Interspeech 2018
2-6 September 2018, Hyderabad

VoiceGuard: Secure and Private Speech Processing

Ferdinand Brasser[1], *Tommaso Frassetto*[1], *Korbinian Riedhammer*[2], *Ahmad-Reza Sadeghi*[1], *Thomas Schneider*[1], *Christian Weinert*[1]

[1]Technische Universität Darmstadt, Germany
[2]University of Applied Sciences Rosenheim, Germany

{ferdinand.brasser, tommaso.frassetto, ahmad.sadeghi}@trust.tu-darmstadt.de,
korbinian@ieee.org, thomas.schneider@cs.tu-darmstadt.de, christian.weinert@crisp-da.de

Abstract

With the advent of smart-home devices providing voice-based interfaces, such as Amazon Alexa or Apple Siri, voice data is constantly transferred to cloud services for automated speech recognition or speaker verification.

While this development enables intriguing new applications, it also poses significant risks: Voice data is highly sensitive since it contains biometric information of the speaker as well as the spoken words. This data may be abused if not protected properly, thus the security and privacy of billions of end-users is at stake.

We tackle this challenge by proposing an architecture, dubbed *VoiceGuard*, that efficiently protects the speech processing task inside a trusted execution environment (TEE). Our solution preserves the privacy of users while at the same time it does not require the service provider to reveal model parameters. Our architecture can be extended to enable user-specific models, such as feature transformations (including fMLLR), i-vectors, or model transformations (e.g., custom output layers). It also generalizes to secure on-premise solutions, allowing vendors to securely ship their models to customers.

We provide a proof-of-concept implementation and evaluate it on the *Resource Management* and *WSJ* speech recognition tasks isolated with Intel SGX, a widely available TEE implementation, demonstrating even real time processing capabilities.
Index Terms: speech recognition, privacy protection, cloud computing

1. Introduction

Devices providing voice-based interfaces are omnipresent in today's world. Amazon Alexa, Apple Siri, Google Assistant, or Microsoft Cortana are available to the more than two billion smartphone users in 2018. Also, there is a steadily increasing number of smart-home devices, like Amazon Echo, Apple HomePod, or Google Home, solely relying on voice-based interaction. Possible application scenarios are not restricted to the consumer market but increasingly cover professional activities, for example enterprise-ready smart assistants guiding through complicated business processes in order to increase productivity.

In any of the aforementioned cases, voice data is constantly transferred to the cloud for remote speech processing, such as automated speech recognition (ASR) or speaker verification. This poses significant security and privacy risks since voice data contains sensitive biometric information as well as the spoken words: in case unprotected voice data gets out of hand, it may be abused, e.g., for impersonation attacks, assembling fake recordings, or simply extracting intimate as well as secret and sensitive content.

A naive solution to these problems is to ship the speech processing code together with corresponding models to the users to run locally. While this might be infeasible for low-end devices anyhow, it also contradicts the business interests of vendors providing such models which represent their intellectual property.

Attempts based on purely cryptographic solutions, i.e., homomorphic encryption (HE) or secure multi-party computation (SMPC), guarantee that neither user nor vendor need to reveal their respective inputs in the clear. However, as we elaborate in our review of related work in §2, these solutions are highly impractical due to their massive overhead in computation time and communication costs. Besides, none of the existing solutions considered user-specific models, i.e., the common practice to train or adapt a separate model for each user that covers deviations from the model to incorporate specific characteristics, e.g., in dialect and pronunciation.

Goals and Contributions. To overcome these limitations, we propose *VoiceGuard* in §5, an architecture that efficiently protects speech processing tasks using a trusted execution environment (TEE). It allows the secure processing of confidential data even in a hostile environment by combining cryptographic techniques with hardware-enforced code and data isolation.

Although the concept of TEEs has been known for many years, they only recently became widely available with Intel's introduction of Software Guard Extensions (SGX). SGX is Intel's implementation of a TEE available in most of their recent CPUs. It generated large interest in both academic research and industry: Signal, for example, a popular instant messaging service similar to WhatsApp, employs Intel SGX to identify the contacts in a new user's address book that are signed up to the service while all other contacts remain private [1]. The deployment of such privacy-preserving services is also facilitated by leading cloud service providers (e.g., Microsoft Azure [2]) making this CPU feature available to customers.

