Developing Privacy-preserving AI Systems: The Lessons learned

Huili Chen
Intel Labs
huili.chen@intel.com

Siam Umar Hussain
UC San Diego
siamumar@ucsd.edu

Fabian Boemer
Intel Corporation
fabian.boemer@intel.com

Emmanuel Stapf
TU Darmstadt
emmanuel.stapf@trust.tu-darmstadt.de

Ahmad Reza Sadeghi
TU Darmstadt
ahmad.sadeghi@trust.tu-darmstadt.de

Farinaz Koushanfar
UC San Diego
farinaz@ucsd.edu

Rosario Cammarota
Intel Labs
rosario.cammarota@intel.com

***Abstract*—Advances in customers' data privacy laws create pressures and pain points across the entire lifecycle of AI products. Working figures such as data scientists and data engineers need to account for the correct use of privacy-enhancing technologies such as homomorphic encryption, secure multi-party computation, and trusted execution environment when they develop, test and deploy products embedding AI models while providing data protection guarantees. In this work, we share the lessons learned during the development of frameworks to aid data scientists and data engineers to map their optimized workloads onto privacy-enhancing technologies seamlessly and correctly.**

I. Introduction

Artificial Intelligence (AI), in particular Machine Learning (ML) algorithms, has advanced various real-world applications due to its unprecedented capability to perform critical tasks such as speech/video recognition, natural language processing, and medical diagnosis, with superior performance compared with both human beings and expert systems.

Given such superior performance, an increasing number of technology companies are providing Machine Learning as a Service (MLaaS) to their customers. In the MLaaS paradigm, the end-customer, e.g., an institution or a private individual, sends his private data to the ML remote service and receives the final output. However, this workflow raises crucial concerns about customers' privacy, since: (i) often times the input data is either personal, particularly in healthcare and medical domain, or carries business-sensitive information, particularly in the financial space; (ii) the customer of a MLaaS has no control over the data handed off to the server. As such, ML service providers are typically assumed to be a trusted authority, constraining the applicability of MLaaS; (iii) the input data itself and its metadata can be used to enrich the model or for other purposes beyond customer's intended use.

Privacy-Enhancing Machine Learning - often referred to as Privacy-Preserving Machine Learning (PPML) - is an emerging field that targets to resolve tensions with data privacy in concurrent MLaaS applications by deploying privacy-enhancing technologies embedded in such services. In a simplified view of the problem, there are two parties involved in a PPML transaction: the customer, e.g., a user, who owns the input data, and the service provider who owns the MLaaS application. The customer holds confidential data and wants to obtain the result of his desired machine learning application deployed by the service provider. The service provider has sufficient computing power to perform the machine learning task (training/inference). The dataflow of a typical PPML application can be divided into three stages: (i) Data pre-processing at the data input owner's side - the processed data is then securely transmitted to the service provider; (ii) the service provider executes the target data application - the resulting data is sent back to the data owner; and (iii) the data owner (i.e., end user) post-processes the result received from the service provider.

Note that the objective of PPML is to provide data-level protection for the customers, either users or third party institutions. Other lines of research focus on protecting the ownership of the pre-trained models for the ML service providers [1], [2]. This paper is centered on summarizing the key points for developing secure, effective, and efficient PPML systems.

II. PPML methods in AI

Various techniques have been developed and optimized to tackle the challenge in MLaaS data sharing. We refer to such techniques as PPML building blocks. We categorize the existing PPML building blocks as follows.

Multi-party Computation (MPC) is a cryptographic primitive that allows the evaluation of an arbitrary function on private inputs from multiple parties. MPC protocols typically require the designer to represent the function as a Boolean circuit. To reduce the computational time of MPC protocols, [3] proposes MPCircuits that generate optimized Boolean circuits for MPC realization using customized libraries and hardware synthesis tools. Garbled Circuit (GC) is a subcategory of MPC that ensures secure two-party computation [4]. MAXelerator [5] is a GC-based hardware accelerator for privacy-preserving Multiply-Accumulate (MAC) or its repetition.

