vertrauenswürdige Ausführungsumgebung (TEE)

Autor und Herausgeber

Alexander Jung*, Department of Computer Science, Aalto University, Espoo, Finland (ORCID 0000-0001-7538-0990)
Editors: Konstantina Olioumtsevits (ORCID 0000-0003-2697-8659), Ekkehard Schnoor (ORCID 0000-0001-8252-1499), Aalto University, Espoo, Finland
*Corresponding author: alex.jung@aalto.fi

Zusammenfassung

Eine vertrauenswürdige Ausführungsumgebung (engl. *trusted execution environment*, TEE) ist ein Teil eines Rechensystems, der Code und Daten von anderer Software isoliert. Selbst das Betriebssystem kann ihren geschützten Speicher nicht direkt lesen oder verändern. Durch Fernattestierung lässt sich gegenüber einer anderen Partei nachweisen, welcher Code darin ausgeführt wird. Prozessorfunktionen wie Arm TrustZone und Intel SGX stellen TEEs auf gewöhnlichen Chips bereit. Beim maschinelles Lernen (ML) schützt eine TEE eingegebene Merkmalsvektoren, gelernte Modellparameter und die Integrität der berechneten Vorhersage. Sie begrenzt jedoch nicht, was die Vorhersage preisgibt; dieses Problem wird durch differenzielle Privatheit (DP) adressiert. Physische und softwarebasierte Seitenkanäle erfordern zusätzliche Gegenmaßnahmen. Im Vergleich zur vollständig homomorphe Verschlüsselung (FHE) arbeitet eine TEE mit nahezu nativer Geschwindigkeit, ersetzt aber eine mathematische Garantie durch das Vertrauen in den Chiphersteller und dessen Hardware.

Definition

Betrachtet sei eine eingebettete Steuereinheit in einem gekühlten Medikamentenschrank. Jede Minute liest sie einen Temperatursensor aus und bildet aus der aktuellen Temperatur und deren jüngsten Änderungen einen Merkmalsvektor $\mathbf{x}$. Eine gelernte Hypothese berechnet die Vorhersage $\widehat{h}(\mathbf{x})$ darüber, ob das Kühlsystem ausfällt. Die Steuereinheit befindet sich in einem versiegelten Metallgehäuse ohne Netzwerkkabel und ohne Funkverbindung. Ihre einzigen Eingänge sind das Sensorkabel und das Stromkabel, und ihr einziger Ausgang ist eine Alarmanzeige. Die Modellparameter $\widehat{\mathbf{w}}$ von $\widehat{h}$ und die aufgezeichneten Temperaturmesswerte verlassen das Gehäuse nie.

Eine TEE ist der Teil des Prozessors, der die logische Isolation bereitstellt. Software außerhalb davon kann ihren geschützten Code und Speicher nicht direkt lesen oder verändern, selbst wenn es sich bei dieser Software um ein privilegiertes Betriebssystem handelt (Sabt et al., 2015). Das Metallgehäuse ist nicht die TEE; es ist eine separate Schicht, die den Prozessor gegen physisches Ausspähen und Fehlerinjektion schützt.

Die meisten TEEs sind ein Modus eines gewöhnlichen Prozessors. Arm TrustZone teilt den Prozessor der Steuereinheit in zwei Teile. Die normale Welt führt

die Anzeigesoftware aus. Die sichere Welt führt den Code für die Vorhersage aus und hält $\widehat{\mathbf{w}}$ sowie die Temperaturmesswerte. Die normale Welt kann den Speicher der sicheren Welt selbst mit Root-Rechten nicht lesen (Pinto and Santos, 2019).

Intel SGX bietet eine ähnliche Isolation für eine Enklave innerhalb eines Prozesses auf einem Server (Costan and Devadas, 2016). Die Fernattestierung ermöglicht es einem Client, einen signierten Prüfwert des Enklavencodes zu verifizieren, bevor er Daten sendet. Die Isolationsgarantie betrifft den direkten Zugriff über Software- und Speicherschnittstellen; Informationen, die durch Ausgaben, Seitenkanäle und physische Angriffe preisgegeben werden, erfordern zusätzlichen Schutz. Abb. 1 zeigt die eingebettete Steuereinheit und was jeder Teil sehen kann.

