TÉCNICO LISBOA

Carnegie Mellon University

Towards End-to-End Private Automatic Speaker Recognition

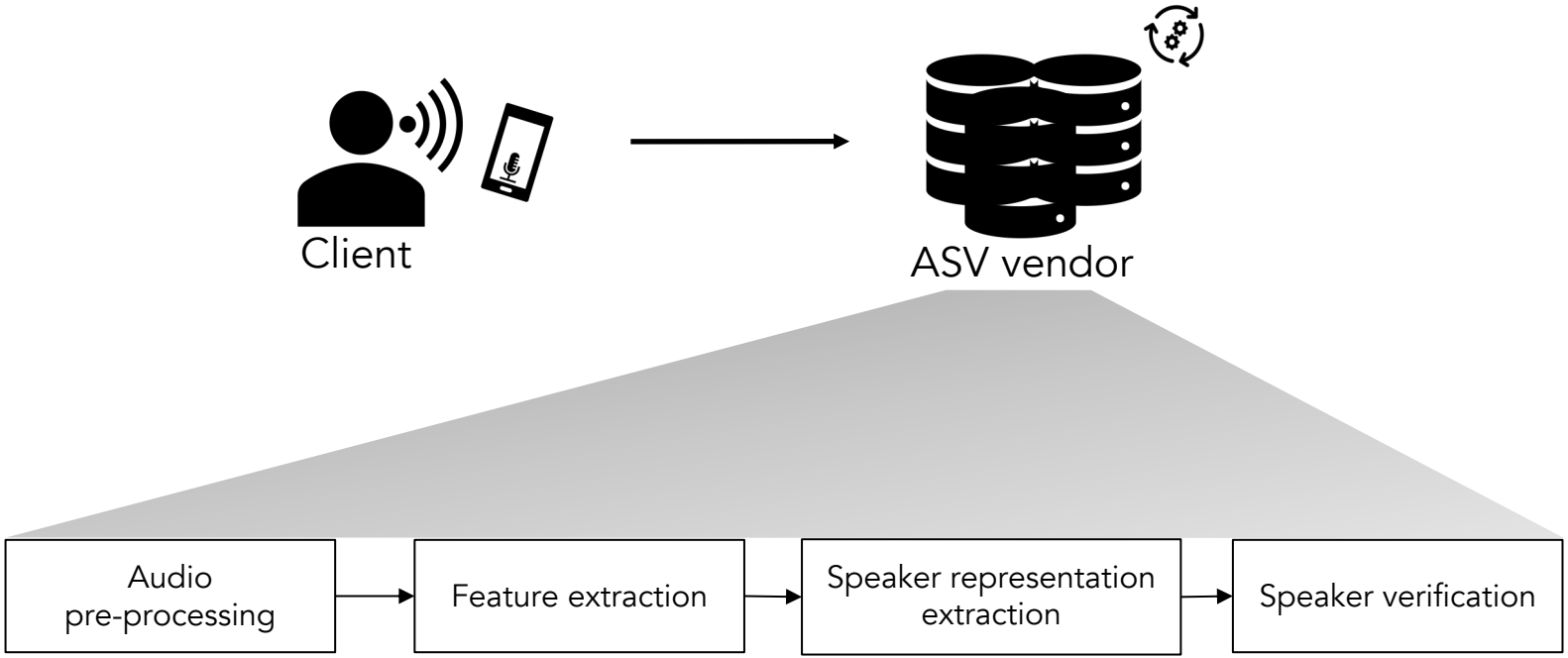
Francisco Teixeira,

Alberto Abad, Bhiksha Raj and Isabel Trancoso

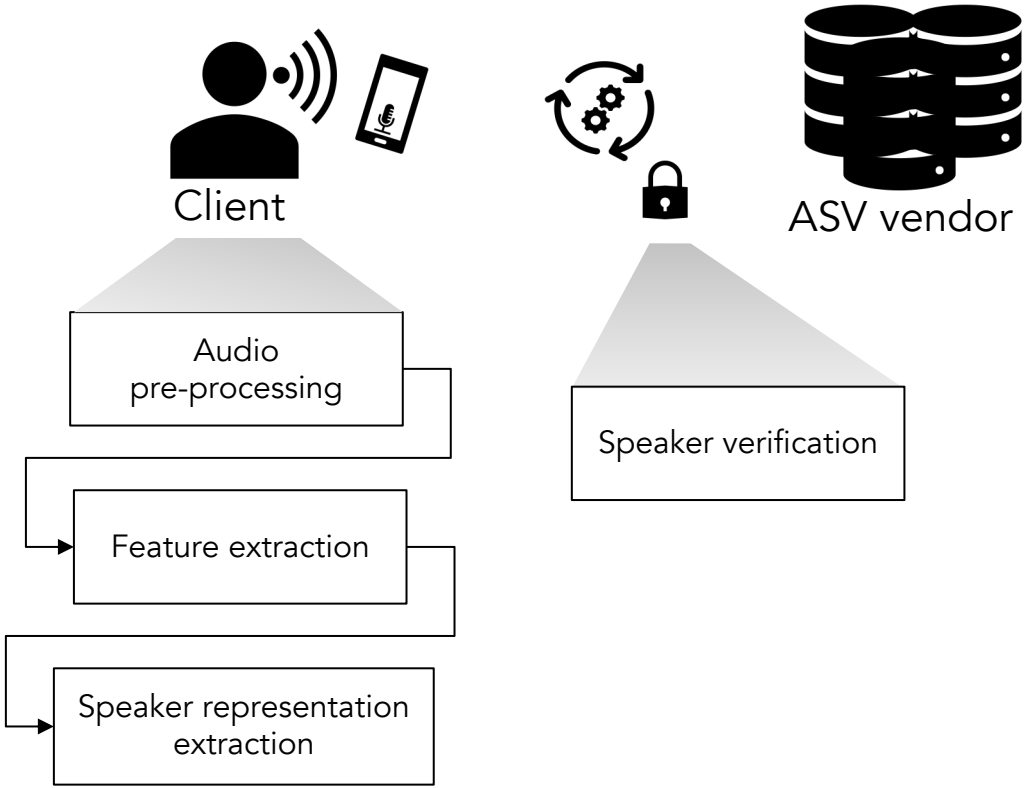
francisco.s.teixeira@tecnico.ulisboa.pt

inesc id lisboa