VoiceGuard enables secure and private speech processing, independent of who actually controls the machine performing the computation. Thus, it could be hosted by the vendor of the speech processing software, a third party service provider, or even the user. The latter on-premise solution could be preferred if it is necessary to comply to certain legal regulations or the user wants to exclude the possibility of a malicious party performing sophisticated hardware attacks.

The architecture of VoiceGuard can easily be extended to enable user-specific models, such as feature transformations (including fMLLR), i-vectors, or model transformations (e.g., custom output layers). We present a fully functional prototype implementation of VoiceGuard for ASR based on the kaldi toolkit [3]. Moreover, we conduct an empirical performance evaluation of the *Resource Management* and *WSJ* speech recognition tasks in §6, thereby demonstrating that the overhead induced by our protection measures is low enough to enable privacy-preserving speech recognition in real time.

10.21437/Interspeech.2018-2032

2. Related Work

In the following, we briefly review general approaches for privacy-preserving machine learning (grouped by the underlying technology) that could be adapted to speech processing tasks which depend on the evaluation of neural networks. Furthermore, we review specialized approaches for various privacy-preserving speech processing tasks.

2.1. Privacy-Preserving Machine Learning

Secure Multi-Party Computation (SMPC). SMPC enables two or more parties to jointly compute a publicly known function without revealing private inputs to each other by executing an interactive cryptographic protocol. Recently, SMPC protocols and frameworks have been applied to both privacy-preserving training of neural networks [4] and corresponding inference [5, 6, 7, 8, 9], mostly for image classification tasks. However, compared to unprotected data processing, SMPC-based solutions require several orders of magnitude higher computation time and communication cost. They are especially impractical for on-the-fly processing due to repeated initialization costs.

Homomorphic Encryption (HE). HE allows performing operations on encrypted data s.t. the decryption of the computation result equals the outcome when performing the same operations on plaintext data. Microsoft CryptoNets [10] was the first attempt to utilize HE for secure evaluation of neural networks, followed by an improvement named CryptoDL [11], which replaces complex activation functions with approximated low-degree polynomials. Nevertheless, the reported performance results indicate that solutions based on heavyweight HE are currently far from suitable for speech recognition in real time.

TEE. SMPC via TEEs has been proposed in [12, 13, 14]. Ohrimenko et al. [15] adapt several machine learning algorithms, including neural networks, to prevent cache-based side-channel attacks in scenarios where multiple institutions use Intel SGX to securely share their datasets for training and evaluation of joint machine learning models. In [16], the authors introduce a similar protection mechanism that is efficient enough for real-time data processing: instead of preventing memory accesses that depend on sensitive data, they add noise to memory traces by accessing dummy data. The very recent Chiron [17] system allows a user to train a model using the computing resources of a cloud service provider while the training data remains hidden and the resulting model can only be accessed as a black box. This machine learning as-a-service (MLaaS) concept differs from our scenario where we assume vendors who provide existing models which should only be evaluated obliviously.

2.2. Privacy-Preserving Speech Processing

Pathak et al. [18] explored how to use the previously mentioned SC and HE techniques for privacy-preserving versions of speech processing tasks such as speech recognition and speaker verification. However, with their prototype implementation based on the Paillier HE scheme, it takes more than 3 hours to encrypt 1 s of audio and to recognize a single word out of a 10 word vocabulary. Admitting the impracticality of this approach, the authors furthermore propose a very efficient solution based on secure string-matching. Unfortunately, this approach can only be used for certain tasks such as speaker verification.

Recently, Glackin et al. [19] proposed an architecture for finding outsourced (encrypted) speech documents that contain given keywords. The architecture works as follows: (I) the client translates audio to phonetic symbols using a CNN-based acoustic model, (II) the encrypted phones and a search index are sent to a server, and (III) the server uses a searchable encryption scheme to deliver outsourced data matching the given keywords. However, this approach requires the vendor to hand the acoustic model to the user in the clear.

