OFFLINE MODEL GUARD: Secure and Private ML on Mobile Devices

Sebastian P. Bayerl*, Tommaso Frassetto†, Patrick Jauernig†, Korbinian Riedhammer*, Ahmad-Reza Sadeghi†, Thomas Schneider†, Emmanuel Stapf†, Christian Weinert†

*Technische Hochschule Nürnberg, Germany, {sebastian.bayerl, korbinian.riedhammer}@th-nuernberg.de

†Technische Universität Darmstadt, Germany, {tommaso.frassetto, patrick.jauernig, ahmad.sadeghi, emmanuel.stapf}@trust.tu-darmstadt.de, {schneider, weinert}@encrypto.cs.tu-darmstadt.de

***Abstract*—Performing machine learning tasks in mobile applications yields a challenging conflict of interest: highly sensitive client information (e.g., speech data) should remain private while also the intellectual property of service providers (e.g., model parameters) must be protected. Cryptographic techniques offer secure solutions for this, but have an unacceptable overhead and moreover require frequent network interaction.**

In this work, we design a practically efficient hardware-based solution. Specifically, we build OFFLINE MODEL GUARD (OMG) to enable privacy-preserving machine learning on the predominant mobile computing platform ARM—even in offline scenarios. By leveraging a trusted execution environment for strict hardware-enforced isolation from other system components, OMG guarantees privacy of client data, secrecy of provided models, and integrity of processing algorithms. Our prototype implementation on an ARM HiKey 960 development board performs privacy-preserving keyword recognition using TensorFlow Lite for Microcontrollers in real time.

***Index Terms*—TEE, TrustZone, private ML, speech processing**

I. INTRODUCTION

An increasing number of applications running on mobile devices like smartphones and tablets relies on machine learning (ML) services to enhance the user experience, e.g., to give an estimate on battery life based on user behavior, improve image quality, or perform speech recognition.

Many of these ML services require frequent cloud interaction, resulting in severe privacy risks for billions of users due to the highly sensitive nature of such remotely processed data. Besides potentially confidential and intimate content, voice recordings, for example, contain unique biometric information that can be abused, e.g., for impersonation attacks and distributing fake recordings.

Privacy breaches in this domain are not fiction: in 2018, a customer requested his recording archive from Amazon, but accidentally got access to 1,700 audio files from a stranger [1]. Furthermore, state authorities ordered Amazon to hand out recordings as they might contain evidence of crime [2]. Media reports also revealed that Apple, among others, sent voice recordings to third party companies in order to improve their service with manual transcriptions. The employees of those companies got to listen to private discussions between doctors and patients, business deals, criminal dealings, and sexual encounters [3]. Moreover, biometric data used for identification was recently leaked at a large scale: the database of a UK government contractor with more than a million fingerprints and facial recognition information was publicly accessible [4].

When relying on online services for mobile ML applications, there are also usability issues to consider: high latency and, therefore, a bad user experience occurs if the user has an unreliable or low-bandwidth network connection, and high roaming fees may apply if the user is abroad.

A trivial solution for all these issues is to process all sensitive user data on the client's device. Previously, this approach was severely limited by the storage space constraints on mobile devices and the storage space requirements of ML models used in practice. Recently, though, Google lifted this limitation by training a recurrent neural network (RNN) model for character-level speech recognition and compressing it to only 80 MB, while delivering the same accuracy as former cloud-based production models with a size of multiple gigabytes [5], [6].

However, deploying such a model in unencrypted form is often not in the interest of the service provider. A production-level model constitutes intellectual property as the underlying training data is usually hard to obtain and creating an accurate while compact model requires extensive expertise [7]. Furthermore, if attackers have unrestricted model access, the privacy of people represented in the training data is even more threatened by, e.g., membership inference attacks [8] and unintended memorization [9].

