A Review of High-Latency Optimization in Privacy-Preserving Computation: Hardware-Algorithm Co-Design, Hierarchical Protection, and Scenario-Aware Deployment

He Wang
School of Computer Science, Engineering and Artificial Intelligence, Wuhan Institute of Technology, Wuhan, 430205, Hubei, China
15731495796@163.com

ABSTRACT

With the continuous expansion of data circulation, federated learning, and privacy-preserving machine learning, cross-institutional and cross-device collaborative data analysis imposes increasingly stringent requirements on security and real-time performance. Existing studies show that the main bottleneck of privacy-preserving computation is not merely the slow execution of a single cryptographic operator, but the combined effect of ciphertext computation complexity, protocol interaction costs, and system-level data movement overhead. Accordingly, this paper reviews high-latency optimization for privacy-preserving computation along the thread of latency sources, optimization paths, and scenario adaptation. It compares representative methods in terms of applicable objects, performance gains, implementation costs, and deployment boundaries, and summarizes the problem as three structural contradictions: hardware rigidity, algorithmic inefficiency, and static protection strategies. On this basis, the paper surveys heterogeneous hardware acceleration, model redesign and compression, selective and hierarchical protection, and cloud-edge-terminal collaborative deployment. The literature indicates that graphics processing units (GPUs), field-programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs) provide important acceleration support for critical paths such as number theoretic transform (NTT), key switching, and bootstrapping; quantization, pruning, ciphertext packing, and gradient compression can reduce redundant computation and communication; and selective encryption, hybrid privacy mechanisms, and dynamic privacy-budget allocation help concentrate limited resources on highly sensitive objects. Overall, future optimization of privacy-preserving computation should not remain at isolated attempts to make a single operator faster. Instead, it should move toward cross-layer collaborative frameworks that account for scenarios, hardware platforms, and privacy mechanisms, thereby promoting privacy-preserving computation from theoretical feasibility toward engineering usability.

Keywords: Privacy-Preserving Computation; Homomorphic Encryption; Federated Learning; Hardware-Algorithm Co-Design; Hierarchical Protection.

1. INTRODUCTION

As the demand for data sharing, collaborative modeling, and intelligent decision-making continues to grow, scenarios such as financial risk control, smart healthcare, industrial Internet, and public-sector governance increasingly rely on cross-entity collaborative data analysis. However, raw data are vulnerable to privacy leakage, model inversion, membership inference, and parameter theft during centralized storage, cross-domain transmission, and federated training. For this reason, “data usability without data visibility” has become an important objective of privacy-preserving computation and privacy-enhancing technologies [1-2]. The key issue is that improved security is often accompanied by substantial time and system costs. This is particularly evident in homomorphic-encryption-driven privacy-preserving machine learning, where ciphertext multiplication, modulus switching, key switching, and bootstrapping significantly amplify computational and memory-access pressure, directly affecting deployability [3-6].

To address these bottlenecks, existing studies mainly reduce the latency of privacy-preserving computation from four directions: hardware co-design [4-8], algorithmic compression [3,6,9,17], hierarchical protection [10,14,18-19], and scenario-aware deployment [11-13,15-16]. These paths mitigate critical-operator execution, redundant ciphertext computation, uniform protection overhead, and resource-constrained deployment, respectively, but their gains and costs depend heavily on specific tasks, hardware platforms, and security boundaries.

Nevertheless, existing research still has two limitations. First, many studies focus on a single operator, platform, or scenario and lack a cross-layer analytical perspective. Second, although some results report performance improvements, they seldom explain the source of the gain, who bears the cost, and where the method is applicable, making it difficult to directly guide engineering selection [4,6,10,12,14].

Based on this observation, this paper does not treat "high-latency optimization" as a single technical solution, but as a typical cross-layer system problem. It reviews the literature around four questions: What are the major sources of high latency in privacy-preserving computation? Which bottlenecks are mitigated by representative methods? What gains and costs do these methods introduce? What are their applicability boundaries in different scenarios? The paper first defines the research scope and analytical framework, then discusses hardware co-design, algorithmic optimization, hierarchical and adaptive protection, and cloud-edge-terminal collaborative deployment, and finally summarizes the main findings and research issues that deserve further attention.

2. SOURCES OF HIGH LATENCY AND ANALYTICAL FRAMEWORK FOR PRIVACY-PRESERVING COMPUTATION

2.1 Technical Spectrum and Scope of This Review

Privacy-preserving computation is not a single technology. It is a technical spectrum composed of homomorphic encryption (HE), especially fully homomorphic encryption (FHE) [3,14,21], secure multi-party computation (SMPC) [16,20], federated learning (FL) [12,14], differential privacy (DP) [14,18], trusted execution environments (TEE) [19], and other mechanisms [1]. These technologies differ significantly in trust assumptions, interaction rounds, supported operations, computational complexity, and deployment conditions. HE emphasizes strong protection by keeping data encrypted throughout computation, but it usually incurs high computation and memory-access costs. SMPC often trades multiple rounds of interaction for security and is therefore more sensitive to link quality and bandwidth. FL reduces the need to centralize raw data, but it does not naturally eliminate gradient leakage or model inversion risks [1,3,16]. Without distinguishing these sources of performance cost and protection boundaries, discussions of why privacy-preserving computation is slow can easily lose specificity.

This paper takes homomorphic-encryption-driven privacy-preserving machine learning [3-4,6,21] and federated learning [12,14,18] as the main thread, while treating SMPC [16,20], DP [14,18], and TEE [19] as complementary mechanisms. The focus is on how these mechanisms jointly affect end-to-end latency and scenario adaptation.

2.2 Major Sources of High Latency: Computation, Communication, and Data Movement

The high latency of privacy-preserving computation first comes from the intrinsic complexity of ciphertext computation. Surveys on homomorphic encryption and its applications generally point out that ciphertext expansion, large-modulus arithmetic, and complex operations over polynomial rings make HE far more expensive than plaintext computation [3,6,21]. In FHE scenarios in particular, ciphertext multiplication, modulus switching, key switching, and bootstrapping often determine the latency upper bound of the entire execution path [4-6,22]. Therefore, bootstrapping and key management should not be viewed as a single slow operator, but as a system bottleneck formed by multiple high-cost subprocesses.

The second source of latency is communication and interaction. Whether the system relies on SMPC or federated learning, once multiple rounds of parameter exchange, secure aggregation, or protocol coordination are required, link quality and the number of participating nodes will significantly affect end-to-end latency [11,20]. Existing studies show that reducing communication rounds, compressing update objects, or adjusting the aggregation location can be as important as reducing the cost of a single encryption operation [11-12,23].