3. Background

For the remainder of the paper, we assume familiarity with state-of-the-art speech processing pipelines and restrict the background to the introduction of Intel SGX.

Intel SGX. Intel Software Guard Extensions (SGX) enables processing of confidential data on untrusted systems [20, 21, 22, 23]. SGX introduces the concept of *enclaves*, which are programs executed in isolation from *all* other software on a system, including privileged software, like the operating system (OS) or a hypervisor.

Enclaves are loaded as part of a host process and are embedded in its virtual memory, like a library. The initial content of an enclave is loaded from unprotected memory, hence, it can be manipulated and is not kept confidential. Therefore, confidential data must be provisioned to an enclave over a secure channel *after* it has been created. However, to ensure that secret data is not sent to a malicious (or maliciously modified) enclave, the integrity and authenticity of an enclave needs to be verified before provisioning secret data. To enable this, SGX provides a security service called remote attestation (RA). With RA, an external party can verify whether an enclave was created correctly, i.e., a cryptographic hash of the initial memory state of an enclave is signed by the platform signing key which is built into the CPU.

Once available inside an enclave, secret data can be encrypted using an enclave-specific key and written to untrusted storage, e.g., the hard disk. This *sealing* mechanism allows an enclave to use secret data across multiple instantiations.

4. Model and Assumptions

In this paper we consider a setting where three parties collaborate to perform secure and private speech processing:

(1) The *user* provides the voice data to be processed. She is concerned about her privacy and does not want the other parties to identify her based on biometric characteristics in her input. Additionally, the user does not want to reveal the content of her input to the other parties, i.e., they should not be able to access the voice data or the processing results. Lastly, the user does not want to be traceable across multiple sessions.

(2) The *vendor* provides the software required for speech processing together with corresponding models. This data constitutes the vendor's intellectual property, hence it must be kept confidential from the other parties.

(3) The *service provider* carries out the actual computations based on the user's and the vendor's inputs. The service provider could be an independent third party, e.g., a cloud service provider. Without loss of generality, the service provider could also be under the control of the user or the vendor.

Adversary Model. The adversary's goal is to extract sensitive information, i.e., the intellectual property of the vendor, the input of the user, or data that allows the adversary to identify or track the user.

We assume that the adversary is in control of the service provider's infrastructure, in particular, all computer systems involved in performing the speech processing task. The adversary has full control over the software in the service provider's infrastructure, including privileged software like the OS or a

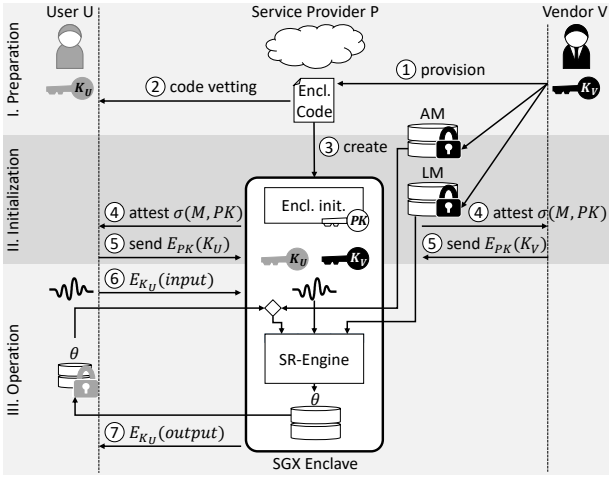

Figure 1: *VoiceGuard architecture. User U establishes a secure channel with the SGX enclave hosted at service provider P and sends sensitive voice data as well as user-specific adaptation data θ. Similarly, vendor V sends the sensitive models AM and LM through a secure channel. P securely processes U's voice data using V's models within an SGX Enclave.*

hypervisor. We assume that the adversary cannot perform invasive hardware attacks like extracting keys from the CPU. We also consider physical side-channel attacks, like differential power analysis [24], out of scope. We assume the enclave developer incorporated appropriate defense mechanism to protect against side-channel attacks leveraging micro-architectural effects [25, 26, 27].[1]

5. VoiceGuard Design

Our architecture VoiceGuard enables privacy-preserving and efficient speech processing on untrusted systems. VoiceGuard supports different deployment scenarios, i.e., the service provider is not necessarily a third party, but could also be the user or the vendor. Common to all scenarios is the basic setup, i.e., at least two *input parties* provide sensitive data while the computing platform is not trusted by at least one of them.

