Interspeech 2022
18-22 September 2022, Incheon, Korea

Towards End-to-End Private Automatic Speaker Recognition

Francisco Teixeira[1], *Alberto Abad*[1], *Bhiksha Raj*[2], *Isabel Trancoso*[1]

[1]INESC-ID/Instituto Superior Técnico, University of Lisbon, Portugal
[2]LTI, Carnegie Mellon University, USA

francisco.s.teixeira@tecnico.ulisboa.pt

Abstract

The development of privacy-preserving automatic speaker verification systems has been the focus of a number of studies with the intent of allowing users to authenticate themselves without risking the privacy of their voice. However, current privacy-preserving methods assume that the template voice representations (or speaker embeddings) used for authentication are extracted locally by the user. This poses two important issues: first, knowledge of the speaker embedding extraction model may create security and robustness liabilities for the authentication system, as this knowledge might help attackers in crafting adversarial examples able to mislead the system; second, from the point of view of a service provider the speaker embedding extraction model is arguably one of the most valuable components in the system and, as such, disclosing it would be highly undesirable. In this work, we show how speaker embeddings can be extracted while keeping both the speaker's voice and the service provider's model private, using Secure Multiparty Computation. Further, we show that it is possible to obtain reasonable trade-offs between security and computational cost. This work is complementary to those showing how authentication may be performed privately, and thus can be considered as another step towards fully private automatic speaker recognition.

Index Terms: privacy, speaker recognition, secure multiparty computation

1. Introduction

Recent years have seen an increase in the number of online services and applications that use speech as a means for authentication and interaction. Among other speech technologies, voice-based authentication systems – or Automatic Speaker Verification (ASV) systems – are becoming more and more a part of our everyday lives. The uniqueness and ubiquitous nature of speech make its use a straightforward manner to protect and grant access to both local and remote systems. However, in the remote case, ASV systems raise multiple privacy concerns. This comes from the fact that speech and the information it carries are extremely sensitive in nature: from speech one can extract information as diverse as the speaker's age, gender, emotional or personality traits, health state, among many others [1, 2]. As such, by sending their voice – or a template thereof – to a remove server, users are risking their privacy. These concerns are reflected in recent regulations, as shown, for instance, by the definition of personal data provided by the European Union's General Data Protection Regulation (GDPR) [3], wherein speech data and the information extracted from it are considered as Personally Identifiable Information, i.e., information that, on its own, allows determining the identity of an individual, and that, as such, is protected under the GDPR [4].

Due to the above, the problem of protecting privacy in the setting of ASV has been a precursor for much of the research done for speech privacy. One of the first strides in this direction is the cryptographic-based work of Pathak et al. [5], who adapted a Gaussian Mixture Model (GMM) to work with Homomorphic Encryption (HE) in order to perform speaker verification. Similarly, Portêlo et al. [6] implemented a privacy-preserving GMM-based speaker verification using Garbled Circuits. More recently, Nautsch et al. and Treiber et al. applied Homomorphic Encryption [7] and Secure Multiparty Computation [8, 9] to the same problem. Differently, Pathak et al. [10], Portêlo et al. [11] and Jiménez et al. [12] have explored the applicability of distance-preserving hashing techniques to privacy-preserving ASV, while Mtibaa et al. have studied cancelable biometric schemes for ASV [13, 14].

However, the above-mentioned works focus mainly on the security of the speaker templates or on how the verification step itself can be performed privately, sharing the assumption that the client locally extracts voice templates. In contrast, we argue that this is extremely undesirable for service providers. Specifically, we argue that the model used to extract voice templates, or speaker embeddings, is one of the, if not the most valuable component in the speaker verification pipeline. This stems from the fact that speaker embedding extractors require large amounts of data and high levels of expertise to be developed. As such, by sharing this model, ASV service providers would relinquish control over their intellectual property, and consequently, lose the value it holds. Further, as noted by Das et al. [15] and Villalba et al. [16], having knowledge of the speaker embedding extractor model may allow attackers to craft adversarial examples that mislead the ASV system, raising security and robustness concerns. For this reason, in this work we show how speaker embeddings can be extracted privately using Secure Multiparty Computation. Specifically, we focus on the private extraction of *x-vector* speaker embeddings [17]. This not only allows the protection of the speaker's voice, as it is never shared with the ASV provider, but also the protection of the speaker embedding extraction model. Moreover, even though we only consider the private extraction of *x-vectors*, our implementation can be directly combined with some of the above-mentioned works for private speaker verification [2, 9], in order to produce a fully end-to-end private speaker verification system that protects both the speaker's voice and the vendor's model.

