Security for Distributed Machine Learning

Laurent Gomez[1], Tianchi Yu[1] and Patrick Duverger[2]
[1]*SAP Security Research, SAP Labs France, Mougins, France*
[2]*City of Antibes, France*

Keywords: Machine Learning, Edge Computing, Intellectual Property, Data Privacy, Privacy Enhancing Technology, Trusted Execution Environment.

Abstract: With the adoption of IoT-like technologies, industrials aim to enhance the business value of their physical assets and improve their operational efficiency. However, IoT devices alone tend to strain enterprise systems with a sheer volume of unstructured and unfiltered data. To overcome this challenge, endowing (smart) devices with AI-based capabilities can significantly enhance enterprise system capabilities. However, deploying AI-based capabilities on potentially insecure edge hardware and platforms introduces new security risks, including AI model theft, poisoning, and data leaks. This paradigm shift necessitates the protection of distributed AI applications and data. In this paper, we propose a solution for safeguarding the Intellectual Property and data privacy of ML-based software. We utilize hardware-assisted Privacy Enhancing Technologies, specifically Trusted Execution Environments. We evaluate the effectiveness of our approach in the context of ML-based motion detection in CCTV cameras. This work is part of a co-innovation project with the Smart City of Antibes, France.

1 MOTIVATION

1.1 Context

The increasing interconnection of physical objects has given rise to trends such as Industrial IoT, Industry 4.0, and Edge Computing. These initiatives aim to maximize business value and enhance operational efficiency by converging Information and Operational Technology (IT/OT). However, IoT technology alone lacks intelligence and floods centralized Enterprise Systems with unstructured data. Combining IoT with AI-based capabilities enables distributed decision-making, reducing latency and costs while improving insights. This has numerous applications in predictive maintenance, traffic management, and production optimization. As IoT technologies are adopted by Enterprise Systems, new security challenges arise. Deploying AI on potentially un-trusted edge hardware and platforms exposes ML models' confidentiality and data privacy to attacks.

1.2 Problem Statement

The Machine Learning Development Life-Cycle (MLDLC) consists of three phases: Data Engineering, AI-based Software Engineering, and AI-based Software Deployment and Execution. This paper focuses on security concerns at the Deployment and Execution phase, specifically data privacy leaks and model theft. Attacks such as membership inference, property inference, or model update attacks put at risk privacy of ML data, while model reconstruction and model extraction attacks aim to steal AI models and intellectual property. Edge computing further intensifies these security challenges due to resource limitations, platform diversity, and increases attack surface on Enterprise Systems.

1.3 Our Approach

This paper proposes a solution for secure deployment and execution of AI-based software on edge devices. We target Intellectual Property (IP) protection and data privacy by leveraging Trusted Execution Environments (TEEs). TEEs provide isolated and secure execution environments - enclaves -, safeguarding code and memory against unauthorized access or modification. Local and remote attestation further validate the integrity of enclaves instantiated on trusted platforms. The protocol involves deploying AI-based software in a secure enclave on an edge

Gomez, L., Yu, T. and Duverger, P.
Security for Distributed Machine Learning.
DOI: 10.5220/0012137700003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 838-843
ISBN: 978-989-758-666-8; ISSN: 2184-7711

device by the model owner (MO). The query data owner (QDO) executes the model using her query input data. The encryption material for query data is protected within the secure enclave and shared with the AI-based software enclave for model evaluation.

1.4 Paper Organization

The remainder of this paper is organized as follows. Section 2 explores the state of the art on security for machine learning at the deployment and execution phase. Section 3 elaborates on our approach, outlining the use of Trusted Execution Environments (TEE). Section 4 evaluates our approach's performance and feasibility through a smart city scenario for risk prevention in public spaces. We conclude and discuss future research directions in Section 5.

2 STATE OF THE ART

Numerous approaches exist for securing ML models and associated data during deployment and execution. These approaches can be categorized as crypto-based PETs and hardware-assisted PETs.

2.1 Crypto-Based PETs

2.1.1 Fully Homomorphic Encryption (FHE)

FHE schemes (Brakerski et al., 2012; Fan and Vercauteren, 2012) enable computations on encrypted data without revealing the underlying data. CryptoNets (Dowlin et al., 2016) was an early FHE-based approach for Neural Network inference. However, FHE incurs high latency and resource overhead. Recent advancements by Chabanne et al.(Chabanne et al., 2017) and Juvekar et al.(Juvekar et al., 2018) have improved performance and accuracy in cryptographic schemes. While crypto-based PETs offer strong security guarantees, they introduce resource overhead and may require model modification and retraining. Moreover, their computational expressiveness is limited.