For the sake of simplicity we explain our solution based on the speech recognition scenario visualized in Fig. 1, where the service provider P is an untrusted third party, e.g., a cloud service provider. The vendor V's private input consists of speech recognition models. The user U's private input is the voice data. In this example, the output is sensitive as well and should only be made available to the user.[2]

VoiceGuard works in three phases: (I) preparation, (II) initialization, and (III) operation. In the first phase, user U and vendor V need to agree on the code to be executed in the enclave ("Encl. Code" in Fig. 1). In the second phase, the enclave code is instantiated. U and V use remote attestation (RA) to establish secure channels with the enclave through which they provision their respective encryption keys to the enclave. In the third phase, the enclave is ready to perform speech processing. Using the keys transmitted in the previous phase, U and V provide their respective inputs to the enclave in encrypted form. The result of the operation phase is encrypted with the user's key, so only she can decrypt it. Next, we describe the individual phases in detail:

Preparation Phase. First, U and V need to agree on the code to be run inside the SGX enclave. While SGX protects enclaves against accesses from the outside, they are nevertheless allowed to output data without any restriction. Therefore, U and V want to make sure that the enclave code only outputs non-sensitive data. The code typically comes from the vendor, i.e., V provisions the enclave code, ① in Fig. 1. Thus, V can easily ensure that no sensitive data will leave the enclave. The code itself is not necessarily confidential and is often open source. However, U has to carefully analyze the enclave code in a *vetting* process ② to verify that it does not contain functions which will leak her sensitive data. The vetting process could also be outsourced to a trusted third party, e.g., a government institution.

Additionally, the vendor provisions its acoustic model AM and language model LM to the service provider. Both are encrypted with the vendor's key K_V s.t. the service provider cannot access the vendor's intellectual property. At this stage, the models are not yet loaded inside an enclave, but are written to untrusted storage, e.g., the hard disk.

Initialization Phase. The enclave is created from the code provisioned by V earlier ③. The creation process is measured by the SGX-enabled CPU, i.e., a cryptographic hash of the initial memory content of the enclave is created and stored securely. If the enclave code is manipulated before or during the creation process, the measurement will produce a different result and the manipulation is detected. After the creation is finished, the code is isolated from all accesses and cannot be changed anymore.

The first operation performed by the enclave is the enclave initialization, during which the enclave generates a key pair for asymmetric cryptography operations like RSA [28],[3] with the public key PK shown in white in Fig. 1.

Next, U and V need to establish a secure channel with the enclave by provisioning their keys K_U and K_V, respectively, to the enclave. We will describe this process for U. The process for V is identical. VoiceGuard uses public key cryptography similar to Transport Layer Security (TLS) [29], which is widely used to secure web sites. The enclave sends its public key PK to U. However, U needs assurance that the received PK comes indeed from the correct enclave, i.e., the authenticity of PK must be established. This is done using the remote attestation (RA) feature of SGX, which generates a digital signature $\sigma(M, PK)$ that binds PK to the measurement M of the enclave, ④ in Fig. 1. In particular, the public key PK, which was generated *inside* the enclave, and the measurement of the initial enclave memory content are signed with the platform key. This signature can be verified using Intel's public key infrastructure (PKI) for SGX.

The user verifies the signature and checks that M matches her expectations, i.e., that the enclave has not been altered before or during creation. If both checks were successful, the user can be sure that PK belongs to the key pair generated by the correct enclave and that information encrypted with PK can only be decrypted inside that enclave. In step ⑤, U encrypts her key K_U with PK and sends the result $E_{PK}(K_U)$ to the enclave.

At the end of the initialization, the enclave shares a symmetric key with the user (K_U, the gray key in Fig. 1) and with the vendor (K_V, the black key in Fig. 1).