The third source of latency is system-level data movement and the storage hierarchy. Research on FHE acceleration indicates that many systems are not simply slow in computation, but slow in movement: large ciphertexts and keys cause frequent off-chip memory access, bus-bandwidth pressure, and limited cache reuse [4-5]. In Internet of Things (IoT) and cloud-edge-terminal collaboration scenarios, this problem is further amplified because ciphertexts must not only move across storage levels but also be transmitted across terminals, networks, and cloud platforms [24-25]. Thus, high latency in privacy-preserving computation should be understood as the joint result of computational complexity, protocol interaction, and data flow rather than as a local defect at a single layer.

2.3 Three Structural Contradictions and Comparison Dimensions

To organize the above phenomena into a comparable analytical framework, this paper summarizes high latency in privacy-preserving computation as three structural contradictions. The first is hardware rigidity: ciphertext computation imposes high requirements on parallelism, bandwidth, and memory-access organization, whereas many acceleration schemes are still tied to fixed parameters, fixed workflows, and fixed schemes, making them difficult to adapt to rapid algorithmic evolution [4-5,22]. The second is algorithmic inefficiency: model structures, numerical representations, and training workflows are often not designed for encrypted environments, resulting in redundant computation, redundant transmission, and redundant encryption [3,6,17]. The third is static strategy: many systems still apply uniform protection strength to all objects, without differentiating according to parameter sensitivity, device status, and latency constraints [10,14,26].

Within this framework, the following sections do not merely ask what methods exist. Instead, they compare methods by asking five questions: What is the problem? What are the representative methods? Where do the gains come from? Who bears the cost? What are the applicability boundaries? The goal is to move the literature review from a list of directions to a basis for method judgment, so that the conclusions can better serve system design and solution selection.

Table 1 summarizes the main optimization paths, typical gains, major costs, and applicable scenarios reviewed in this paper, and can serve as an overview for the following sections.

Table 1. High-latency optimization strategies for privacy-preserving computation.

Optimization layer	Main bottleneck	Representative methods	Typical gains	Major costs	More suitable scenarios	Evaluation judgment
Hardware acceleration	High computation and memory-access pressure in key operators such as NTT, key switching, and bootstrapping	GPU parallelization; FPGA customized pipelines; ASIC accelerators	GPU key switching achieves up to about 380x acceleration over traditional SEAL; dedicated accelerators can improve throughput and energy efficiency [4,8]	Strong parameter coupling, high development cost, and limited portability	Stable technical routes, clear hotspot operators, and high-volume batch workloads	Suitable for the slowest-operator problem, but hardware alone cannot remove redundancy at model and strategy layers
NTT/bootstrapping co-optimization	Long critical paths; data exchange and memory access become bottlenecks	NTT topology optimization; parallel bootstrapping; general bootstrapping paths	General bootstrapping can reduce level consumption; hypercube topology can improve NTT data exchange [22,28]	High design complexity and sensitivity to specific schemes and parameters	FHE core libraries, low-level execution engines, and long-term maintained infrastructure	A low-level performance breakthrough point, but gains depend on joint software-hardware redesign rather than single-point acceleration

Table 1. (continued) High-latency optimization strategies for privacy-preserving computation.

Optimization layer	Main bottleneck	Representative methods	Typical gains	Major costs	More suitable scenarios	Evaluation judgment
Model redesign and lightweighting	Original models are not HE-friendly, with high multiplicative depth and nonlinear costs	Polynomial approximation; structural pruning; quantization-aware training; HE-friendly encoding	Quantized MNIST execution time decreases by about 80%, CIFAR by about 40%; THOR reduces some BERT key-switching operations [3,29-30]	Retraining is required and may introduce accuracy loss and generalization risks	Inference tasks, redundant models, and tasks that can tolerate structural redesign	A fast-acting optimization approach, but with clear boundaries for high-precision and strongly nonlinear tasks
Communication compression and parameter pruning	Full-gradient uploading and full encryption are too costly in federated learning	Top-K gradients; quantization coding; model pruning; single-round secure updates	QuanCrypt-FL achieves up to about 9x encryption acceleration, 16x decryption acceleration, and 3x training-time compression; low-communication schemes reduce decentralized FL traffic by about 49% [17,23]	May affect convergence stability and parameter expressiveness	Communication-dominated federated learning and decentralized training	Essentially reduces the scale of protected objects and is suitable for repetitive communication-heavy tasks
Selective encryption	Different parameters have different sensitivities, while uniform strong protection wastes resources	Sensitivity maps; important-parameter filtering; HE only for highly sensitive parts	Selective HE concentrates high-cost protection on critical parameters or intermediate results and reduces full-encryption burden [10]	Additional sensitivity assessment is required; misclassification may cause underprotection or overprotection	Large-model training and federated learning with uneven parameter sensitivity	Suitable for security-efficiency trade-offs, provided sensitivity identification is reliable

Table 1. (continued) High-latency optimization strategies for privacy-preserving computation.

Optimization layer	Main bottleneck	Representative methods	Typical gains	Major costs	More suitable scenarios	Evaluation judgment
Hybrid privacy mechanisms	Single HE is difficult to use to simultaneously achieve high security, low latency, and high adaptability	HE+DP; HE+lightweight protocols; HE+TEE; blockchain+HE+DP	Strong protection can be concentrated on critical paths, reducing the total overhead of purely cryptographic schemes [14,18-19]	Higher system complexity, trust assumptions, and audit difficulty	Cloud-edge-terminal collaboration, industrial Internet, and cross-institutional collaboration	Suitable for engineering deployment, but trust boundaries of added components must be explicit
Dynamic adaptive strategies	Static thresholds cannot adapt to device heterogeneity, link fluctuation, and task-stage changes	Personalized privacy budgets; online scheduling; closed-loop feedback control	LAPA allocates differentiated privacy budgets by round and optimizes aggregation using communication quality [26]	Requires online monitoring, feedback control, and strategy auditing, with high engineering barriers	Heterogeneous edge networks, dynamic task loads, and long-running systems	A valuable future direction, but more suitable for system-level research than simple module stacking in the short term
Cloud-edge-terminal collaborative deployment	Single-node computing is limited, and improper task placement causes high end-to-end latency	Edge preprocessing; hierarchical aggregation; cloud-edge-terminal task partitioning; TEE-assisted deployment	Transforms single-node bottlenecks into system-level task-placement problems and improves real-time performance in resource-constrained environments [11-12,15,19]	High deployment and operation complexity; difficult task partitioning and trust-boundary design	Medical edge intelligence, industrial IoT, and resource-constrained terminals	The system layer that carries the preceding optimization paths into real applications

3. HARDWARE-ALGORITHM CO-DESIGN: BREAKING HARDWARE RIGIDITY

3.1 Hardware Mapping Requirements of Critical Ciphertext Operators

In homomorphic-encryption-based privacy-preserving computation, the latency upper bound is often determined not by the complete application but by a small number of critical operators, such as NTT, ciphertext multiplication, key switching, and bootstrapping [4,6]. These operators are both compute-intensive and memory-intensive. On the one hand, they perform a large number of multiply-add and relinearization operations over large moduli and high-dimensional vector spaces. On the other hand, they frequently read large keys and intermediate ciphertexts [4-5]. In this situation, general-purpose central processing units (CPUs) can satisfy functional correctness, but they often cannot provide sufficient parallelism and bandwidth after the system scale increases.