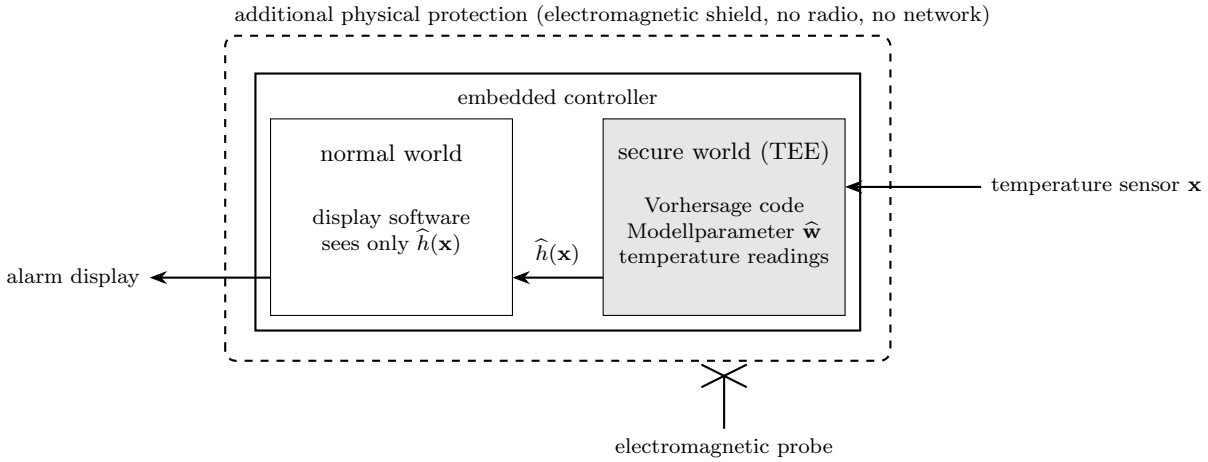

Abbildung 1: Eine TEE in einer eingebetteten Steuereinheit zur Temperaturüberwachung. Die sichere Welt (die TEE) hält den Code für die Vorhersage, die Modellparameter und die aufgezeichneten Temperaturmesswerte; die normale Welt erhält lediglich die Vorhersage für die Alarmanzeige. Unabhängig davon reduziert das versiegelte Gehäuse die physischen Seitenkanäle, die einer Sonde außerhalb des Prozessors zur Verfügung stehen

Eine TEE schützt drei Werte eines maschinelles Lernen (ML)-Systems. Der erste sind die Daten: Der Temperaturverlauf ist eine Betriebsaufzeichnung, die der Betreiber des Schranks weder offenlegen noch durch einen Angreifer verändern lassen darf. Der zweite ist die gelernte Hypothese. Ihre Modellparameter $\widehat{\mathbf{w}}$ sind das Ergebnis des Trainings auf einem kostspieligen Datensatz. Ein Angriff, der sie ausliest, erlangt die Hypothese ohne den Datensatz. Der dritte ist die Integrität der Vorhersage: Niemand kann den Code so verändern, dass er einen Alarm unterdrückt, wenn das Kühlsystem ausfällt.

Eine TEE schützt nicht das, was die Vorhersage preisgibt. Ein Membership-

Inference-Angriff, der lediglich Vorhersagen beobachtet, bleibt davon unberührt; für diesen Teil ist differenzielle Privatheit (DP) zuständig.

Eine TEE schützt für sich genommen auch nicht gegen physische Seitenkanäle. Der Stromfluss und das elektromagnetische Feld des Prozessors hängen von den verarbeiteten Daten ab. Eine nahe an den Chip gehaltene Sonde kann aus diesen Spuren geheime Schlüssel rekonstruieren (Kocher et al., 1999). Dieselbe Sonde rekonstruiert die Modellparameter und die Schichtstruktur eines auf einem eingebetteten Prozessor laufenden künstliches neuronales Netz (KNN)s, Multiplikation für Multiplikation (Batina et al., 2019). Ein kurzer elektromagnetischer Impuls oder eine Störung der Versorgungsspannung kann außerdem einen Befehl überspringen, z. B. die Prüfung, die unsignierten Code abweist (Bar-El et al., 2006).

