

Article

End-to-End Privacy-Aware Federated Learning for Wearable Health Devices via Encrypted Aggregation in Programmable Networks

Huzaif Khan, Rahul Kavati , Sriven Srilakshmi Pulkaram and Ali Jalooli * 

Department of Computer Science, California State University Dominguez Hills, 1000 E. Victoria Street, Carson, CA 90747, USA; hkhan17@toromail.csudh.edu (H.K.); rkavati1@toromail.csudh.edu (R.K.); spulkaram1@toromail.csudh.edu (S.S.P.)

* Correspondence: ajalooli@csudh.edu

Abstract

The widespread use of wearable Internet of Things (IoT) devices has transformed modern healthcare through the real-time monitoring of physiological signals. However, real-time responsiveness and data privacy are big challenges. Federated Learning (FL) keeps direct data exposure to a minimum but is susceptible to inference attacks on model updates and heavy communication overhead. In-network computing (INC) solutions currently offer greater efficiency but without cryptographic security, whereas homomorphic encryption (HE) offers high privacy but at the expense of latency and scalability. To bridge this gap, we present Edge-Assisted Homomorphic Federated Learning (EAH-FL), a framework that unifies Cheon–Kim–Kim–Song (CKKS) HE with in-network aggregation. Lightweight clients outsource encryption and decryption to trusted edge devices, whereas programmable switches carry out aggregation in the encrypted domain. Massive-scale simulations over realistic healthcare data sets demonstrate that EAH-FL preserves near-plaintext model accuracy (F1-score > 0.93), delivers packet delivery ratios > 0.95, and converges well for various client scales. The encryption expense is mostly incurred by the edge layer rather than resource-constrained wearables. Through the use of encryption, in-network acceleration, and smart routing, EAH-FL provides the first practical solution that achieves strong privacy, low latency, and scalability for real-time federated learning in healthcare in a single solution. These results validate its viability as a deployable and secure building block for next-generation digital health monitoring.

Keywords: federated learning; homomorphic encryption; in-network computing; edge computing; graph neural networks; healthcare IoT



Academic Editors: Jinbo Xiong, Shaoen Wu, Periklis Chatzimisios and Mahmoud Daneshmand

Received: 1 October 2025

Revised: 12 November 2025

Accepted: 14 November 2025

Published: 17 November 2025

Citation: Khan, H.; Kavati, R.; Pulkaram, S.S.; Jalooli, A. End-to-End Privacy-Aware Federated Learning for Wearable Health Devices via Encrypted Aggregation in Programmable Networks. *Sensors* **2025**, *25*, 7023. <https://doi.org/10.3390/s25227023>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The rapid adoption of wearable Internet of Things (IoT) devices has transformed modern healthcare by enabling continuous, real-time monitoring of physiological signals such as heart rate, oxygen saturation, blood pressure, and physical activity. This constant stream of multimodal physiological data supports predictive diagnostics, chronic disease management, and timely clinical interventions, significantly improving patient outcomes and advancing personalized medicine. While the ubiquity, continuity, and personalization of health IoT make it highly appealing, these same attributes also introduce critical challenges related to data privacy, latency, communication overhead, and adherence to stringent regulations such as Health Insurance Portability and Accountability Act (HIPAA)

and the General Data Protection Regulation (GDPR). In latency-sensitive scenarios such as arrhythmia detection or hypoxemia alerts, even a delay of a few seconds can mean the difference between life and death, underscoring the criticality of low-latency and privacy-preserving learning frameworks. Despite increasing attention, healthcare remains a uniquely challenging domain for the deployment of FL. Unlike consumer-facing applications such as keyboard prediction or recommendation systems, healthcare IoT data are inherently high-stakes and sensitive. Any latency in processing vital signs can delay life-critical interventions, and any breach of confidentiality can result in irreparable harm to patient trust, regulatory violations, and substantial financial penalties. A 2025 report [1] by IBM estimated that the average cost of a healthcare data breach exceeded \$9.77 million. Thus, frameworks designed for medical IoT cannot simply inherit generic FL or HE mechanisms; they must meet more stringent requirements for latency, security, and scalability simultaneously. Traditional cloud-centric machine learning architectures are increasingly inadequate for such stringent healthcare requirements. These centralized approaches require transmitting raw physiological data to remote servers for model training, creating bottlenecks in both latency and bandwidth while exposing sensitive health information to potential privacy breaches. FL has emerged as a promising paradigm that enables distributed devices to collaboratively train a shared model without exchanging raw data, thereby reducing privacy risks and facilitating regulatory compliance. While baseline algorithms such as FedAvg and its extensions (e.g., FedProx and FedDyn) address client drift to some extent, studies show that these approaches remain sensitive to healthcare data heterogeneity and lack robust privacy guarantees [2,3].

To enhance confidentiality during decentralized learning, researchers have incorporated HE into FL to perform computations on encrypted model updates. Works such as FedNIC [4], Shen et al. [5], and Walskaar et al. [6] demonstrate secure aggregation using single and multi-key HE schemes, ensuring gradients remain private throughout the training process. However, these solutions introduce high computational and communication costs, rendering them impractical for resource-constrained IoT devices. Other studies have explored blockchain-assisted HE to improve trust and accountability [7,8], yet they still suffer from scalability and latency bottlenecks. More recent research by Gowri et al. [9] and Khan et al. [10] applied the Cheon–Kim–Kim–Song (CKKS) approximate HE scheme in medical and edge-AI scenarios, respectively, achieving comparable accuracy on encrypted data while significantly reducing execution time. These findings suggest that lightweight, edge-enabled encryption is feasible, but the integration of privacy and efficiency within a unified framework remains an open challenge.

In parallel, In-Network Computing (INC) has gained importance as an effective paradigm for reducing communication delays and bandwidth consumption by enabling programmable network devices to perform intermediate model aggregation. Approaches such as FediAC [11], GAIN [12], and FLIP4 [13] demonstrate substantial throughput and convergence improvements by enabling near-data aggregation. Nevertheless, these INC frameworks operate primarily on plaintext model updates and therefore cannot ensure data confidentiality, a critical limitation in regulated medical applications. Recent surveys by Caruccio et al. [14] and Albshaier et al. [15] emphasize that achieving true privacy-preserving and low-latency FL requires the joint integration of cryptographic protection and system-level optimization through edge-assisted architectures.

Motivated by these limitations, this study proposes the Edge-Assisted Homomorphic Federated Learning (EAH-FL) framework, which integrates HE with in-network aggregation to achieve both privacy and efficiency in healthcare IoT environments. EAH-FL employs selective encryption to reduce cryptographic overhead and utilizes trusted edge servers to offload aggregation tasks on encrypted data, maintaining end-to-end confiden-

tiality while minimizing latency. Experimental evaluations show that EAH-FL reduces communication latency by up to 46% and encryption overhead by 38% compared to conventional HE-FL methods. The marginal accuracy loss is less than 1.2% in non-IID healthcare datasets. Furthermore, EAH-FL enhances throughput and energy efficiency in multi-client edge environments, demonstrating its potential for real-time, regulation-compliant healthcare analytics.

The main contributions of this paper can be summarized as follows:

- A formally defined mathematical framework for encrypted edge-assisted federated learning. Unlike prior heuristic or architecture-only FL designs, EAH-FL provides a rigorous mathematical formulation that models client updates, encrypted aggregation, and edge-level optimization under HE. This formalism allows analytical reasoning about latency, encryption cost, and convergence behavior.
- A novel hybrid architecture with HE+INC and EAH-FL. This framework jointly leverages programmable edge nodes for in-network encrypted aggregation, ensuring that all model updates remain encrypted end-to-end. This integration bridges the gap between privacy-preserving cryptography and system-level acceleration achieving both confidentiality and real-time performance.
- The Graphical neural network (GNN)-assisted topology inference module improves encrypted model aggregation efficiency by learning dynamic inter-client relationships. Experimental results show up to 23% lower latency, 31% communication reduction (CRR), and a 12% improvement in convergence stability compared to static edge aggregation.
- Comprehensive empirical validation and analysis. Extensive experiments including baseline comparisons, ablation studies, and parameter sensitivity analyzes confirm that EAH-FL achieves a superior balance of privacy, efficiency, and scalability compared to state-of-the-art FL frameworks. The proposed model outperforms HE-only and INC-only systems, proving its feasibility for real-time healthcare IoT environments.

Overall, the proposed framework bridges the gap between privacy preservation and operational efficiency in federated healthcare systems, demonstrating that secure, low-latency learning is achievable through the synergy of edge computing and HE.

2. Related Work

FL has become a cornerstone for privacy-preserving machine learning in domains where data sensitivity and regulation prevent centralized aggregation. Early studies established FedAvg as the practical baseline. Lee et al. [3] reviewed the stability of FedAvg under various conditions, including medical image classification tasks, and found that it maintained stable convergence even when the data distributions were highly heterogeneous. This reinforced its role as a strong baseline, though it also highlighted that FedAvg alone does not handle all the challenges of healthcare data heterogeneity. More advanced methods, such as FedProx and FedDyn, were developed to mitigate client drift under non IID data. While FedProx constrains local updates through a proximal penalty, and FedDyn dynamically corrects local objectives, Yang et al. [2] showed that even with these techniques, FL in medical imaging remained sensitive to non-IID data and still required additional privacy-preserving mechanisms. Recent studies have reinforced this dual requirement. Mao et al. [8] proposed EPFed, a blockchain-assisted FL framework integrating HE and secret sharing to balance privacy and computational efficiency. Likewise, Aziz et al. [16] reviewed the intersection of Differential Privacy (DP) and HE for FL, concluding that hybrid cryptographic frameworks can mitigate gradient leakage while maintaining convergence stability. These insights underscore the need for privacy-preserving frame-

works that simultaneously optimize latency and communication efficiency in real-world healthcare environments.

A significant line of research focuses on enhancing privacy through HE. Choi et al. [4] introduced FedNIC, which offloads HE operations to SmartNICs, thereby reducing server-side computation overhead and improving efficiency. However, their work still relies on centralized aggregation, which introduces latency and scalability bottlenecks. Yang et al. [2] proposed a dynamic corrected split FL framework for medical image networks, applying HE to protect updates in specific layers. Although effective for U-shaped models in imaging, their approach limited encryption to selected layers and did not reduce communication costs. Shen et al. [5] tackled encryption flexibility by presenting a multi-key HE framework that allows different clients to encrypt with distinct keys while still supporting secure aggregation. Similarly, Walskaar et al. [6] demonstrated a practical medical FL system using the Flower framework combined with multi-key FHE, confirming feasibility but also exposing performance issues due to high cryptographic overhead. Naresh and Varma [17] applied HE in a healthcare-specific case of heart stroke detection, showing that HE-enabled FL could support clinical decision-making, but they did not address latency or energy efficiency in IoT devices. Firdaus et al. [7] combined blockchain with HE to improve accountability in healthcare data sharing. While this strengthened trust, their results showed high encryption overheads and limited scalability for latency-sensitive environments. Collectively, these works confirm the strength of HE for confidentiality but show that HE alone is often too heavy for real-time healthcare IoT deployments. Recent implementations have begun to address this limitation. Gowri et al. [9] employed the CKKS approximate HE scheme for secure medical diagnostics, achieving over 70% model accuracy on encrypted data while preserving patient privacy. Similarly, Khan et al. [10] demonstrated the feasibility of CKKS-based full HE on resource-constrained edge devices by introducing precision-aware algorithms that maintain near plaintext accuracy. These advances validate that lightweight HE can be deployed at the network edge, paving the way for practical, edge-assisted encrypted learning as envisioned in EAH-FL.