The remainder of this paper is organised as follows: in Section 2 we provide the necessary background on Secure Multiparty Computation (SMC); Section 3 specifies the setting and threat models assumed for our task; in Section 4 we describe the experimental setup, while in Section 5 we present and discuss the results obtained. Finally, Section 6 presents our conclusions and topics for future work.

This work was supported by Portuguese national funds through Fundação para a Ciência e a Tecnologia (FCT), with references UIDB/50021/2020 and CMU/TIC/0069/2019.

10.21437/Interspeech.2022-10672

2. Secure Multiparty Computation

Secure Multiparty Computation (SMC) is an umbrella term for protocols designed to allow several parties to jointly and securely compute functions over their data, while keeping all inputs private. SMC protocols are usually built over some form of Secret Sharing (e.g. Shamir's Secret Sharing [18], GMW [19], BGW [20]), or Garbled Circuits (e.g., Yao's GCs [21], BMR [22]), and are often combined with cryptographic primitives like public-key encryption, symmetric encryption, Homomorphic Encryption (HE) or Oblivious Transfers (OTs) to perform specific functionalities, each lending different levels of security, computational and communication costs [23]. Our approach for the private extraction of *x-vectors* relies on two forms of secret sharing briefly described below.

2.1. Secret Sharing

Secret sharing is a basic primitive for SMC protocols. It allows parties to represent and share their data with other parties in such a way that each of the parties participating in the computation will only have access to a random-looking *share* (here denoted as $\langle\cdot\rangle$) of the original value. Considering an additive secret sharing scheme in the n-party case, a value x secret shared among several parties by a dealer, is defined as:

$$x = \langle x\rangle_1 + \langle x\rangle_2 + ... + \langle x\rangle_n, \tag{1}$$

where $\langle x\rangle_1$, ..., $\langle x\rangle_n$ represent random-looking shares of x held by each party, generated as $\langle x\rangle_n = x - \sum_{i=1}^{n-1} s_i$, where each s_i is chosen uniformly at random. When a value is represented in this way, a single party is not able to reconstruct the *secret* without the remaining parties. Further, this representation allows for parties to interactively compute any operation over their secret data. For instance, due to the associative property of addition, adding two shared values simply amounts to each party adding the shares they hold that correspond to the two values, without requiring any communication with the other parties.

On the other hand, in the case of multiplications, additive secret sharing schemes require additional pre-computed random shared values called *Multiplication Triples (MTs)* (or *Beaver Triples* [24]). These values are shares of the form $\langle a\rangle$, $\langle b\rangle$ and $\langle c\rangle$, where $\langle c\rangle = \langle a\rangle \times \langle b\rangle$. To perform a multiplication between shared values x and y, each party sets its shares to $\langle e\rangle_i = \langle x\rangle_i - \langle a\rangle_i$ and $\langle f\rangle_i = \langle y\rangle_i - \langle b\rangle_i$ and exchanges the results with the other parties, so that each party holds e and f. The resulting share is given by [25]:

$$\langle z\rangle_i = i \cdot e \cdot f + f \cdot \langle a\rangle_i + e \cdot \langle b\rangle_i + \langle c\rangle_i \tag{2}$$

It can then be shown that by adding the z_i we obtain $x \times y$.