Zero Knowledge Proof (ZKP) is a cryptographic technique where a *prover* convinces a *verifier* whether a certain mathematical statement is true or false without revealing any underlying data. The nature of ZKP determines that it is suitable for the usage in PPML. Drynx [6] adopts ZKP to

978-1-7281-1085-1/20/$31.00 ©2020 IEEE

ensure computation integrity and validate that the encrypted data are within the given range.

Homomorphic Encryption (HE) allows computation to be directly performed on encrypted data (ciphertext) without decrypting it [7]. HE schemes support two types of computation: HE addition, and HE multiplication. CryptoNet [8] is the first attempt of deploying the HE primitive for PPML. It approximates the non-linear activation functions in the neural network with square polynomial functions. The SEAL library [9] by Microsoft is used for HE computation.

Differential Privacy (DP) is a rigorous mathematical definition of privacy in the context of statistical and ML analysis. By definition, an algorithm or a process is called deferentially private if an observer cannot tell whether or not an individual's data is included in the original dataset by inspecting the output. In the context of PPML, DP typically works by injecting noise to the training database [10].

Federated Learning (FL) is a variant of decentralized learning where local clients have their own proprietary data. The ML service provider creates a global model and broadcasts it to the selected clients. The local models are then updated via back-propagation using the local dataset. The global model is updated by aggregating the local updates. Since the client's data never leaves his local platform, FL ensures the privacy of the local participants.

Trusted Execution Environment (TEE) architectures are hardware-assisted security architectures that guarantee a level of protection for sensitive services beyond that of commodity operating systems (OS). In particular, TEE architectures provide strongly isolated execution contexts, also called *TEEs* or *enclaves*, protected from higher privileged code, including the OS and even the hypervisor. TEEs are built around hardware security primitives which are configured by microcode or a small trusted code base. Besides runtime isolation, TEEs also offer security functionalities such as local or remote attestation, secure data provisioning or secure storage to their enclaves. Thus, TEE can be utilized to build PPML solutions by isolation the ML algorithms and privacy-sensitive user data inside of an enclave.

III. Supporting Infrastructure for PPML

The effectiveness and efficiency of PPML techniques rely on both software algorithm and hardware architecture. We give an overview of each component and also discuss the co-design of PPML infrastructures in this section.

A. Software Infrastructure

GC Optimizations. Various optimization techniques have been proposed to improve the efficiency of GC, including Free-XOR [11], Row Reduction [12], Half Gate [13], and Garbling With a Fixed-key Block Cipher [14]. These techniques are effective since they alleviate the main bottleneck of GC: generating the optimal Boolean circuit. Take Free-XOR as an example, it allows for the evaluation of XOR, XNOR, and NOT gates without any communication or encryption.

HE Optimizations. As for HE, integer encoding schemes [15] are initially adopted [8]. Later on, the CKKS scheme [16] and Chinese Remainder Theory (CRT) [17] are developed to support computation on fixed-point, large numbers. Ciphertext packing is a technique that encrypts a vector of plaintext in a single ciphertext. As such, homomorphic operations can be applied to the vectors component-wise in a Single Instruction Multiple Data (SIMD) manner.

DP Optimizations. In the context of differential privacy, the Laplacian mechanism adds a random variable to the query answer to limit the information leakage. The exponential mechanism randomly selects the query answer from a set of possible answers based on the parameterized distribution. Note that traditional techniques that deal with missing data can be used in conjunction with DP mechanisms to deal with scenarios where a clean dataset is not available.

FL Optimizations. Aggregation of multiple local clients can be performed via element-wise average or element-wise median. It has been shown that the adversary can reconstruct training samples from the model parameters [18]. Secure aggregation is a class of secure multi-party computation which ensures that only the target collaborative computation result can be learned by the participants. As such, secure aggregation can be deployed in FL to guarantee that the client's private data will not be leaked via his model gradients update [19].

