

trusted execution environment (TEE)

Author and editors

Alexander Jung*, Department of Computer Science, Aalto University, Espoo, Finland

Editors: Konstantina Olioumtsevs, Ekkehard Schnoor, Aalto University, Espoo, Finland

*Corresponding author: alex.jung@aalto.fi

Synonyms

secure enclave

Abstract

A trusted execution environment (TEE) is a part of a computing system that runs code and holds data in isolation from the rest of the system. Neither the operating system nor the operator of the hardware can read or alter them, and the TEE can prove to another party which code it runs. Processor features such as Arm TrustZone and Intel SGX provide a TEE on an ordinary chip; a sealed, shielded embedded device without network connection is the physical extreme. In machine learning (ML), a TEE protects three assets: the feature vectors handed to a learned hypothesis, the model parameters of that hypothesis, and the integrity of the computed prediction. It hides nothing about what the prediction reveals, which is the part that differential privacy (DP) addresses, and it protects against physical side channels only with hardware countermeasures. Compared with fully homomorphic encryption (FHE), a TEE runs at near-native speed but rests on trust in the chip vendor rather than on a mathematical proof.

Definition

Consider a Raspberry Pi in a clinic that reads the signal of an electrocardiograph and computes, from the feature vector $\mathbf{x}$ of each heartbeat, the prediction $\hat{h}(\mathbf{x})$ of a learned hypothesis that flags an arrhythmia. The board sits in a sealed metal box with no network cable and no radio. The only inputs are the sensor cable and the power cable, and the only output is a display. The model parameters $\hat{\mathbf{w}}$ of $\hat{h}$ and the recorded heartbeats never leave the box. A TEE is the part of a computing system that provides this kind of protection: code and data inside it cannot be read or altered from outside, and a party outside can verify which code runs inside (Sabt et al., 2015). The sealed box is the physical extreme of a TEE. Most TEEs are instead a mode of an ordinary processor: Arm TrustZone splits the processor of the Raspberry Pi into a normal world, which runs Linux and the display software, and a secure world, which runs the prediction code and holds $\hat{\mathbf{w}}$ and the heartbeats; the normal world cannot read the memory of

the secure world even with root privileges (Pinto and Santos, 2019). Intel SGX provides the same isolation for an enclave inside a process on a server (Costan and Devadas, 2016), and remote attestation lets a client verify a signed digest of the enclave’s code before it sends any data. Fig. 1 shows the Raspberry Pi and what each part can see.

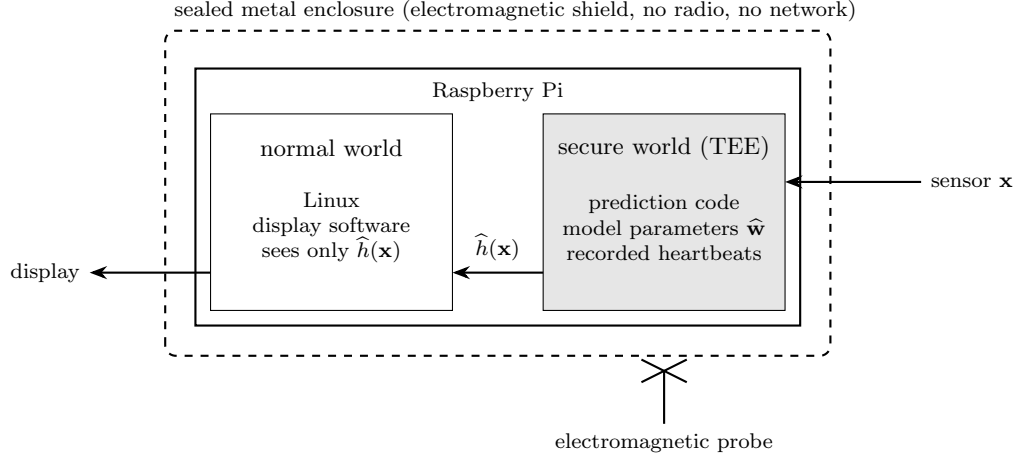


Figure 1: A TEE on a Raspberry Pi. The secure world holds the prediction code, the model parameters and the recorded heartbeats; the normal world receives only the prediction for the display. The sealed enclosure closes the side channels that a probe outside the processor could read

