SHAPER: A General Architecture for Privacy-Preserving Primitives in Secure Machine Learning

***Abstract*—Secure multi-party computation and homomorphic encryption are two primary security primitives in the privacy-preserving machine learning, whose wide adoption is, nevertheless, constrained by the computation and network communication overheads. This paper proposes a hybrid Secret-sharing and Homomorphic encryption Architecture for Privacy-pERsevering machine learning (*SHAPER*). SHAPER protects sensitive data in the encrypted or randomly shared domains instead of relying on a trusted third party. The proposed algorithm-protocol-hardware co-design methodology explores techniques such as plaintext SIMD and fine-grained scheduling, to minimize end-to-end latency in various network settings. SHAPER also supports secure domain computing acceleration and the conversion between mainstream privacy-preserving primitives, making it ready for general and distinctive data characteristics. SHAPER is evaluated by FPGA prototyping with a comprehensive hyper-parameter exploration, demonstrating 129× speed-up over CPU clusters on large-scale logistic regression training tasks.**

***Index Terms*—Privacy-Preserving Machine Learning, Multi-Party Computation, Additive Homomorphic Encryption, Hardware Accelerator**

I. Introduction

Cross-agency data collaboration maximizes the accuracy of Machine learning (ML) models. Nonetheless, from the perspective of user privacy and business interests, concerns about data privacy and security arise [1]. Privacy-preserving machine learning (PPML) [25] allows participants to co-operate on training and inference procedures by applying privacy-preserving computing techniques, *e.g.* multi-party computation (MPC) [26], homomorphic encryption (HE) [9], and trusted execution environment (TEE) [6]. These security primitives prevent the raw data, model weights, and gradient values from revealing to any other participants. Since the algorithms and protocols of PPML heavily depend on the data characteristics, scale, ownership, and security model, debates on technical roadmap never stop.

MPC covers a series of privacy-preserving techniques that support secure computation protocols on mathematically masked data. Garble circuit (GC) is a secure two-party logical computing protocol, where evaluating an *AND* gate requires transferring a ciphertext look-up table. Secret sharing (SS) guarantees information-theoretic security by randomly sharing the raw data. However, arithmetic on the SS domain relies on intensive in-order data interaction. Even though MPC is versatile to different PPML scenarios, network overhead always hinders further development of MPC-based PPML with complex models in real-time applications.

HE-based schemes support several operators on encrypted data. Fully HE (FHE) schemes can ideally support any multiplication level by refreshing its noise budget with bootstrapping. Nevertheless, ciphertext evaluation and bootstrapping always require complex modular operations, which introduce tremendous computation overhead. Existing academic FHE accelerators are still costly and only feasible on small-scale training and inference scenes [23]. On the other hand, additive HE (AHE) provides partial linear operators except for ciphertext-to-ciphertext multiplication with affordable overhead. However, purely AHE-based two-party schemes need a trusted party to generate and manage the secret key [13].

Recent works show that hybrid SS-AHE solutions achieve 130× speedup with practical dataset and network bandwidth [5], [16]. The key insight is to keep the characteristics of the training set, *e.g.* sparsity, from being masked in the SS domain or encrypted in the AHE domain. This is achieved by always holding samples as plaintext within their owner and only transferring small-size HE domain intermediate values. Participants can evaluate the layer functions with sparse operations and share the result in the SS domain other than revealing any sensitive values to one participant or a third party.

When the complex HE algorithm combines SS protocol in the end-to-end PPML solutions, new architecture considerations and methodologies are necessary. We observe that the computation and communication complicity, which are the bottlenecks of the two primitives, can complement each other. The standalone SS and HE approaches present a highly polarized communication-computation ratio, for which latency hiding between the data transfer and execution units gets little return. We optimize the hybrid approach following the intuition that, in an ideal case, a well-balanced and parallel communication-computation flow can fold latency by 50%. This observation can also make the architecture less sensitive to the network bandwidth, which typically dominates MPC performance. At the algorithm level, it is helpful to tune hyper-parameters, such as an overflow-free pack level, to relieve ciphertext explosion. Considering practical computing power settings are also vital for PPML, a ready-to-use architecture should be suitable for FPGA platforms while flexible to ASIC design as well.

This paper provides a general architecture that can efficiently execute the SS-AHE hybrid PPML protocols on the industrial-level large training dataset. The proposed design preserves privacy in either the SS domain or AHE domain, without any reliance on trusted hardware manufacturers. In summary, this paper makes the following contributions:

- A hybrid Secret-sharing and Homomorphic encryption Architecture for Privacy-pERsevering ML (*SHAPER*) with algorithm-protocol-hardware co-optimization.
- A hardware implementation supporting linear and approximated non-linear tasks with the help of efficient conversion protocols between primary security primitives.
- Vectorized high-performance modular multiplication (MM) engines to improve the efficiency of encryption, decryption, and ciphertext domain evaluation.
- SHAPER shows universal performance improvement on micro-operations and reduces the end-to-end latency by 129× on large-scale logistic regression training tasks.

II. Background and Related Work

Various PPML schemes have been proposed to ensure the security of data and models with different cryptographic primitives. Actually, MPC-based PPML schemes [14], [15], [27] divide ML models into fragments of circuits, and then engage multiple parties to cooperatively execute circuit computations, including arithmetic and binary circuits, without additional privacy leaking. Afterwards, the

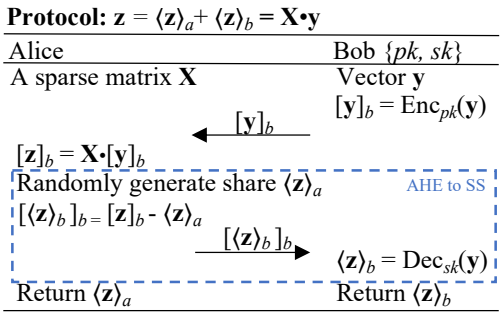

Fig. 1. SS-AHE-based secure matrix-vector multiplication.