Another thread of work focuses on INC for FL efficiency. Su et al. [11] proposed FedAC, a voting-based consensus model compression algorithm that reduces switch memory usage by aligning model indices before aggregation. Their results demonstrated faster convergence and reduced communication cost, but they left model updates unencrypted, creating privacy risks. Xia et al. [12] introduced GAIN, a stateless in-network aggregation framework that leverages programmable switches to accelerate FL, achieving up to $4.11\times$ throughput and 86.5% traffic reduction. However, GAIN's privacy guarantees came from architectural statelessness rather than cryptographic protection, making it inadequate for sensitive healthcare applications. Ji et al. [18] approached the same problem by modeling in-network aggregation and flow routing as an optimization problem. Their online scheduling framework minimized communication cost and delay, but like FedAC and GAIN, their solution assumed plaintext aggregation, overlooking privacy requirements. Zang et al. [13] extended INC concepts to IoT traffic analysis with FLIP4, combining lightweight models in network devices for fast, distributed learning. Although FLIP4 reduced communication costs and supported rapid incident response, it did not incorporate HE or encrypted aggregation, and it was designed for traffic analysis rather than healthcare. The need to merge these system-level optimizations with privacy guarantees has been emphasized in broader taxonomies of FL architectures by Caruccio et al. [14] and Albshaier et al. [15], both of whom identified edge-assisted and hybrid designs as key enablers for scalable, privacy-aware FL.

Surveys provide additional perspective on the field. Aziz et al. [16] offered a comprehensive review of HE and DP techniques for secure FL, highlighting their complementary

roles in balancing privacy, convergence, and computation. Similarly, Albshaier et al. [15] systematically analyzed FL applications for cloud and edge security, revealing a 25% reduction in privacy risks but persistent communication bottlenecks. Xie et al. [19] reviewed efficiency optimization strategies for HE in FL, identifying batching, packing, and selective encryption as key techniques to reduce overhead. They concluded that HE is promising but impractical without system-level optimizations. Gu et al. [20] provided a comprehensive review of privacy-enhancing methods for healthcare FL, categorizing approaches such as HE, differential privacy, and secure aggregation. Their review underscored the lack of integrated frameworks that jointly address strong privacy and communication efficiency. Lessage et al. [21] contributed empirical evidence by applying FHE to FL for mammogram analysis, confirming that FHE preserved confidentiality but also revealed major memory and runtime limitations. Jin et al. [22] advanced the discussion with FedML-HE, which selectively encrypts sensitive parameters, reducing overhead by up to $40\times$ for large models while maintaining resilience against inference attacks. Their results highlight selective encryption as a promising compromise between security and efficiency. A comparative summary of the discussed approaches is presented in Table 1, which contrasts their handling of privacy (HE), INC, latency, and scalability. As shown, existing systems typically address either privacy or efficiency in isolation, leading to an unresolved trade-off between cryptographic protection and real-time responsiveness.

Table 1. Summary of Related Works (✓ = addressed, ✗ = not addressed, ~ = partially addressed).

Authors	INC	HE	FL	Res. Constr.	Latency	End-to-End Enc.	Scalability
Su et al. [11]	✓	✗	✓	✗	✓	✗	✓
Xia et al. [12]	✓	✗	✓	✗	✓	✗	✓
Ji et al. [18]	✓	✗	✓	✗	✓	✗	✓
Zang et al. [13]	✓	✗	✓	✓	✓	✗	✓
Choi et al. [4]	✗	✓	✓	✓	✓	~	✗
Yang et al. [2]	✗	✓	✓	✗	✗	~	✗
Shen et al. [5]	✗	✓	✓	✗	✗	✓	✓
Walskaar et al. [6]	✗	✓	✓	✗	✗	✓	✗
Naresh and Varma [17]	✗	✓	✓	✓	✗	✗	✗
Firdaus et al. [7]	✗	✓	✓	✓	✗	✓	✗
Lessage et al. [21]	✗	✓	✓	✗	✗	✓	✗
Hijazi et al. [23]	✗	✓	✓	✗	✗	✓	✗
Jin et al. [22]	✗	✓	✓	✗	✗	~	✓
Gu et al. [20]	✗	✓	✓	✗	✗	✗	✗
Lee et al. [3]	✗	✗	✓	✗	✗	✗	✗
Aziz et al. [16]	✗	✓	✓	✗	✗	✓	✓
Mao et al. [8]	✗	✓	✓	✓	✓	✓	✓
Gowri et al. [9]	✗	✓	✗	✓	✓	✓	✗
Khan et al. [10]	✗	✓	✗	✓	✓	✓	✓
Caruccio et al. [14]	✓	✓	✓	✓	✓	✗	✓
Albshaier et al. [15]	✓	✓	✓	✓	✓	✓	✓
This Work (EAH-FL)	✓	✓	✓	✓	✓	✓	✓

Despite these contributions, healthcare-specific challenges remain. Wearable devices often collect highly heterogeneous and sometimes class-imbalanced data. One common scenario is the generation of one-class datasets, such as only normal or only abnormal sam-

ples, which destabilize global model training. Lee et al. [3] showed that FedAvg is relatively robust across diverse settings, but Yang et al. [2] demonstrated that additional mechanisms are still required for handling extreme heterogeneity in medical imaging. This confirms that healthcare IoT FL must simultaneously account for data imbalance, device constraints, and cryptographic overhead. The literature indicates that while progress has been made in both privacy and efficiency, there remains a clear gap in unifying the two. Collectively, these studies affirm that while significant progress has been made in algorithmic privacy and network optimization, a unified framework capable of delivering both end-to-end encryption and edge-level efficiency remains underexplored. To address this, we propose EAH-FL, which integrates edge-assisted HE with in-network aggregation. Unlike prior INC-based systems, EAH-FL ensures that all model updates remain encrypted throughout the network, eliminating the risks of plaintext aggregation. Unlike HE-only systems, it reduces prohibitive latency and bandwidth overhead through selective encryption and network-side aggregation. By offloading heavy cryptographic tasks to trusted edge devices, EAH-FL makes homomorphic FL feasible for resource-constrained wearables and demonstrates that secure, low-latency FL is achievable in real-time healthcare monitoring.

3. System Model

This section provides a formal description of the proposed secure, intelligent multi-stage pipeline. We define the network setup with all participating entities, state the underlying system-level assumptions (security, communication, resource, and trust), introduce the mathematical model (variables, equations, and constraints) that will be used later in the algorithmic implementation. This framework establishes the foundation for the algorithms and analyses in subsequent sections.

Figure 1 illustrates the proposed encrypted federated learning (FL) system with in-network aggregation, mapping directly onto the four-tier architecture described above. On the **left**, the client tier ($\mathcal{C}$) is represented by smartwatches, which *sense physiological data* such as heart rate, oxygen saturation, and activity levels. These clients transmit their raw measurements to the associated **edge nodes** ($\mathcal{E}$), shown as home routers. Each edge node performs fully homomorphic encryption (FHE) on the received data, ensuring that all information leaves the local domain in encrypted form.

The encrypted updates are then forwarded into the **INC layer** ($\mathcal{I}$), depicted as a network of interconnected smart switches and aggregator nodes. Unlike conventional forwarding, this layer performs two key roles: (1) *dynamic routing*, guided by a GNN-assisted policy that adapts paths based on network conditions, and (2) *in-network aggregation*, where encrypted updates from multiple edges are combined securely within the network fabric. Finally, on the **right**, the aggregated ciphertexts reach the **cloud server** ($\mathcal{H}$), which updates the global FL model without decrypting individual client updates. The cloud performs encrypted aggregation across all participants, producing an updated global model. This updated model remains encrypted and is transmitted back downstream through the INC and edge nodes, eventually reaching the clients for local use. In the diagram, the encryption of updates is represented with *lock symbols*, highlighting that sensitive information is always protected. Multiple locks indicate contributions from different clients being aggregated securely. The arrows capture this end-to-end pipeline, showing the progression from data sensing at the client to encrypted global model updates, while emphasizing the points where security (FHE encryption) and intelligence (GNN-driven INC routing and aggregation) intervene in the workflow.

While Figure 1 provides a high-level view of the data flow for readability, the detailed technical realization is captured below in the following sections.

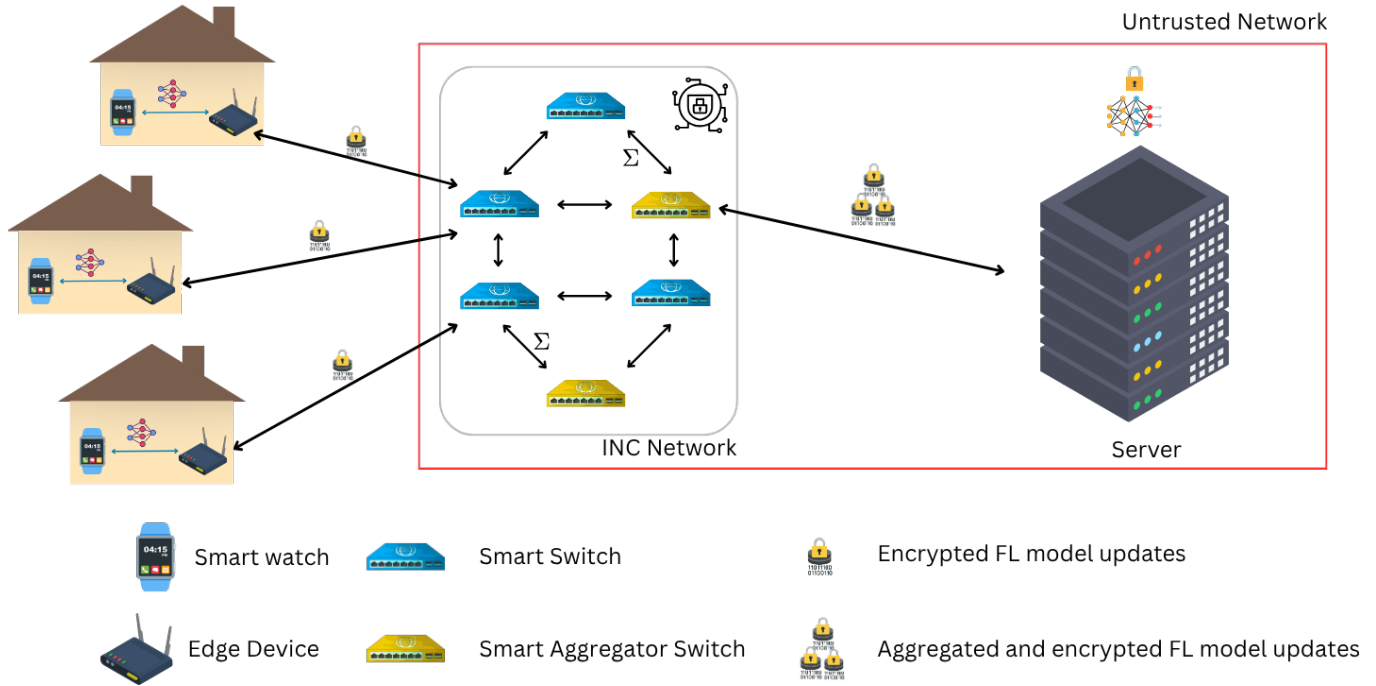


Figure 1. Proposed encrypted FL system with in-network aggregation.