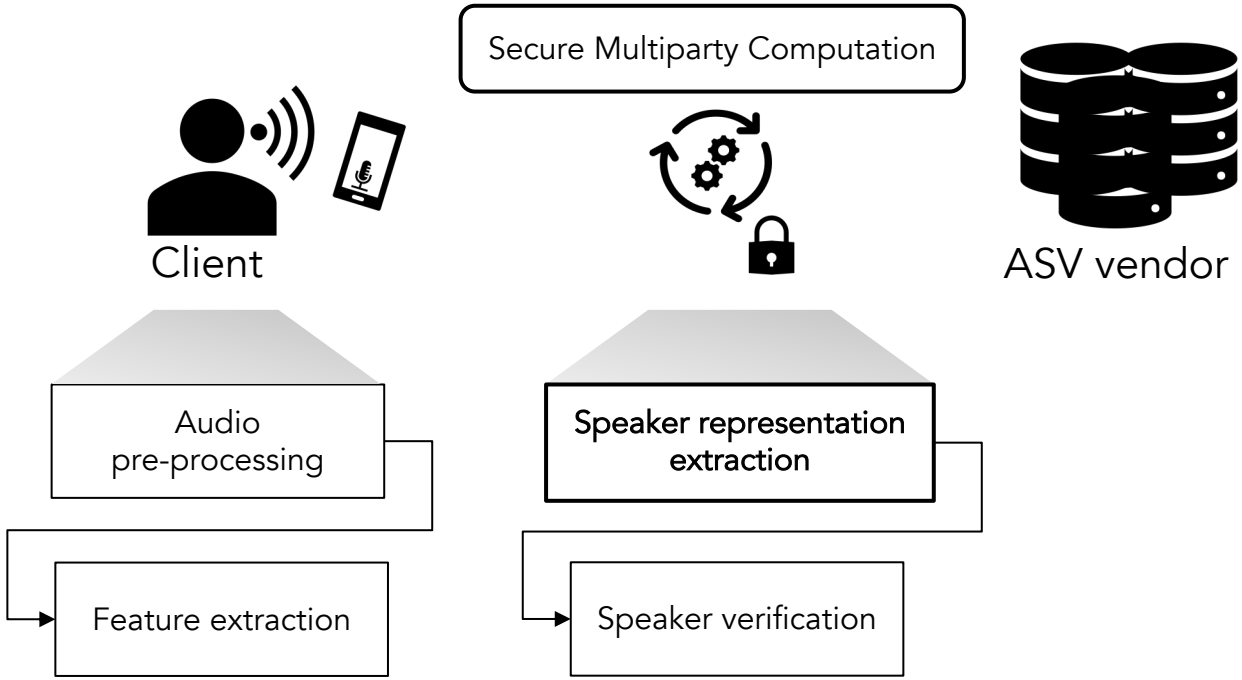
Motivation: Problem setting



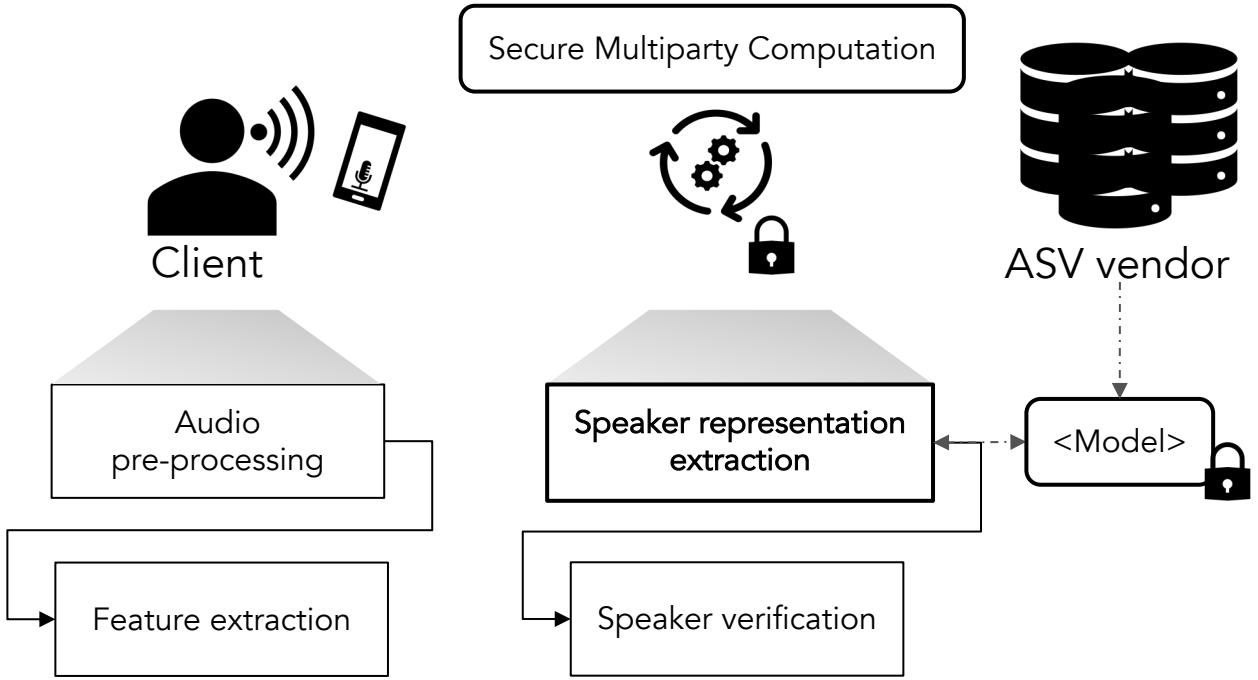
Motivation: State-of-the-art



Motivation: Proposed solution



Motivation: Proposed solution



Outline

- Motivation
- Secure Multiparty Computation
- Privacy-preserving speaker embedding extraction: security settings
- Implementation and experimental setup
- Results
- Conclusions and future work

Secure Multiparty Computation (SMC)

- **Family of cryptographic protocols** that allow two or more parties to interactively (and privately) compute functions, e.g.:
  - Arithmetic Secret Sharing
  - Boolean Secret Sharing
  - Garbled Circuits