Cryptographic techniques like homomorphic encryption (HE) and secure multi-party computation (SMPC) provide solutions for this conflict of interest: with HE, private inputs can be securely processed under encryption by the client or the service provider, whereas with SMPC, client and server can jointly compute any function on private inputs in a provably secure protocol. Unfortunately, the computational overhead for HE when performing complex ML tasks is impractical for the given mobile scenario, whereas the amount and the frequency of required network communication is the bottleneck for SMPC protocols. Thus, we explore hardware-assisted solutions to deliver secure and private ML on mobile devices in offline scenarios while providing practical efficiency.

Our Contributions. In this work, we build OFFLINE MODEL GUARD (OMG), a generic architecture that efficiently protects machine learning tasks on mobile devices like smartphones and tablets, and demonstrate its practicality using offline keyword recognition as an example application.

978-3-9819263-4-7/DATE20/©2020 EDAA

OMG leverages unprivileged (normal-world) user-space enclaves on ARM platforms to execute ML tasks in a hardware-protected environment that is two-way isolated from all other system components to minimize the attack surface. Utilizing TrustZone functionality, OMG can securely access peripherals like the microphone to protect sensitive information directly from the source. As a result, OMG guarantees complete privacy of client data, secrecy of the provided ML models, and integrity of processing algorithms.

We provide a fully functional prototype implementation of OMG on an ARM HiKey 960 development board for offline keyword recognition based on TensorFlow Lite for Microcontrollers [10]. As TrustZone on ARM does not provide user-space enclaves, we leverage SANCTUARY [11] for our implementation. Our performance evaluation demonstrates that secure and private offline speech processing is possible in real time even with strong protection guarantees. As we developed our prototype with TensorFlow compatibility in mind, our implementation can easily be extended to network architectures used for other related tasks such as end-to-end continuous speech recognition, speaker verification, and emotion recognition.

II. Related Work

In the following, we review existing works that preserve privacy in machine learning. The goal there is usually to train a model on the server side without allowing the server to see training data in the clear, or to obliviously classify input data without leaking the model (inference). Proposed solutions either rely entirely on cryptography or build on TEEs.

For protecting only the IP of ML models there also exist orthogonal works for model watermarking [12] and fingerprinting [13] that do not consider the privacy of client inputs.

A. Cryptography

The cryptographic techniques used for privacy-preserving machine learning are homomorphic encryption (HE) and secure multi-party computation (SMPC). Also, combinations of these techniques are being studied. HE allows to perform operations directly on encrypted data, but generally incurs a high computational overhead. SMPC allows multiple parties to jointly perform secure computations on shared data. This works by obliviously evaluating a Boolean or arithmetic circuit representation of the desired functionality, but results in a high communication overhead and for some protocols requires interaction for each layer of the circuit.

For cryptographic protocols it is possible to formally prove security with respect to input privacy. However, many protocols and corresponding implementations assume that both client and server honestly follow the protocol description. This assumption is unrealistic in real-world scenarios since mobile clients might run modified applications. Securing such protocols against malicious parties comes at additional cost.

Privacy-preserving neural network inference via HE and SMPC was studied in [14]–[16]. Thereafter, many frameworks for privacy-preserving machine learning have been developed, e.g., [17]–[22]. They allow at least for secure deep/convolutional neural network inference and are usually benchmarked with standard image classification tasks.

Using such cryptographic frameworks requires expert knowledge and thus they are hardly accessible for ML experts. However, recently there are efforts to integrate cryptographic protocols into standard ML tools: for TensorFlow there are HE [23] and SMPC [24] implementations, and for Intel's ngraph compiler there exists HE support [25].

Unfortunately, the current performance results discourage from actual deployment and scaling them to more involved speech processing tasks seems unrealistic [26]. Addressing all outlined disadvantages, with OMG we propose a computation- and communication-efficient hardware-assisted design for secure and private ML on mobile devices that enforces correct execution of the algorithms and can easily be used by ML experts due to TensorFlow Lite compatibility.

B. Trusted Execution Environments (TEEs)

Compared to cryptographic techniques, trusted execution environment (TEE) architectures provide several orders of magnitude better performance for protecting ML services [27]. Most of the existing works rely on Intel SGX as the dedicated TEE architecture to protect ML services.