Representative methods address this issue by decomposing the operational structure of critical operators and mapping it onto heterogeneous hardware that is more suitable for parallel execution and high-bandwidth access [4-5,7-8]. The gain of this path is clear: if critical paths can be stably supported by hardware, system throughput and energy efficiency may improve substantially. The cost is equally clear: the more hardware design is tailored to a specific operator and parameter set, the more likely it is to lose generality when algorithms change. Therefore, the premise of hardware acceleration is not to move all tasks onto accelerators, but to identify which parts are structurally stable and repeatedly invoked.

3.2 Comparison of GPU, FPGA, and ASIC Acceleration Paths

GPUs, FPGAs, and ASICs are the three major acceleration platforms, but there is no universally optimal choice [4]. GPUs provide abundant parallel cores, mature software ecosystems, and low prototyping costs, making them suitable for quickly testing batched and parallel operators in homomorphic encryption. Studies have shown that for three key-switching methods, GPU implementations can achieve up to about 380x execution-time improvement over the Microsoft Simple Encrypted Arithmetic Library (SEAL) [27]. This result suggests that GPUs are well suited to workloads with high parallelism and relatively regular memory access.

The representative value of FPGAs lies in reconfigurability. Compared with GPUs, FPGAs can bind data paths, pipelines, and on-chip cache layouts more closely to target operators, and therefore can often process NTT, polynomial multiplication, and some parameter-switching operations with lower latency and higher energy efficiency [4,7]. Their gains come from structural customization, while their costs include higher development barriers, more complex resource planning, and stronger requirements for software-hardware co-design. ASICs pursue extreme throughput and energy efficiency and are particularly suitable for stable technical routes and large-scale deployment, but their inflexibility is also the strongest. Once parameter settings or scheme structures change, the migration cost of ASICs is much higher than that of GPUs and FPGAs.

3.3 Co-Optimization for NTT and Bootstrapping

If platform selection determines the basic form of acceleration, NTT and bootstrapping determine whether acceleration truly reaches the core bottleneck. NTT is frequently invoked in polynomial multiplication, key switching, and bootstrapping, and is therefore a typical low-level infrastructure component [4-5,28]. A common conclusion in related studies is that co-optimizing dataflow organization, processing-element interconnection, and memory-access scheduling can mitigate both computation and data-exchange pressure, but it also increases the complexity of structural design and parameter adaptation [28].

Bootstrapping optimization further reflects the collaborative logic that algorithms must make way for hardware and hardware must also be rewritten for algorithms. Existing work mainly reduces critical-path latency or improves overall throughput through FPGA customization, parallelized execution, and general bootstrapping path reconstruction [7-8,22]. The value of these methods does not lie in restating internal workflows, but in showing that bootstrapping optimization must jointly handle the trade-offs among algorithmic structure, hardware pipelines, and scheme compatibility.

Figure 1 illustrates the correspondence among critical operators, heterogeneous hardware platforms, and major benefits, helping readers understand the overall path of hardware co-design.

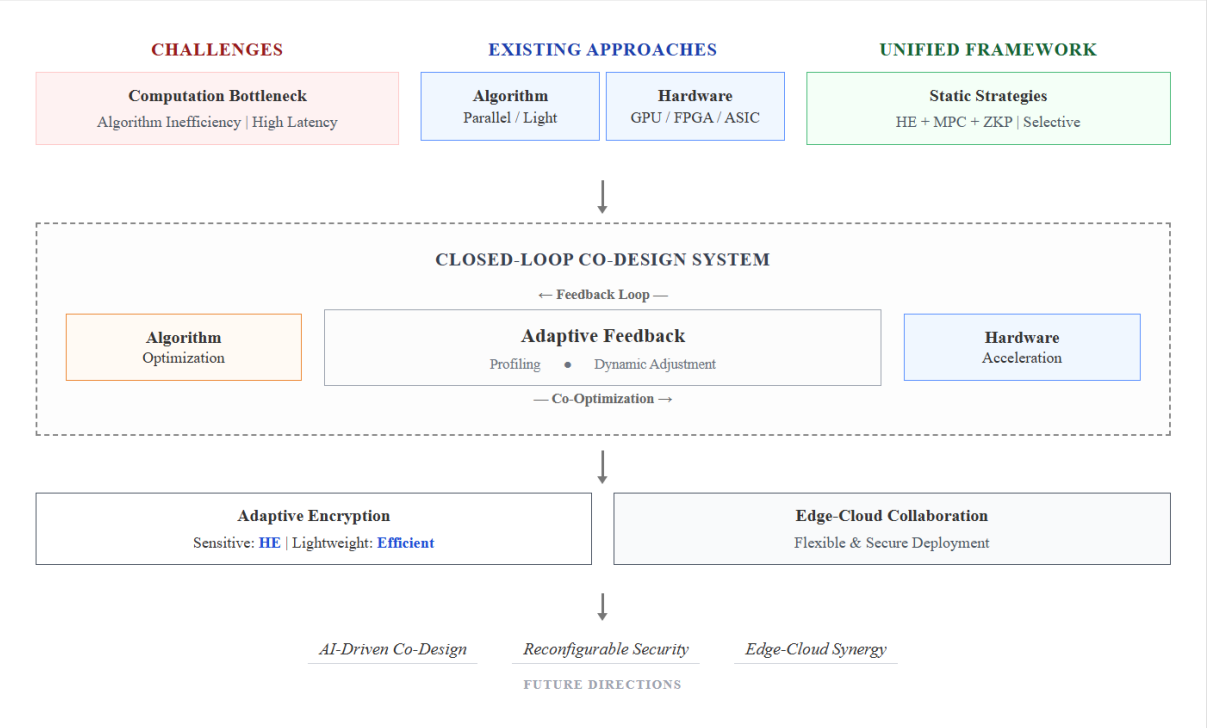

Figure 1. Mapping between critical privacy-preserving computation operators and heterogeneous hardware.

3.4 From Static Accelerators to Hardware-Aware Co-Design

Many acceleration studies have shown that heterogeneous platforms can effectively reduce the latency of privacy-preserving computation, but many schemes remain at the stage of static accelerators, optimizing only fixed parameters, fixed workloads, and fixed schemes [4,7-8]. Such methods often perform well in experimental environments, but they may not suit real systems because model size, encryption strength, batch size, and device load can all change dynamically. In other words, local speed does not necessarily imply system-level optimality.