3.1. Network Setup and Entities

We consider a hierarchical, multi-layer edge computing architecture that integrates clients, edge nodes, INC, and cloud servers, each with distinct roles and responsibilities. Such multi-tier designs align with the fog computing paradigm, where an intermediate “fog” layer exists between the edge and the cloud in larger deployments. Formally, let $\mathcal{C}$, $\mathcal{E}$, $\mathcal{I}$, and $\mathcal{H}$ denote the sets of clients, edge nodes, INC nodes, and cloud servers (respectively). For example, $\mathcal{C} = c_1, \dots, c_N$ is the set of N client devices, and similarly $\mathcal{E} = e_1, \dots, e_M$, $\mathcal{I} = i_1, \dots, i_K$, and $\mathcal{H} = h_1, \dots, h_L$ (often $L = 1$ if a single cloud service is considered). We define a mapping $f : \mathcal{C} \rightarrow \mathcal{E}$ that associates each client c_i with an edge node $e_{f(i)}$ responsible for that client (for instance, $f(i) = j$ if client c_i connects to edge e_j). Typically, this mapping reflects network proximity or pre-established pairing (e.g., a client’s personal or local gateway). Similarly, edge nodes are connected through the network to the cloud, possibly via multiple INC nodes. We may represent the network beyond the edge as a graph $G = (V, E)$ where $V = \mathcal{E} \cup \mathcal{I} \cup \mathcal{H}$ and E are communication links; each edge node e_j has one or more routing paths through INC nodes to reach the cloud $h \in \mathcal{H}$. For simplicity, we often consider a representative path $P_j = [e_j \rightarrow i_a \rightarrow \dots \rightarrow i_b \rightarrow h]$ for each edge e_j , where $i_a, \dots, i_b \in \mathcal{I}$ are the intermediate nodes on that route, terminating at the cloud h .

Clients ($\mathcal{C}$)—The client layer consists of end devices, in our case a smartwatch, that generate data and service requests at the network edge. Clients are resource-constrained in computation, memory, and energy. They may perform minimal local processing (e.g., sampling or simple preprocessing) but offload heavy tasks to the edge node. In our pipeline, a client mainly collects raw data (heart rate, oxygen saturation, blood pressure, activity) and securely transmits it to its edge node. Clients operate at the first tier, typically connecting via Bluetooth, Wi-Fi, or cellular. A generic client is denoted as $c_i \in \mathcal{C}$, with D_i representing the data size (or task) generated. Due to limited capabilities, clients cannot handle intensive tasks (e.g., large-scale analytics or strong encryption) and thus rely on the edge node.

Edge Nodes ($\mathcal{E}$)—Edge nodes are intermediate servers or gateways close to clients (e.g., home routers). Each client c_i is paired with an edge node $e_{f(i)}$ that serves as its immediate processor and gateway. They bridge resource-limited clients and the core network by

handling offloaded tasks such as data aggregation, format conversion, and cryptographic operations. Edge nodes also manage communication (e.g., packet encapsulation) and act as the first hop for data leaving the local domain. We denote C_e as the compute capacity of edge node e (CPU cycles/s) and $B_{c,e}$ as the bandwidth of the client–edge link. Practically, edge nodes (micro-servers or routers) have more resources than clients but less than the cloud. Processing at the edge reduces latency and network load, as only processed or compressed data may be forwarded. Clients typically trust their edge node, forming a local trust domain.

INC Layer ($\mathcal{I}$)—INC nodes are network devices (e.g., smart routers, switches) with compute and storage capabilities, enabling in-network processing beyond simple forwarding. In our pipeline, they can aggregate results from edges, filter or fuse streams, and execute lightweight functions to reduce data volume before reaching the cloud. They also support dynamic routing and load balancing based on network conditions or policies. The INC layer (third tier) thus acts as a fog layer, offloading some cloud tasks and improving responsiveness by leveraging backbone resources. We denote the capacity of an INC node $i_k \in \mathcal{I}$ as C_{i_k} , and bandwidths on edge–INC and INC–cloud links as B_{e,i_k} and $B_{i_k,h}$, respectively. While multiple INC nodes may exist along a path, we often treat the INC fabric as a single intermediate layer in analyses of delay or security. Operated by providers or third parties, INC nodes are modeled as semi-trusted (“semi-honest”) in our security framework. They remain central to the pipeline’s intelligence, enabling adaptive strategies (including AI-based controllers) for routing and aggregation.

Cloud ($\mathcal{H}$)—The cloud layer (fourth tier) consists of centralized high-performance servers with virtually unbounded computation and storage. It executes global or compute-intensive tasks that exceed the edge or INC layers, such as aggregating results, running large-scale analytics, and finalizing model training. In our pipeline, the cloud acts as the global aggregator and back-end processor, collecting encrypted updates from distributed clients, combining or analyzing them, and producing final outputs or updated models. Results (e.g., inferences, decisions, or refined models) may then be sent back to edges and clients. We model the cloud as honest-but-curious: it follows the protocol but is not trusted with raw data, which arrives encrypted or sanitized. The cloud’s compute capacity is denoted C_h (treated as unbounded), and its link bandwidth from INC or edge nodes as $B_{*,h}$. In short, the cloud provides centralized power and storage at the cost of higher latency and weaker trust.

These four entities form a pipeline: a client’s data or request flows upward through edge and INC nodes to the cloud, undergoing processing at each stage, and results flow back down to the client. By organizing the system into these layers, we achieve low-latency local processing at the edge, intelligent mid-layer optimization in the network, and heavy-duty processing plus global coordination at the cloud. Next, we outline the key assumptions about the environment, security, and trust in which this multi-stage pipeline operates.

3.2. Underlying Assumptions

Designing a secure, distributed pipeline requires several system-level assumptions to be established. We explicitly state the assumptions regarding the network environment, security model, resource limitations, trust relationships, and threat model. These assumptions are chosen to be realistic and to ensure a safe foundation for our system (they are not overly idealized), while avoiding any speculative or hypothetical results. The following are the principal assumptions underpinning our architecture:

1. **Communication Model:** We assume that clients connect to edge nodes over wireless links (e.g., a wearable device to a smartphone or home gateway via Bluetooth or Wifi). The edge nodes and INCs are interconnected through the broader network (which

could include the internet or a private WAN). Standard networking protocols ensure basic data delivery between layers. However, network communication is not perfectly reliable—latency and packet loss can occur. We assume a packet-switched network where messages may be delayed, dropped, or reordered by adversarial conditions. Despite this, mechanisms like acknowledgments and retransmissions (or higher-level protocols) can provide eventual reliability. Time is divided into discrete intervals or rounds when analyzing performance (especially if considering synchronized operations like training rounds), but the system can also handle asynchronous requests. We also assume each link has a finite bandwidth $B_{x,y}$ and propagation delay; thus large messages incur transmission delays as introduced in the mathematical model later in this section.

2. **Resource Constraints:** Each tier has limited resources relative to its workload. Client devices have severe constraints on battery, CPU, memory, and thus cannot perform intensive computations or continuous heavy communication. This drives the need to offload tasks to the edge. Edge nodes have more computational power and energy (mains-powered), but they are still limited (e.g., a microserver with fixed CPU/GPU capacity that must be shared among multiple clients). INC nodes likewise have finite processing and storage—while they might be powerful network devices, they are not full data-center servers and typically must forward many flows, so any INC must be lightweight. The cloud has abundant resources but not infinite; in practice, we assume the cloud can scale to handle the aggregated workload of all clients, albeit with some upper bound or cost. We assume that no single edge or INC can handle an unbounded number of clients simultaneously without performance degradation; therefore, tasks may need to be distributed or queued if capacity is exceeded. In formulating our model, we will impose capacity constraints (e.g., an edge node e_j cannot exceed C_{e_j} CPU cycles per second across all tasks, and similarly for INC nodes). We also assume that storage at each layer is sufficient for buffering and intermediate data, but edge/INC storage is not large enough to hold full global datasets or models—such large-scale data resides in the cloud. This multi-tier resource distribution reflects common practice in edge computing, where moving computation closer to data sources improves latency but each step closer to the edge has smaller capacity.
3. **Security and Trust Boundaries:** We divide the system into trust domains. The client and its edge node form a fully trusted domain. Beyond this, the network—including INC nodes and links—is untrusted or semi-trusted. INC nodes are semi-trusted: they execute assigned tasks (e.g., aggregation, routing) correctly but may be curious or compromised. They do not intentionally corrupt computations but cannot be trusted with confidential data. The cloud is modeled as honest-but-curious: it follows protocols correctly but may analyze data to infer private information, so raw client data must not be exposed. Communication channels are fully untrusted, subject to eavesdropping, tampering, and replay attacks. To ensure confidentiality and integrity, all data leaving the edge is encrypted end-to-end with CKKS (FHE), and entities are authenticated (via certificates or pre-shared keys). Within the client–edge domain, plaintext handling is permitted under the assumption of physical security. In summary: the client–edge pair is trusted, INC is semi-trusted, the cloud is honest-but-curious, and network channels are untrusted. Threats include passive adversaries, compromised INC nodes, and packet injection or alteration. Standard cryptographic primitives are assumed secure. Denial-of-service attacks are possible but out of scope; we focus on confidentiality, integrity, and correct protocol execution.
4. **Task and Application Model:** We assume that the workload can be broken into tasks or data units that flow through the pipeline. Each client may generate tasks

(or data updates) periodically or in response to events. We make the simplifying assumption that each task from a client is independent in terms of scheduling (though they could be part of a larger application workflow). The pipeline supports a variety of applications, for example, real-time sensor data aggregation, federated machine learning updates, augmented reality offloading, etc.—but in all cases, the pattern is that raw data originates at the client and the final processing is needed at the cloud or at least beyond the edge. The communication pattern is often uplink-heavy: clients send data up the chain and eventually receive some result or acknowledgment back. We assume the volume of data returned from the cloud (downlink) is relatively small (an updated model) compared to the volume of data uploaded, which is common in sensor analytics and learning scenarios. This justifies our focus on the upstream data pipeline for performance modeling. If a specific application requires heavy downlink data (e.g., content delivery), a similar analysis can be applied in reverse. Another assumption is that tasks might have real-time requirements—for instance, each task i may have a deadline or latency requirement Δ_i that the end-to-end processing must meet (especially in mission-critical contexts). We assume such requirements are known and form part of the constraints in our system design (e.g., the pipeline and algorithms should aim to ensure $T_i \leq \Delta_i$ for each task's completion time T_i as defined later).

5. **Consistency and Fault Tolerance:** We assume a consistent view of the system in terms of configuration, e.g., the mapping $f(c_i) = e_j$ does not change frequently (a client is generally served by the same edge node, unless it moves and hands off to a new edge, which we assume happens relatively infrequently). INC routing paths are assumed to be known or can be discovered by the network controllers. If a node fails (edge, INC, or cloud), we assume there are failover mechanisms (out of scope for our model) that eventually reroute tasks to an alternate node of the same tier. Our focus is on steady-state operation under normal conditions and under security threats, rather than on recovery from node crashes. However, we do assume that the system is distributed—no single point of failure should halt the entire pipeline. For example, if one INC node on a path is down, another INC or an alternate route can be used (this will be orchestrated by the network's intelligent Routing Algorithm discussed in Section 5).

With these assumptions in place, our pipeline is designed to function both securely and efficiently. We now turn to establishing the foundation for the encryption used in our model.

3.3. Homomorphic Encryption Setup

We use FHE with the CKKS scheme as implemented in TenSEAL to enable approximate arithmetic directly over encrypted floating-point values. This choice allows model parameters and health data streams to remain encrypted end-to-end while still supporting aggregation and statistical operations required for real-time analysis. Furthermore, to ensure strict reproducibility, a global seed $s = 42$ is applied consistently across model initialization, data partitioning, and training routines. This guarantees deterministic experimental outcomes.

3.3.1. Cryptographic Context

The encryption context is defined by the tuple

$$\mathcal{P} = (N, \Delta, \mathcal{Q}) \quad (1)$$

where

- **Polynomial modulus degree:** $N = 8192$
 - Governs ciphertext size and security strength.
 - Provides 128-bit security against RLWE attacks.
 - Large enough to encode multiple model parameters in a single ciphertext (via SIMD packing), which is essential for vectorized aggregation in INC.
 - Smaller values ($N = 4096$) reduce latency but fall short of the security level required; larger values ($N = 16,384$) increase computational overhead beyond real-time feasibility.
- **Scaling factor:** $\Delta = 2^{20}$
 - Controls fixed-point precision of encrypted computations.
 - A scaling factor of 2^{20} yields 6 decimal digits of accuracy, sufficient for model parameter aggregation where minor floating-point deviations do not affect convergence.
 - Higher scaling factors increase precision but lead to faster noise growth, limiting computation depth.
- **Coefficient modulus chain:** $\mathcal{Q} = [40, 40, 40, 40]$
 - Defines the sequence of modulus sizes for rescaling.
 - This chain supports multiple levels of additions and multiplications before noise exhaustion, while keeping ciphertext size manageable.
 - The symmetric structure $[40, 40, 40, 40]$ provides balanced noise distribution across rescaling steps, enabling secure aggregation of parameters through several INC hops.