Ohrimenko et al. [28] protect ML algorithms and models in SGX enclaves. They consider a scenario where sensitive data from multiple data providers is aggregated on a remote server while SGX enclaves are used to protect the training process. However, the enclaves might leak information to the untrusted software on the server through data-dependent access patterns, which can be exploited in controlled-channel attacks [29], [30]. Therefore, the authors develop data-oblivious variants of standard ML techniques, e.g., support vector machines, neural networks, and decision trees, which guarantee that all memory accesses do not depend on secret data.

In Chiron [31], an ML-as-a-Service (MLaaS) scenario is considered where sensitive data is collected from customers and used for training without revealing the data to the MLaaS provider. This is achieved by performing the training process in a Ryoan [32] sandbox (based on SGX), which protects sensitive customer data but still offers the service provider the possibility to freely select, configure, and train the models.

Myelin [33] provides security guarantees similar to [28] as it relies on data-oblivious deep learning algorithms: every model owner compiles its deep learning model into a privacy-preserving model graph, which is then trained on a remote server (inside an SGX enclave) on sensitive data.

In [34], the authors introduce an alternative protection mechanism against controlled-channel attacks that is more efficient and suitable for real-time data processing. The authors propose to add noise to memory traces by accessing dummy data instead of enforcing data-oblivious memory accesses.

VoiceGuard [35] targets the use case of privacy-preserving speech processing. For this, sensitive voice recordings are collected from user devices, e.g., smart home devices like Amazon Echo, Google Home, and Apple HomePod, and are sent

via secure channels to a service provider. The service provider performs speech recognition using proprietary models provided by ML specialists in an SGX enclave, thereby protecting the user data as well the proprietary models. The inference results are then securely sent back to the user device. Very recent work [36] also enables efficient private online speech recognition but uses obfuscation techniques and the notion of differential privacy, which significantly degrades accuracy.

In contrast, MLCapsule [37] considers an offline MLaaS scenario where the trained model is used on the client side for inference while being protected using an SGX enclave.

None of the previous works considers the challenge of how user data can be securely collected on the user device. Intel SGX, which is mostly used as the dedicated TEE architecture, is not able to provide a secure communication channel from enclaves to system peripherals, e.g., the microphone or camera [38]. Thus, sensitive user data is endangered as it could be exfiltrated by malicious software running on the client device. With OMG, we present the first TEE architecture that provides protection for proprietary ML models and privacy-sensitive user input at the same time. Furthermore, while Intel SGX is a TEE widely available in recent Intel CPUs, most mobile devices like smartphones and tablets come with CPUs based on the ARM platform. This prevents using the previously proposed SGX-based solutions for securing relevant use cases on mobile devices, e.g., offline speech recognition. Thus, in this work, we present OMG for ARM-based devices and as an example application demonstrate privacy-preserving offline keyword recognition in real time.

III. Background

In the following, we introduce relevant details regarding the ARM TrustZone TEE implementation and the SANCTUARY security architecture [11] for user-space enclaves.

A. ARM TrustZone

Trusted execution environments (TEEs) combine memory isolation techniques [39]–[41] and attestation [42] with isolated execution to provide protected execution of security-critical code. For mobile devices, the predominant computing platform is ARM, which provides a TEE implementation called ARM TrustZone [43]. A chip with TrustZone capabilities simultaneously runs two security contexts (or "worlds") as virtual processors: a "normal world" and an isolated "secure world" (cf. Fig. 1). While the normal world executes a commodity OS (e.g., Android) and ordinary applications, the secure world forms a TEE for running security-critical code on a trusted OS.

A major assumption of TrustZone is that an attacker cannot compromise code running in the secure world. Unfortunately, the TrustZone design is flawed in this aspect: the isolation between applications in the secure world is rather weak and the attack surface is massively increased the more applications run therein [44]. Thus, the secure world with its privileged platform access is an attractive target for adversaries.