2.2 Hardware-Assisted PETs

Trusted Execution Environments (TEE) like Intel SGX, ARM TrustZone, and AWS Nitro provide secure enclaves for isolated code execution. TEE-based approaches have been proposed, such as SLALOM (Tramèr and Boneh, 2018) and CHEMIX (Gupta and Raskar, 2018). However, existing hardware-assisted PETs primarily focus on input data privacy and do not address model protection. Key management for encryption is also left open. In summary, securing ML models at deployment and execution involves a trade-off between security and resource consumption. Crypto-based PETs provide strong security but introduce major overhead, while hardware-assisted PETs offer hardware-based security with limitations in model protection.

3 SECURITY FOR DISTRIBUTED MACHINE LEARNING

We present a solution to protect the Intellectual Property (IP) of distributed AI-based software and ensure privacy of associated data. Our approach leverages hardware-assisted Privacy Enhancing Technologies (PETs), specifically Trusted Execution Environments (TEEs). We extend the SLALOM approach introduced in Section 2, enabling privacy of query input data and secure inference within secure enclaves.

3.1 SLALOM Protocol Limitations

SLALOM uses a "slicing technique" to divide the neural network into smaller sub-networks for parallel execution. Linear layers are delegated to a co-located GPU, while non-linear layers are executed within the TEE. However, SLALOM lacks means to safeguard the IP of ML-based software, as model parameters are transferred in plain text to untrusted co-located GPU for acceleration.

3.2 Overall Protocol

Depicted in Figure 1, our protocol involves two actors: the ML model owner (MO) and the query data owner (QDO). The AI-based software is sealed in a secure enclave by the MO and securely shared with the edge device TEE through remote attestation.

Prior to any AI-based model evaluation, QDO self-generates encryption keys, embedded within a key enclave, ans deployed along remote attestation on the edge device. When the QDO queries the AI-based software, their input data is encrypted using that self-generated encryption key and sent to the evaluation enclave. At model evaluation, the data is decrypted within the secure evaluation enclave and processed based on the MO-QDO agreement. The processing result can be encrypted and sent to the MO, QDO, or both, based on agreed access control policy.

Our protocol achieves the following objectives:

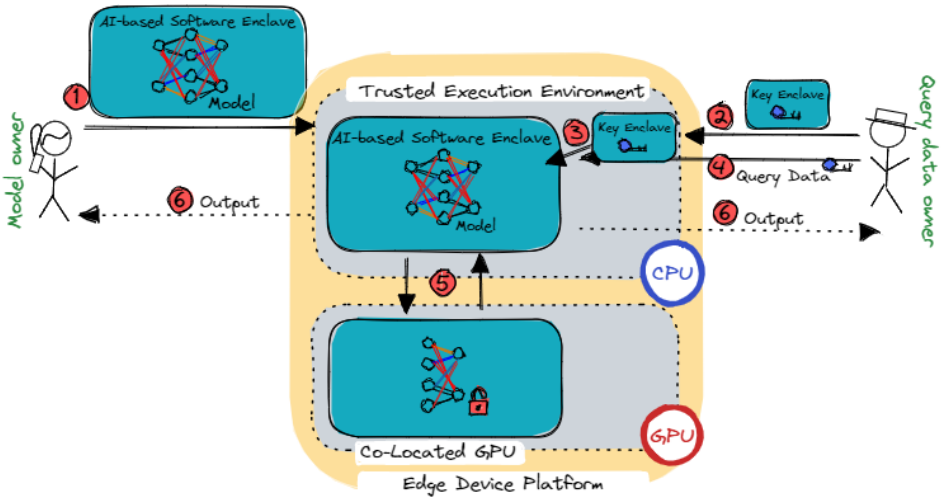

Figure 1: Overall Protocol.

- **AI-Based Software IP Safeguarding:** We deploy the AI-based software on TEE to protect the model's IP and leverage co-located GPU acceleration extending SLALOM protocol.
- **Data Privacy:** Input and output data of the AI-based software are encrypted to ensure data privacy.

3.3 Distributed QDO Key Management

Our approach requires each QDO to encrypt their data using their own cryptographic material. To ensure proper key management and control, we encapsulate the key material within dedicated secure enclaves co-located with the AI-based software enclave. The cryptographic material is securely deployed on the edge device through local attestation between the cryptographic material enclaves and the AI-based software enclave.

