t, \mu_i, \Sigma_i\right)\right) = \overline{x_t}^T\overline{G_i}\overline{x_t}$, with $\overline{x_t} = [x_t|1]^T$. The expression for obtaining the log-likelihood between each Gaussian and the feature vector is further simplified into computing a single scalar product:

$$\log P(x_t|i) = \log\left(w_i\,\mathcal{N}\left(x_t, \mu_i, \Sigma_i\right)\right) = \hat{x_t}^T\hat{G}_i, \quad (4)$$

where $\hat{x_t}$ is an extended feature vector composed by pairwise products $\overline{x_t}^k\overline{x_t}^l$ and $\hat{G}_i$ is a linearization of $\overline{G_i}$, both with size $L = (N+1)^2$.

The privacy-preserving method just described is able to achieve interesting speaker verification results, with only slight degradation when compared to their non-private counterpart. However, because of intrinsic characteristics of homomorphic encryption systems, this approach is very inefficient. This happens because both private protocols must be split into several steps, each requiring successive encryptions, decryptions and input randomizations. Naturally, this leads to substantial computation overheads and large execution times.

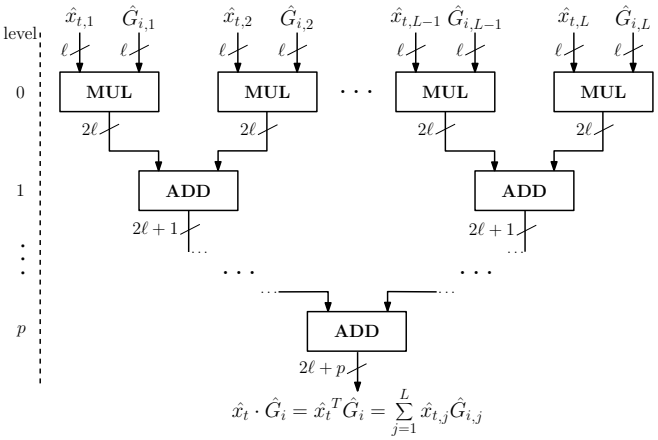


Fig. 2. Scalar product using Garbled Circuits.

We are aware of more sophisticated techniques for performing speaker verification, in particular using *i-vectors* [8] instead of the UBM-GMM approach. In fact, we have previously experimented with i-vectors together with a distance-preserving hashing technique called Secure Binary Embeddings (SBE) [9] for obtaining a privacy-preserving speaker verification system [10]. However, in this work we decided to illustrate our Gabled Circuit approach by comparing it with the work presented in [5].

4. PRIVACY-PRESERVING SPEAKER VERIFICATION USING GARBLED CIRCUITS

This section describes how Garbled Circuits can be used to implement a speaker verification system where each of the participating parties keep their personal data secret from the others. In our technique the user is responsible for generating the GCs and the system is responsible for evaluating them and deciding on whether or not to authenticate the user. We start by presenting the scalar product operation followed by the logsum operation, then we show how they are combined to perform a GMM evaluation, and finally we describe how the log-likelihood ratio can be computed.

4.1. Scalar product using Garbled Circuits

The scalar product is a simple linear operation consisting of multiplications and additions. A diagram illustrating the scalar product between two arrays $\hat{x_t}$ and $\hat{G}_i$ is presented in Figure 2. Notice that if each element of each input array is represented using ℓ bits, the output value $\hat{x_t}^T\hat{G}_i$ needs in theory at most $2\ell + p$, $p = \lceil \log L \rceil$, bits to be represented. However, in many real situations this is often not the case, and we will address this problem in Section 5.

4.2. Logsum operation using Garbled Circuits

The logsum is a cornerstone operation to many signal processing scenarios, and it is required to compute the occurrence probabilities of events in situations where underflow or overflow problems are likely to appear. The expression for computing the logsum of an array $\hat{x_t}^T\hat{G}$ of M elements is presented in Equation 5:

$$LS\left(\hat{x_t}^T\hat{G}\right) = \log\left(\sum_{i=1}^{M}\exp\left(\hat{x_t}^T\hat{G}_i\right)\right)$$

$$= m_X + \log\left(\sum_{i=1}^{M}\exp\left(\hat{x_t}^T\hat{G}_i - m_X\right)\right), \quad (5)$$

where $m_X = \max_i\left(\hat{x_t}^T\hat{G}_i\right)$. The logsum over all the elements of the array can be further simplified by splitting it into partial sums of two elements each. Since the logsum is a non-linear operation, in order to make it practical to implement using Garbled Circuits, each partial sum must be cast into a linear piecewise approximation, leading to the expression in Equation 6:

$$LS\left(\hat{x_t}^T\hat{G}_i, \hat{x_t}^T\hat{G}_j\right) = m_X + \log(1+\exp(m_N - m_X))$$

$$\approx m_X + (n_1 \cdot (m_N - m_X) + n_2), \quad (6)$$

where m_X and m_N are the maximum and minimum of $\left(\hat{x_t}^T\hat{G}_i, \hat{x_t}^T\hat{G}_j\right)$, respectively, and n_1, n_2 are the linearization parameters. Similarly to the previous section, here the increase in the number of bits ℓ is also a relevant matter, and it will be further discussed in Section 5. A more complete and detailed analysis of the logsum operation using GC can be found in [11].