Therefore, a more valuable next step is to make hardware design aware of algorithmic and scenario changes. On the algorithm side, designs should explicitly consider parallel granularity, memory-access patterns, and cache reuse, so that model pruning, quantization, and operator fusion can be mapped to low-level acceleration units [6]. On the hardware side, resource allocation should be adjusted elastically according to workload, bootstrapping demand, and resource status rather than fixed once offline [4-5]. This means that future hardware acceleration should not only answer how fast a certain operator can be, but also whether the system can maintain stable gains under algorithmic evolution and changing scenarios.

4. ALGORITHM-LEVEL OPTIMIZATION: MITIGATING ALGORITHMIC INEFFICIENCY

4.1 Parallelization, Packing, and Ciphertext Batching

Algorithmic inefficiency first appears as using too many encrypted operations to accomplish too little useful work. In homomorphic encryption scenarios, if element-wise processing from plaintext environments is directly adopted, the system will be slowed by many repeated ciphertext operations and unnecessary interactions [3,6]. The most basic representative method is ciphertext packing and batching: multiple plaintext values are encoded into different slots of the same ciphertext, so that multiple groups of data can be processed in parallel through one homomorphic operation [3,6]. For structured tasks such as matrix-vector multiplication, convolution, and aggregation, this approach directly reduces the number of homomorphic operations and improves effective throughput per unit time.

The benefit of this path is direct because it can reduce average computational cost without new hardware, making it suitable for systems that first optimize at the software and compiler layers. Its cost lies in data layout and scheduling complexity. If the packing strategy is poorly designed, it may save computation while increasing data rearrangement and memory-

access costs [3,6]. Thus, packing and batching are the starting point of algorithmic optimization, not the endpoint. Their gains become more stable only when they are considered together with model restructuring and hardware mapping.

4.2 HE-Oriented Model Redesign and Lightweight Design

The second source of algorithmic inefficiency is that existing machine-learning models were not originally designed for encrypted environments. Traditional deep models rely heavily on floating-point operations, complex activation functions, and deep network structures, whereas homomorphic encryption supports addition, multiplication, and bounded-depth composite operations more efficiently [3,6]. Directly migrating an original model into an HE environment often leads to excessive multiplicative depth, activation functions that are difficult to evaluate homomorphically, and large intermediate representations. Representative methods include replacing nonlinear functions with polynomial approximations, pruning network depth, reducing high-cost layers, optimizing encoding methods, and reorganizing matrix computation [3,6,29].

Such redesign has produced concrete gains in quantization and model lightweighting. Studies show that quantization-aware training can reduce integer bit-width and lower the execution time of Brakerski/Fan-Vercauteren (BFV)-based Modified National Institute of Standards and Technology (MNIST) digit-classification networks by about 80% and Canadian Institute for Advanced Research (CIFAR) image-classification networks by about 40% without significant accuracy loss [30]. For Transformer inference, THOR reduces the number of key-switching operations in the linear layers of Bidirectional Encoder Representations from Transformers (BERT)-base attention modules by up to 14.5x through diagonal-major encoding, compact ciphertext packing, and nonlinear optimization [29]. These results show that model redesign is not simply shrinking the model, but redesigning the computational path according to operations efficiently supported by HE. The cost is that the model must be retrained, approximation errors must be controlled, and generalization performance must be revalidated; therefore, not all tasks can be losslessly migrated to HE-friendly structures.

4.3 Top-K, Quantization Pruning, and Communication Compression

In federated learning and its privacy-preserving variants, algorithmic inefficiency also appears as too many objects being encrypted and transmitted. If every training round encrypts, uploads, and aggregates all gradients and parameters, the system is quickly overwhelmed by communication and encryption/decryption overhead [9,17]. Representative methods include Top-K gradient selection, quantization coding, model pruning, and compressed aggregation [9,17]. Their common idea is not to make every encryption operation faster, but to reduce the amount of information that truly needs protection and transmission.

This idea has yielded clear quantitative results. QuanCrypt-FL combines low-bit quantization, model pruning, and homomorphic encryption, and uses mean clipping to suppress quantization overflow. It achieves up to about 9x encryption acceleration, 16x decryption acceleration, and 3x training-time compression while maintaining competitive accuracy [17]. In decentralized federated learning, compressing the secure update protocol into a single communication round and combining it with functional encryption can reduce total communication by about 49% compared with existing HE baselines [23]. These results indicate that algorithmic optimization is often effective not because it protects everything faster, but because it protects less of what is not worth high-cost protection. However, the cost is also clear: excessive compression may harm convergence speed, parameter expressiveness, and some privacy-protection boundaries, so it must be evaluated together with task-accuracy requirements.

Figure 2 summarizes three algorithmic optimization paths-model redesign, parameter compression, and communication compression-as well as their benefit-cost relationships.

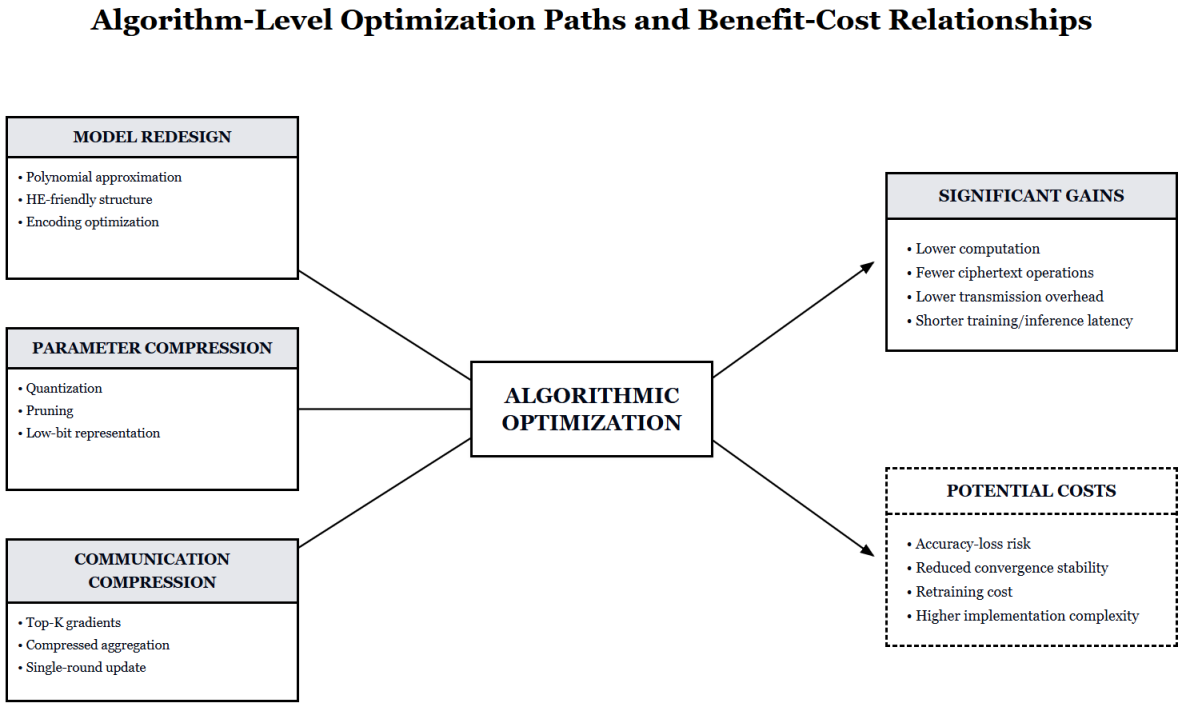

Figure 2. Algorithm-level optimization paths and benefit-cost relationships.