3.4 Model and Query Data Protection

During the evaluation phase, we adopt the "slicing network" technique, as described in SLALOM, to split the execution of the AI-based software model. Non-linear layers are processed within the enclave on the CPU, while linear layers are delegated to a co-located GPU. To ensure data privacy, we encrypt the input of the linear layers using a pre-computed pseudo-random stream. In SLALOM, only the data inputs delegated to the GPU are encrypted, while the model parameters (e.g., weights, bias) are sent in clear text.

In our proposed approach, we guarantee privacy of model parameters by enhance SLALOM algorithm as follows:

1: Linear Layer Evaluation:
2: $u1 = r1 * W$ (TEE; pre-computed)
3: $u2 = r1 * r2$ (TEE; pre-computed)
4: $u3 = r2 * x$ (TEE)
5: $\bar{x} = x + r1; \bar{W} = W + r2$ (TEE)
6: $\bar{y} = \bar{x} * \bar{W}$ (GPU)
7: $y = \bar{y} - u1 - u2 - u3$ (TEE)
8: **assert** $Freivalds(y, x, W)$ (TEE)
9: Non-Linear Layer Evaluation:
10: $x = F_{non-linear}(y)$

In this enhanced algorithm, for linear layer evaluation, both the model parameters W and the input data x are encrypted within the TEE (lines 1-5), with $r1$ and $r2$ pseudo-random streams. The evaluation of linear layers is then delegated to the GPU with the encrypted inputs (line 6). The resulting output is decrypted within the TEE (line 7).

While this approach ensures model and data privacy during the delegation of linear layers, it introduces additional overhead due to the decryption process within the AI-based software enclave. The impact of this overhead is evaluated in Section 4.

3.5 Query Output Access Control

The AI-based software evaluates the access control policy on evaluation outcome, as agreed between the QDO and MO. The overhead introduced by AES-256 encryption is negligible, as it is performed only once per model evaluation.

4 EVALUATION

In this section, we discuss the evaluation of a demonstration implementing our approach to a smart city use case for risk prevention in public spaces. We elaborate more on our technical architecture and performance evaluation of this demonstrator.

4.1 Risk Prevention in Public Spaces

We assess the performance and feasibility of embedding AI-based motion detection within a CCTV camera in the city of Antibes for risk prevention in public spaces. The encrypted video stream is processed locally on the camera. Motion detection classification (e.g., jumping, falling, cycling) triggers automated alerts to public safety and security forces. Our aim is to optimize the use of Public Safety & Security resources and maximize the value of existing CCTV systems. Following the previously introduced terminology on Model Owner and Query Data Owner, we consider the ML-based motion detection software provider as the MO and the smart city as the QDO (e.g video stream owner).

4.2 ST-GCN-Based Motion Detection

Motion detection is crucial for analyzing human activities and identifying incidents in hazardous areas. In our demonstrator, we employ a Spatial-Temporal Graph Convolutional Network (ST-GCN) (Yan et al., 2018). ST-GCN combines Graph Convolutional Networks (GCNs) and Temporal Convolutional Networks (TCNs) to enhance motion detection precision. Notable ST-GCN models include ASGCN (Shi et al., 2019), and ST-TR (Plizzari et al., 2021).

4.3 Overall Protocol

In Figure 3, motion detection process is broken down into two main steps: (i) extraction of human skeletons as graph of joints, (ii) ST-GCN-based human motion detection (e.g., standing, sitting, walking).

4.3.1 Graph-Based Skeleton Extraction

As implemented by (Yan et al., 2018), we employ OpenPose (Cao et al., 2019) to extract graph-based skeletons from video streams. OpenPose is a real-time multi-person keypoint detection library that combines pose estimation and image processing techniques for estimating 2D poses of multiple individuals in an image. The keypoints represent joints and body parts (e.g., head, elbows, hips, or knees), and OpenPose generates a graph-based representation of these interconnected keypoints through post-processing. We perform graph-based skeleton detection using OpenPose outside of the TEE due to the complexity of implementation and dependencies on third-party libraries, which deter code migration into enclaves.

4.3.2 ST-GCN Implementation

Our evaluation is made on the deployment of the ST-GCN model within an enclave on Jetson Xavier AGX (NVIDIA, 2019). The ST-GCN model comprises Convolution Layers and an average pooling layer. GCN performs Spatial Graph Convolution by taking feature maps and a normalized weighted adjacency matrix as inputs. It multiplies the feature maps with the adjacency matrix to process the data in each frame. TCN operates like typical CNNs, extracting features. The Residual Networks' output is bitwise added to the result of GCN+TCN, addressing gradient issues, preserving inference accuracy, and preventing network degradation. Figure 2 illustrates the ST-GCN model architecture.