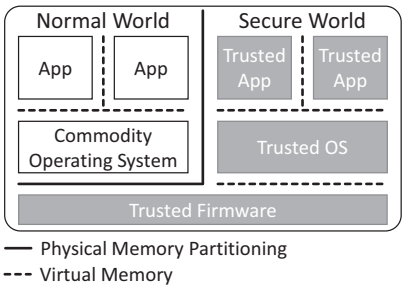

Fig. 1: ARM TrustZone architecture overview.

B. SANCTUARY

SANCTUARY [11] is a security architecture that circumvents the previously explained flaws of ARM TrustZone without requiring hardware extensions, heavy modifications of existing code bases, or major changes in the commodity OS. In particular, it allows to run security-critical code in user-space enclaves or so-called SANCTUARY Apps (SAs). SAs are executed in a normal-world environment that is protected via strict hardware-enforced two-way isolation from all other system components to minimize the attack surface. This is achieved by leveraging TrustZone's address space controller (TZASC) to exclusively bind memory to a (temporarily) dedicated CPU core running an SA.

The life cycle when running an SA is as follows:

1) Setup: Memory for the SA instance is prepared by loading the SANCTUARY library (SL), which is implemented using the Zircon microkernel [45], and the SA. The TZASC is securely configured to isolate this memory region and the least busy CPU core is shut down. Besides the isolated memory, additional memory regions are shared with the commodity OS and the secure world, which allows the SA to access the secure world and (untrusted) OS services.
2) Boot: The memory is attested and the CPU core is booted with the SL providing a basic execution environment.
3) Execution: The SA runs as a normal-world user process, potentially using services provided by the commodity OS or secure world code.
4) Teardown: The CPU core is shut down, data in the first level cache (L1) is invalidated, the SA memory is cleaned and unlocked, and finally the CPU core is handed back to the commodity OS.

SANCTUARY provides code and data integrity as well as data confidentiality, is secure against malicious SAs, and has no negative impact on the user experience due to the wide availability of multicore chips for mobile devices. Furthermore, side-channel attacks that extract secrets from caches can be prevented easily since the L1 cache is core exclusive and the shared second level cache (L2) can be excluded from SANCTUARY memory without severe performance impact [11].

SANCTUARY extends TrustZone to provide an arbitrary number of user-space enclaves. Additionally, SANCTUARY inherits many useful features from TrustZone like secure boot or DMA attack protection. Moreover, TrustZone allows to assign sensitive peripherals exclusively to the secure world.

An SA can use this feature by sending communication requests to the secure world code. After checking the permission rights of the SA, the secure world reads from the sensitive data and directly stores it in the memory region shared with the SA. Thus, performance overhead is only produced by the additional world switches between the SA and the secure world.

IV. Security Model and Assumptions

In this paper, we consider two parties collaborating to perform ML tasks on sensitive data provided by one party while protecting the intellectual property of the other party.

The *user U* provides input data to be processed. She is concerned about the privacy of the content to be processed (i.e., her inputs as well as outputs) and biometric characteristics potentially used throughout processing. Lastly, the user does not want to be traceable across multiple sessions.

The *vendor V* (who might act as the service provider) provides ML algorithms including corresponding models. The models constitute the vendor's intellectual property, hence the user must not be able to reverse engineer, share, or break the license check of these models.

Adversary Model. The adversary's goal is to extract sensitive information, i.e., the intellectual property of the vendor, the input and output 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 user's device. The adversary has full control over the software running in the normal world of the user's device, including privileged software like the commodity OS. We assume that the adversary cannot perform hardware attacks, e.g., a physical side channel to extract secret keys. For the enclave we assume that all of SANCTUARY's defense mechanisms are in place, including hardware cache partitioning (for a detailed discussion see [11]).

V. OMG Design

OMG enables privacy-preserving and efficient offline execution of ML algorithms on untrusted ARM-based systems. For the sake of simplicity, we explain our solution based on the speech recognition scenario visualized in Fig. 2.