A TEE protects three assets of a machine learning (ML) system. The first is the data: the heartbeats are personal data that the clinic must not disclose. The second is the learned hypothesis: the model parameters $\hat{\mathbf{w}}$ are the outcome of training on a costly dataset, and an attack that reads them out obtains the hypothesis without the dataset. The third is the integrity of the prediction: nobody can alter the code so that it flags fewer arrhythmias. A TEE does not protect what the prediction reveals. A membership inference attack that only observes predictions is unaffected by it, and differential privacy (DP) is the tool for that part. A TEE also does not, by itself, protect against side channels. The current drawn by the processor and the electromagnetic field it emits depend on the data it processes, and a probe held near the chip recovers secret keys from these traces (Kocher et al., 1999). The same probe recovers the model parameters and the layer structure of an artificial neural network (ANN) running on an embedded processor, one multiplication at a time (Batina et al., 2019). A short electromagnetic pulse or a glitch on the supply voltage can also skip an instruction, e.g., the check that rejects unsigned code (Bar-El et al., 2006). The metal enclosure of the Raspberry Pi attenuates the emitted field and blocks the pulse, tamper sensors erase $\hat{\mathbf{w}}$ when the box is opened, and constant-

time code makes the power trace independent of the data. Server TEEs face a software side channel instead: the memory access pattern of an enclave is visible to the operating system, so ML code inside an enclave must be data-oblivious, with access patterns that do not depend on the data (Ohrimenko et al., 2016), and speculative execution has leaked enclave memory outright (Van Bulck et al., 2018).

Two uses of TEEs beyond the sealed box are prediction in the cloud and federated learning (FL). A hospital that rents a server can require an SGX enclave, verify by attestation that it runs the agreed prediction code, and send its feature vectors encrypted to the enclave’s key, so that the cloud provider never sees them (Ohrimenko et al., 2016). In FL, an enclave on the server implements secure aggregation: the clients encrypt their model parameters updates to the enclave, which decrypts and sums them, and the server operator sees only the sum; the TrustZone secure world of each phone protects the local update in turn (Mo et al., 2021).

A TEE, fully homomorphic encryption (FHE) and DP protect against different parties by different means. FHE hides the data from the computing party by mathematics, at a cost of orders of magnitude in time and memory, and needs no trust in the hardware. A TEE hides the data by hardware isolation at near-native speed, and rests on trust in the chip vendor and in the absence of side channels. DP bounds what the prediction reveals about any single data point and is combined with either.

Cross References

FHE, secure aggregation, DP, privacy protection, privacy attack, membership inference attack, attack, FL, edge device, cloud computing, ML system, trustworthy AI

References

1. Bar-El et al. (2006). The Sorcerer’s Apprentice Guide to Fault Attacks. *Proceedings of the IEEE*, 94(2), pp. 370–382.
2. Batina et al. (2019). CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In: *Proc. 28th USENIX Security Symposium*, pp. 515–532.
3. Costan and Devadas (2016). Intel SGX Explained. *IACR Cryptology ePrint Archive*, Report 2016/086.
4. Kocher et al. (1999). Differential Power Analysis. In: *Advances in Cryptology – CRYPTO ’99*, pp. 388–397. Springer.
5. Mo et al. (2021). PPFL: Privacy-preserving Federated Learning with Trusted Execution Environments. In: *Proc. 19th Annual Int. Conf. Mobile Systems, Applications, and Services (MobiSys)*, pp. 94–108.

6. Ohrimenko et al. (2016). Oblivious Multi-Party Machine Learning on Trusted Processors. In: Proc. 25th USENIX Security Symposium, pp. 619–636.
7. Pinto and Santos (2019). Demystifying Arm TrustZone: A Comprehensive Survey. ACM Computing Surveys, 51(6), pp. 130:1–130:36.
8. Sabt et al. (2015). Trusted Execution Environment: What It is, and What It is Not. In: Proc. IEEE Trustcom/BigDataSE/ISPA, pp. 57–64.
9. Van Bulck et al. (2018). Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In: Proc. 27th USENIX Security Symposium, pp. 991–1008.