4.4 Benefit Boundaries and Use Criteria of Algorithmic Optimization

Compared with hardware design, algorithmic optimization is usually easier to deploy and less costly to modify, and is therefore often the first effective means of improving system performance [3,6]. However, it has clear boundaries. First, many gains rely on assumptions such as model redundancy, compressible parameters, and tolerance for some accuracy loss. When tasks require high precision, involve high-risk decisions, or use tightly coupled model structures, compression and approximation may no longer be safe [3,6]. Second, although algorithmic optimization can reduce computation and communication, it may not fundamentally eliminate rigid low-level costs such as bootstrapping, key switching, and large-scale memory access. In other words, the algorithm layer reduces burden rather than changes the underlying cost structure.

Algorithmic optimization is therefore more suitable for three types of tasks: inference tasks with clear model redundancy; federated learning tasks dominated by communication burden; and scenarios that can tolerate approximate computation and structural redesign. For such tasks, algorithmic optimization can often obtain substantial gains at relatively low cost. By contrast, for tasks with very deep multiplicative depth, strong nonlinear dependence, or strict accuracy constraints, algorithmic compression alone is usually insufficient and must be considered together with hardware and strategy layers.

5. HIERARCHICAL AND ADAPTIVE PROTECTION: OVERCOMING STATIC STRATEGIES

5.1 Why Full Uniform Encryption Is Inefficient

Many early schemes apply uniform protection strength to all data, parameters, and intermediate results by default. This approach is simple to implement, has clear boundaries, and is easy to analyze, making it suitable for theoretical proof and prototype systems [10,14]. The problem is that privacy value and business value are not uniformly distributed in real applications: different gradients, layer parameters, and data blocks have different sensitivities and attack-exposure surfaces. If a system applies the strongest protection to all objects, substantial computation and communication resources may be spent on low-value objects [10,18].

From a benefit-cost perspective, the main benefit of full uniform encryption is that the scheme is clearly defined, whereas the cost is overly coarse resource allocation. It assumes that all scenarios have the same latency constraints and computing conditions, while medical edge devices, industrial sensor nodes, and online financial risk-control systems clearly face different time budgets [11,14-15]. Therefore, the reason many systems become difficult to run after encryption is not that the

security objective is wrong, but that the protection strategy is not aligned with object sensitivity, device conditions, and business deadlines.

5.2 Selective Encryption and Sensitivity-Driven Parameter Protection

Selective encryption provides a more realistic path for the above problem. Its core idea is not to abandon strong protection, but to concentrate strong protection on parameters or intermediate results that are truly high-risk and more strongly related to raw data [10]. In federated learning, different gradients and weights contribute differently to privacy leakage. Therefore, applying HE or stronger protocols only to highly sensitive parts can substantially reduce computation and communication overhead while maintaining the security boundary [10].

Related research has produced persuasive results. Selective homomorphic encryption identifies key parameters through sensitivity maps and encrypts only highly sensitive parts, thereby reducing HE burden while maintaining accuracy and privacy protection [10]. FedML-HE embeds selective parameter encryption into federated learning systems, reducing training overhead by about 10x in Residual Network (ResNet)-50 scenarios and up to about 40x in BERT scenarios. The benefit is clear: limited resources are preferentially allocated to truly important objects. The cost is also non-negligible: the system must build an additional sensitivity-evaluation mechanism. If the evaluation is inaccurate, highly sensitive objects may be underprotected or low-sensitivity objects may still be overprotected. Therefore, the key to selective encryption is not encrypting less content, but reliably identifying what deserves priority protection.

5.3 Hierarchical Encryption and Hybrid Privacy Mechanisms

If selective encryption answers the question of what to protect, then hierarchical and hybrid mechanisms answer how to protect it. Increasing evidence shows that a single privacy technology is difficult to use to simultaneously achieve high security, low latency, and high adaptability [14]. Representative methods therefore combine HE, DP, SMPC, and TEE. For example, HE can protect critical aggregation paths [14,18], DP can control overall leakage risk [14,18], lightweight protocols can support fast edge-side processing [11], and TEE can replace part of the high-cost cryptographic computation [19].

The greatest benefit of such methods is that they change the system from placing all pressure on HE to using different protection tools at different layers, thereby improving the overall cost structure. In industrial Internet scenarios, blockchain, HE, and differential privacy have been combined for trusted aggregation and multi-level protection [18]. In edge federated learning, lightweight protocols, weight masking, or edge secure aggregation can substantially reduce centralized cloud-side processing burden [11,13,15]. In cross-TEE federated learning, Flatee shows that compared with schemes relying entirely on cryptographic operations, TEE can provide isolated execution while significantly reducing training and communication time [19]. However, hybrid mechanisms are not cost-free. They usually introduce more system components, more complex trust assumptions, and higher engineering debugging costs, and they may also increase audit and operation complexity. Thus, hybrid mechanisms are more suitable for engineering-oriented application environments that can tolerate a certain level of system complexity.

5.4 From Static Rules to Dynamic Adaptive Strategies

Selective and hierarchical protection are already better than uniform encryption, but if thresholds and invocation logic remain fixed, the problem of static strategies is not truly solved. In real systems, data sensitivity, network status, device load, and task stage may all change continuously, so rules hard-coded in advance can easily lead to configuration imbalance [10,14,26]. For example, the importance of the same batch of parameters may differ between early and late training, and the same device should not use exactly the same protection strength under high-bandwidth and low-bandwidth conditions.

The representative idea of dynamic adaptive strategies is to adjust privacy budgets and execution paths during runtime according to device heterogeneity, communication quality, and task state. Lightweight Adaptive Privacy Allocation (LAPA)-based dynamic privacy optimization can allocate personalized privacy budgets to different devices by round and combine this with communication-quality-aware transmission strategies, thereby improving convergence while satisfying privacy constraints [26]. The benefit is that protection and scheduling can be linked, avoiding long-term reliance on a static threshold set. The cost is that the system must support online monitoring, feedback control, and strategy auditing, making engineering implementation significantly more difficult than static schemes. Therefore, the prerequisite for real deployment is not adding another optimization module, but building an interpretable, traceable, and stable closed-loop scheduling framework.

