The High-Level Practical Overview of Open-Source Privacy-Preserving Machine Learning Solutions

Konrad Kuźniewski, Krystian Matusiewicz, Piotr Sapiecha

Abstract—This paper aims to provide a high-level overview of practical approaches to machine-learning respecting the privacy and confidentiality of customer information, which is called Privacy-Preserving Machine Learning. First, the security approaches in offline-learning privacy methods are assessed. Those focused on modern cryptographic methods, such as Homomorphic Encryption and Secure Multi-Party Computation, as well as on dedicated combined hardware and software platforms like Trusted Execution Environment - Intel® Software Guard Extensions (Intel® SGX), Combining the security approaches with different machine learning architectures leads to our Proof of Concept in which the accuracy and speed of the security solutions will be examined. The next step was exploring and comparing the Open-Source Python-based solutions for PPML. Four solutions were selected from almost 40 separate, state-of-the-art systems: SyMPC, TF-Encrypted, TenSEAL, and Gramine. Three different Neural Network architectures were designed to show different libraries’ capabilities. The POC solves the image classification problem based on the MNIST dataset. As the computational results show, the accuracy of all considered secure approaches is similar. The maximum difference between non-secure and secure flow does not exceed 1.2%. In terms of secure computations, the most effective Privacy-Preserving Machine Learning library is based on Trusted Execution Environment, followed by Secure Multi-Party Computation and Homomorphic Encryption. However, most of those are at least 1000 times slower than the non-secure evaluation. Unfortunately, it is not acceptable for a real-world scenario. Future work could combine different security approaches, explore other new and existing state-of-the-art libraries or implement support for hardware-accelerated secure computation.

Keywords—Privacy-Preserving Machine Learning, Homomorphic Encryption, Secure Multi Party Computation, Trusted Execution Environment

I. INTRODUCTION

Recent advances in Machine Learning (ML) or Deep Learning (DL) techniques have demonstrated outstanding performance on various tasks, including organ recognition from medical images, classification of interstitial lung diseases, detection of lung nodules, medical image reconstruction, and segmentation of brain tumors. The advantage of DL models over humans has resulted in the development of computer-aided diagnosis systems - for example, the United States Food and Drug Administration recently approved the approval of an intelligent diagnosis system for medical images that do not require human intervention [1], [2].

Nowadays, deep model training and evaluation are frequently outsourced to clouds, referred to in the literature as Machine Learning as a Service (MLaaS). Cloud providers like Google, Microsoft Azure, or Amazon Web Services offer these services. Despite the impressive performance of DL algorithms, numerous recent studies have raised concerns about the security and robustness of machine learning models [3]–[5]. Moreover, the security of such algorithms’ execution environments is being questioned [6]–[8]. The realization that DL models are neither safe nor resilient considerably complicates their practical implementation in security-critical applications such as predictive healthcare, which is basically life-critical. As a result, maintaining the integrity and security of deep learning models and health data is critical to the industry’s wider adoption of ML and DL. However, significant research has been conducted recently to resolve this challenge using different cryptography techniques. CryptoNets [9], which showed Privacy-Preserving Machine Learning (PPML) prediction using Homomorphic Encryption (HE) cryptography in 2016 was one of the first publicly publicized benefits. As shown in the 2021 publications [10], [11], the use of multi-party computing techniques demonstrates breakthroughs in the privacy-preserving analysis of large amounts of medical data. While MLaaS is utilized in an insecure cloud environment, hardware-accelerated alternatives such as Trusted Execution Environment (TEE) or Field Programmable Gate Arrays (FPGA) accelerated computations are also used.

II. SECURITY OVERVIEW FOR MACHINE LEARNING

An Artificial Intelligence (AI) system’s ML components include data, models, and methods for training, testing, and validation. In general, ML data-driven approach poses additional security risks throughout the training and inference phases of ML operations. These security concerns include the possibility of malicious manipulation of training data and adversarial exploitation of model sensitivities to degrade ML classification and regression performance. Adversarial Machine Learning (AML) is focused on the development of secure ML algorithms, the analysis of attacker capabilities, and the comprehension of attack repercussions. Malicious adversaries launch attacks and ML security refers to protections designed to avoid or mitigate the results of such attacks. The NIST NISTIR 8269 [12] draft is one of the publications that summarize the vocabulary and security-related concepts associated with AML. It discusses the various kinds of attacks, their defenses, and their associated...
implications in terms of AML. The privacy of ML solutions is violated if an adversary obtains personal information about one or more individual and legitimate model inputs, either included in the training data or not. An example would be when an adversary acquires or extracts an individual’s medical records in violation of privacy policies. One of the defenses against testing (inference) attacks can be based on applying the following security models: Homomorphic Encryption, Secure Multi-Party Computation, use of Trusted Execution Environment.