The vendor V's private input consists of a ML model. The user U's private input consists of voice recordings. In this example, the ML model is the vendor's intellectual property and any information about its architecture or trained weights must never be disclosed. The only output is the transcription, which is sent to the user.

OMG works in three phases: (I.) preparation, (II.) initialization, and (III.) operation. In the preparation phase, the enclave (containing the SL and SA) is loaded and attested to user U and vendor V. Then, V provides the encrypted ML model to the enclave. In the initialization phase, V sends the decryption key for the ML model so that the enclave can decrypt the model. Finally, in the operation phase, the enclave is ready to perform offline speech recognition. U sends her voice recordings to the enclave and receives respective textual output (which can be further processed into an action, as with virtual assistants). Next, we detail the individual phases:

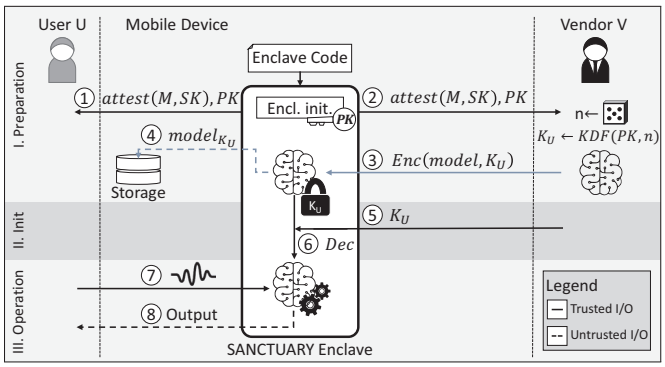

Fig. 2: OMG overview. Once the encrypted model is stored locally, steps in gray are optional until a model update.

I. Preparation Phase. First, the enclave needs to be run on U's device. The enclave contains the environment required to apply the ML model to input data. The enclave code can be open source, since it does not contain any vendor secrets (e.g., it may just consist of a TensorFlow environment), and can be distributed by the device manufacturer via regular distribution channels. To load the enclave, its code is first copied to memory and locked to a dedicated SANCTUARY CPU core so it cannot be changed anymore by the commodity OS (cf. § III-B). Then, the enclave is attested ("measured") by SANCTUARY, 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 the creation process, the measurement will produce a different result and the manipulation will be detected.

SANCTUARY then assigns an unique asymmetric key pair to this enclave, e.g., by using RSA [46] (the public key *PK* is shown in Fig. 2). This key pair is derived from the platform certificate issued by the device vendor, effectively creating a certificate hierarchy similar to SSL certificates. To assure to U that the correct enclave code has been loaded, an attestation report is generated (i.e., the cryptographic hash of the initial memory content is signed using the secret key *SK* corresponding to *PK*) and sent to U using the secure output functionality of SANCTUARY ①. Such an attestation report is also sent to V using a secure connection (e.g., via TLS) directly from the enclave ②.

Note that the attestation report includes the enclave's public key *PK*. V uses *PK* and a nonce n to derive a symmetric encryption key K_U used only for this respective enclave and version of the model. V encrypts the ML model using K_U and securely provisions the model to the enclave ③.

The enclave then stores the model locally in unprotected storage ④. As the model can be loaded from untrusted local storage, after running the preparation phase once, steps ③ and ④ can be omitted until the vendor's model is updated.

II. Initialization Phase. Thanks to never making the decrypted model directly accessible to U, the initialization phase can be kept simple while providing strong guarantees to V. V can actively manage the access of U to the model by either sending or not sending the symmetric key K_U. In case

of, e.g., an expired license, V can stop sending K_U to the enclave, making it fail to decrypt the locally stored model. If V decides that U should be allowed to use the model, V securely sends K_U ⑤ to the enclave and the enclave decrypts the model ⑥. As the key K_U depends on the nonce n, this also prevents rollback attacks for U's locally stored model.