- **Arithmetic Secret Sharing:**

P1	P2
Input: x	Input: y
$\langle x \rangle_1 = r_x$	$\langle y \rangle_1 = r_y$
$\langle x \rangle_2 = x - r_x$	$\langle y \rangle_2 = y - r_y$

Secure Multiparty Computation: Arithmetic Secret Sharing

- *n*-party setting: the owner of the data generates *n-1* random values, with each *secret share* being defined as:

$$\langle x \rangle_n = x - \sum_{i=1}^{n-1} r_i \qquad x = \sum_{i=1}^{n} \langle x \rangle_i$$

- It is then possible to perform operations over multiple shared values:
  - Additions can be performed locally;
  - Multiplications require *multiplication triples* (also called *Beaver triples*).

Secure Multiparty Computation: *Multiplication Triples*

We want to compute $\langle z \rangle = \langle x \rangle \times \langle y \rangle$

- Assume we have *pre-computed* secret-shared values $\langle a \rangle$, $\langle b \rangle$ and $\langle c \rangle$, such that:

$$\langle c \rangle = \langle a \rangle \times \langle b \rangle$$

- To perform a *multiplication* each party sets local shares $\langle e \rangle_i, \langle f \rangle_i$ as:

$$\langle e \rangle_i = \langle x \rangle_i - \langle a \rangle_i \qquad \langle f \rangle_i = \langle y \rangle_i - \langle b \rangle_i$$

- All parties then interact to reconstruct *e* and *f*, and set their share of $\langle z \rangle$ to

$$\langle z \rangle_i = i \cdot e \cdot f + f \cdot \langle a \rangle_i + e \cdot \langle b \rangle_i + \langle c \rangle_i$$

Secure Multiparty Computation: Boolean Secret Sharing

- The previous representation is easily adaptable to binary representations:

$$\langle x \rangle_n = x \oplus \sum_{i=1}^{n-1} s_i \qquad x = \langle x \rangle_1 \oplus \langle x \rangle_2 \oplus \ldots \oplus \langle x \rangle_n$$

- Similarly to Arithmetic Secret Sharing:
  - XORs can be computed locally;
  - AND operations require *multiplication triples.*

- We can convert between Arithmetic and Boolean domains using pre-computed values shared in both domains (e.g. *daBits*, *edaBits*).

Secure Multiparty Computation: Replicated Secret Sharing

- A (more efficient) variant of the previous secret sharing protocol:
  - Instead of having a single share, each party holds a *set* of shares per value.

e.g. consider three parties, and a secret-shared value x:

$$x = \sum_{i=1}^{3} \langle x \rangle_i$$

P1 will hold: $\langle x \rangle_1, \langle x \rangle_2$

P2 will hold: $\langle x \rangle_2, \langle x \rangle_3$

P3 will hold: $\langle x \rangle_3, \langle x \rangle_1$

Secure Multiparty Computation: Replicated Secret Sharing

- A (more efficient) variant of the *vanilla* secret sharing protocol:
  - Additions are performed locally by each party.
  - Multiplications **no longer** require *multiplication triples.*

e.g. Simple multiplication protocol for three parties ($\langle z \rangle = \langle x \rangle \times \langle y \rangle$):

- Each party multiplies the shares it holds for each value locally and obtains:

$$z_1 = \langle x \rangle_1 \langle y \rangle_1 + \langle x \rangle_1 \langle y \rangle_2 + \langle x \rangle_2 \langle y \rangle_1$$
$$z_2 = \langle x \rangle_2 \langle y \rangle_2 + \langle x \rangle_2 \langle y \rangle_3 + \langle x \rangle_3 \langle y \rangle_2$$
$$z_3 = \langle x \rangle_3 \langle y \rangle_3 + \langle x \rangle_3 \langle y \rangle_1 + \langle x \rangle_1 \langle y \rangle_3$$

- *Re-sharing* protocol is required.

Secure Multiparty Computation: Security models

- *Honest-but-curious* model:
  - Parties are assumed to follow the protocol, but to try to get as much information as possible.

- *Malicious* model:
  - Parties are assumed to try to thwart the protocol to gain more information.