4.4 Hardware Setup

We implement our prototype on an NVIDIA Jetson AGX Xavier platform (NVIDIA, 2019). This edge device is equipped with an ARM-based TEE, based on TrustZone Technology. TEE functionalities for enclave generation & execution, local and remote attestation are abstracted through the software abstraction layer Trusty (Android Open Source Project, 2019). Trusty is part of the Android Open Source Project (AOSP). Hardware-wise, Jetson Xavier AGX is equipped with 64 GPU tensor cores, 8 CPU tensor cores. With 32 GB of RAM, this device can allocate up to 128 MB of RAM to Trusty, for secure enclave development. This limits the processing capabilities within enclaves. Further technical specifications can be found here (NVIDIA, 2019).

4.5 Technical Architecture

4.5.1 Distributed QDO Key Management

To ensure query data confidentiality and privacy, each Query Data Owner (QDO) encrypts their data symmetrically using self-generated AES-128 keys, embedded in a key enclave. Remote attestation, with RSA-2048 encryption, is used for key enclave deployment on the edge device (step 2 in Figure 1). AES-128 keys are hard-coded in the key enclaves for simplicity

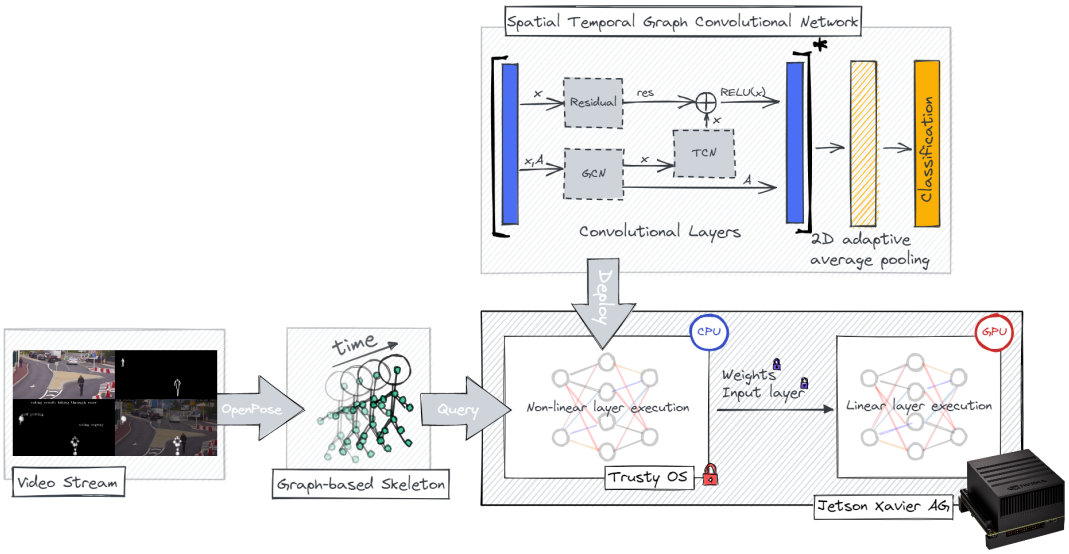

Figure 2: Architecture.

in our experiment. Key enclaves guarantee key confidentiality and integrity. The evaluation enclave decrypts QDO inputs (step 3 in Figure 1) by extracting cryptographic materials from key enclaves. Local attestation, based on RSA-2048, enables key exchange between key and model evaluation enclaves.

Communication overhead from key exchange is minimal, occurring once per ML model evaluation. It is insignificant compared to encryption and communication iterations between the evaluation enclave and the GPU.

4.5.2 Protection of AI-Based Software Intellectual Property

Figure 1 depicts the secure deployment and evaluation of the ST-GCN model within an enclave for model integrity and confidentiality on the edge device. In step 4, we use the "slicing network" technique to accelerate model evaluation with the co-located GPU. Linear layers (convolutional layers) are executed on the co-located GPU for hardware acceleration, while non-linear layers (RELU, Batch Normalization, and Average Pooling) are executed in the AI-based Software Enclave.

To protect the privacy and confidentiality of model parameters and input data, we employ a pseudo-random stream encryption technique. It encrypts delegated data and parameters using random data streams generated uniformly, with element-wise addition to the plaintext. Using a secure enclave and co-located GPUs for data privacy and IP protection increases communication overhead between the enclave and GPU, impacting performance due to additional encryption and decryption cycles and random data generation.