HE is a type of encryption that enables the computation over encrypted data. Homomorphism is a mathematical notion that refers to preserving structure and correctness across a computation. The most important scheme supports both additions and multiplications of the ciphertext data which are called Fully Homomorphic Encryption - FHE. Their security relies on the hard mathematical problems based on number lattices like Shortest Vector Problem. One widely adopted schemes are BGV, BFV and CKKS.

Secure Multi-Party Computation (SMPC) computing may be extended to several parties with SMPC, in which processing is carried out on encrypted data shares that are shared among the parties so that no single party can recover the complete data on their own. The outcome of the calculation can be published without any party ever seeing the data, which would be retrieved only by agreement. A conceptual illustration of SMPC is a ballot, in which the outcome is required, but the individual voters’ choices are not. The basic protocol for secure two-party computation is Yao’s garbled circuit protocol and its underlying security is based on 1-out-of-2 Oblivious Transfer protocol. Other than that, we can find more modern and complex approaches including three or more parties, other than that, we can find more modern and complex approaches including three or more parties, other possibilities to share the data (arithmetic or Boolean sharing), and problem-specific optimizations.

TEE is a tamper-resistant processing environment that runs on a dedicated, hardened subsystem. It guarantees the authenticity of the executed code, the integrity of the runtime states (e.g., central processing unit registers, memory, and sensitive I/O), and the confidentiality of its code, data and runtime states stored on persistent memory. In addition, it should be able to provide a remote attestation that proves its trustworthiness for third parties. One of the examples of TEE technology is Intel® Software Guard Extensions (Intel® SGX).

In other words, we need to ensure Data Confidentiality in MLaaS Cloud Computational Environment:

- **Input:** Datasets for digits image MNIST and recognition issue using different Neural Network (NN) architectures;
- **Constraints:** Data Confidentiality in the Cloud Computational Environment;
- **Output:** PPML predictions for a dataset;
- **Security approaches:** HE, SMPC, TEE.

As a part of problem definition, we define three research questions as follows:

- How secure environment affect the accuracy of computation?
- What overhead is associated with ensuring security in terms of neural network prediction time?
- What are efficient ways to transform neural networks from the non-secure computation to a secure one?

### Tool selection criteria

In our solution, we mainly focused on two important criteria for tool selection:

- Several solutions will be compared. As a result, the solution should be Open-Source, with the source code accessible to the public.
- The majority of ML solutions and libraries are written in Python. To facilitate library efficient adoption, it is noted whether a particular solution is available as a Python package that includes support for either Tensorflow or PyTorch - the two most popular ML libraries available in Python.

### Tool selection procedure

We compared tools that meet our security approaches requirement, those should be based on HE, SMPC or TEE.

**Homomorphic Encryption Based Tools**

Regarding HE libraries - see Table I, the most promising ones are nGraph-HE2 [14] (an extension of nGraph-HE [15]) and TenSEAL [16].

<table>
<thead>
<tr>
<th>Library</th>
<th>Tensorflow or PyTorch support</th>
<th>Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>nGraph-HE2 [14]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>nGraph-HE [15]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TenSEAL [16]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CryptoNets [9]</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Cingulata [17]</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>TFHE [18]</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>MLwithHE [19]</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>CHET [20]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>CryptoDL [21]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Chimera [22]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Glyph [23]</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