III. Operation Phase. In the operation phase, the actual ML task takes place. U can directly and securely provide voice recordings to the enclave as SANCTUARY allows secure input from peripherals like the microphone ⑦ by utilizing TrustZone features as described in § III-B. The speech data is then processed using the model, the output can be presented to the user or made available to other applications ⑧.

Once in the operation phase, the system can be queried repetitively, thereby avoiding repeated preparation and initialization costs as well as interaction with V. To do this, after a query is processed, the SANCTUARY core can be reallocated to the commodity OS while the memory is still locked such that no device or core is able to access it. When receiving a new query, a new SANCTUARY core is allocated and the locked memory is mapped to it for performing the ML task.

VI. Evaluation

We demonstrate the practicality of our approach by providing a fully functional prototype implementation of OMG on an ARM HiKey 960 development board based on TensorFlow Lite for Microcontrollers [10] and evaluating our prototype with an offline keyword recognition application.

The ARM HiKey 960 development board is equipped with an ARMv8 octa-core SoC (4 cores @ 2.4 GHz, 4 cores @ 1.8 GHz) with 3 GB of RAM, which closely resembles the specifications of today's mobile devices. We use such a development board instead of an off-the-shelf device since most vendors restrict developer access to TrustZone, which prevents us from setting up SANCTUARY (cf. § III-B). As our offline keyword recognition application is just a proof of concept, following [35], we do not focus on best accuracy, but study whether accuracy and runtime are affected when providing strong security guarantees.

The models are trained and evaluated on the Speech Command dataset [47] consisting of 105,000 WAVE audio files of people saying 30 different words. The recordings were post-processed to be a single word per file at a fixed 1 s duration.

We follow the TensorFlow Lite example recipe [10]: Features are computed using a 256 bin fixed point FFT across 30 ms windows (20 ms shift), averaging 6 neighboring bins, resulting in 43 values per frame. The 49 frames for each recording are concatenated, forming a fixed 49×43 compressed spectrogram ("fingerprint") per utterance.

The network architecture resembles [48], but is simplified to better match embedded requirements. The `tiny_conv` architecture feeds the audio fingerprint to a 2D convolutional layer (8 filters, 8×10, x and y stride of 2), followed by ReLU activation and a regular layer that maps to the output labels. During training, dropout is applied after the convolution layer.

TABLE I: Accuracy and runtime results for running the keyword recognition with and without OMG protection.

Model	Accuracy	Runtime
TensorFlow Lite "micro"	75 %	379 ms
TensorFlow Lite "micro" (OMG)	75 %	387 ms

We trained a system for a 12-class problem: *silence*, *unknown*, "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go". The model is first trained using TensorFlow and subsequently converted to a TensorFlow Lite and "micro" model. The resulting compressed model is about 49 kB in size.

We evaluated the "micro" model on a subset of the published test set comprising 10 examples for each class, excluding the two rejection classes "silence" and "unknown", since sensitivity for those would typically be tuned for production.

Inference was run on a 2.4 GHz core of the ARM development board both with and without OMG protection. Tab. I shows the overall accuracy for the 10 classes, and the respective runtimes in milliseconds. The accuracy with and without OMG protection is 75 %, confirming the correctness of the setup. The runtimes are very close when executed with and without OMG protection due to the fact that the hardware-enforced two-way isolation provided by SANCTUARY adds no additional overhead during execution. Since the overall duration of the test set is 100 s, the real-time factor is 0.004x.

The runtime measurements do not include the overhead for collecting the input data from the on-device microphone. As described in § V, OMG uses the capabilities from SANCTUARY to securely connect to sensors. Thus, only the world switch from an SA to the secure world to request the sensor data and the switch back to the SA introduce some overhead. As presented in [11], the switch from an SA to the secure world takes around 0.3 ms. Therefore, even in the short-running speech processing use case presented in this paper, the performance overhead introduced by reading sensor data via the secure world is negligible.