Figure 3 illustrates the evolution from uniform protection to selective protection and dynamic adaptive protection.

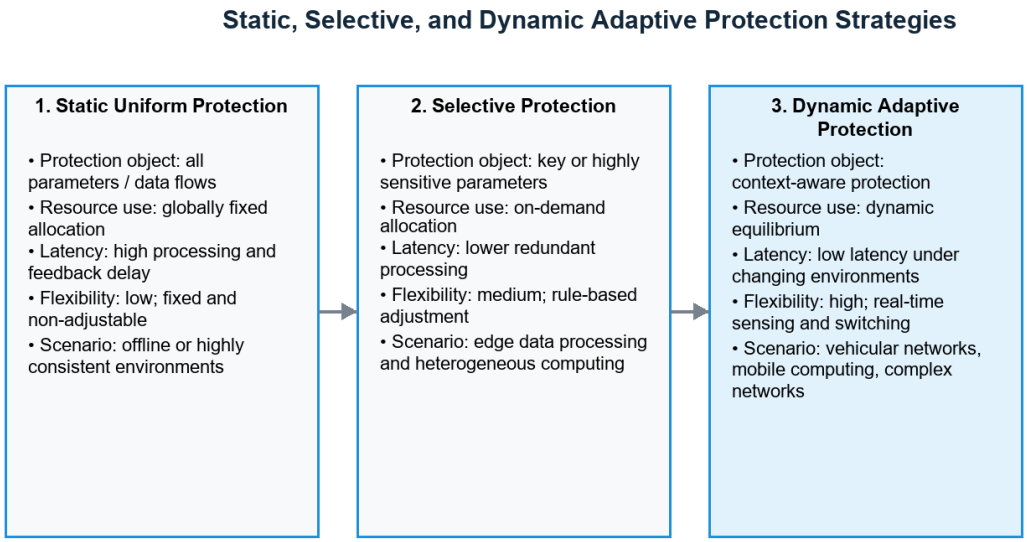

Figure 3. Static, selective, and dynamic adaptive protection strategies.

6. CLOUD-EDGE-TERMINAL COLLABORATION AND SCENARIO-AWARE DEPLOYMENT

6.1 Requirements for Privacy-Preserving Learning in Edge Environments

With the widespread connection of mobile terminals, wearable devices, industrial sensors, and smart medical devices, increasing amounts of data are generated at the network edge and participate in model training [11-12,15]. Edge environments are closer to data sources and can shorten transmission paths and improve real-time response. However, this does not make privacy protection easier. On the contrary, edge nodes often have limited computing power, limited memory, tight energy supply, link fluctuations, and device heterogeneity, making high-complexity HE or multi-round secure protocols more likely to become deployment bottlenecks [11-12,15].

Therefore, the key question in edge scenarios is no longer whether protection is needed, but how to complete protection within an acceptable time budget. From the perspective of method judgment, simply pursuing the strongest cryptographic protection is often difficult to deploy, whereas solutions combining model lightweighting, task hierarchy, and collaborative deployment are more realistic [12,15]. In other words, edge environments force privacy-preserving computation to move from theoretical optimality to system usability.

6.2 Differentiated Requirements in Healthcare, Finance, Industrial Internet of Things (IIoT), and Mobile Scenarios

Different application scenarios vary greatly in their tolerance of high latency. Healthcare scenarios first emphasize regulatory compliance, patient privacy, and model trustworthiness. Electronic health records, medical images, and Internet of Medical Things (IoMT) data are highly sensitive, so protection strength cannot be weak; however, in edge diagnosis and remote monitoring, inference latency must also remain low, making lightweight models and near-end collaborative deployment practically valuable [15-16]. Financial scenarios place greater emphasis on high concurrency, low latency, and high reliability. Many risk-control and fraud-detection tasks lie on millisecond-level decision chains, so they have lower tolerance for multi-round interaction and high-cost ciphertext computation. IIoT scenarios face heterogeneous devices, multi-source data, and unstable links, and therefore rely more on lightweight protocols, hierarchical aggregation, and edge preprocessing [12-13,18]. In highly mobile distributed learning scenarios such as the Internet of Vehicles, defenses against gradient leakage must also be balanced with latency constraints. Leveled HE can strengthen protection, but it also introduces additional computation and transmission burden.

These differences show that privacy-preserving computation optimization cannot adopt a one-size-fits-all strategy. If a scheme has only been validated in a stable cloud environment, it may not be suitable for edge healthcare. If it has only

been tested on small-scale nodes, it may not support high-concurrency financial systems. Thus, scenario differences are not merely deployment details, but prerequisites for method selection.

Figure 4 compares healthcare, finance, and IIoT in terms of security, real-time requirements, and resource constraints.

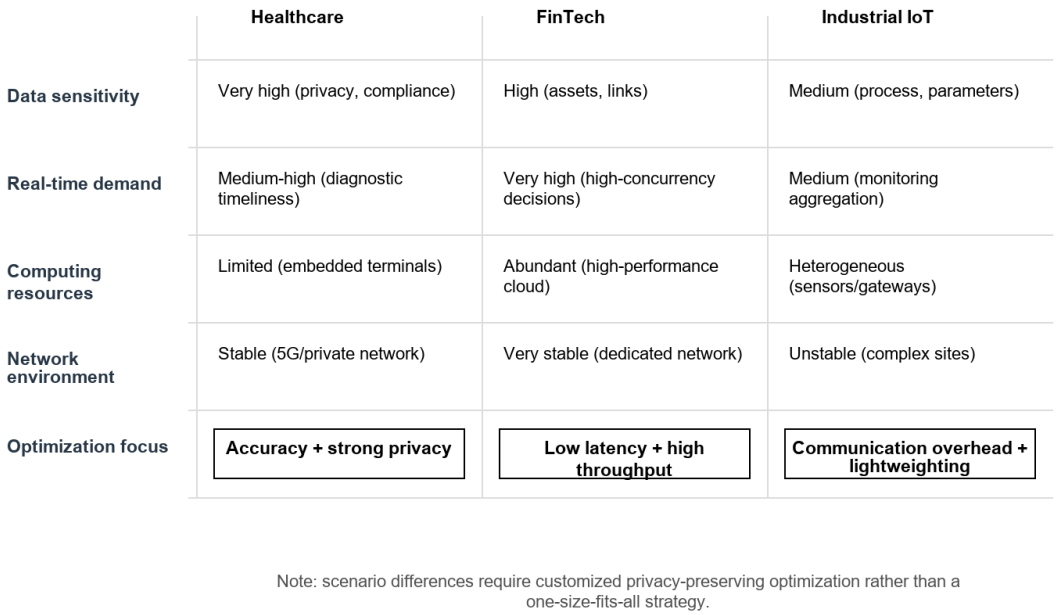

Figure 4. Demand differences among healthcare, finance, and IIoT scenarios.