4.5.3 Query Output Fairness Access

In our use case, the model evaluation outcome is limited to the city and intended for police forces and public safety services. We encrypt the outcome using an AES-128 encryption key owned by the city of Antibes, embedded within the evaluation enclave. The impact on prototype performance is minimal, occurring only once.

4.6 Results

We assess the performance of our demonstrator in terms of processing time, communication cost, and model accuracy to evaluate the feasibility of our approach in a real use case. We evaluate our prototype dynamically, without pre-computed optimizations. All random number generation and matrix operations are performed at run-time.

4.6.1 Processing Overhead

Excluding OpenPose execution, we measure the processing overhead of graph-based skeleton evaluation using the ST-GCN model. Specifically, we focus on GPU-based linear layer execution, TEE-based non-linear layer execution, enclave-GPU communication, and encryption/decryption rounds. Our TEE-based implementation adds 28.7 seconds to the evaluation time, compared to the original ST-GCN model's 0.8 to 1.2 seconds for a 10-second video stream. The

breakdown of processing time is as follows: 2% for linear layer execution, 55% for non-linear layer execution, 15% for communication, and 20% for data encryption/decryption. The lack of GPU acceleration for executing non-linear layers within the secure enclave contributes to over half of the processing overhead.

4.6.2 Communication Cost

The "slicing technique" necessitates frequent communication between the secure enclave and GPU. When the model evaluation enclave delegates linear layer processing to the GPU, the layer parameters and input data must be transferred to the GPU. After processing, the GPU-based application sends the output back to the evaluation enclave. Communication between the enclaves and GPU is limited to a maximum payload size of 4 pages or 4KB. Considering that the average size of a linear layer is 3.375 MB, delegating a single linear layer leads to around 863 communication rounds between the TEE and GPU. This high number of communication rounds highlights the potential overhead of the "slicing technique".

4.6.3 Accuracy Loss

We evaluate motion detection accuracy by comparing the top-5 predicted labels and mean losses of our implementation to the original model. Our implementation achieves a 98% match with the top-5 predictions, demonstrating similar accuracy to the original model. Additionally, we assess the mean loss of our implementation compared to the original model. We observe an average accuracy loss of 5% resulting from approximating the use of 2-digit floating points to reduce communication rounds between the TEE and GPU.

5 CONCLUSION

In this paper, we propose a solution to protect AI based software's Intellectual Property and preserve data privacy. We evaluate our approach in a real-world scenario of risk prevention in public spaces, embedding ML-based motion detection within CCTV cameras. Despite increased processing time, our approach demonstrates feasibility without significant accuracy loss. We also identify potential optimization on communication rounds between the secure enclave and the GPU, such as pre-computation of random stream generation. As future work, we plan to extend our approach to cloud confidential computing, addressing security threats in edge device TEE and improving processing time. Initial experiments in this direction show promising results.

REFERENCES

Android Open Source Project (2019). Trusty. "source.android.com/docs/security/features/trusty".

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2012). Fully homomorphic encryption without bootstrapping. In *3rd Innovations in Theoretical Computer Science Conference*.

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., and Sheikh, Y. A. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Chabanne, H., de Wargny, A., Milgram, J., Morel, C., and Prouff, E. (2017). Privacy-preserving classification on deep neural network. *IACR Cryptol. ePrint Arch.*, page 35.

Dowlin, N., Gilad-Ba, R., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2016). Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML'16: Proceedings of the 33rd International Conference on International Conference on Machine Learning*.

Fan, J. and Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*.

Gupta, O. and Raskar, R. (2018). Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8.

Juvekar, C., Vaikuntanathan, V., and Chandrakasan, A. (2018). Gazelle: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium*.

NVIDIA (2019). Jetson XAVIER AGX. "https://www.nvidia.com/fr-fr/autonomous-machines/embedded-systems/jetson-agx-xavier/".

Plizzari, C., Cannici, M., and Matteucci, M. (2021). Spatial temporal transformer network for skeleton-based action recognition. In *Pattern Recognition. ICPR International Workshops and Challenges*, pages 694–701, Cham. Springer International Publishing.

Shi, L., Zhang, Y., Cheng, J., and Lu, H. (2019). Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tramèr, F. and Boneh, D. (2018). Slalom: Fast, verifiable and private execution of neural networks in trusted hardware.

Yan, S., Xiong, Y., and Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the Thirty-Second Annual Conference on Innovative Applications of Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.