The above works for any number of parties greater than or equal to two. However, for a number of parties strictly larger than two, it is possible to instantiate more efficient schemes. Replicated Secret Sharing (RSS) schemes [26] are such an example. While in additive secret sharing, each party holds a single share per value in the computation, with RSS each party holds a set of shares per value. Considering for instance the three party case and a shared value $y = \sum_{i=1}^{3} \langle y\rangle_i$, party p_1 will hold shares $\langle y\rangle_1, \langle y\rangle_2$, party p_2 will hold shares $\langle y\rangle_2, \langle y\rangle_3$ and party p_3 will hold shares $\langle y\rangle_3, \langle y\rangle_1$. In this case, addition will work as before, and each party can simply perform the operation locally. Multiplication, on the other hand, may work differently. A possible implementation of the multiplication operation would be for each party to locally multiply the shares it holds for each of the secret shared values. In this way, party p_1 will obtain $z_1 = \langle x\rangle_1\langle y\rangle_1 + \langle x\rangle_1\langle y\rangle_2 + \langle x\rangle_2\langle y\rangle_1$; party p_2, $z_2 = \langle x\rangle_2\langle y\rangle_2 + \langle x\rangle_2\langle y\rangle_3 + \langle x\rangle_3\langle y\rangle_2$ and party p_3, $z_3 = \langle x\rangle_3\langle y\rangle_3 + \langle x\rangle_3\langle y\rangle_1 + \langle x\rangle_1\langle y\rangle_3$. As above, it can be shown that adding the resulting shares will yield the correct result. Still, at the end of the computation, each party only holds a single share of the value, and a *re-sharing* protocol is required, so that each party holds the same set of shares as before [27].

The above-mentioned schemes are described with regard to arithmetic operations, but also hold for binary computations, with minor modifications [19]. This is important, as performing operations in each of these domains may prove to be more efficient for different operations, or may even allow performing different functionalities. Depending on the SMC protocol, the conversion between domains may take different forms. In some protocols, it is possible to convert between domains locally, with minimal interaction between parties [28, 29]. However, other protocols may need to use pre-computed values that are shared in both domains such as *daBits* [30] or *edaBits* [31].

The generation of *Multiplication Triples*, *daBits*, *edaBits*, as well as other auxiliary secret shares – called *correlated randomness* – requires the participation and interaction of all the parties involved in the computation. However, since the generation of these auxiliary shares is not dependent on input data, this step can be moved to what is called an *offline*, or *pre-processing* phase. This phase can be performed at any time before the *online* data-dependent phase. Many protocols are hence designed to have the most expensive operations within the *offline* phase, making the *online* phase much more efficient.

2.2. Fixed-point numbers

An important detail of SMC protocols is the fact that secret shared values are integers, whereas for most real-world applications, values are floating-point numbers. While floating-point representations exist within SMC, fixed-point representations are much more efficient. In this work, we adopt the fixed-point representation of [29], where a value x is represented as $x = y \cdot 2^f$, where y is an integer, and f is the fixed precision. While this approximation does not affect additions, for multiplications, one needs to first multiply the two integers, and then truncate by f. This can be implemented as a binary left shift operation [32], or via probabilistic truncation [33, 34, 31].

2.3. Security

The shared nature of SMC protocols demands making threat assumptions about the participating parties. The threat model of an SMC protocol is extremely important as it significantly affects its security, computational and communication performance, and thus, its range of applications.

The most common security (or threat) models include the *semi-honest* adversary model (also called *honest-but-curious* adversary or *passive* security) and the *malicious* adversary model (or *active* security). The *honest-but-curious* model is the simplest model possible. In this model, the adversaries are assumed to follow the established protocol, but are also assumed to pry into and record the data that is visible to them. In this way, no party will be able to obtain information other than that which it is allowed to, resulting in very efficient implementations. On the other hand, the *malicious* model assumes that adversaries will attempt to thwart the protocol, demanding additional proof that each party is behaving correctly. This can be done in different ways, depending on the protocol and phase of the computation, through Zero Knowledge (ZK) proofs, cut-and-choose methods, Message Authentication Codes (MACs),

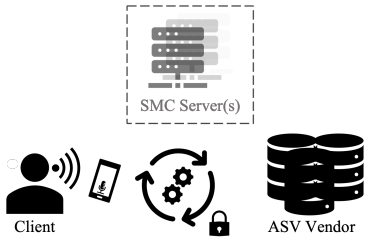

Figure 1: *Privacy-preserving extraction of speaker embeddings.*

among others [35, 36, 29]. Although more secure, this model significantly increases the protocol's computational cost.

Besides the behaviour of individual parties, one can also define the security of the protocol in terms of whether a *majority* of the parties will behave correctly or not – *honest majority* vs *dishonest majority*, and whether a subset of parties might collaborate – or *collude* – to obtain more information than they are allowed to. If a majority of parties are assumed to be honest, protocols that take into account *malicious* behaviour can be made much more efficient [37, 28]. On the other hand, the highest possible level of security is achieved by assuming malicious adversaries in a dishonest majority. However, this comes at very high computational and communication costs [35, 36].