B. Hardware Support

MPC Acceleration. Various accelerators have been proposed to speedup MPC protocols. MAXelerator [5] is a GC computing framework specifically optimized for Multiply-Accumulate operation, which is the basic block of ML workloads. MAXelerator leverages two optimization techniques to improve the efficiency of GC: (i) sequential GC where the same netlist is garbled for multiple rounds with updated encryption keys; (ii) static function analysis to decide the most optimized netlist to garble in every round. While MAXelerator achieves high throughput for MAC, it has limited application scenarios. FASE [20] is a generic GC accelerator that is proposed to address the shortcoming of MAXelerator.

HE Acceleration. HE primitives are computation-bounded since they typically involve complex lattice operations. Besides the advancement and optimizations of HE schemes mentioned in Section III-A, researchers also devote effort to designing hardware architectures for accelerating HE computation. HEAX [21] presents a novel architecture for number-theoretic transform (NTT) which is frequently used in HE schemes. Furthermore, they leverage ciphertext-level as well as fine-grained modular-level arithmetic optimization to introduce the first architecture design for the CKKS scheme.

TEE-based PPML. Besides cryptographic primitives, TEE architectures can also be used to construct PPML systems. Most of the research on TEE-based PPML systems was conducted making use of Intel SGX, whereas most of the authors target a MLaaS scenario. In Chiron [22], ML model training is performed in a Ryoan sandbox, which is based on SGX, that offers the MLaaS provider the possibility to freely select, configure and train the ML models. In MlCapsule [23], the ML computations are performed in an SGX enclave on

the ML customer site, thereby supporting an offline MLaaS scenario. Other works [24], [25] combine SGX with privacy-preserving ML algorithms in order to protect the privacy-sensitive user data from controlled side-channel attacks to which SGX enclaves were shown to be vulnerable [26]. OMG [27] focuses on an offline MLaaS scenario on mobile devices using Sanctuary. In contrast to the aforementioned approaches, OMG provides a mechanism to securely collect privacy-sensitive user data, e.g., biometric data, from sensors on the device and to transmit it to the enclave.

All aforementioned approaches protect the ML algorithms in enclaves on the main CPU. However, offloading ML tasks to hardware accelerators, e.g., GPUs or Neural Processing Units (NPU) yields better performance. Graviton [28] offloads ML tasks from SGX enclaves to the GPU at the cost of hardware changes. Recent work from Intel [29] demonstrates how SGX enclaves can move ML computations from the CPU to custom FPGA accelerators when introducing novel AES-GCM crypto engines at the accelerator side.

C. Software/Hardware Co-design

We discuss the software-level and hardware-level support for PPML solutions in Section III-A and III-B, respectively. It is worth noticing that the design of these two components shall be put in the same loop instead of tackled separately to achieve the optimal performance.

MPC. Here, we use MPCircuits [3] as an example to demonstrate how software/algorithm-level optimizations can be taken into account during hardware design. The key idea of MPCircuits is developing customized synthesis libraries to realize MPC protocols. The target function is first described with Hardware Description Language (HDL). The optimization objective of MPCircuits is to minimize the number of non-XOR gates $n_{non-XOR}$ in the circuit. As such, [3] defines the problem of minimizing $n_{non-XOR}$ as a special case of logic optimization that can be solved using existing synthesis tools.

HE. Take HE-based PPML solutions as an example, compiler frameworks have been developed to enable efficient HE computation. CHET [30] is a domain-specific optimizing compiler that intends to facilitate the development of HE implementations. To be more detailed, CHET proposes an ISA extension for homomorphic evaluation (HISA) and tensor description to abstract the data flow of the FHE scheme used in the PPML system. Such abstraction is further leveraged by the compiler to optimize the HE computation. Encrypted Vector Arithmetic (EVA) [31] is a new FHE language that aims to hide the complexity of HE from program developers. Compared to CHET, EVA is designed to work as an intermediate representation (IR) that can be used as a target when compiling other high-level domain-specific languages such as CHET. As a result, EVA reduces the runtime overhead of the original CHET implementation by $5.3\times$.