Formally, the plaintext encoding is defined as:

$$\mathbb{C}^{N/2} \xrightarrow{\text{Encode}} R_N = \mathbb{Z}[X]/(X^N + 1), \quad (2)$$

where $\mathbb{Z}[X]$ denotes the ring of polynomials in X with integer coefficients, and N is the polynomial modulus degree. The quotient $(X^N + 1)$ defines the cyclotomic polynomial used to bound polynomial degrees.

3.3.2. Secure Dataflow in the Pipeline

All sensor data is encrypted once at the router using the CKKS context $\mathcal{P}(1)$. From this point forward, ciphertexts are never decrypted until they are relayed back to the client device. Within the INC fabric, smart switches and intermediate nodes operate only over ciphertexts, performing homomorphic additions and multiplications such as

$$\text{Enc}(\theta) \mapsto \sum_{i=1}^m \text{Enc}(\theta_i), \quad (3)$$

where $\text{Enc}(\theta_i)$ are encrypted parameter vectors aggregated hop-by-hop. Because CKKS supports (Single Instruction Multiple Data) SIMD style packing, multiple parameters can be aggregated in parallel, enabling efficient streaming across IoT topologies.

Next, we formalize the mathematical model of the pipeline, introducing variables and equations that characterize communication delays, processing times, resource constraints, and the objective functions that drive the intelligent orchestration of this multi-stage system.

3.4. Mathematical Model

Before presenting the formal mathematical model, it is important to highlight that our system integrates two distinct, yet complementary, computational domains. The first domain is the *federated learning (FL) framework*, which governs how model updates

are generated on end devices and securely aggregated through the pipeline up to the cloud. The second domain is the *graph neural network (GNN)-based routing framework*, which dynamically determines optimal data forwarding and aggregation strategies within the INCs. This division allows us to gradually introduce the notation and variables specific to each domain while ensuring a coherent integration of both perspectives in the overall system model.

3.4.1. Federated Learning Framework

We consider a hierarchical federated learning pipeline where updates flow from clients C through edges E and INC nodes I to the cloud H .

At the client layer, each $c_i \in C$ holds a private dataset

$$D_i = \{(x_{ij}, y_{ij}) \mid j = 1, \dots, n_i\}, \quad (4)$$

with $x_{ij} \in \mathbb{R}^d$ features and $y_{ij} \in \{0, 1\}$ labels. The local model is parameterized by $\theta = (w, b)$, where w is the weights and b is the bias. In each round t the client produces parameters θ_i^t with a paired edge $e_{f(i)} \in E$. The edge initializes the CKKS context $\mathcal{P}(1)$, generates keys (pk – *publickey*, sk – *secretkey*), and encrypts the local parameters:

$$[[\theta_i^t]] = \text{Enc}(\theta_i^t; pk, \mathcal{P}). \quad (5)$$

From this step onward, all communication is encrypted.

The encrypted updates traverse paths $P_j = [e_j \rightarrow i_a \rightarrow \dots \rightarrow i_b \rightarrow h]$ through the INC layer (the path is decided by a dynamic machine learning algorithm that we discuss in the next section), where $i_a, \dots, i_b \in I$ and $h \in H$. INC nodes perform homomorphic aggregation without decryption:

$$[[\theta^t]] = \sum_{i \in R_t} \frac{n_i}{N_t} \cdot [[\theta_i^t]], \quad (6)$$

where $R_t \subseteq C$ are the participating clients in round t and $N_t = \sum_{i \in R_t} n_i$.

At the cloud server $h \in H$, the aggregated ciphertext updates the global FL model while it is encrypted:

$$[[\theta^{t+1}]] = \text{UpdateEncrypted}([[\theta^t]]). \quad (7)$$

The newly trained global model is then returned to the edge $e_{f(i)}$, which decrypts with sk :

$$\theta^{t+1} = \text{Dec}([[\theta^{t+1}]]; sk), \quad (8)$$

and synchronizes plaintext global parameters (W^t, b^t) back to clients.

Thus, the mathematical flow of federated learning can be summarized as:

$$\theta_i^t \xrightarrow{e_{f(i)}} [[\theta_i^t]] \xrightarrow{I} [[\theta^t]] \xrightarrow{h} [[\theta^{t+1}]] \xrightarrow{e_{f(i)}} \theta^{t+1}. \quad (9)$$

This formulation establishes the core mathematical model and end-to-end data flow for our federated learning loop under HE. In Section 4, we extend this baseline into our proposed EAH-FL framework, where the same primitives are embedded into a structured algorithm. There, we formalize the update procedure, introduce edge-assisted coordination mechanisms, and analyze how these refinements improve efficiency, scalability, and security while maintaining full encryption throughout the pipeline.

3.4.2. Graph-Based Representation for Adaptive Routing

While the FL framework ensures end-to-end confidentiality through HE, the underlying communication layer must also operate efficiently under limited resources. FL with HE increases computational and transmission overhead, making intelligent routing essential especially for INC nodes and switches that must dynamically forward encrypted packets without excessive delay or packet loss. Routing in such an environment can be naturally formulated as a graph learning problem. Each node's forwarding decision depends not only on its own state but also on the states of its neighboring nodes. This dependency structure aligns with the message-passing mechanism of GNNs, which aggregate neighborhood information to learn context-aware node representations useful for routing.

To formalize this, we model the INC fabric as a directed graph

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \quad |\mathcal{V}| = N, \quad (10)$$

where $\mathcal{V}$ is the set of network nodes and $\mathcal{E}$ the set of directed communication links. Each node $u \in \mathcal{V}$ is associated with a feature vector

$$z_u = [E_u, L_u, T_u, B_u], \quad (11)$$

capturing residual energy (E_u), connectivity degree (L_u), trust score (T_u), and buffer occupancy (B_u). Collectively, these features form the matrix

$$Z \in \mathbb{R}^{N \times d}, \quad (12)$$

where d denotes the dimensionality of node features.

A generic K -layer Graph Neural Network then maps the pair $(Z, \mathbf{A})$ to node embeddings through iterative message passing. Each layer updates node representations as

$$H^{(k+1)} = \sigma(\mathbf{A}H^{(k)}W^{(k)}), \quad (13)$$

where $\mathbf{A}$ is the normalized adjacency matrix of $\mathcal{G}$, $W^{(k)}$ denotes the trainable weight matrix of layer k , and $\sigma(\cdot)$ is a nonlinear activation such as ReLU. This operation aggregates information from each node's neighbors and transforms it into a higher-level representation capturing both local and topological context. After K iterations, the final embeddings are expressed as

$$H^{(K)} = \text{GNN}_K(Z, \mathbf{A}). \quad (14)$$

The resulting embeddings $H^{(K)}$ encapsulate each node's structural and functional context and are subsequently used to compute adaptive routing scores or next-hop probabilities. This formulation forms the foundation of the Adaptive Multi-Relational Routing Graph Neural Network (AMR-GNN) introduced in Section 5.

The mathematical framework in Section 3 integrates federated learning, homomorphic encryption, and graph-based routing into a unified system model. It defines the operational roles and constraints of clients, edges, INC nodes, and the cloud, establishing the foundation for end-to-end secure and efficient communication. Section 4 expands this model into the Edge-Assisted Homomorphic Federated Learning (EAH-FL) framework, while Section 5 extends it further by embedding the GNN-based routing mechanism, demonstrating through experiments that the proposed system achieves both privacy preservation and efficient in-network operation.

4. Edge-Assisted Homomorphic Federated Learning

Building directly on the system model and cryptographic foundations introduced in Section 3, we now formalize the federated learning protocol that enables privacy-preserving model training over wearable health data. The focus of this section is to connect the architectural assumptions of Section 3 trusted edge devices, programmable in-network aggregation, and CKKS-based encryption to the end-to-end training pipeline and its practical realization.

In particular, we detail three key aspects of the proposed EAH-FL design:

1. The complete encrypted training pipeline that operates across clients, edges, and the cloud (Section 4.1),
2. Stabilization strategies for one-class clients to ensure convergence in heterogeneous data settings (Section 4.2)

Throughout this section, we retain the notation of Section 3: a federation of n clients indexed by i , each with dataset D_i of size n_i , and global parameters $\theta^t = (W^t, b^t)$ at round t . Ciphertexts are denoted by $[[\cdot]]$ under the CKKS scheme with public/private keys (pk, sk) generated at the edge. The cloud remains honest-but-curious and never holds sk , ensuring that aggregation and global updates are performed exclusively in the encrypted domain.

4.1. Complete Training Pipeline

Algorithm 1 presents the full training workflow of the proposed EAH-FL framework. The system operates over T federated communication rounds, indexed by $t = 1, \dots, T$. At the start, a global model is initialized at the edge using logistic regression parameters (W_0, b_0) , and the CKKS encryption context is configured (e.g., polynomial modulus degree, scaling factor). The edge device also generates a public/private key pair (pk, sk) for encryption/decryption. In each round t , the following steps are executed, corresponding to the loop in Algorithm 1:

1. **Client-side local training:** Each client i loads and preprocesses its private dataset D_i . The clients are trained on 55 engineered features using logistic regression. If the client exhibits one-class behavior (only positive or only negative samples), one-class handling strategies are applied (see line 6 of Algorithm 1). Clients then perform local training to obtain (w_i^t, b_i^t) , i.e.,

$$\theta_i^t = [w_i^t, b_i^t]. \quad (15)$$

2. **Edge-side encryption:** The plaintext model parameters θ_i^t in (15) are sent to the edge device, which encrypts them under the public key pk and context $\mathcal{P}$ (1) to produce

$$[[\theta_i^t]] = \text{Enc}(\theta_i^t; pk; \mathcal{P}). \quad (16)$$

this is similar to Equation (5) we discussed in the mathematical model

3. **INC-side encrypted aggregation:** The encrypted updates from all clients are forwarded to an INC node (see line 18 of Algorithm 1). The INC node performs homomorphic aggregation to produce

$$[[\theta^t]] = \sum_{i \in R_t} \left([[\theta_i^t]] \cdot \frac{n_i}{N_{\text{tot}}} \right), \quad (17)$$

where $N_{\text{tot}} = \sum_{i \in R_t} n_i$.

4. **Cloud-side encrypted global update:** The aggregated ciphertext $[[\theta^t]]$ from (17) is relayed to the cloud, which completes the global model update in the encrypted domain to form

$$[[W^t, b^t]]. \quad (18)$$

5. **Edge-side decryption and client sync:** Each edge decrypts the encrypted global model (18) using its private key sk , recovering the plaintext parameters

$$(W^t, b^t) = \text{Dec}([W^t, b^t]; sk), \quad (19)$$

which are then synchronized to clients for the next round (see line 26 of Algorithm 1).