Our evaluation of a keyword recognition task using spectral fingerprints and a basic CNN lays the groundwork to port larger and recurrent architectures as well as to study training tasks. Since our implementation has no inherent memory limitations, it also allows to securely run more complex end-to-end systems, such as the recently released TensorFlow-based dictation model by Google [6], making it highly practical.

VII. Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 850990 PSOTI). It was supported by the DFG (HWSec, project A.1 within the RTG 2050 "Privacy and Trust for Mobile Users", and P3, S2, and E4 within CROSSING), by the BMBF and HMWK within CRISP, and by the Intel Collaborative Research Institute for Collaborative Autonomous & Resilient Systems (ICRI-CARS).

REFERENCES

[1] "Amazon Alexa User Receives 1,700 Audio Recordings of a Stranger through 'Human Error'," https://www.washingtonpost.com/technology/2018/12/20/amazon-alexa-user-receives-audio-recordings-stranger-through-human-error/, 2018.

[2] "Amazon Ordered to Give Alexa Evidence in Double Murder Case," https://www.independent.co.uk/life-style/gadgets-and-tech/news/amazon-echo-alexa-evidence-murder-case-a8633551.html, 2018.

[3] "Apple contractors 'regularly hear confidential details' on Siri recordings," https://www.theguardian.com/technology/2019/jul/26/apple-contractors-regularly-hear-confidential-details-on-siri-recordings, 2019.

[4] "Major breach found in biometrics system used by banks, UK police and defence firms," https://www.theguardian.com/technology/2019/aug/14/major-breach-found-in-biometrics-system-used-by-banks-uk-police-and-defence-firms, 2019.

[5] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein, "Streaming End-to-end Speech Recognition for Mobile Devices," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[6] J. Schalkwyk, "An All-Neural On-Device Speech Recognizer," https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html, 2019.

[7] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel," in *USENIX Security*. USENIX, 2019.

[8] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in *IEEE S&P*. IEEE, 2017.

[9] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The Secret Sharer: Measuring Unintended Neural Network Memorization & Extracting Secrets," *CoRR*, vol. abs/1802.08232, 2018.

[10] "TensorFlow Lite for Microcontrollers," https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/experimental/micro.

[11] F. Brasser, D. Gens, P. Jauernig, A.-R. Sadeghi, and E. Stapf, "SANCTUARY: ARMing TrustZone with User-space Enclaves," in *NDSS*. Internet Society, 2019.

[12] B. D. Rouhani, H. Chen, and F. Koushanfar, "DeepSigns: An End-to-End Watermarking Framework for Ownership Protection of Deep Neural Networks," in *ASPLOS*. ACM, 2019.

[13] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, "DeepMarks: A Secure Fingerprinting Framework for Digital Rights Management of Deep Learning Models," in *International Conference on Multimedia Retrieval (ICMR)*. ACM, 2019.

[14] C. Orlandi, A. Piva, and M. Barni, "Oblivious Neural Network Computing via Homomorphic Encryption," *EURASIP Journal on Information Security*, 2007.

[15] A.-R. Sadeghi and T. Schneider, "Generalized Universal Circuits for Secure Evaluation of Private Functions with Application to Data Classification," in *International Conference on Information Security and Cryptology (ICISC)*. Springer, 2008.

[16] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Privacy-Preserving ECG Classification With Branching Programs and Neural Networks," *Trans. Information Forensics and Security (TIFS)*, vol. 6, no. 2, 2011.

[17] 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 *ICML*. JMLR, 2016.

[18] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," in *IEEE S&P*. IEEE, 2017.

[19] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious Neural Network Predictions via MiniONN Transformations," in *CCS*. ACM, 2017.

[20] 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 *ASIACCS*. ACM, 2018.

[21] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A Low Latency Framework for Secure Neural Network Inference," in *USENIX Security*. USENIX, 2018.

[22] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. E. Lauter, and F. Koushanfar, "XONN: XNOR-based Oblivious Deep Neural Network Inference," in *USENIX Security*. USENIX, 2019.