### III. PROBLEM STATEMENT

This paper aims to provide a high-level overview of practical approaches to machine-learning respecting the privacy and confidentiality of customer information, in short - Privacy-Preserving Machine Learning. In order to achieve it, we need to check and verify different security approaches as well as their implementation to meet privacy preserving ML prediction’s goal. As one of the ML tasks, we consider image recognition problem using Modified National Institute of Standards and Technology (MNIST) [13] dataset. Having this ML problem set, we will use different libraries with their security approaches to solve it and compare its accuracy and efficiency.
The $n$Graph-HE2 has the interface to support Tensorflow. TenSEAL is supported via PyTorch. However, using TenSEAL is less complex to use. The other promising solutions, such as CryptoNets [9], Cingulata [17], TFHE [18], MLwithHE [19] were indeed open-source but written in C or C++. Other alternatives like CHET [20], CryptoDL [21], Chimera [22], Glyph [23] that have been developed are not open source and do not support Tensorflow or PyTorch.

TenSEAL [24] is the HE library that makes use of the CKKS technique. It employs relinearization, rescaling, and modulus flipping by default. The polynomial modulus degree is set at 8192 for the $\lambda = 128$ bit security level, with primes scaled to 26 bits. As it’s core implementation it utilizes the Microsoft SEAL library.

Despite the continued study, however, the HE cryptosystems do not provide direct division and maximum operations [25], [26]. As a result, the use of contemporary NN topologies is constrained. For instance, the activation function of the NN do not provide direct division and maximum operations [25], [26]. As it’s core implementation it utilizes the Microsoft SEAL library.

Secure Multi-Party Computation Based Tools

Dalskov [28] is the SMPC system that satisfies the requirements (see Table II).

<table>
<thead>
<tr>
<th>Library</th>
<th>Tensorflow or PyTorch support</th>
<th>Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dalskov [28]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SyMPC [29]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CryptoFlow [30]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CryptoFlow2 [31]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SIRNN [32]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cкрыт [33]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TF-Encrypted [34]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TASTY [35]</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>ABY3 [36]</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>SecureNN [37]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cerebro [38]</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>FALCON [39]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XONN [40]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Chameleon [41]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Sadeghi [42]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Barni [43]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>SecureML [44]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Tetrad [45]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>BLAZEN [46]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>SWIFT [47]</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

However, the solution was designed in a way that does not support installation via Python package. CryptoTF [30], CryptoTF [31], and SIRNN [32] are all components of the broader ExPC [48] library. While some components of the system support Tensorflow, integration with the standard machine learning Python flow is quite challenging due to its complexity.

SyMPC [29], which is included in PySyft [49] can run low-level protocols such as ABY3 [36] or FALCON [39] and supports PyTorch. TF-Encrypted [34] could also be used with the Tensorflow libraries, and it’s working with the SecureNN [37] protocol.

Other libraries did not conform to the specified requirements. However, it is worth mentioning current findings in SMPC subjects such as Tetrad [45], BLAZEN [46], and SWIFT [47].

SyMPC [29] is a solution that uses the AriaNN [50] protocol for semi-honest two-party computing. Due to the library’s ability to utilize the ReLU activation function, multi-layer perceptron NN architectures are feasible. However, convolutional NNs are limited in their use since they only support the MaxPool building component [27]. Additionally, Dropout does not work as a construction block [27]. The SyMPC is distributed under MIT Licence.

TF-Encrypted [51] is a SMPC that is based on the SecureNN [37] - three-party malicious-aware computation protocol. Consequently, even if one of the parties is a malicious actor, the computation may still succeed. The TF-Encrypted is distributed under Apache Licence 2.0.

Trusted Execution Environment Based Tools

Three solutions (see Table III) that match the required criteria are Gramine, [52], TF-Trusted [53] and SLALOM [54].

<table>
<thead>
<tr>
<th>Library</th>
<th>Tensorflow or PyTorch support</th>
<th>Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gramine [52]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TF-Trusted [53]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SLALOM [54]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PPFL [55]</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>DarknetZ [56]</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>BigDL PPMIL [57]</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Eleos [58]</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>TensorSCONE [59]</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>PERUN [60]</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Fleece [61]</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

However, SLALOM [54], like Dalskov [28], the solution was designed in a way that does not support installation via Python package. On the other hand, TF-Trusted [53] is based on an older Intel® Software Guard Extensions SDK for Linux® OS (Intel® SGX SDK for Linux® OS) version which is nearly impossible to run on more recent versions. The other alternatives are either proprietary or do not support Tensorflow or PyTorch.