Operation Phase. The user sends encrypted inputs $E_{K_U}(input)$, i.e., audio samples, to the service provider. Since the input is encrypted with U's key, it can only be accessed by the enclave ⑥. If applicable, U also sends her user-specific adap-

[1] Our evaluation is performed without such protection mechanisms and thus does *not* reflect their impact on the performance results.

[2] The output could also be provided to one or multiple other parties.

[3] This process leverages the hardware random number generator of the CPU and can therefore not be influenced from outside the enclave.

tation parameters θ (e.g., i-vectors), which are also encrypted with K_U, to the enclave.

Inside the enclave, U's input is decrypted and passed to the speech recognition engine ("SR-Engine" in Fig. 1). The SR-Engine has two additional inputs, the acoustic model AM, typically a deep neural network (DNN), and the language model LM, typically a decoding graph. AM is provided by V and already stored encrypted at P. When AM is used, it is loaded into the enclave and decrypted using V's key K_V. Similarly, any adaptation parameters θ and the language model LM are loaded by the enclave, decrypted, and passed to the SR-Engine.

On-demand loading of AM or LM could leak sensitive information about their structure by observing access patterns. This can be prevented by storing this data in a randomized order, i.e., preventing an observer from learning useful information from observed access patterns [30].

The result of the speech processing is encrypted with K_U and sent back to the user ⑦.[4] Additionally, the SR-Engine may produce updated adaptation parameters θ, which are then encrypted with K_U and sent back to U.[5]

Once in the operation phase, the system can be queried repetitively by the user, thereby avoiding repeated preparation and initialization costs.

6. Evaluation

To show the effectiveness of VoiceGuard, we created a proof-of-concept implementation which embeds kaldi [3] in an SGX enclave using the Graphene library OS [31]. We ran experiments on two representative corpora: DARPA Resource Management (RM) [32] and Wall Street Journal (WSJ) [33]. Note that the purpose of these experiments is not to show improvements for certain training algorithms, but rather to prove that both regular and VoiceGuard decoding yield the exact same results with acceptable overhead. We chose RM and WSJ since they are well-established baseline recipes in kaldi which result in very different net and graph sizes for performance analysis.

For RM, we train on the speaker independent training and development set (about 4 000 utterances) and test on the six DARPA test runs: Mar and Oct'87, Feb and Oct'89, Feb'91, and Sep'92 (about 1 500 utterances in total), as a joint set. We use kaldi's `rm/s5` recipe and train the `nnet2_online` system with i-vectors. The resulting DNN is about 3 MB (9 hidden layers, 750 k parameters), the uni- and bigram decoding graphs are 0.5 MB and 2 MB, respectively. For details of the recipe, refer to kaldi at commit `cd6562`.

For WSJ, we train on the full SI284 set (about 60 h) and test on the Dec'93 development, Nov'92, and Nov'93 test sets. We use kaldi's `wsj/s5` recipe and also train the `nnet2_online` system with i-vectors. The resulting DNN is about 14 MB (15 hidden layers, 3.6 M parameters), the pruned trigram decoding graph is about 641 MB; since this is not about accuracy, no LM rescoring is applied. For details of the recipe, refer to kaldi at commit `ec98e7`.

In order to determine the overhead induced by VoiceGuard, we run kaldi on an Intel Core i7-7700 CPU @ 3.6 GHz over every corpus and report the run time of each test in Table 1. The overhead of VoiceGuard is between 39 % and 49 % for RM and between 98 % and 104 % for WSJ. The higher overhead for WSJ is due to its larger model (graph) size. In the current version of SGX, enclaves can only use up to 96 MB memory and rely on swapping to access additional data. Table 1 also shows the word error rate (WER) of each test, which is identical for VoiceGuard and baseline kaldi since they execute the same code on the same models resulting in identical lattices and transcriptions.