[23] T. van Elsloo, G. Patrini, and H. Ivey-Law, "SEALion: A Framework for Neural Network Inference on Encrypted Data," *CoRR*, vol. abs/1904.12840, 2019.

[24] M. Dahl, J. Mancuso, Y. Dupis, B. Decoste, M. Giraud, I. Livingstone, J. Patriquin, and G. Uhma, "Private Machine Learning in TensorFlow using Secure Computation," *CoRR*, vol. abs/1810.08130, 2018.

[25] F. Boemer, A. Costache, R. Cammarota, and C. Wierzynski, "nGraph-HE2: A High-Throughput Framework for Neural Network Inference on Encrypted Data," in *Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC)*, 2019, to appear.

[26] 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.

[27] F. Tramèr and D. Boneh, "Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware," in *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[28] 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*. USENIX, 2016.

[29] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-Level Cache Side-Channel Attacks are Practical," in *IEEE S&P*. IEEE, 2015.

[30] Y. Xu, W. Cui, and M. Peinado, "Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems," in *IEEE S&P*. IEEE, 2015.

[31] 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.

[32] T. Hunt, Z. Zhu, Y. Xu, S. Peter, and E. Witchel, "Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data," *Transactions on Computer Systems (TOCS)*, vol. 35, no. 4, 2018.

[33] N. Hynes, R. Cheng, and D. Song, "Efficient Deep Learning on Multi-Source Private Data," *CoRR*, vol. abs/1807.06689, 2018.

[34] S. Chandra, V. Karande, Z. Lin, L. Khan, M. Kantarcioglu, and B. Thuraisingham, "Securing Data Analytics on SGX with Randomization," in *ESORICS*. Springer, 2017.

[35] F. Brasser, T. Frassetto, K. Riedhammer, A.-R. Sadeghi, T. Schneider, and C. Weinert, "VoiceGuard: Secure and Private Speech Processing," in *INTERSPEECH*. ISCA, 2018.

[36] S. Ahmed, A. R. Chowdhury, K. Fawaz, and P. Ramanathan, "Preech: A System for Privacy-Preserving Speech Transcription," *CoRR*, vol. abs/1909.04198, 2019.

[37] L. Hanzlik, Y. Zhang, K. Grosse, A. Salem, M. Augustin, M. Backes, and M. Fritz, "MLCapsule: Guarded Offline Deployment of Machine Learning as a Service," *CoRR*, vol. abs/1808.00590, 2018.

[38] V. Costan and S. Devadas, "Intel SGX Explained," *IACR Cryptology ePrint Archive*, vol. 2016/086, 2016.

[39] T. Frassetto, P. Jauernig, C. Liebchen, and A.-R. Sadeghi, "IMIX: In-Process Memory Isolation EXtension," in *USENIX Security*. USENIX, 2018.

[40] S. Weiser, M. Werner, F. Brasser, M. Malenko, S. Mangard, and A.-R. Sadeghi, "TIMBER-V: Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V," in *NDSS*. Internet Society, 2019.

[41] S. Crane, C. Liebchen, A. Homescu, L. Davi, P. Larsen, A.-R. Sadeghi, S. Brunthaler, and M. Franz, "Readactor: Practical Code Randomization Resilient to Memory Disclosure," in *IEEE S&P*. IEEE, 2015.

[42] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig, "TrustVisor: Efficient TCB Reduction and Attestation," in *IEEE S&P*. IEEE, 2010.

[43] "ARM Security Technology - Building a Secure System using TrustZone Technology," http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, 2009.

[44] "Trust Issues: Exploiting TrustZone TEEs," https://googleprojectzero.blogspot.com/2017/07/trust-issues-exploiting-trustzone-tees.html, 2017.

[45] "Zircon Microkernel," https://fuchsia.googlesource.com/zircon.

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

[47] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *CoRR*, vol. abs/1804.03209, 2018.

[48] T. Sainath and C. Parada, "Convolutional Neural Networks for Small-Footprint Keyword Spotting," in *INTERSPEECH*. ISCA, 2015.