Das Metallgehäuse dämpft das abgestrahlte Feld und blockiert den Impuls. Manipulationssensoren löschen $\widehat{\mathbf{w}}$, wenn das Gehäuse geöffnet wird. Zeitkonstanter Code macht den Stromverlauf unabhängig von den Daten.

Server-TEEs sind jedoch zusätzlich Seitenkanälen auf Software- und Prozessorebene ausgesetzt. Das Betriebssystem kann das Speicherzugriffsmuster einer Enklave einsehen. Daher muss ML-Code innerhalb einer Enklave datenunabhängig (engl. *data-oblivious*) sein. Seine Zugriffsmuster dürfen nicht von den Daten abhängen (Ohrimenko et al., 2016). Spekulative Ausführung hat ebenfalls dazu geführt, dass Enklavenspeicher direkt abfloss (Van Bulck et al., 2018).

Zwei Anwendungen von TEEs jenseits des versiegelten Gehäuses sind die Vorhersage in der Cloud und das föderiertes Lernen (FL). Ein Krankenhaus, das einen Server mietet, kann eine SGX-Enklave verlangen. Es nutzt zunächst die Attestierung, um den vereinbarten Code für die Vorhersage zu verifizieren. Anschließend verschlüsselt es seine Merkmalsvektoren für die Enklave, sodass der Cloud-Anbieter sie nie zu Gesicht bekommt (Ohrimenko et al., 2016).

Beim FL kann eine Server-Enklave eine hardwarebasierte Implementierung der sichere Aggregation bereitstellen. Clients verschlüsseln die Aktualisierungen ihrer Modellparameter für die Enklave. Diese entschlüsselt und summiert die Aktualisierungen, während der Serverbetreiber nur die Summe sieht. Unabhängig davon kann die sichere TrustZone-Welt jedes Telefons dessen lokale Aktualisierung vor der Übertragung schützen; ein System kann einen der beiden Schutzmechanismen nutzen oder beide kombinieren (Mo et al., 2021).

Eine TEE, vollständig homomorphe Verschlüsselung (FHE) und DP bieten unterschiedliche Garantien mit unterschiedlichen Mitteln. FHE verbirgt die Daten während der Berechnung, ohne dem Prozessor zu vertrauen, allerdings auf Kosten von Größenordnungen an Zeit und Speicherplatz. Eine TEE erreicht nahezu native Geschwindigkeit, indem sie dem Chiphersteller, der Hardwareimplementierung und den Gegenmaßnahmen gegen Seitenkanäle vertraut. DP begrenzt, was die veröffentlichte Vorhersage über jeden einzelnen Datenpunkt preisgibt (Dwork and Roth, 2014, Def. 2.4), und lässt sich mit beiden Ansätzen kombinieren.

Synonyme: sichere Enklave.

Siehe auch: FHE, sichere Aggregation, DP, Datenschutz, Datenschutzangriff, Membership-Inference-Angriff, Angriff, FL, Edge-Gerät, Cloud Compu-

ting, ML-System (ML-System), vertrauenswürdige künstliche Intelligenz (vertrauenswürdige KI).

Querverweise

Literatur

1. Dwork and Roth (2014). The Algorithmic Foundations of Differential Privacy. Foundations and Trends®in Theoretical Computer Science, 9(3–4), pp. 211–407.

2. Bar-El et al. (2006). The Sorcerer's Apprentice Guide to Fault Attacks. Proceedings of the IEEE, 94(2), pp. 370–382.

3. 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.

4. Costan and Devadas (2016). Intel SGX Explained. IACR Cryptology ePrint Archive, Report 2016/086.

5. Kocher et al. (1999). Differential Power Analysis. In: Advances in Cryptology – CRYPTO '99, pp. 388–397. Springer.

6. 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.

7. Ohrimenko et al. (2016). Oblivious Multi-Party Machine Learning on Trusted Processors. In: Proc. 25th USENIX Security Symposium, pp. 619–636.

8. Pinto and Santos (2019). Demystifying Arm TrustZone: A Comprehensive Survey. ACM Computing Surveys, 51(6), pp. 130:1–130:36.

9. Sabt et al. (2015). Trusted Execution Environment: What It is, and What It is Not. In: Proc. IEEE Trustcom/BigDataSE/ISPA, pp. 57–64.

10. 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.