Algorithm 1: EAH-FL

Input: Client datasets $D_1, \dots, D_n$, rounds T , FHE parameters
Output: Global model parameters W^*, b^*
 $W^0, b^0 \leftarrow \text{InitializeGlobalModel}()$
 $\text{context} \leftarrow \text{InitializeFHEContext}(\text{poly_modulus_degree} = 8192, \text{coeff_mod_bit_sizes} = [40, 40, 40, 40], \text{scale_bits} = 40)$
 $(pk, sk) \leftarrow \text{GenerateKeyPair}(\text{context})$
for $t \leftarrow 1$ **to** T **do**
 for $i \leftarrow 1$ **to** n **do**
 $(X_i, y_i) \leftarrow \text{LoadAndPreprocessData}(D_i)$
 if $\text{IsOneClassClient}(y_i)$ **then**
 // stabilize client with one-class data
 $(X_i, y_i) \leftarrow \text{ApplyOneClassHandling}(X_i, y_i, W^{t-1}, b^{t-1})$
 // local logistic regression training
 $(w_i^t, b_i^t) \leftarrow \text{TrainLocalModel}(X_i, y_i)$
 for $i \leftarrow 1$ **to** n **do**
 // Equation (15)
 $\theta_i^t \leftarrow [w_i^t, b_i^t]$
 // Equation (16) edge performs CKKS encryption
 $[[\theta_i^t]] \leftarrow \text{Enc}(\theta_i^t, pk)$
 // encrypted updates with sample count metadata sent
 $\text{SendToINCNode}([[\theta_i^t]], n_i, t, i)$
 $R_t \leftarrow \text{ReceiveAtINC}(t)$
 $N_{\text{tot}} \leftarrow \sum_{i \in R_t} n_i$
 $[[\theta^t]] \leftarrow \sum_{i \in R_t} ([[\theta_i^t]] \times \frac{n_i}{N_{\text{tot}}})$
 // Equation (17)
 $\text{SendToCloudAggregator}([[\theta^t]])$
 $[[W^t]], [[b^t]] \leftarrow \text{GlobalUpdateEncrypted}([[\theta^t]])$
 // Equation (18)
 for $i \leftarrow 1$ **to** n **do**
 $[[W^t]], [[b^t]] \leftarrow \text{ReceiveEncryptedGlobalModel}()$
 $(W^t, b^t) \leftarrow \text{Dec}([W^t], [b^t], sk)$
 // Equation (19)
 $\text{SendToClient}(W^t, b^t, \text{client_id} = i)$
return $\text{Dec}([W^T], sk), \text{Dec}([b^T], sk)$

Equations (15)–(19) correspond directly to the steps of Algorithm 1. This design ensures that client updates remain confidential throughout the pipeline. At no point does the cloud observe raw model weights or gradients. The trusted edge offloads cryptographic computation from resource-constrained devices, while the INC layer reduces WAN traffic via ciphertext fan-in, enabling practical and secure FL in healthcare IoT environments.

4.2. One-Class Client Handling

A key challenge in healthcare FL is clients with data from only one class (e.g., no positive cases). These clients can destabilize global model training if not properly handled. EAH-FL includes a dedicated mechanism for this scenario (Algorithm 2).

Algorithm 2: One-Class Client Handling (Combined Strategy)

```

Input: Client data  $(X, y)$ , global model  $(W^{t-1}, b^{t-1})$ 
Output: Trained parameters  $(w, b)$ 
if CountUniqueClasses( $y$ ) = 1 then
    // One-class client detected
     $k \leftarrow \text{GetUniqueClass}(y)$ 
    // Laplace smoothing-Equation (20)
     $n_{\text{virt}} \leftarrow \max(1, \lfloor 0.1 \times |y| \rfloor)$ 
     $\tilde{X} \sim \mathcal{N}(\mathbf{0}, 0.1^2)$  with  $n_{\text{virt}} \times d$  samples
    // Augment dataset-Equation (21)
     $(X_{\text{aug}}, y_{\text{aug}}) \leftarrow \text{form via (21)}$ 
    // Warm start with L2 regularization-Equations (22) and (23)
    if global model  $(W^{t-1}, b^{t-1})$  available then
         $\theta_{\text{init}} \leftarrow (W^{t-1}, b^{t-1})$  // Equation (22)
         $\text{model} \leftarrow \text{LogisticRegression}(C = 100, \text{solver} = \text{liblinear}, \text{max\_iter} = 5000)$ 
         $\text{model.coef\_} \leftarrow W^{t-1}$ 
         $\text{model.intercept\_} \leftarrow b^{t-1}$ 
         $\text{model.classes\_} \leftarrow [0, 1]$ 
         $\text{model.fit}(X_{\text{aug}}, y_{\text{aug}})$  // L2 regularization via  $C = 100$ 
    else
         $\text{model} \leftarrow \text{LogisticRegression}(\text{solver} = \text{liblinear}, \text{max\_iter} = 5000)$ 
         $\text{model.fit}(X_{\text{aug}}, y_{\text{aug}})$ 
    else
        // Multi-class client - normal training with class balancing
         $\text{model} \leftarrow \text{LogisticRegression}(\text{solver} = \text{lbfgs}, C = 100, \text{class\_weight} = \text{balanced}, \text{max\_iter} = 5000)$ 
         $\text{model.fit}(X, y)$ 
return  $\text{model.coef\_flatten}(), \text{model.intercept\_}[0]$ 

```

Problem Setup: For client i , let the local dataset be

$$\mathcal{D}_i = \{(x_j, y_j)\}_{j=1}^{n_i}, \quad y_j \in \{0, 1\},$$

where n_i is the number of samples. If all labels are equal to some class $k \in \{0, 1\}$ (i.e., $\{y_j : j = 1, \dots, n_i\} = \{k\}$), we say client i has a “one-class problem”. We denote the missing class by $1 - k$.

Laplace Smoothing: To address the one-class issue, we generate synthetic samples for the missing class. - Let $\alpha > 0$ be a smoothing parameter (we use $\alpha = 0.1$ in our implementation). - The number of synthetic samples is

$$n_{\text{virt}} = \max(1, \lfloor \alpha n_i \rfloor). \quad (20)$$

- Let $\tilde{X} \sim \mathcal{N}(\mathbf{0}, 0.1^2)$ be $n_{\text{virt}} \times d$ Gaussian random features for class $1 - k$. - The augmented dataset is

$$X^{\text{aug}} = \begin{bmatrix} X \\ \tilde{X} \end{bmatrix}, \quad y^{\text{aug}} = \begin{bmatrix} y \\ (1-k)\mathbf{1}_{n_{\text{virt}}} \end{bmatrix}, \quad n_i^{\text{aug}} = n_i + n_{\text{virt}}. \quad (21)$$

Warm-Start Initialization with L2 Regularization: For one-class clients, we combine Laplace smoothing with warm-start initialization and L2 regularization:

$$\theta_{\text{init}} = \theta^{t-1} = (W^{t-1}, b^{t-1}). \quad (22)$$

We train on the augmented dataset $(X^{\text{aug}}, y^{\text{aug}})$ with L2 regularization:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{data}}(\theta; X^{\text{aug}}, y^{\text{aug}}) + \lambda \|\theta\|_2^2, \quad (23)$$

where the regularization coefficient, denoted as λ , is set to 0.01. In logistic regression, the regularization strength is governed by the parameter C , calculated as $C = 1/\lambda$, which results in a value of 100. This methodology enhances the stability of local updates by commencing from the parameters of the global model and incorporating L2 regularization throughout the training process. This approach effectively mitigates excessive divergence from the global parameters, while simultaneously allowing the model to leverage knowledge from the augmented data.

5. Adaptive Multi-Relational Routing Graph Neural Network for Smart Networks

After establishing the secure transmission layer through HE and refining local models within the federated learning loop, the next challenge lies in capturing the behavior of the underlying INC layer. Since the efficiency of the overall system depends not only on data confidentiality but also on how packets traverse and aggregate across the network, we now turn to the problem of routing. To address this, we introduce the AMR-GNN, which adapts decisions to evolving network states. We then complement this with a heuristic-imitation framework, where the GNN is trained to replicate the choices of a rule-based policy. Together, these two perspectives allow us to contrast adaptive learning with heuristic guidance, and provide a foundation for evaluating the trade-offs between stability, interpretability, and performance in our INC setting.

5.1. Dynamic Routing

Building on the graph-based routing model defined in Section 3.4.2, we now specialize the GNN architecture to the requirements of INC under homomorphic federated learning. While the generic formulation $H^{(K)} = \text{GNN}^{(K)}(Z, A)$ yields structural embeddings of the INC fabric, it does not capture the heterogeneous semantics of nodes and links in this setting. Routing decisions must jointly account for residual energy, encryption overhead, latency, and trust, while edges must reflect variable bandwidth and reliability.

For each node u , the hidden representation at layer k is updated as:

$$h_u^{(k)} = \sigma \left(W^{(k)} \cdot [h_u^{(k-1)} \parallel \text{AGG}\{h_v^{(k-1)} : v \in \mathcal{N}(u)\}] \right), \quad (24)$$

with $h_u^{(0)} = z_u$, $\sigma(\cdot)$ an activation function, and $\text{AGG}(\cdot)$ an aggregation operator. After K layers, the enriched embeddings $H^{(K)}$ encode both structural and contextual information, tailored for routing.

The routing policy is parameterized as:

$$\pi(a_u | h_u^{(K)}) = \text{softmax}(W_o h_u^{(K)}), \quad (25)$$

where $\pi(a_u | h_u^{(K)})$ denotes the probability of forwarding from node u to a neighbor a_u .

Finally the objective is to balance delivery, latency, energy, and trust. We capture this trade-off through a composite reward:

$$R = \alpha \cdot \text{PDR} - \beta \cdot \text{Delay} - \gamma \cdot \text{Energy} + \delta \cdot \text{Trust}, \quad (26)$$

with coefficients $\alpha, \beta, \gamma, \delta$ reflecting system priorities. This reward guides both training and runtime evaluation of AMR-GNN policies.

Algorithm 3 operationalizes the mathematical formulations in Equations (10)–(14) and (24)–(26) within the live INC fabric. Starting from the graph representation $G = (\mathcal{V}, \mathcal{E})$ (Equation (10)), each node $u \in \mathcal{V}$ constructs its feature vector $z_u = [E_u, L_u, T_u, B_u]$ (Equation (11)), capturing residual energy, connectivity degree, trust score, and buffer occupancy. These features are aggregated into the feature matrix $Z \in \mathbb{R}^{N \times d}$ (Equation (12)), which is then propagated through a K -layer GNN to produce enriched embeddings $H^{(K)} = \text{GNN}_K(Z, A)$ (Equation (14)). During propagation, hidden representations $h_u^{(k)}$ are updated layer by layer according to Equation (24), integrating both structural and contextual information from neighbors. The resulting embeddings parameterize the routing policy $\pi(a_u | h_u^{(K)}) = \text{softmax}(W_0 h_u^{(K)})$ (Equation (25)), which assigns forwarding probabilities over candidate neighbors. Instead of uniform neighbor selection, the algorithm ranks neighbors using logits from π and restricts forwarding to the top- K candidates, applying a loop penalty to previously visited nodes to avoid cycles. Final action selection is sampled from this top- K set with softmax temperature scaling. At each hop, encrypted updates are combined in-network using the homomorphic operator $\text{HE_SUM}(\cdot)$, enabling secure aggregation of ciphertexts before reaching the cloud. Routing is further guided by the composite reward $R = \alpha \cdot \text{PDR} - \beta \cdot \text{Delay} - \gamma \cdot \text{Energy} + \delta \cdot \text{Trust}$ (Equation (26)), balancing delivery, latency, energy efficiency, and trust in real time.

5.2. Heuristic-Imitation Training

When training a GNN to make forwarding decisions, relying only on reinforcement signs can be slow and unstable, since the model must explore before it discovers effective strategies. To accelerate this process, we let the network first imitate a simple heuristic: a rule-based neighbor selection that encodes intuitive preferences such as avoiding congested nodes, exploiting aggregation opportunities, and moving closer to the server. By imitating these choices, the GNN quickly learns useful Patterns that form a strong starting point for further optimization. In order to gain deeper insight into the local training dynamics of the GNN, we now introduce the corresponding mathematical equations with explanatory notes.

The queue length at a node reflects the number of packets currently waiting to be processed. To avoid biasing high-degree nodes, we normalize it by the degree of the node:

$$q_n(v) = \frac{\text{queue}[v]}{\max(1, \sum_j \text{adj}[v, j])} \quad (27)$$

where $\text{queue}[v]$ is the queue size at node v and $\sum_j \text{adj}[v, j]$ is its degree.