3. Privacy-preserving speaker embedding extraction

We consider two parties to be involved in the extraction of speaker embeddings in the context of ASV: the *client*, who wants to be able to access a given system and the *ASV Vendor*, who provides the authentication system as a service. If we were considering a full ASV system, we would also need to consider the *ASV Controller*, the party who holds the set of speaker templates who are allowed to access the system. However, since in this work we only focus on the extraction of speaker embeddings, we will not take this party into account. Nonetheless, in some cases the *vendor* and *controller* may be same party.

3.1. Threat models

Our goal is to have the *vendor* and *client* collaborate to privately extract a speaker embedding from a speech sample belonging to the *client*, using an extraction model belonging to the *vendor*. We consider that both the *client* and *vendor* are interested in protecting the privacy of their own data – the *client* for the sensitive nature of their speech data, and the *vendor* due to the value and security of its model. To this end, we consider four scenarios.

In the first scenario, the *vendor* and *client* are the only parties involved in the private extraction of the speaker embedding, and are both assumed to be *semi-honest*. This is the weakest security model, as either the *vendor* or the *client* might thwart the protocol to obtain information about the other's data.

In the second scenario, we consider adding a trusted non-colluding SMC server to the computation. In a real world setting, this party would correspond to a company providing servers for SMC. Since such a company would need to rely on its reputation for its business, we argue that it would always follow protocol, and would never collude with any other party involved in the computation [38]. By adding this trusted non-colluding server, and since the *client* and *vendor* have no incentive to collude – the SMC server does not have data to be stolen – this allows us to instantiate the honest majority 3-party RSS SMC protocol of Araki et al. [26]. As discussed in Section 2.1, RSS schemes are much more efficient than additive secret sharing protocols while keeping the same level of security [26].

For our third scenario, we consider adding a second trusted non-colluding SMC server. This allows us to instantiate a 4-party honest-majority RSS SMC protocol. Specifically, we can instantiate the 4-party protocol of Dalskov et al. [28], which is secure against one malicious party. In this way, if either the *client* or the *vendor* behave maliciously, the protocol will abort. Since the *client* and the *vendor* will not collude, and the SMC servers are assumed to be trusted, this setting will be more secure than the previous. However, in this case the non-collusion assumption of the SMC server is much stronger.

In our fourth scenario we return to the 2-party setting, and assume that either the *vendor* or the *client* might behave maliciously. This is the setting with the highest level of security, however, it will also incur the highest computational and communication costs. A diagram of the aforementioned computational settings can be found in Figure 1.

3.2. Privacy-preserving *x-vector* extraction using SMC

Originally proposed by Snyder et al. [17], the *x-vector* architecture is a neural network trained to discriminate between a large number of speakers. Within this context, *x-vectors* correspond to latent representations extracted from an intermediate layer of the network. This network is composed of three main blocks: the first block is a set of time-delay (TDNN) layers that operate at the frame level with a small temporal context. These layers work as 1D dilated convolutions, with a kernel size corresponding to the temporal context, which alternate with ReLU nonlinearities; the second block, a statistical pooling layer, aggregates the information across the time dimension and outputs the per-feature mean and standard deviation for the entire speech segment; the third block is a set of fully connected layers, from which *x-vector* embeddings are extracted after the network is trained for speaker classification.

To implement this extractor with SMC, we need to take into account the type of operations required by each layer in the network. 1D dilated convolutional layers are linear transformations and can be implemented using either the basic arithmetic operations of the SMC protocol, or with specific protocols for inner product computations [29]. ReLU activations require the computation of a comparison, which can be done through the secure comparison protocol of [33]. The Statistical Pooling layer involves computing the mean and standard deviation of the input. To compute the standard deviation, we will need to compute a square-root, which can be done through the protocol of [39].

All the protocols mentioned above work for the fixed-point number representation of Section 2.2, making them directly compatible with the weights and inputs of neural networks, after these have also been converted to a fixed-point representation.