4.3. GMM evaluation using Garbled Circuits

Given the algorithms for computing the scalar product and the logsum with GC, performing a GMM evaluation becomes a problem trivial to solve. The diagram showing it can be done when $M = 4$ Gaussians are considered is presented in Figure 3. The 4-LOGSUM block illustrates how the logsum of M elements can be computed as the expense of several logsums of two elements each.

4.4. Log-likelihood ratio using Garbled Circuits

The last step is to perform the ratio between the log-likelihoods of sample $\mathbf{x}$ with the UBM, λ_U, and the user-adapted GMM, λ_S. Since both of them are represented as logarithms, for obtaining $\log(P(\mathbf{x}|\lambda_{S/U}))$ one must sum all the values $\log(P(x_t|\lambda_{S/U}))$ (instead of multiplying them) and the ratio must be computed as a subtraction (instead of a division).

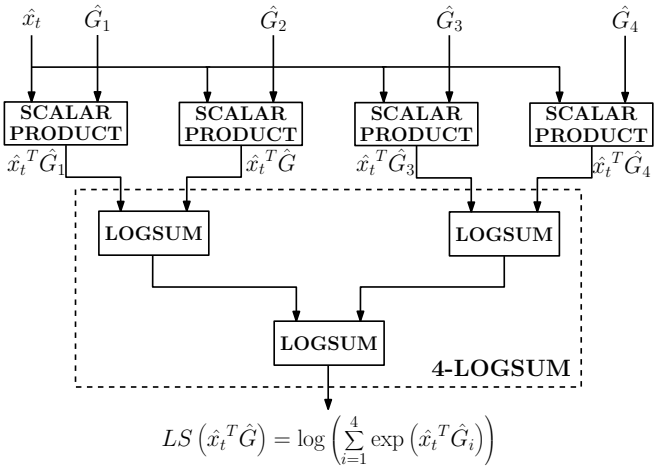


Fig. 3. Evaluation of a Garbled GMM for $M = 4$.

#Gaussians	16	32	64	128	256
non-private approach	5.18	4.04	2.99	2.03	1.43
GC approach ($\ell = 16$)	6.00	4.73	3.06	2.09	1.60

Table 2. Speaker verification results, EER (%age).

5. EXPERIMENTAL RESULTS

For the sake of comparison with [5], we ran experiments on the YOHO Speaker Verification Corpus [12], consisting of short utterances produced by 138 different speakers. Each utterance contains a sequence of three two-digit numbers. The recordings were sampled at 8kHz and stored as 16-bit words. The corpus is split into two sets: enrollment and verification. The enrollment set contains 96 utterances from each speaker, totaling 14.54 hours of audio, and the verification set contains 40 utterances from each speaker, totaling 6.24 hours. All speech signals were parameterized into MFCC features extracted from 25ms frames at the rate of 100 frames per second. For each audio frame, we extracted 13 MFCC coefficients augmented with temporal Δ and $\Delta\Delta$, resulting in 39 features per frame. The UBM was trained using all the enrollment set, and each speaker-specific model was obtained by adapting the UBM to the enrollment data from each speaker. We evaluated the verification set against the UBM, the correct speaker-adapted model (positive trials) and all other speaker models (negative trials). Afterwards, the obtained values were used to compute the log-likelihood ratio.

The results obtained in terms of EER using both the regular speaker verification approach and our privacy-preserving approach using GC are presented in Table 2. It is straightforward to verify that increasing the total number of Gaussians leads not only to reduced error rates but also to smaller gaps between our approach and the baseline. Since in both approaches the computations performed are exactly the same, these gaps can only be justified by the number of bits ℓ considered and/or by the additional bits that are ignored after the

#Gaussians	16	32	64	128	256
t_{mean}	0.089	0.179	0.360	0.723	1.436
t_{max}	0.092	0.185	0.376	0.777	1.476

Table 3. Speaker verification results, execution time (sec).

scalar products and the logsum operations. As mentioned in Sections 4.1 and 4.2, each of these operations introduces many additional bits to the output. Regarding the scalar product, for the corpus and feature extraction process we considered, preliminary experiments revealed that all additional bits could be discarded, as many most significant bits had the value '0' and ignoring some least significant bits did not significantly change the output values. As for the logsum, according to the work presented in [11], ignoring all but one of the additional bits does not seem to visibly affect the results. Therefore, it is reasonable to assume that the difference between the results is due to $\ell = 16$ bit numbers being considered instead of a full 32-bit floating point representation.