  - Requires specific protocols to ensure all parties are "behaving" correctly, e.g.:
    - Cut-and-choose methods;
    - Zero-Knowledge (ZK) proofs;
    - Message Authentication Codes (MACs).

- Honest majority vs dishonest majority

Privacy-preserving speaker embedding extraction

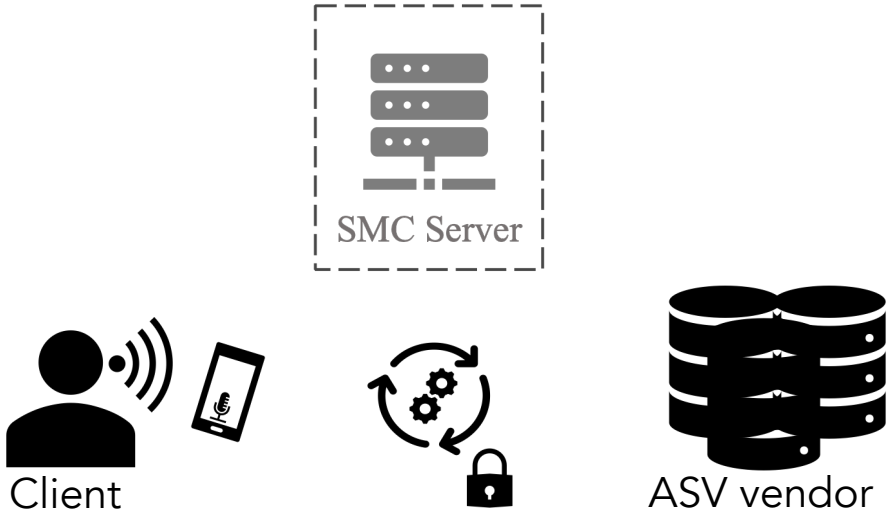
Privacy-preserving speaker embedding extraction: Security setting

2-party setting

Simplest/most natural setting

- Honest-but-curious security
- Malicious security

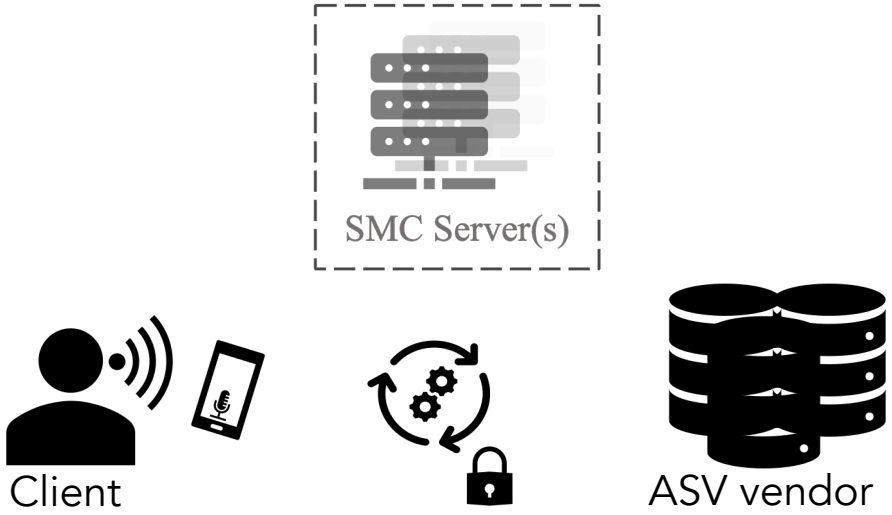
Privacy-preserving speaker embedding extraction: Security setting



3-party setting:

- Allows the instantiation of more efficient protocols

Privacy-preserving speaker embedding extraction: Security setting



4-party setting:

- 4-party Replicated Secret Sharing Protocol of Dalskov et al. [1].
- Provides honest-majority security against one malicious party.