4. Experimental Setup

4.1. Corpora

The Voxceleb corpus was used to train the *x-vector* extractor and the Probabilistic Linear Discriminant Analysis (PLDA) model described below. This corpus includes recordings of 7,363 speakers of multiple ethnicities, accents, occupations and age groups. It is composed of short clips taken from interviews uploaded to YouTube [40, 41]. The corpus is composed of two parts, *VoxCeleb 1 and 2*, both subdivided into *dev* and *test*.

Table 1: *Results obtained for each protocol in terms of computational performance and communication cost.*

Protocol	Security Model	Time (s)			Communication (MB)		
		Pre-processing	Online	Total	Pre-processing	Online	Total
2-party Semi$_{2^k}$ [36]	DM/SH	8,423.00. $\pm$ 165.36*	18.92 $\pm$ 0.22	8,441.93 $\pm$ 165.36	1,662,300.60*	12,919.40	1,675,220.00
3-party RSS [26]	HM/SH	0.18 $\pm$ 0.20*	10.68 $\pm$ 0.15	10.85 $\pm$ 0.14	15.04*	118.02	133.06
4-party RSS [28]	HM/Mal	1.21 $\pm$ 0.29*	16.76 $\pm$ 0.21	17.97 $\pm$ 0.21	27.14*	333.16	360.30
2-party SPDZ$_{2^k}$ [36]	DM/Mal	147,799.68 $\pm$ 1,016.82*	126.32 $\pm$ 1.16	147,926.00 $\pm$ 1,016.82	21,870,489.60*	27,810.40	21,898,300.00

4.2. Speaker embeddings

For our experiments we used the pre-trained *x-vector* model made available by SpeechBrain [42]. This model follows the architecture of [17]. A description of the layers used for extraction can be found in Table 2. The model was trained using the *dev* partitions of Voxceleb 1 and 2. As a baseline reference for computational cost, extracting a single *x-vector* from a 3-second long speech sample with this model, using a CPU, takes $\sim$0.03s.

Table 2: *x-vector extractor architecture.*

#	Layer	Input	Output	Kernel	Dilation
1	TDNN 1	24	512	5	1
2	TDNN 2	512	512	3	2
3	TDNN 3	512	512	3	3
4	TDNN 4	512	512	1	1
5	TDNN 5	512	1500	1	1
6	Statistics Pooling	1500	3000	-	-
7	Linear	3000	512	-	-

A Probabilistic Linear Discriminant Analysis (PLDA) model was used to score pairs of *x-vectors* when performing verification [43]. The full pipeline achieves 3.2% Equal Error Rate (EER) on the Voxceleb 1 test set (Cleaned) [41, 42].

4.3. Privacy-preserving implementation

The network described in the previous subsection was implemented using the MP-SPDZ library [29]. We tested our implementation using four different protocols with different levels of security, as detailed in Section 3.1. Specifically, we tested our implementation over the following protocols: the 2-party *semi-honest* (SH) version of the SPDZ$_{2^k}$ scheme for dishonest majority (DM), denoted as Semi$_{2^k}$ [36]; the 3-party RSS scheme described in [26], which provides *semi-honest* security, in the *honest majority* (HM) setting; the 4-party RSS scheme of [28], which provides *malicious* (Mal) security in the *honest majority* setting against one corrupted party; and the 2-party *malicious* version of the SPDZ$_{2^k}$ scheme [36].

For 3 and 4-party RSS and for the *semi-honest* version of SPDZ$_{2^k}$, we used local share conversions [28] to improve efficiency. For 3 and 4-party RSS, we also used probabilistic truncation as proposed by [34, 28], instead of regular truncation, to further improve efficiency. Our experiments assume the default security parameters for each protocol, namely 40-bit security for 3 and 4-party RSS, and Semi$_{2^k}$, and 64-bit security for SPDZ$_{2^k}$. We used the library's fixed-point number representation adopting the default configuration of 16 bits for the decimal part, and 15 bits for the fractional part. All tested protocols perform computations modulo 2^k, where $k = 64$. Experiments were performed on a machine with 24 Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz processors and 250GB of RAM.

5. Results

Table 1 includes the results obtained for our experiments in terms of computational performance and communication cost. All results correspond to the extraction of a single *x-vector* using a 3-second long speech sample. Online results for all protocols, and for full results for 3 and 4-party RSS, were obtained by averaging over 100 runs. Full results for Semi2k and SPDZ2k were computed over 10 runs, due to their high computational cost. Values denoted with * were estimated by computing the difference between the full protocol and the online phase.