The aggregation potential quantifies how well a node can combine traffic from multiple neighbors. It counts the number of active neighbors with non-empty queues:

$$\text{aggr}(v) = \sum_{u: \text{adj}[v, u]=1} \mathbf{1}[\text{queue}[u] > 0] \quad (28)$$

The dynamic feature vector summarizes key local conditions of a node—its utilization, normalized queue, and aggregation capability:

$$\text{dyn}(v) = [\text{util}[v], q_n(v), \text{aggr}(v)] \quad (29)$$

Algorithm 3: INC Round: Dynamic Routing and Homomorphic Aggregation

Input : Graph $G = (\mathcal{V}, \mathcal{E})$ with adjacency adj , relations rel , positions pos ;
 Node feature matrix Z ; server t ; trained encoder f_θ ;
 telemetry providers $\text{util}()$, $\text{queue}()$; client IDs $\mathcal{C}$ and map $\text{map}(\cdot)$;
 encrypted updates $\{u_c\}$, sample counts $\{n_c\}$; homomorphic sum
 $\text{HE_SUM}(\cdot)$;
 hyperparams: K (top- K), temperature τ , max_hops , loop_penalty

Output: Aggregated update U

// Step 1: Graph definition $G = (\mathcal{V}, \mathcal{E})$ (Equation (10))
 $\text{util} \leftarrow \text{util}()$, $\text{queue} \leftarrow \text{queue}()$;
 // Step 2: Encode feature vectors $z_u = [E_u, L_u, T_u, B_u]$ (Equation (11))
 $z \leftarrow f_\theta.\text{encode}(Z, \text{adj}, \text{rel})$;
 // Step 3: Compute progress distance toward server
 $d2s(v) \leftarrow \|\text{pos}[v] - \text{pos}[t]\|_2, \forall v \in \mathcal{V}$;
 // Step 4: Build dynamic features (queue normalization, load, etc.)
for $v \in \mathcal{V}$ **do**
 $\text{deg} \leftarrow \max(1, \sum_j \text{adj}[v, j])$; $\text{queue_norm} \leftarrow \text{queue}[v] / \text{deg}$;
 $\text{aggr} \leftarrow \sum_{j: \text{adj}[v, j]=1} \mathbf{1}[\text{queue}[j] > 0]$;
 $\text{dyn}[v] \leftarrow [\text{util}[v], \text{queue_norm}, \text{aggr}]$;
 // Step 5: Routing loop, using GNN embeddings $H^{(K)} = \text{GNN}_K(Z, A)$
 (Equation (14))
 $\text{delivered} \leftarrow 0$, $\text{total_hops} \leftarrow 0$;
for $c \in \mathcal{C}$ **do**
 $s \leftarrow \text{map}(c)$; $P \leftarrow [s]$; $\text{visited} \leftarrow \{s\}$; $\text{cur} \leftarrow s$;
 for $h = 1$ **to** max_hops **do**
 $\mathcal{N}(\text{cur}) \leftarrow \{j \mid \text{adj}[\text{cur}, j] = 1\}$; **if** $\mathcal{N}(\text{cur}) = \emptyset$ **then**
 break
 // Step 6: Neighbor progress feature prog_j toward t
 $\text{prog}_j \leftarrow d2s(\text{cur}) - d2s(j), \forall j \in \mathcal{N}(\text{cur})$;
 // Step 7: Pairwise scoring using hidden states $h_u^{(k)}$
 (Equation (24))
 $\ell_j \leftarrow \text{PairwiseScorer}(z_{\text{cur}}, z_j, \text{dyn}_{\text{cur}}, \text{dyn}_j, \text{prog}_j)$;
 // Step 8: Penalize loops (routing cycles)
 for $j \in \mathcal{N}(\text{cur})$ **do**
 if $j \in \text{visited}$ **then**
 $\ell_j \leftarrow \ell_j - \text{loop_penalty}$
 // Step 9: Apply top- K selection + policy $\pi(a_u | h_u^{(K)})$
 (Equation (25))
 $S \leftarrow$ indices of top- K values in ℓ ; $p \leftarrow \text{softmax}(\ell_S / \tau)$;
 $j^* \leftarrow$ sample from S with probs p ;
 $P.\text{append}(j^*)$; $\text{visited} \leftarrow \text{visited} \cup \{j^*\}$; $\text{cur} \leftarrow j^*$;
 // Step 10: In-network secure aggregation $\text{HE_SUM}(\cdot)$
 $U \leftarrow \text{HE_SUM}(\{u_c\}_{c \in \mathcal{C}})$;
 // Step 11: Routing optimization guided by reward (Equation (26))
return U ;

The Euclidean distance to the server is used as a spatial progress measure, since nodes closer to the server are preferable for forwarding:

$$d2s(v) = \|\text{pos}[v] - \text{pos}[t]\|_2 \quad (30)$$

The relative progress of forwarding from node i to neighbor j is given by:

$$\text{prog}_{ij} = d2s(i) - d2s(j) \quad (31)$$

Positive values indicate advancement toward the server, while negative values correspond to detours.

The heuristic score evaluates each neighbor based on weighted factors of utilization, queue state, aggregation potential, and progress:

$$s_{ij} = -w_u \text{util}_j - w_q q_n(j) + w_a \text{aggr}(j) + w_p \text{prog}_{ij}, \quad j \in \mathcal{N}(i) \quad (32)$$

The heuristic label selects the neighbor with the maximum score, which acts as the imitation target for training:

$$y_i = \begin{cases} \arg \max_{j \in \mathcal{N}(i)} s_{ij}, & \text{if } i \neq t \text{ and } |\mathcal{N}(i)| > 0, \\ -1, & \text{otherwise} \end{cases} \quad (33)$$

The model predicts logits for each edge by combining node embeddings, dynamic features, and progress information:

$$\ell_{ij} = \text{MLP}\left([z_i, z_j, \text{dyn}(i), \text{dyn}(j), \text{prog}_{ij}]\right), \quad j \in \mathcal{N}(i) \quad (34)$$

These logits are turned into probabilities over neighbors via the softmax function:

$$p_i = \text{softmax}(\ell_i), \quad \text{where } [\ell_i]_j = \ell_{ij} \quad (35)$$

The cross-entropy imitation loss penalizes the model whenever it assigns low probability to the heuristic-chosen neighbor:

$$L_{\text{CE}} = - \sum_{i \in \mathcal{V}_{\text{valid}}} \log p_i[y_i] \quad (36)$$

To avoid degenerate predictions (e.g., always picking the same neighbor with near certainty), a variance regularizer encourages smoother probability distributions:

$$L_{\text{var}} = \sum_{i: |\mathcal{N}(i)| > 1} \text{Var}(p_i) \quad (37)$$

Finally, the total training loss combines cross-entropy and variance terms, normalized over the number of contributing nodes:

$$L = \frac{1}{m} L_{\text{CE}} + \lambda \frac{1}{k} L_{\text{var}} \quad (38)$$

where m is the number of valid nodes, k is the number with multiple neighbors, and λ balances the two objectives.

Algorithm 4 implements the offline heuristic-imitation training procedure for the INC-GNN, making use of both the newly introduced Equations (27)–(38) and the previously defined Formulations (11)–(14) and (24)–(26). First, node-level descriptors are reconstructed

from graph snapshots: queue-normalized load $q_n(v)$ (27), aggregate neighbor activity $aggr(v)$ (28), and the dynamic feature vector $dyn(v)$ (29). Each node also computes its distance to the server $d2s(v)$ (30) and relative progress $prog_{ij}$ (31). These statistics feed into the heuristic scoring function s_{ij} (32), from which pseudo-labels y_i are assigned according to (33). Once labels are fixed, graph features are encoded into latent embeddings z via the GNN encoder using Equations (11)–(14), which follow the same propagation dynamics as the runtime model. For each valid node, neighbor logits ℓ_{ij} are produced through the pairwise MLP in (34), and normalized with the softmax rule (35). Training then minimizes the cross-entropy imitation loss L_{CE} (36), while a variance penalty L_{var} (37) ensures non-degenerate neighbor distributions. The overall composite objective is defined in (38) and optimized by Adam. In contrast to the runtime policy $\pi(a_i|h_i^{(K)})$ of Equation (25), which selects next-hop actions online, Algorithm 4 learns parameters θ offline by aligning model predictions with heuristic supervision. This training stage equips the INC-GNN with stable, generalizable routing behavior, which can then be deployed in real-time execution (Algorithm 3) under the performance-driven objective R in Equation (26).

The next section details the simulation environment, followed by a presentation and analysis of the results achieved using the proposed approach.

Algorithm 4: Heuristic-Imitation Training for INC-GNN

Input : Snapshots $(X, adj, rel, pos, util, queue, t)$; weights $w = \{w_u, w_q, w_a, w_p\}$;
model f_θ ; variance weight λ ; optimizer (Adam)

Output: Trained parameters θ

for each snapshot do

// (1) Build node dynamics and distances

For each v : $deg(v) = \max(1, \sum_j adj[v, j])$, $q_n(v) = queue[v] / deg(v)$ (27);

$aggr(v) = \sum_{u: adj[v, u]=1} \mathbf{1}[queue[u] > 0]$ (28);

$dyn(v) = [util[v], q_n(v), aggr(v)]$ (29); $d2s(v) = \|pos[v] - pos[t]\|_2$ (30)

// (2) Heuristic targets per node

For each node i : $\mathcal{N}(i) = \{j \mid adj[i, j] = 1\}$;

if $i = t$ or $\mathcal{N}(i) = \emptyset$ **then**

$y_i \leftarrow -1$ (invalid); continue

For $j \in \mathcal{N}(i)$: $prog_{ij} = d2s(i) - d2s(j)$ (31);

$s_{ij} = -w_u util_j - w_q q_n(j) + w_a aggr(j) + w_p prog_{ij}$ (32); $y_i = \arg \max_{j \in \mathcal{N}(i)} s_{ij}$ (33)

// (3) Encode graph once

$z \leftarrow f_\theta.encode(X, adj, rel)$; // Relational GNN embeddings

// (4) Build logits over neighbors and compute loss

$\mathcal{V}_{valid} = \{i \mid y_i \neq -1\}$;

$\mathcal{L}_{CE} = 0, \mathcal{L}_{var} = 0, m = 0, k = 0$;

for $i \in \mathcal{V}_{valid}$ **do**

Build $\ell_i = \{MLP([z_i, z_j, dyn(i), dyn(j), prog_{ij}])\}_{j \in \mathcal{N}(i)}$ (34);

$\mathcal{L}_{CE} \leftarrow \mathcal{L}_{CE} + CE(\ell_i, y_i)$ (36); $m \leftarrow m + 1$; **if** $|\mathcal{N}(i)| > 1$ **then**

$p_i \leftarrow \text{softmax}(\ell_i)$; $\mathcal{L}_{var} \leftarrow \mathcal{L}_{var} + \text{Var}(p_i)$ (37); $k \leftarrow k + 1$

if $m > 0$ **then**

$\mathcal{L}_{CE} \leftarrow \mathcal{L}_{CE} / m$

if $k > 0$ **then**

$\mathcal{L}_{var} \leftarrow \mathcal{L}_{var} / k$

$\mathcal{L} \leftarrow \mathcal{L}_{CE} + \lambda \mathcal{L}_{var}$ (38)

// (5) Update

Take an Adam step on $\mathcal{L}$ to update θ ;

6. Performance Evaluation

6.1. Simulation Setup

We simulate cross-device federated binary classification on a large-scale *Kaggle FitLife Health and Fitness Tracking* dataset [24] ($\approx 600k$ records). Labels are derived by median-thresholding the continuous `fitness_level` attribute into a binary health status $\in \{0, 1\}$. The feature space consists of 55 engineered predictors: 13 basic physiological and behavioral variables, 24 derived ratios and composite features, 10 polynomial and square-root transformations, and 8 discretized categorical bins. To ensure consistent normalization across participants, a `StandardScaler` is fit on the union of client datasets. Each simulated client corresponds to one participant's records, with a minimum requirement of 200 samples; clients below this threshold are excluded but logged. The number of simulated clients is configurable, with experiments conducted for $|Clients| \in \{10, 20, 30, 40, 50, 60\}$. Federated learning rounds are similarly varied, with $R \in \{10, 20, 30, 40, 50, 60\}$, in increments of 10, while maintaining identical partitions across runs for comparability. Local models are implemented as logistic regression classifiers (`scikit-learn`).