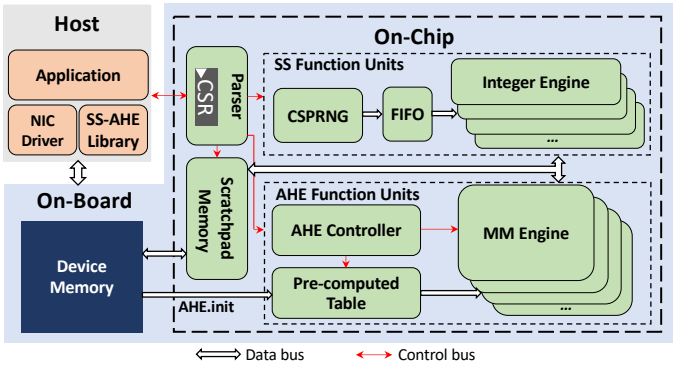

Fig. 2. SHAPER Architecture Overview – The orange and red rounded boxes represent software and hardware components, respectively.

results of these fragments of circuits are gathered from the parties to construct the full result of complex computation tasks. Historically speaking, MPC was proposed in [26], which solved the "Millionaire Problem" with GC. Not long after that, SS-based MPC [12] was proposed for better performance. Compared to GC, SS requires smaller computation and communication overheads, and outperforms GC in most scenarios. In the perspective of computation, addition and multiplication are two major operations in SS-based MPC. Its additions can be easily handled by both arithmetic and binary SS with little computation and no communication overhead. However, due to the data exchange required by each MPC multiplication, the communication overhead dominates the overall performance for SS-based MPC and grows significantly for large-scale datasets.

HE-based PPML [11], [17] provides the capability to execute operations on encrypted data to protect privacy. Compared to public-key cryptography, apart from key generation, encryption, and decryption, HE additionally supports addition/multiplication over ciphertexts without private keys, and therefore no information about the corresponding plaintexts is revealed. Therefore, HE-based PPML requires less communication but more computation than MPC-based PPML for expensive HE encryption/decryption.

Fig. 1 describes a typical linear function in PPML, the sparse matrix is kept by its owner, Alice, and the result is shared between the participants Alice and Bob. Since AHE protects the confidentiality of vector $\mathbf{y}$, Alice can not recover the plaintext from the ciphertext $[\mathbf{y}]_b$. On the other hand, Alice shares $[\mathbf{z}]_b$ in line 5, which guarantees Bob can only learn a masked result $\langle \mathbf{z} \rangle_b$. Recent works on hybrid SS-AHE PPML [5] framework achieve 130× speedup compared with MPC-based schemes. AHE supports additions between ciphertexts and multiplications between ciphertexts and plaintexts. However, most existing AHE algorithms, such as Paillier [21], DGK [7], OU [20], depend on large integer modular operations, especially modular multiplications and exponentiations, which cost heavy computational overheads. Therefore, the overall performance of SS-AHE hybrid PPML is greatly dominated by the efficiency of the basic modular multiplications. Montgomery modular multiplication [19] is the most classic method while new modular algorithms have also been proposed recently [16].

III. Architecture Design

To accelerate the hybrid SS-AHE framework, SHAPER proposes an instruction set and explores efficient design methodologies of AHE, SS, and conversion functions.

A. Architecture overview

The overview of our proposed SHAPER architecture is shown in Fig. 2, which includes both software and hardware implementations. The host application controls the start and convergence condition of training tasks, it also consults hyper-parameters between participants. The application calls the SS-AHE library, which supports execution flows encapsulated as kernel functions. The kernel functions update algorithm parameters and architecture flags by setting control and status registers (CSRs), and implement the security primitives with customized instructions summarized in Table I. SHAPER analyzes the dependency of control flow and packs the instructions as VLIW-style, ensuring the packed ones in a VLIW instruction can execute in parallel. Since the instructions execute in order and deterministically, memory allocation is scheduled in a static manner. To communicate with other participants, all network interaction is handled by a network interface card (NIC). The host application always waits for interruption from the NIC and SHAPER. Since runtime and driver layers are common components in the HW/SW co-design, we omit them in Table I for concise.

On the SHAPER hardware, the parser unpacks the instructions and dispatches them to the corresponding function units. **AHE.init** reloads data from the device memory in the offline phase when the host updates its key pair. Other AHE instructions consist of a series of MM operations, handled by the AHE controller. **SS.gen** returns a vector of random shares sampled from the cryptographic-secure pseudorandom number generator (CSPRNG). **Int.add** and **Int.mul** execute a batch of integer arithmetic in a continuous address space. The memory hierarchy consists of on-board device memory and on-chip scratchpad, which is managed with **DM.ld/st** and **SPM.ld/st**.

B. Algorithm-Protocol Co-Optimization

Fig. 3 describes the methodology to analyze and explore the PPML solutions. We map a task to the coordinate point according to the computation and communication overhead. The network bandwidth is represented as a dotted guideline, points on which have the same communication and computation latency. The solutions above the guideline (*e.g. SecureML* [18]) are communication dominated. On the other hand, the communication-less FHE solutions (*e.g. CraterLake* [23]) cost most of the time on ciphertext evaluation. The location of SS-AHE-based solutions depends on computation power, especially the performance of cryptographic engines. In our work, the following optimizations are applied to explore an optimal solution.

1) Data Characteristic: In real-world scenes, the training dataset is sparse due to incomplete user information and one-hot encoding [5]. Since SS-AHE schemes keep the data sparsity, the number of instructions significantly reduces.