Our results show that RSS schemes significantly outperform the *semi-honest* and *malicious* versions of SPDZ$_{2^k}$, both in terms of computational and communication performance. Further, since for the *semi-honest* version of SPDZ$_{2^k}$, preprocessing takes >2h and >1TB of data, and since the preprocessing for the malicious version takes >41h and >20TB of data, it is clear that the private extraction of *x-vectors* with these protocols, particularly for a high level of security, is currently infeasible. Contrarily, the results for the RSS schemes can be deemed feasible, particularly when considering the fact that no modifications were made to reduce the size of the *x-vector* extraction network. When comparing the 3 and 4 party RSS protocols, while the 3-party *semi-honest* version is more efficient in terms of computational cost and communication, we argue that the added security of the 4-party RSS protocol is a reasonable trade-off for the $\sim$7s and $\sim$230MB difference in the total computational and communication costs. Still, we need to consider that to implement the 4-party RSS protocol we need strong assumptions about the honest behaviour of the SMC servers.

Finally, our experiments showed that the SMC implementation yielded negligible degradation, with the average Mean Squared Error distance between 100 *x-vectors* extracted with the original and SMC implementations being just $\sim 1\%$ of the total magnitude of the vector.

6. Conclusions

In this work we have shown that it is possible to extract *x-vector* speaker embeddings at a reasonable level of security and computational and communication costs, while protecting the privacy of both the *client*'s data and the *ASV Vendor*'s model, using SMC, particularly when deploying on 3 and 4-party RSS protocols. This problem had been unexplored so far, as other privacy-preserving works for ASV assumed that speaker embeddings to be extracted by the client. This makes this work complementary to others in the literature and another step towards fully private ASV pipelines.

As future work, we consider that it would be important to explore ways to reduce the size of the *x-vector* extraction network, to improve efficiency. Moreover, it would be interesting to also consider protocols following the *covert* security model, wherein adversaries may have a malicious behaviour, but where there is a probability that in doing so, they may be discovered.

7. References

[1] R. Singh, *Profiling humans from their voice*. Springer, 2019, ISBN: 978-981-13-8403-5.

[2] A. Nautsch, A. Jiménez, A. Treiber, J. Kolberg, C. Jasserand *et al.*, "Preserving privacy in speaker and speech characterisation," *Computer Speech & Language*, vol. 58, pp. 441–480, 2019.

[3] European Parliament and Council, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 - General Data Protection Regulation." 2016.

[4] A. Nautsch, C. Jasserand, E. Kindt, M. Todisco, I. Trancoso, and N. Evans, "The GDPR & Speech data: Reflections of legal and technology communities, first steps towards a common understanding," *arXiv preprint*, vol. 1907.03458, 2019.

[5] M. A. Pathak and B. Raj, "Privacy-Preserving Speaker Verification and Identification Using Gaussian Mixture Models," *IEEE TASLP*, vol. 21, no. 2, pp. 397–406, 2013.

[6] J. Portêlo, B. Raj, A. Abad, and I. Trancoso, "Privacy-preserving speaker verification using garbled gmms," in *22nd (EUSIPCO)*. IEEE, 2014, pp. 2070–2074.

[7] 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, pp. 16–23.

[8] A. Nautsch, J. Patino, A. Treiber *et al.*, "Privacy-Preserving Speaker Recognition with Cohort Score Normalisation," in *Proc. Interspeech 2019*, 2019, pp. 2868–2872.

[9] A. Treiber, A. Nautsch, J. Kolberg, T. Schneider, and C. Busch, "Privacy-preserving PLDA speaker verification using outsourced secure computation," *Speech Communication*, vol. 114, pp. 60–71, 2019.

[10] M. A. Pathak and B. Raj, "Privacy-preserving speaker verification as password matching," in *ICASSP*, 2012, pp. 1849–1852.

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

[12] A. Jiménez, B. Raj, J. Portêlo, and I. Trancoso, "Secure Modular Hashing," in *WIFS*. IEEE, 2015, pp. 1–6.

[13] A. Mtibaa, D. Petrovska-Delacretaz, and A. B. Hamida, "Cancelable speaker verification system based on binary gaussian mixtures," in *4th ATSIP*, 2018, pp. 1–6.