Experimental setup

- Pre-trained SpeechBrain *x-vector* speaker embedding extraction model [2]:
  - 3.2% EER on Voxceleb 1 test set
- MP-SPDZ library [3]:
  - Implements linear and non-linear operations required for the *x-vector* extraction network.
- Protocols used:
  - *Semi*$_2^k$: 2-party semi-honest protocol [4]
  - 3-party RSS: semi-honest protocol (Araki et al. [5])
  - 4-party RSS: malicious protocol w/ honest majority (Dalskov et al. [1])
  - SPDZ$_2^k$: 2-party malicious protocol [4]

#	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	-	-

Table 1: *x-vector* extractor architecture

Experimental setup: Fixed-point representations

- In neural networks, weights are **floating-point** numbers.
- Shares in Arithmetic Secret Sharing protocols are **integers.**
  - It is not possible to compute random *real* numbers **uniformly** over an interval.
- In our implementation we use MP-SPDZ's **fixed-point** number representation:

$$x = y \cdot 2^f$$

  - *f* represents a fixed precision
  - *y* is a secret-shared value
- Additions can be computed without changes.
- Multiplications require an extra division/truncation by *f*.
  - Implemented as binary left-shift operation or specific *probabilistic truncation* protocol.

Results

Protocol	Security model	Time (s)		Communication (MB)	
		Pre-processing	Online	Pre-processing	Online
2-party Semi$_2^k$ [4]	Semi-honest	>2 hours	≅19	≅1.6 TB	≅12.6 GB
2-party SPDZ$_2^k$ [4]	Malicious	>1 day	≅126	≅21 TB	≅27 GB
3-party RSS [5]	Semi-honest w/ honest majority	≅0.18	≅11	15	118
4-party RSS [1]	Malicious w/ honest majority	≅1.2	≅17	27	333

Table 2: Computational and communication costs for the extraction of speaker embeddings from 3-second long speech recordings.

Results

Protocol	Security model	Time (s)		Communication (MB)	
		Pre-processing	Online	Pre-processing	Online
2-party Semi$_2^k$ [4]	Semi-honest	>2 hours	≅19	≅1.6 TB	≅12.6 GB
2-party SPDZ$_2^k$ [4]	Malicious	>1 day	≅126	≅21 TB	≅27 GB
3-party RSS [5]	Semi-honest w/ honest majority	≅0.18	≅11	15	118
4-party RSS [1]	Malicious w/ honest majority	≅1.2	≅17	27	333

Table 2: Computational and communication costs for the extraction of speaker embeddings from 3-second long speech recordings.

Results

Protocol	Security model	Time (s)		Communication (MB)	
		Pre-processing	Online	Pre-processing	Online
2-party Semi$_2^k$ [4]	Semi-honest	>2 hours	≅19	≅1.6 TB	≅12.6 GB
2-party SPDZ$_2^k$ [4]	Malicious	>1 day	≅126	≅21 TB	≅27 GB
3-party RSS [5]	Semi-honest w/ honest majority	≅0.18	≅11	15	118
4-party RSS [1]	Malicious w/ honest majority	≅1.2	≅17	27	333

Table 2: Computational and communication costs for the extraction of speaker embeddings from 3-second long speech recordings.

Conclusions & Future Work

- In this work we have shown that it is possible to extract *x-vector* speaker embeddings using Secure Multiparty Computation.
- By using SMC we are able to protect both the privacy of the speaker's voice as well as the ASV vendor's model.
- As future work it would be important to explore:
  - Techniques to reduce the size of the *x-vector* extraction network
  - Other security models that better fit real world scenarios (e.g. *covert security*).
- This work has also been recently applied to Automatic Speaker Diarization in the context of the CMU Portugal project PrivaDia.

TÉCNICO LISBOA

Carnegie Mellon University

inesc id lisboa

Thank you!

francisco.s.teixeira@tecnico.ulisboa.pt

References

[1] Dalskov, A., Escudero, D. and Keller, M. "Fantastic Four: Honest-Majority Four-Party Secure Computation With Malicious Security." 30th USENIX Security Symposium (USENIX Security 21), 2021.

[2] Parcollet, T., et al. "Speechbrain: A general-purpose speech toolkit." (2022).

[3] Keller, M. "MP-SPDZ: A versatile framework for multi-party computation." Proceedings of the 2020 ACM SIGSAC conference on computer and communications security. 2020.

[4] Cramer, R., Damgärd, I., Escudero, D., Scholl, P. and Xing C., “SPD Z2k : Efficient MPC mod 2k for dishonest majority,” in Annual International Cryptology Conference. Springer, 2018, pp. 769– 798.

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