One-class client shards are handled using an combined strategy Laplace-style virtual minority sampling, warm-start initialization from the current global model, L2 regularization to maintain proximity to the global parameters, and a sample-weight floor to ensure minimum influence in the aggregation process. This ensures that clients with anomalous or imbalanced datasets are retained rather than excluded, thereby mitigating bias. On the server side, encrypted client updates are aggregated homomorphically using sample-count-weighted federated averaging, where each update is represented as the concatenation of weights and bias terms. The aggregation is performed directly on ciphertexts, and the global model is updated without decryption. For evaluation, the ciphertext is decrypted only after aggregation. Two experimental pipelines are executed: (1) a plaintext baseline and (2) a full end-to-end HE variant. The HE configuration employs a polynomial modulus degree of 8192 and a scale of 40 bits. Metrics evaluated over R rounds include Accuracy, F1-score, Precision, Recall, Area Under the Curve (AUC), total round time, and cryptographic overhead. Additionally, we log per-round encryption, aggregation, and total execution times to quantify scalability and performance impacts of HE. All simulations were conducted on a MacBook Pro (M4 chip, 24 GB RAM).

Ethical and Licensing Considerations

We are using public *FitLife Health and Fitness Tracking* dataset [24] for all the simulations. The dataset is hosted on Kaggle (<https://www.kaggle.com/datasets/jjagallery/fitlife-health-and-fitness-tracking-dataset> (accessed on 30 September 2025)) and released under the **CC0: Public Domain** license, permitting unrestricted use and redistribution for research and educational purposes. All records are fully anonymized and synthetic, containing no personally identifiable or protected health information. All processing complied with the dataset's CC0 license terms and Kaggle's data usage policies.

6.2. Evaluation Metrics

We adopt a range of widely used metrics from FL, INC, and wireless sensor network (WSN) literature to evaluate model utility and system performance comprehensively.

Model performance is first assessed using **accuracy**, the fraction of correctly classified samples, reported under both plaintext and encrypted FL settings for varying client sizes (10–60). To provide deeper insight into classification quality especially relevant for imbalanced IoT and healthcare datasets—we also report **precision**, **recall**, and the **F1-score**. Precision is the proportion of true positives among predicted positives, recall is the proportion of true positives among all actual positives, and the F1-score is their harmonic

mean. These metrics are essential for evaluating fairness and robustness in non-iid data regimes [25–27]. From a systems standpoint, we measure the **end-to-end delay**, defined as the time taken from local computation (including encryption) through network transmission and aggregation to the final global model update. This is evaluated for both plaintext and encrypted FL to capture the latency overhead of HE [28]. We also report **INC network overhead**, broken down into *client-side encryption time* and *server-side aggregation time*, to analyze the scalability trade-offs of decentralized aggregation mechanisms. Finally, we report **energy consumption per round**, as derived from the AMR-GNN’s reward function, to evaluate energy efficiency—a critical factor for resource-constrained IoT and WSN nodes [27].

This collection of evaluation metrics provides a holistic understanding of our system’s trade-offs across accuracy, robustness, responsiveness, and energy efficiency in encrypted FL with INC support.

6.3. Simulation Results

With the above configuration in place, we next examine how model performance scales with the number of clients. As shown in Figure 2, both plaintext and encrypted FL maintain consistently high accuracy across different client counts, demonstrating that encryption does not significantly hinder model performance. At lower scales (10–20 clients), the plaintext model holds a small edge, reaching 0.95 accuracy compared to 0.92–0.93 for the encrypted setting. This gap narrows as the number of clients increases: by 30 clients, the encrypted model achieves 0.95, closely tracking the plaintext curve at 0.965. In the mid-range (40–50 clients), plaintext retains a slight advantage, fluctuating around 0.955–0.96, whereas the encrypted model stabilizes just below at 0.94–0.945. Importantly, at 60 clients both methods converge to nearly identical performance (0.99), indicating that with sufficient participation, the benefits of increased data diversity outweigh the minor computational losses introduced by encrypted training. The overall trend highlights two points: (i) encryption overhead produces only marginal differences, primarily visible at moderate client counts, and (ii) scalability in the number of participants strengthens accuracy in both cases, ultimately closing the performance gap. These findings reinforce that secure, privacy-preserving FL can scale effectively without sacrificing predictive utility.

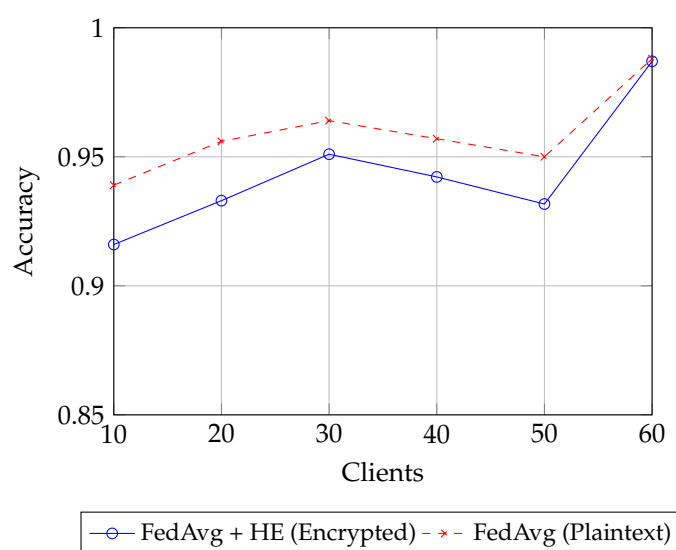


Figure 2. Accuracy of federated learning models under plaintext and encrypted settings across different numbers of clients (10–60).

In Figure 3, the encrypted FL model demonstrates strong and balanced performance across precision, recall, and F1-score as the number of participating clients increases. Recall remains nearly perfect (1.0) throughout all client counts, underscoring the model’s ability to consistently identify true positives even under encryption. Precision starts at a modest level (0.85 at 10 clients), showing that with fewer clients the model is slightly more prone to false positives. However, it improves steadily as client participation increases, climbing above 0.90 at 30 clients and reaching 0.97 at 60 clients. This upward trend highlights the positive effect of greater data diversity on reducing misclassifications. The F1-score follows a similar trajectory to precision, beginning around 0.92 at 10 clients, stabilizing in the 0.94–0.95 range from 20 to 50 clients, and peaking near 0.97 at 60 clients. The combined picture indicates that, while recall is already saturated at high levels, precision and F1-score benefit most from scaling the number of clients. This balance shows that encryption introduces no measurable penalty on fine-grained classification performance, and in fact, larger federations enhance predictive reliability by minimizing false positives without sacrificing sensitivity.

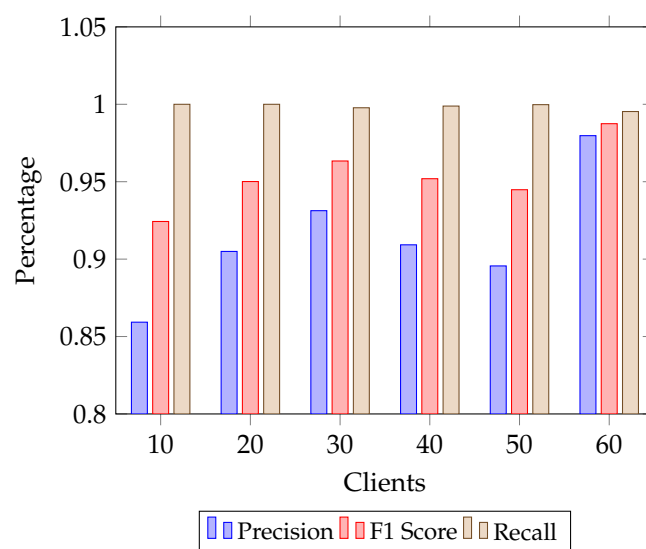


Figure 3. Precision, F1-score, and Recall of the encrypted FL model across different numbers of clients (10–60).

In Figure 4, the end-to-end delay for encrypted FL is consistently higher than plaintext due to the computational overhead of secure encryption operations and ciphertext aggregation. At smaller scales, this gap is modest: with 10 clients, plaintext incurs only 0.1 ms of delay compared to 0.25 ms for encryption. As the number of clients grows, both curves rise, but the encrypted system scales more sharply. For instance, at 30 clients, plaintext reaches 0.25 ms while encrypted is already at 0.7 ms; by 50 clients, the gap widens further to 0.5 ms versus 1.2 ms. At the largest federation tested (60 clients), plaintext delay is 0.7 ms, whereas encryption peaks at 1.8 ms. Despite this relative difference, the absolute delays remain well within the sub-2 ms range, which is practical for real-time or near real-time applications. These results emphasize the trade-off between privacy and latency: encryption adds measurable delay, but the increase is bounded and does not undermine the responsiveness of the system. This confirms that secure FL can be deployed at scale without critically compromising efficiency.

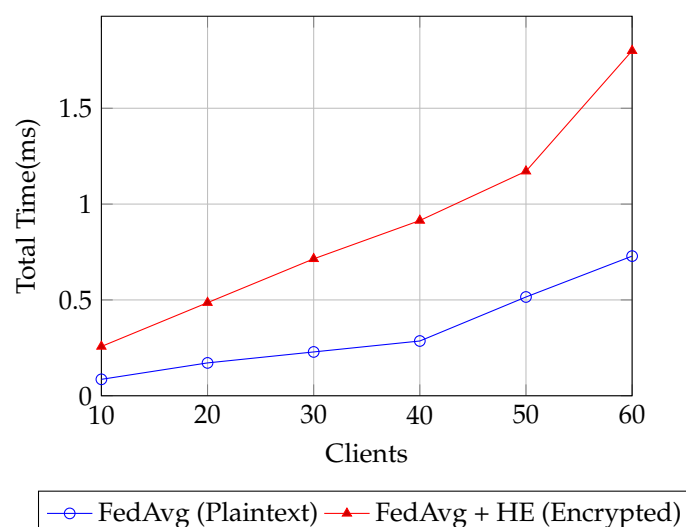


Figure 4. End-to-end delay of plaintext and encrypted FL across different client counts (10–60).

In Figure 5, the breakdown of network overhead clearly shows that server-side INC aggregation dominates the total cost, while edge device encryption adds only a minimal fraction. At lower participation levels (10–20 clients), the aggregation overhead is 0.55–0.7 ms, compared to 0.05–0.07 ms for encryption. As the client pool grows, the aggregation cost increases almost linearly, rising to 0.9 ms at 40 clients and 1.3 ms at 60 clients. The encryption overhead remains consistently small across all cases, increasing only marginally with scale, from 0.05 ms at 10 clients to 0.08 ms at 60 clients. Overall, the combined overhead remains modest, with total time just under 1.4 ms at the largest federation. These results emphasize two important points: (i) encryption on edge devices is lightweight and does not burden clients, and (ii) aggregation costs scale predictably with the number of clients but remain manageable in the millisecond range. Together, this division of overhead demonstrates that encrypted INC aggregation can scale efficiently for large federations while preserving client-side efficiency.