FL. In the context of FL, [32] presents a system-level design for FL deployment on mobile device platforms. In particular, various practical problems are considered, including device availability that affects the local data distribution, unreliable communication between devices and the server, limited on-device storage, orchestration of lock-step execution across devices. These challenges are tackled via communication protocol, device management, and server levels.

IV. Qualitative assessment of PPML frameworks

We present a quantitative comparison between existing PPML solutions in the following sections.

GC. GC has been explored for constructing PPML systems due to its provable security property. MiniONN [33] is a mixed oblivious inference protocol using HE, GC, and Secret Sharing (SS). Chameleon [34] is also a mixed-protocol framework that deploys GMW and GC for non-linear functions, and additive SS for linear functions. Gazelle [35] is a hybrid approach that leverages additive HE for linear operations in a DNN and GC for non-linear operations. Compared to the above techniques, XONN [36] achieves the minimal inference latency while preserving comparable task accuracy as shown in Figure 1. The reduction of XONN's runtime overhead is derived from the efficiency of binarizing the DNN and transforming matrix multiplication operations into XNORs that are free in GC.

Framework	Runtime (s)	Comm. (MB)	Acc. (%)
MiniONN	544	9272	81.61
Chameleon	52.67	2650	81.61
EzPC	265.6	40683	81.61
Gazelle	15.48	1236	81.61
XONN	5.79	2599	81.85

Fig. 1: Comparison of existing PPML methods with GC primitive component on CIFAR-10 benchmark.

HE. CyrptoNet [8] is the first attempt of adopting HE for PPML inference. Using polynomial approximation and YASHE with SIMD packing, CryptoNet achieves a throughput of 58982 per hour on MNIST benchmark. While demonstrating the feasibility of HE-based PPML, CryptoNets relies on SEAL library [9] that is written in C++. As a result, adapting CryptoNets to different DL benchmarks incurs large engineering cost. nGraph-HE [37], [38] presents a backend for Intel's nGraph compiler stack [39] that enables HE-aware optimizations that are compatible with TensorFlow. Building on top of Intel's DL compiler PlaidML [40], the work [41] PlaidML-HE proposes to develop a customized operator library for HE-specific computation to realize PPML. The compiler-based approach of PlaidML-HE not only automate the generation of kernel libraries for HE operations on the target hardware, but also enables various optimization techniques provided by PlaidML compiler.

V. The lessons Learned

We provide an overview of existing PPML solutions in Section II and categorize them based on their underlying privacy-preserving primitives. One can see that different techniques have their own pros and cons in terms of security, effectiveness, efficiency, and scalability. We believe that a *hybrid PPML scheme* can enjoy the benefits of each component, thus provisioning the optimal trade-off between ML task performance and privacy-preserving overhead.

Meanwhile, we summarize the infrastructure support for PPML techniques in Section III from both software-level and hardware-level. As corroborated by the quantitative results in Section IV, we emphasize that *software/hardware co-design* principle plays an important role in developing a fully optimized PPML system.

REFERENCES

[1] B. Darvish Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 485–497.

[2] H. Chen, C. Fu, B. D. Rouhani, J. Zhao, and F. Koushanfar, "Deepattest: an end-to-end attestation framework for deep neural networks," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 487–498.

[3] M. S. Riazi, M. Javaheripi, S. U. Hussain, and F. Koushanfar, "Mpcircuits: Optimized circuit generation for secure multi-party computation." *IACR Cryptology ePrint Archive*, vol. 2019, p. 275, 2019.

[4] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.

[5] S. U. Hussain, B. D. Rouhani, M. Ghasemzadeh, and F. Koushanfar, "Maxelerator: Fpga accelerator for privacy preserving multiply-accumulate (mac) on cloud servers," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[6] D. Froelicher, J. R. Troncoso-Pastoriza, J. S. Sousa, and J.-P. Hubaux, "Drynx: Decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets," *arXiv preprint arXiv:1902.03785*, 2019.

[7] C. Gentry *et al.*, "Fully homomorphic encryption using ideal lattices." in *Stoc*, vol. 9, no. 2009, 2009, pp. 169–178.