Since we wanted our approach to be as practical as possible to implement in real-life situations, we also analyzed it in terms of execution time. The results we obtained are presented in Table 3. The results presented correspond to mean and maximum time it takes, in seconds, to evaluate a Garbled GMM, i.e., to compute a single $\hat{x_t}^T \hat{G}$, when $\ell = 16$ is considered. They were obtained by running the experiments on an Intel Core i7-3630QM CPU @ 2.40GHz. We notice that in all experiments the execution times are extremely fast, given that a privacy-preserving approach is considered. Also, the execution time scales linearly with the number of Gaussians. We only present the execution times for evaluating individual frames since all the required evaluations are independent from each other and may be computed in parallel. However, even if we consider that only a single computer core is available, the overall execution time would be approximately 300 seconds (evaluating $\hat{x_t}^T \hat{G}^U$ and $\hat{x_t}^T \hat{G}^S$ for all the frames extracted from a 4-second file sampled at 100 frames per second) for the situation where 64 Gaussians are considered, which is still much faster than the approach presented in [5].

6. CONCLUSIONS AND FUTURE WORK

In this work we presented a privacy-preserving system using Garbled Circuits for performing a speaker verification task. Our approach is able to obtain similar results to the non-private counterpart, but at the same time guarantees that each of the participants in the protocol does not reveal his/her private information to others. A major advantage of our approach is that it does not require large computational overheads and is very efficient in terms of execution times. For future work we plan to analyze the impact of changing the control parameters in our approach, as well as experimenting with a larger number of Gaussians for representing the models and other corpora [13].

7. ACKNOWLEDGEMENTS

José Portêlo, Alberto Abad and Isabel Trancoso were supported by FCT grants SFRH/BD/71349/2010, PTDC/EIA-CCO/122542/2010 and PEst-OE/EEI/LA0021/2013. Bhiksha Raj was supported by NSF grant number 1017256.

REFERENCES

[1] D. A. Reynolds, T. F. Quatieri and R. B. Dunn, "Speaker Verification using Adapted Gaussian Mixture Models", in *Digital Signal Processing*, Vol. 10, Issue 1, pp. 19–41, January 2000.

[2] ——, "Secure Hash Standard", *FIPS PUB 180-4*, National Institute of Standards and Technology (NIST), 2012. Available at `www.csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf`.

[3] A. Yao, "Protocols for Secure Computation", in *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, Chicago, Illinois, USA, 1982, pp. 160–164.

[4] M. Naor and B. Pinkas, "Oblivious Transfer and Polynomial Evaluation", in *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC)*, Atlanta, Georgia, USA, 1999, pp. 245–254.

[5] M. Pathak and B. Raj, "Privacy-Preserving Speaker Verification and Identification using Gaussian Mixture Models", in *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 21, No. 2, pp. 397–406, February 2013.

[6] P. Paillier, "Public-key Cryptosystems based on Composite Degree Residuosity Classes", in *Proc. Eurocrypt*, Prague, Czech Republic, 1999, pp. 223–238.

[7] P. Smaragdis and M. Shashanka, "A Framework for Secure Speech Recognition", in *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 4, pp. 1404–1413, May 2007.

[8] L. Burget, O. Plchot, S. Cumani, O. Glembek, P. Matějka and N. Brümmer, "Discriminatively Trained Probabilistic Linear Discriminant Analysis for Speaker Verification", in *Proc. ICASSP*, Prague, Czech Republic, 2011, pp. 4832–4835.

[9] P. Boufounos and S. Rane, "Secure Binary Embeddings for Privacy Preserving Nearest Neighbors", in *Proc. Workshop on Information Forensics and Security (WIFS)*, Iguaçu Falls, Brazil, 2011, pp. 1–6.

[10] J. Portêlo, A. Abad, B. Raj and I. Trancoso, "Secure Binary Embeddings of Front-end Factor Analysis for Privacy Preserving Speaker Verification", in *Proc. Interspeech*, Lyon, France, 2013, pp. 2494–2498.

[11] J. Portêlo, B. Raj and I. Trancoso, "Logsum using Garbled Circuits", submitted to *PLoS ONE*, Public Library of Science.

[12] J. P. Campbell, "Testing with the YOHO CD-ROM Voice Verification Corpus", in *Proc. ICASSP*, Detroit, Michigan, USA, 1995, pp. 341–344.

[13] A. Martin and C. Greenberg, "The NIST 2010 Speaker Recognition Evaluation", in *Proc. Interspeech*, Makuhari, Japan, 2010, pp. 2726–2729.