Gramine [52] solution allows running unmodified Linux binaries on Intel® SGX. The Gramine is distributed under GNU GPLv3.

Comparison Conclusion

HE, SMPC and TEE were addressed throughout the solution search. Only a few of the nearly 40 privacy-preserving technologies examined are open-source and could be effectively
integrated into the Python ML code base. Those are: SyMPC [29], TF-Encrypted [51], TenSEAL [24] and Gramine [52].

IV. SYSTEM ARCHITECTURE

The following modules were created as part of the implementation:

- Training Module - optimizes a given NN architecture in terms of weights for a specific dataset classification or regression issue - see Figure 1.
- Plaintext Module - this module computes the solution to a given issue using a trained NN model and a provided dataset - see Figure 2.
- Encrypted Module - with a given dataset and trained NN, utilizes a PPML library and conducts the encrypted computation - see Figure 3.
- Benchmark Module - this module provides the average time required to compute the prediction for a given computation in either the Plaintext or Encrypted Modules - see Figure 4.

For a given dataset and NN architecture model, the application flow goes as follows:

- Using the Training Module, the NN model is trained for a given training dataset and issue type. The training parameters may be tailored to a particular issue. The module, in particular, enables the user to choose the training criterion and optimizer parameters such as learning rate, number of epochs, and batch size.
- Then the Plaintext Module verifies the model training metrics for a test dataset. Those are specific to the issue type but could be parametrized within the same problem type. For a classification problem, it computes the Accuracy.
- The Encrypted Module uses one of the PPML libraries to perform the encrypted computation with the same parameters as the Plaintext Module. After the evaluation, the same metric as for the Plaintext Module shows to compare specific library quality.
- The Benchmark Module is used in conjunction with the Plaintext or Encrypted Modules. It shows the elapsed time for the one instance of the test dataset to perform its computation. By default, an average of 20 runs is returned in this module.

All application modules are specific to a given library implementation. As stated in previous chapters, the application will use the following libraries: TenSEAL [24], SyMPC [29], TF-Encrypted [51], Gramine [52]. From the implementation point of view, the TenSEAL and SyMPC are PyTorch-based library but TF-Encrypted and Gramine are Tensorflow based. However, from the security point of view, TenSEAL is HE based, SyMPC and TF-Encrypted are SMPC based and Gramine is Intel® SGX based.

TF-Encrypted and Gramine should support even complex NN architectures including modern convolution building blocks such as Conv2D layers, Average and Max Pooling [27].

V. NEURAL NETWORK ARCHITECTURE MODELS

After reviewing the features of each library, let us discuss the NN architectural models that were constructed. All training was carried out with the Adam optimizer, and the learning phase consists of 30 epochs with a learning rate of 0.001 using momentum 0.9 and $10^{-7}$ epsilon. The CrossEntropy criterion was used to address the multi-class classification challenge.

Model A this model was built primarily to showcase the TenSEAL library’s capabilities. It consists of two fully connected layers with 128 neurons each. $f(x) = x^2$ was used to activate the function. All libraries support this network architecture.

Model B is the multi-layer perceptron model too. It comprises two fully linked layers of 128 neurons that activate
using the ReLU function. The final activation function varies according to the kind of problem. The Softmax function was utilized for multiclass classification, and the Sigmoid function was employed for binary classification. The Identity function was used to define the regression type. TenSEAL does not support this type of architecture since it requires using ReLU, Softmax, and Sigmoid functions.

Model C this model was used to validate the convolutional NN’s performance on the image classification task. Its architecture is similar to LeNet [62]: Conv2D 5x5, Max Pooling 2x2, Conv2D 5x5, Max Pooling 2x2, Fully connected layer 120, Fully connected layer 84, Fully connected layer 10, Softmax activation.

Max Pooling was employed to enable the inclusion of the SyMPC library, as it does not support other pooling layers. Additionally, the design may be executed in TF-Encrypted, or Gramine.