[8] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.

[9] Microsoft, "Microsoft seal library." https://github.com/microsoft/SEAL, 2019.

[10] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1895–1912.

[11] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free xor gates and applications," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2008, pp. 486–498.

[12] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *Proceedings of the 1st ACM conference on Electronic commerce*, 1999, pp. 129–139.

[13] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 220–250.

[14] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 478–492.

[15] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *IMA International Conference on Cryptography and Coding*. Springer, 2013, pp. 45–64.

[16] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.

[17] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full rns variant of approximate homomorphic encryption," in *International Conference on Selected Areas in Cryptography*. Springer, 2018, pp. 347–368.

[18] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[19] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[20] S. U. Hussain and F. Koushanfar, "Fase: Fpga acceleration of secure function evaluation," in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2019, pp. 280–288.

[21] M. S. Riazi, K. Laine, B. Pelton, and W. Dai, "Heax: High-performance architecture for computation on homomorphically encrypted data in the cloud," *arXiv preprint arXiv:1909.09731*, 2019.

[22] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 123–142, 2018.

[23] 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," *arXiv preprint arXiv:1808.00590*, 2018.

[24] N. Hynes, R. Cheng, and D. Song, "Efficient deep learning on multi-source private data," *arXiv preprint arXiv:1807.06689*, 2018.

[25] S. Tople, K. Grover, S. Shinde, R. Bhagwan, and R. Ramjee, "Privado: Practical and secure dnn inference," *arXiv preprint arXiv:1810.00602*, 2018.

[26] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 640–656.

[27] S. P. Bayerl, T. Frassetto, P. Jauernig, K. Riedhammer, A.-R. Sadeghi, T. Schneider, E. Stapf, and C. Weinert, "Offline model guard: Secure and private ml on mobile devices," *DATE 2020*, 2020.

[28] S. Volos, K. Vaswani, and R. Bruno, "Graviton: Trusted execution environments on gpus," in *13th USENIXSymposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 681–696.

[29] S. Ghosh, L. S. Kida, S. J. Desai, and R. Lal, "A¿ 100 gbps inline aes-gcm hardware engine and protected dma transfers between sgx enclave and fpga accelerator device."

[30] R. Dathathri, O. Saarikivi, H. Chen, K. Laine, K. Lauter, S. Maleki, M. Musuvathi, and T. Mytkowicz, "Chet: an optimizing compiler for fully-homomorphic neural-network inferencing," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 2019, pp. 142–156.

[31] R. Dathathri, B. Kostova, O. Saarikivi, W. Dai, K. Laine, and M. Musuvathi, "Eva: An encrypted vector arithmetic language and compiler for efficient homomorphic computation," *arXiv preprint arXiv:1912.11951*, 2019.

[32] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[33] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 619–631.

[34] 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 *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 707–721.

[35] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "{GAZELLE}: A low latency framework for secure neural network inference," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1651–1669.

[36] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "{XONN}: Xnor-based oblivious deep neural network inference," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1501–1518.

[37] F. Boemer, Y. Lao, and C. Wierzynski, "ngraph-he: A graph compiler for deep learning on homomorphically encrypted data," *arXiv preprint arXiv:1810.10121*, 2018.

[38] F. Boemer, A. Costache, R. Cammarota, and C. Wierzynski, "ngraph-he2: A high-throughput framework for neural network inference on encrypted data," *arXiv preprint arXiv:1908.04172*, 2019.

[39] I. AI, "ngraph - open source c++ library, compiler and runtime for deep learning ." https://github.com/NervanaSystems/ngraph, 2019.

[40] PlaidML, "Plaidml: A platform for making deep learning work everywhere." https://github.com/plaidml/plaidml, 2019.

[41] H. Chen, R. Cammarota, F. Valencia, and F. Regazzoni, "Plaidml-he: Acceleration of deep learning kernels to compute on encrypted data," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*. IEEE, 2019, pp. 333–336.