6.3 Lightweight and Low-Latency Deployment under Cloud-Edge-Terminal Collaboration

The core of cloud-edge-terminal collaboration is not simply splitting a task into three segments. Instead, it places training, encryption, aggregation, and inference at more appropriate locations according to task complexity, data sensitivity, and resource conditions [12-13]. In such architectures, privacy protection is no longer bound to a single node, but becomes a system capability that can be migrated, layered, and differentially configured. Related studies show that introducing homomorphic aggregation, lightweight secure protocols, or parameter preprocessing at edge nodes can reduce centralized cloud-side processing burden [11,13]. In resource-constrained scenarios, combining lightweight models, hierarchical protection, and trusted execution paths is more likely to achieve low-latency deployment than simply increasing cryptographic strength at a single layer [12,15,19].

In terms of benefits, cloud-edge-terminal collaboration transforms the problem of computing pressure that a single node cannot bear into the question of where tasks should be placed most cost-effectively, thereby providing new degrees of freedom for system optimization. Its cost lies in increased deployment complexity, requiring the redesign of cross-node collaboration, fault tolerance, and security boundaries. Therefore, the importance of cloud-edge-terminal collaboration lies not only in multi-layer deployment itself, but in elevating privacy-preserving computation optimization from local operator acceleration to system-level resource scheduling.

6.4 Future Evolution toward Closed-Loop Collaboration

Although cloud-edge-terminal collaboration has opened new optimization space, many current systems still remain at the stage of static layered deployment and have not formed true closed-loop collaboration [12,26]. Future research should focus on unified benchmarks, cross-layer scheduling, and scenario-aware validation, so that different privacy mechanisms, hardware platforms, and deployment locations can be compared within the same evaluation framework and protection strength and execution paths can be dynamically adjusted according to real-time states.

Figure 5 summarizes the relationship among cloud-edge-terminal collaboration, dynamic scheduling, and closed-loop feedback.

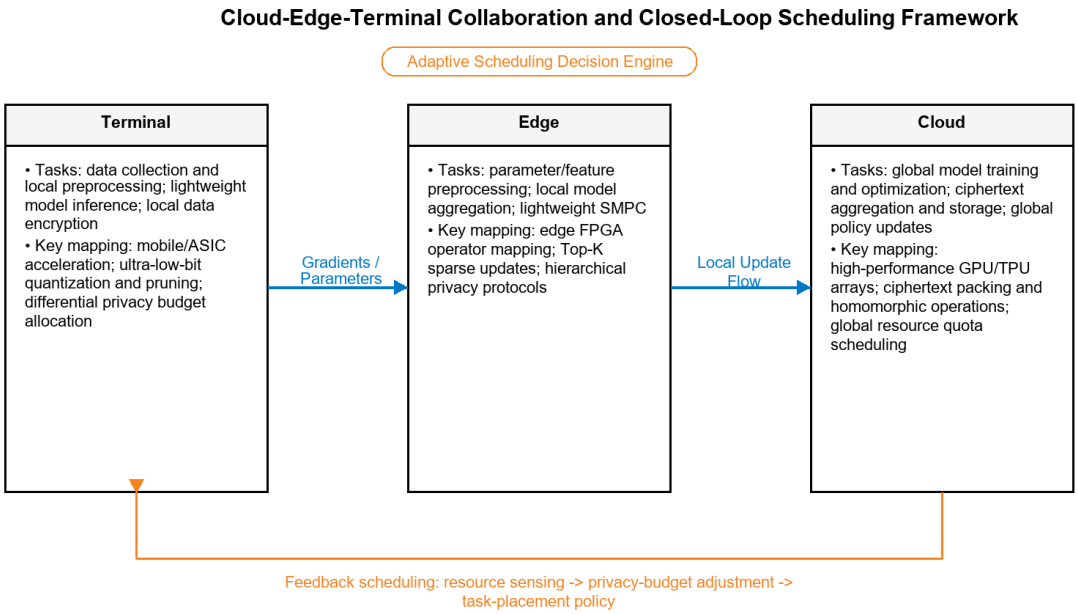

Figure 5. Cloud-edge-terminal collaborative deployment and closed-loop scheduling framework.

From the perspective of this review, the most promising future research is not to continue pursuing the extreme speed of an isolated component, but to unify hardware mapping, algorithmic compression, protection strength, and deployment location into a system optimization process that can sense state, balance cost, and provide continuous feedback. Only when privacy-preserving computation has such cross-layer closed-loop capability can it stably serve complex scenarios such as healthcare, finance, industrial Internet, and mobile intelligence.

7. CONCLUSION

This paper reviews hardware acceleration, algorithmic optimization, hierarchical and adaptive protection, and cloud-edge-terminal collaborative deployment for the high-latency problem in privacy-preserving computation. The review yields three findings. First, high latency is essentially the result of computational complexity, protocol interaction, and data movement acting together. Second, different optimization paths correspond to different bottlenecks: hardware co-design is suitable for stable and frequently invoked operators; algorithmic optimization is suitable for reducing model and communication redundancy; and hierarchical protection and cloud-edge-terminal collaboration are suitable for addressing resource constraints and scenario differences. Third, engineering-usable methods must simultaneously explain the source of performance gains, implementation costs, and applicability boundaries. Future research should advance unified benchmarks, cross-layer collaboration, and scenario-aware validation, so that privacy-preserving computation can move further from theoretical feasibility to engineering usability.

REFERENCES

[1] Heurix, J., Zimmermann, P., Neubauer, T., & Fenz, S. (2015). A taxonomy for privacy enhancing technologies. Computers & Security, 53, 1-17. https://doi.org/10.1016/j.cose.2015.05.002

[2] Fantaye, J. (2023). An introduction and overview of privacy-enhancing technologies for data processing and analysis [Bachelor's thesis, Technical University of Munich]. Software Engineering for Business Information Systems, Technical University of Munich. https://www.cs.cit.tum.de/fileadmin/w00cfj/sebis/thesis/230816_Fantaye_Thesis.pdf

[3] Podschwadt, R., Takabi, D., Hu, P., Rafiei, M. H., & Cai, Z. (2022). A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption. IEEE Access, 10, 117477-117500. https://doi.org/10.1109/ACCESS.2022.3219049

[4] Bian, S., Mao, R., Zhu, Y., Fu, Y., Zhang, Z., Ding, L., Zhang, J., Zhang, B., Chen, Y., Dong, J., & Guan, Z. (2024). A survey on software-hardware acceleration for fully homomorphic encryption. Journal of Electronics & Information Technology, 46(5), 1790-1805. https://doi.org/10.11999/JEIT230448