VI. COMPUTATION ENVIRONMENT

The computation environment consists of:

- Intel® Xeon® Platinum 8358 2 CPU 2.60GHz processors with 128 threads; 512GB RAM
- Linux OS - Ubuntu 20.04.03 (kernel 5.16.5)
- TenSEAL 0.3.6 with Microsoft SEAL support for Intel® HXEL
- SyMPC commit hash 634396
- TF-Encrypted version 0.5.9
- Gramine version 1.1 with Intel® SGX SDK for Linux* OS 2.11 version

For all libraries, Python 3.7 was used. Because of specific implementation for serialization and deserialization in chosen libraries, each has its instance of NN architecture model.

VII. EXECUTION & TEST PLAN

The test plan consists of two activities to answer questions in our problem statement. For the correctness question, for each library and its trained model, the accuracy will be compared for secure and non-secure flow. Accuracy is a measure of correct decisions normalized by the size of the whole space of decisions. For the performance question, the arithmetic mean of 20 runs for each prediction will be compared for secure and non-secure flow (measured execution time will be in seconds). The computation is based on MNIST dataset. It is a collection of handwritten digits as 28x28 pixels black-and-white image divided into ten classes. It contains 60000 training examples and 10000 test instances. In terms of pre-processing the dataset, all pixels were normalized.

 VIII. SYSTEM EVALUATION

To conclude, the difference between non-secure and secure accuracy for TenSEAL was 0.63%, but the biggest noted was for the SyMPC. For Architecture A it gave: 1.14% difference and for Architecture B: 1.01% and C: 1.1% respectively. TF-Encrypted noted the maximum difference of 0.03%, Gramine gave the same results. The performance results are summarized in Table IV. As the results are shown for the first test, the accuracy of the computation is preserved. The maximum difference between non-secure and secure solutions is no more than 1.2%. However, the differences in execution time. TenSEAL appeared to secure evaluation NN the slowest.

Due to its security model, we cannot design architectures other than Architecture A. In FHE systems we can define only functions based on the polynomials. That is why we mathematically cannot express any other activation function like ReLU or Softmax. It is the most significant limitation of the FHE solutions.

Other libraries are based on SMPC. One of them, SyMPC, still computes Architecture A faster than the FHE one. However, it fails to compute other architectures efficiently. This may be connected to the fact that the library’s implementation uses only two threads for computation (in contrast to other solutions where all 128 threads are used, except for Intel® SGX where 20 threads are used). Other SMPC solutions such as TF-Encrypted were much more effective in their computations. It takes under a second to compute simple architecture networks. However, it is quite a challenge for convolutional network evaluation - for TF-Encrypted it lasts less than 3 seconds. TF-Encrypted is Tensorflow based, and its runtime configuration heavily uses the Tensorflow server’s configuration. The best results showed TEE solutions and Gramine. For simple architectures, it is only 10 times slower compared to non-secure computation, and for the convolutional network - it is 30 times slower. However, when we do not have access to TEE the computation is at least 1000 times slower. On the other hand, as is shown by the results, all libraries that evaluate CNN architectures are significantly slower than MLP (network complexity).

IX. SUMMARY

In overall conclusion, the computational results show that the most effective PPML library is based on TEE. However, when one does not access such platforms, SMPC solutions give a similar accuracy ratio but with reduced performance. The worst performance results were achieved for the HE solution. To emphasize this fact, as demonstrated by the results, the
security costs in terms of evaluation time are very high. Most of the encrypted computations are 1000 times slower than the plaintext evaluation. It is a costly operation, but if we want to achieve an adequate security level, there is no other option. Unfortunately, it is not yet acceptable for a real-world scenario. Multiple future research directions are possible from this point. First, one could try combining multiple security models, for instance, combining operations using SMPC or FHE and offloading more complex ones using TEE. The second one is to have a broader library comparison and include support for libraries written in C/C++. It would be more complex to adapt to real-world scenarios, but it will be closer to the original implementation of underlying protocols, thus achieving better performance. The other future research could be improving SMPC protocols implementation using hardware accelerators. Similarly to what has been achieved with FHE TenSEAL library and Intel HEXL hardware accelerator.

X. NOTICE & DISCLAIMERS

We want to sincerely and deeply thank our colleagues Marcin Kolaśiński and Grzegorz Gerka for their help and support during our computation execution. Technologies may require enabled hardware, software or service activation. No product or component can be absolutely secure. Your costs and results may vary. ©Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

REFERENCES