Table 1: *Performance of VoiceGuard w.r.t. baseline kaldi.*

Test	WER	Baseline (s)	VoiceGuard (s)	Overhead
RM-bigram	2.3 %	351	522	48.5 %
RM-unigram	15.4 %	585	815	39.3 %
WSJ (dev93)	18.1 %	1 427	2 854	100.1 %
WSJ (eval92)	13.4 %	876	1 736	98.2 %
WSJ (eval93)	15.5 %	518	1 058	104.3 %

We also differentiate between the time required to initialize the SR-Engine and to process a single file. The model setup time in the baseline is 0.04 s (RM-bigram) and 0.31 s (WSJ), while the setup time for the enclave and the models in VoiceGuard is 0.95 s (RM-bigram) and 23.55 s (WSJ). Note that this overhead is due to the initialization of enclave memory, occurs only once when the enclave is started, and is thus not repeated across multiple queries. The processing with RM-bigram of a 2.79 s audio file takes 0.32 s in the baseline and 0.50 s in VoiceGuard; with WSJ, the processing of a 6.12 s audio file takes 1.90 s in the baseline and 4.06 s in VoiceGuard. Thus, the overhead for the processing of one file is 56 % for RM-bigram and 114 % for WSJ, similarly to the overheads measured for the various batches, which indicates that the enclave setup overhead is amortized across multiple queries. Even though processing time is doubled in some cases, our results show that VoiceGuard enables privacy-preserving speech processing even in real time.

7. Conclusion

We proposed VoiceGuard, a novel architecture for privacy-preserving and efficient speech processing that supports user-specific models and can be deployed either in the cloud or on-premise. The evaluation of our prototype implementation demonstrates applicability for speech recognition in real time. Besides speech recognition, VoiceGuard's generic architecture works for related tasks such as speaker verification or voice biometrics, including emotion recognition and medical speech processing.

One core aspect to take into consideration when implementing this architecture in production systems is the model size: both AM and LM need to be loaded into the secure enclave, in turn causing computational overhead both at initialization and at run time. While small models such as RM (or models for speaker verification) require almost no memory, typical high-accuracy ASR systems would use much larger models than the WSJ models evaluated in this experiment.

Thus, as part of future work, we will explore distributing the processing across multiple SGX-enabled nodes and optimize performance for more accurate models with larger memory requirements. We will also determine the overhead incurred by employing protection mechanisms against side-channel attacks.

8. Acknowledgments

This work was co-funded by the DFG as part of projects P3, S2, and E4 within CROSSING, by the German Federal Ministry of Education and Research (BMBF) and the Hessen State Ministry for Higher Education, Research and the Arts (HMWK) within CRISP, and by the Intel Collaborative Research Institute for Collaborative Autonomous & Resilient Systems (ICRI-CARS).

[4] The result could also be sent to a different party, even a third party.

[5] U can decrypt θ and re-encrypt it to make individual requests from the same user unlinkable.

9. References

[1] M. Marlinspike, "Technology preview: Private contact discovery for Signal," https://signal.org/blog/private-contact-discovery/, September 2017.

[2] M. Russinovich, "Introducing Azure confidential computing," https://azure.microsoft.com/en-us/blog/introducing-azure-confidential-computing/, September 2017.

[3] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi Speech Recognition Toolkit," in *Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE Signal Processing Society, 2011.

[4] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," in *Symposium on Security and Privacy (S&P)*. IEEE, 2017.

[5] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "DeepSecure: Scalable Provably-Secure Deep Learning," *CoRR*, vol. abs/1705.08963, 2017.

[6] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious Neural Network Predictions via MiniONN transformations," in *Conference on Computer and Communications Security (CCS)*. ACM, 2017.

[7] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications," in *Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 2018, to appear.

[8] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, "EzPC: Programmable, Efficient, and Scalable Secure Two-Party Computation," *IACR Cryptology ePrint Archive*, vol. 2017/1109, 2017.

[9] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A Low Latency Framework for Secure Neural Network Inference," *IACR Cryptology ePrint Archive*, vol. 2018/073, 2018.

[10] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy," in *International Conference on Machine Learning (ICML)*. JMLR, 2016.

[11] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep Neural Networks over Encrypted Data," *CoRR*, vol. abs/1711.05189, 2017.