[14] A. Mtibaa, D. Petrovska-Delacrétaz, J. Boudy, and A. B. Hamida, "Privacy-preserving speaker verification system based on binary i-vectors," *IET Biometrics*, vol. 10, no. 3, pp. 233–245, 2021.

[15] R. K. Das, X. Tian, T. Kinnunen, and H. Li, "The Attacker's Perspective on Automatic Speaker Verification: An Overview," in *Proc. Interspeech 2020*, 2020, pp. 4213–4217.

[16] J. Villalba, Y. Zhang, and N. Dehak, "x-Vectors Meet Adversarial Attacks: Benchmarking Adversarial Robustness in Speaker Verification," in *Proc. Interspeech 2020*, 2020, pp. 4233–4237.

[17] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. ICASSP*, Calgary, AB, Canada, April 2018.

[18] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[19] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. 19th STOC*. ACM, 1987, pp. 218–229.

[20] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proc. 20th STOC*. ACM, 1988, pp. 1–10.

[21] A. C. Yao, "How to generate and exchange secrets," in *27th SFCS*, 1986, pp. 162–167.

[22] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols," in *Proc. 22nd STOC*, 1990, pp. 503–513.

[23] Y. Lindell, "Secure multiparty computation (MPC)." *IACR Cryptology ePrint Archive*, vol. 2020, p. 300, 2020.

[24] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *CRYPTO*. Springer, 1991, pp. 420–432.

[25] D. Demmler, T. Schneider, and M. Zohner, "ABY - a framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[26] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *SIGSAC*. ACM, 2016, p. 805–817.

[27] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: Honest-majority maliciously secure framework for private deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 1, pp. 188–208, 2021.

[28] A. Dalskov, D. Escudero, and M. Keller, "Fantastic four: Honest-majority four-party secure computation with malicious security," in *30th USENIX Security Symposium*, 2021.

[29] M. Keller, "Mp-spdz: A versatile framework for multi-party computation," *Cryptology ePrint Archive*, vol. Report 2020/521, 2020.

[30] D. Rotaru and T. Wood, "Marbled circuits: Mixing arithmetic and boolean circuits with active security," in *International Conference on Cryptology in India*. Springer, 2019, pp. 227–249.

[31] D. Escudero, S. Ghosh, M. Keller, R. Rachuri, and P. Scholl, "Improved primitives for mpc over mixed arithmetic-binary circuits," in *Annual International Cryptology Conference*. Springer, 2020, pp. 823–852.

[32] O. Catrina and S. d. Hoogh, "Improved primitives for secure multiparty integer computation," in *International Conference on Security and Cryptography for Networks*. Springer, 2010, pp. 182–199.

[33] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *International Conference on Financial Cryptography and Data Security*. Springer, 2010, pp. 35–50.

[34] A. Dalskov, D. Escudero, and M. Keller, "Secure evaluation of quantized neural networks," *Proceedings on Privacy Enhancing Technologies*, vol. 4, pp. 355–375, 2020.

[35] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits," in *European Symposium on Research in Computer Security*. Springer, 2013, pp. 1–18.

[36] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing, "Spd $\mathbb{Z}_{2^k}$: efficient mpc mod 2^k for dishonest majority," in *Annual International Cryptology Conference*. Springer, 2018, pp. 769–798.

[37] J. Furukawa, Y. Lindell, A. Nof, and O. Weinstein, "High-throughput secure three-party computation for malicious adversaries and an honest majority," in *EUROCRYPT*. Springer, 2017, pp. 225–255.

[38] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *European Symposium on Research in Computer Security*. Springer, 2008, pp. 192–206.

[39] A. Aly and N. P. Smart, "Benchmarking privacy preserving scientific operations," in *International Conference on Applied Cryptography and Network Security*. Springer, 2019, pp. 509–529.

[40] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," *Interspeech 2017*, Aug 2017.

[41] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *Interspeech 2018*, Sep 2018.

[42] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell *et al.*, "SpeechBrain: A general-purpose speech toolkit," 2021, arXiv:2106.04624.

[43] P. Kenny, T. Stafylakis, P. Ouellet, M. J. Alam, and P. Dumouchel, "PLDA for speaker verification with utterances of arbitrary duration," in *ICASSP*. IEEE, 2013, pp. 7649–7653.