[5] Kim, S.-P. (2025). Bootstrapping-oriented software and hardware solutions for accelerating fully homomorphic encryption [Doctoral dissertation, Seoul National University]. S-Space. https://hdl.handle.net/10371/228461

[6] Wang, Z., Zhao, L., Cheng, J., Hou, R., & Meng, D. (2025). A survey on privacy-preserving neural networks based on homomorphic encryption. Journal of Cyber Security. Advance online publication. https://doi.org/10.19363/J.cnki.cn10-1380/tn.2025.04.01

[7] Xu, Z., Ye, T., Kannan, R., & Prasanna, V. K. (2025). FAST: FPGA acceleration of fully homomorphic encryption with efficient bootstrapping. In Proceedings of the 2025 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (pp. 159-170). ACM. https://doi.org/10.1145/3706628.3708879

[8] Agrawal, R., Chandrakasan, A., & Joshi, A. (2024). HEAP: A fully homomorphic encryption accelerator with parallelized bootstrapping. In 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA) (pp. 756-769). IEEE. https://doi.org/10.1109/ISCA59077.2024.00060

[9] Yu, S., & Chen, Z. (2023). Efficient secure federated learning aggregation framework based on homomorphic encryption. Journal on Communications, 44(1), 14-28. https://doi.org/10.11959/j.issn.1000-436x.2023015

[10] Korkmaz, A., & Rao, P. (2025). A selective homomorphic encryption approach for faster privacy-preserving federated learning [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2501.12911

[11] He, C., Liu, G., Guo, S., & Yang, Y. (2022). Privacy-preserving and low-latency federated learning in edge computing. IEEE Internet of Things Journal, 9(20), 20149-20159. https://doi.org/10.1109/JIOT.2022.3171767

[12] Zhan, S., Huang, L., Luo, G., Zheng, S., Gao, Z., & Chao, H.-C. (2025). A review on federated learning architectures for privacy-preserving AI: Lightweight and secure cloud-edge-end collaboration. Electronics, 14(13), Article 2512. https://doi.org/10.3390/electronics14132512

[13] Zhu, B., & Niu, L. (2025). A privacy-preserving federated learning scheme with homomorphic encryption and edge computing. Alexandria Engineering Journal, 118(4), 11-20. https://doi.org/10.1016/j.aej.2024.12.070

[14] Aziz, R., Banerjee, S., Bouzefrane, S., & Le Vinh, T. (2023). Exploring homomorphic encryption and differential privacy techniques towards secure federated learning paradigm. Future Internet, 15(9), Article 310. https://doi.org/10.3390/fi15090310

[15] Wang, R., Lai, J., Zhang, Z., Li, X., Vijayakumar, P., & Karuppiah, M. (2023). Privacy-preserving federated learning for Internet of Medical Things under edge computing. IEEE Journal of Biomedical and Health Informatics, 27(2), 854-865. https://doi.org/10.1109/JBHI.2022.3157725

[16] Naresh, V. S., Venkata Raju, A., & Srinivasa Rao, O. (2025). Secure multiparty computation for privacy-preserving machine learning in healthcare: A comprehensive survey. WIREs Computational Statistics, 17(3), e70046. https://doi.org/10.1002/wics.70046

[17] Mia, M. J., & Amini, M. H. (2024). QuanCrypt-FL: Quantized homomorphic encryption with pruning for secure federated learning [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2411.05260

[18] Jia, B., Zhang, X., Liu, J., Zhang, Y., Huang, K., & Liang, Y. (2022). Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT. IEEE Transactions on Industrial Informatics, 18(6), 4049-4058. https://doi.org/10.1109/TII.2021.3085960

[19] Mondal, A., More, Y., Rooparaghunath, R. H., & Gupta, D. (2021). Flatee: Federated learning across trusted execution environments [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2111.06867

[20] Zhou, I., Tofigh, F., Piccardi, M., Abolhasan, M., Franklin, D. R., & Lipman, J. (2024). Secure multi-part computation for machine learning: A survey. IEEE Access, 12, 53881-53899. https://doi.org/10.1109/ACCESS.2024.3388992

[21] Shao, H., Gao, S., Zhong, L., Fu, Z., Meng, D., & Li, X. (2022). Homomorphic encryption protocols and applications in privacy preserving computation. Information and Communications Technology and Policy, 48(8), 75-88. https://doi.org/10.12267/j.issn.2096-5931.2022.08.012

[22] Kim, A., Deryabin, M., Eom, J., Choi, R., Lee, Y., Ghang, W., & Yoo, D. (2024). General bootstrapping approach for RLWE-based homomorphic encryption. IEEE Transactions on Computers, 73(1), 86-96. https://doi.org/10.1109/TC.2023.3318405

[23] Zhang, J., Zhang, J., Wang, Y., Pang, K., Han, R., Ma, Z., & Ma, J. (2025). Privacy-preserving decentralized federated learning scheme with low communication overhead based on functional encryption. Journal on Communications, 46(11), 1-16. https://doi.org/10.11959/j.issn.1000-436x.2025205

[24] Peralta, G., Cid-Fuentes, R. G., Bilbao, J., & Crespo, P. M. (2019). Homomorphic encryption and network coding inIoT architectures: Advantages and future challenges. Electronics, 8(8), 827. https://doi.org/10.3390/electronics8080827

[25] Kumar, M., Sethi, M., Rani, S., Sah, D. K., AlQahtani, S. A., & Al-Rakhami, M. S. (2023). Secure data aggregation based on end-to-end homomorphic encryption in IoT-based wireless sensor networks. Sensors, 23(13), 6181. https://doi.org/10.3390/s23136181

[26] Sun, P., Liu, E., Ni, W., Wang, R., Geng, Y., Lai, L., & Jamalipour, A. (2025). LAPA-based dynamic privacy optimization for wireless federated learning in heterogeneous environments [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2505.19823

[27] Ozcan, A. S., & Savas, E. (2025). GPU implementations of three different key-switching methods for homomorphic encryption schemes. Cryptology ePrint Archive, Paper 2025/124. https://eprint.iacr.org/2025/124

[28] Hu, J., Fang, Y., & Dai, W. (2024). Number-theoretic transform architecture for fully homomorphic encryption from hypercube topology. Cryptology ePrint Archive, Paper 2024/498. https://eprint.iacr.org/2024/498

[29] Moon, J., Yoo, D., Jiang, X., & Kim, M. (2024). THOR: Secure transformer inference with homomorphic encryption. Cryptology ePrint Archive, Paper 2024/1881. https://eprint.iacr.org/2024/1881

[30] Legiest, W., Turan, F., Van Beirendonck, M., D'Anvers, J.-P., & Verbauwhede, I. (2023). Neural network quantisation for faster homomorphic encryption. In 2023 IEEE 29th International Symposium on On-Line Testing and Robust System Design (IOLTS) (pp. 1-3). IEEE. https://doi.org/10.1109/IOLTS59296.2023.10224890