[12] P. Koeberl, V. Phegade, A. Rajan, T. Schneider, S. Schulz, and M. Zhdanova, "Time to Rethink: Trust Brokerage Using Trusted Execution Environments," in *Trust and Trustworthy Computing (TRUST)*, ser. LNCS, vol. 9229. Springer, 2015.

[13] K. A. Küçük, A. Paverd, A. Martin, N. Asokan, A. Simpson, and R. Ankele, "Exploring the Use of Intel SGX for Secure Many-Party Applications," in *System Software for Trusted Execution (SysTEX)*. ACM, 2016.

[14] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A.-R. Sadeghi, G. Scerri, and B. Warinschi, "Secure Multiparty Computation from SGX," in *Financial Cryptography and Data Security (FC)*, ser. LNCS, vol. 10322. Springer, 2017.

[15] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious Multi-Party Machine Learning on Trusted Processors," in *USENIX Security Symposium*. USENIX, 2016.

[16] S. Chandra, V. Karande, Z. Lin, L. Khan, M. Kantarcioglu, and B. Thuraisingham, "Securing Data Analytics on SGX with Randomization," in *European Symposium on Research in Computer Security (ESORICS)*, ser. LNCS, vol. 10492. Springer, 2017.

[17] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving Machine Learning as a Service," *CoRR*, vol. abs/1803.05961, 2018.

[18] M. A. Pathak, B. Raj, S. Rane, and P. Smaragdis, "Privacy-Preserving Speech Processing: Cryptographic and String-Matching Frameworks Show Promise," *IEEE Signal Processing Magazine*, vol. 30, no. 2, 2013.

[19] C. Glackin, G. Chollet, N. Dugan, N. Cannings, J. Wall, S. Tahir, I. G. Ray, and M. Rajarajan, "Privacy preserving encrypted phonetic search of speech data," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017.

[20] Intel, "Intel Software Guard Extensions Programming Reference," 2014. [Online]. Available: https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf

[21] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative Instructions and Software Model for Isolated Execution," in *Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. ACM, 2013.

[22] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo, "Using Innovative Instructions to Create Trustworthy Software Solutions," in *Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. ACM, 2013.

[23] I. Anati, S. Gueron, S. P. Johnson, and V. R. Scarlata, "Innovative Technology for CPU Based Attestation and Sealing," in *Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*. ACM, 2013.

[24] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology (CRYPTO)*. Springer, 1999.

[25] M.-W. Shih, S. Lee, T. Kim, and M. Peinado, "T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs," in *Network & Distributed System Security Symposium (NDSS)*. Internet Society, 2017.

[26] S. Chen, X. Zhang, M. K. Reiter, and Y. Zhang, "Detecting Privileged Side-Channel Attacks in Shielded Execution with Déjà Vu," in *Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 2017.

[27] F. Brasser, S. Capkun, A. Dmitrienko, T. Frassetto, K. Kostiainen, U. Müller, and A. Sadeghi, "DR.SGX: Hardening SGX Enclaves against Cache Attacks with Data Location Randomization," *CoRR*, vol. abs/1709.09917, 2017.

[28] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Communications of the ACM (CACM)*, vol. 21, no. 2, 1978.

[29] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," Internet Requests for Comments, RFC 5246, 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5246.txt

[30] B. Fuhry, R. Bahmani, F. Brasser, F. Hahn, F. Kerschbaum, and A.-R. Sadeghi, "HardIDX: Practical and Secure Index with SGX," in *Conference on Data and Applications Security and Privacy (DBSec)*, ser. LNCS, vol. 10359. Springer, 2017.

[31] C. Tsai, D. E. Porter, and M. Vij, "Graphene-SGX: A practical library OS for unmodified applications on SGX," in *USENIX Annual Technical Conference (USENIX ATC)*. USENIX, 2017.

[32] P. Price, W. Fisher, J. Bernstein, and D. Pallett, "Resource Management RM1 2.0," Linguistic Data Consortium, Philadelphia, USA, 1993.

[33] J. Garofalo, D. Graff, D. Paul, and D. Pallett, "CSR-I,II (WSJ0,1) Complete," Linguistic Data Consortium, Philadelphia, USA, 2007.