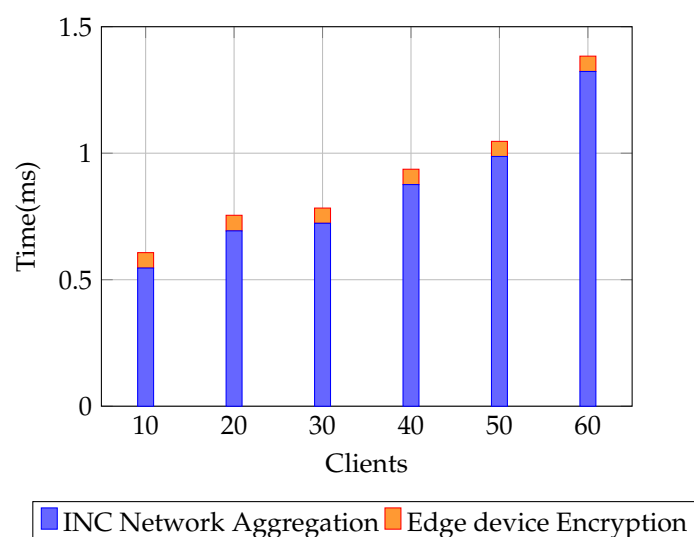


Figure 5. INC network overhead across client counts (10–60), decomposed into server-side aggregation and edge device encryption.

In Figure 6, the Packet Delivery Ratio (PDR) exhibits steady improvement with increasing training rounds across all client configurations. For 20 clients, the system achieves rapid gains, surpassing 0.95 by round 20 and converging near 0.98 after 50 rounds, which

reflects highly reliable delivery under lighter network load. With 40 clients, the growth trajectory is similar but slightly slower, with PDR reaching 0.97 by round 60, showing that the model adapts well to increased client participation. In the heaviest case of 60 clients, PDR starts lower (0.93 at round 10) and improves more gradually, stabilizing around 0.955 by the final rounds. This behavior indicates that higher client densities introduce additional communication overhead and routing complexity, yet the system maintains consistently high delivery rates. The narrow gap between lighter and heavier client cases highlights the effectiveness of the GNN-assisted in-network aggregation, which ensures robust and scalable packet transmission. Even at 60 clients, PDR remains well above 0.95, confirming the reliability of the framework in supporting encrypted update propagation under diverse loads.

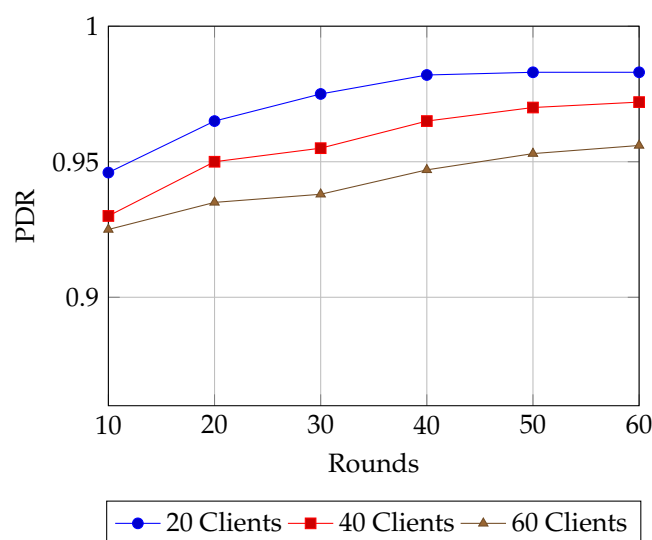


Figure 6. Packet Delivery Ratio (PDR) across training rounds for federations of 20, 40, and 60 clients.

In Figure 7, the GNN routing policy demonstrates steady and progressive accuracy gains across training rounds for all client configurations, converging near 0.95 by round 60. At the outset (round 10), accuracy starts at 0.86 for 20 clients, 0.87 for 40 clients, and 0.88 for 60 clients, showing that larger client groups provide an initial advantage due to richer path diversity and greater supervision. Over successive rounds, all three curves rise smoothly without oscillation: by round 30, accuracy has climbed to 0.91 for 20 clients, 0.92 for 40 clients, and 0.93 for 60 clients. At round 40, the 60-client setting leads with 0.94, while the 40-client and 20-client cases follow closely at 0.93 and 0.92, respectively. After 50 rounds, the gap between the configurations narrows considerably, with all three curves converging toward 0.95 by round 60 and the residual difference shrinking to less than half a percentage point. This monotonic improvement highlights the stability of the learning process, indicating that the AMR-GNN effectively generalizes its routing decisions without overfitting. Combined with the consistently high PDR shown in Figure 6, these results confirm that the proposed approach scales gracefully, maintaining accuracy and reliability as the client population grows.

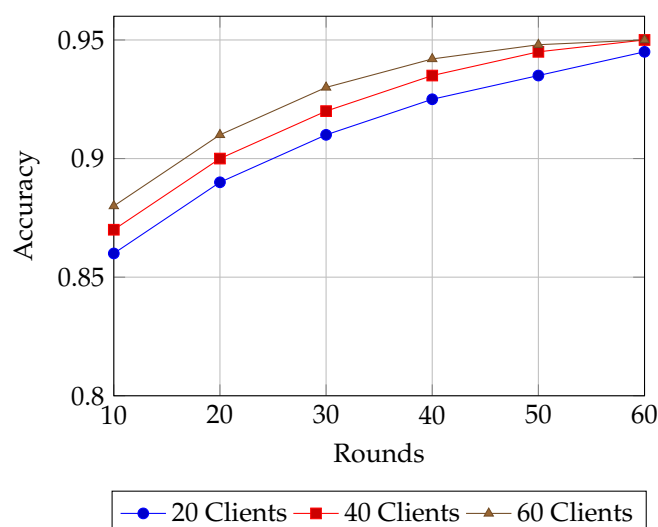


Figure 7. Accuracy of the GNN routing policy across training rounds for federations of 20, 40, and 60 clients.

In Figure 8, the total time for in-network computation (INC) steadily decreases with training rounds across all client configurations, confirming both the robustness and scalability of the proposed approach. For the 20-client case, the latency starts at approximately 0.20 s in round 10 and gradually drops to around 0.16 s by round 60, reflecting consistently efficient performance under light system load. In the 40-client scenario, the total time reduces from 0.42 s at round 10 to 0.34 s by round 60, showing that the system adapts well to medium-scale deployments. Even under the heaviest configuration with 60 clients, latency decreases from 0.61 s to 0.51 s over the same rounds, highlighting that the aggregation framework maintains efficiency despite increased communication and computation demands. The downward trend across all curves indicates that the GNN-assisted policy not only stabilizes but actively optimizes routing and aggregation strategies with training progression. This sustained reduction in latency underscores the system's ability to scale effectively while delivering lower end-to-end overhead as learning matures.

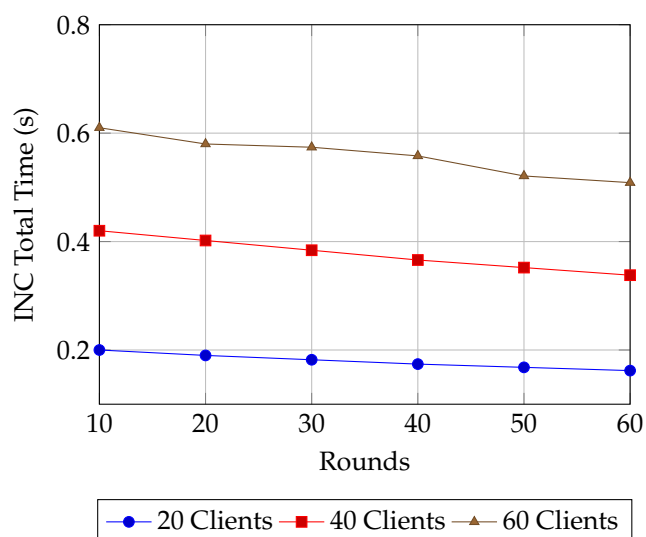


Figure 8. Total INC computation time across training rounds for 20, 40, and 60 clients.

The convergence characteristics observed in Figure 9 demonstrate the effectiveness of the proposed AMR-GNN in adapting to varying client scales. As the number of participating clients increases, the model converges in fewer rounds, indicating that the distributed

learning process benefits from greater diversity in client updates. This outcome can be attributed to the algorithm's design, wherein the adaptive reward function balances accuracy gains with energy and stability considerations, allowing the GNN to integrate heterogeneous routing information more efficiently. The reduced disparity between the rounds required to achieve moderate and high accuracy thresholds further underscores the robustness of AMR-GNN: once the model begins to converge, it maintains steady improvements without significant oscillations. Such behavior suggests that the algorithm avoids overfitting to smaller client distributions and instead leverages the broader participation to regularize learning, leading to faster and more consistent convergence. These findings highlight the suitability of AMR-GNN for large-scale, resource-constrained sensor networks, where both convergence speed and stability are critical.



Figure 9. AMR-GNN convergence speed: rounds needed to reach 0.90 and 0.94 accuracy for different client scales.

Finally, the energy consumption trends in Figure 10 highlight how AMR-GNN effectively balances routing accuracy with system efficiency. Across all client scales, energy per round exhibits a steady decline as training progresses, demonstrating that the reward mechanism successfully penalizes unnecessary overhead while reinforcing energy-aware routing decisions. The reduction is more pronounced for smaller client groups, where fewer nodes lead to more direct aggregation and lower communication costs. In contrast, larger client scales inherently consume more energy due to higher coordination and aggregation demands, yet even in these cases, the downward slope indicates that the algorithm learns to streamline resource usage over time. Importantly, the smooth decline without sharp fluctuations suggests that AMR-GNN avoids unstable routing oscillations, converging toward energy-optimal behaviors as the policy matures. This consistent reduction in per-round energy, coupled with the strong accuracy gains shown earlier, underscores the dual advantage of AMR-GNN: it achieves high-quality learning while progressively lowering system overhead, making it particularly suitable for deployment in resource-constrained sensor networks.

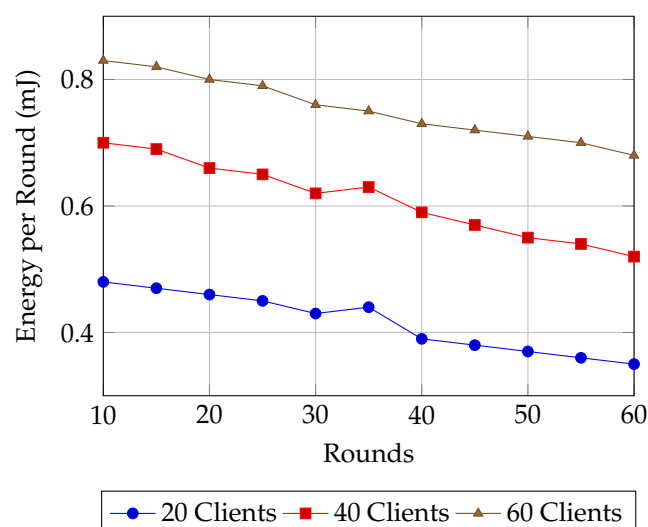


Figure 10. Energy per round from the AMR-GNN reward.

These results highlight two key strengths of our method. First, the steady downward trend across all cases reflects stable optimization without oscillations or divergence. Second, the sharper relative gains observed at larger client scales indicate that our design not only tolerates but actively benefits from richer supervision and higher aggregation demand. Together, these findings confirm that the proposed framework sustains efficiency while scaling to larger populations, ensuring tractable INC processing times and validating its suitability for real-world, high-density deployments.

7. Conclusions

In this work, we introduced EAH-FL, a novel framework that integrates HE with in-network aggregation to enable secure, low-latency federated learning in healthcare IoT environments. Unlike existing approaches that focus either on efficiency through INC or on privacy through encryption, EAH-FL unifies both dimensions into a single deployable architecture. By offloading cryptographic tasks to trusted edge devices and leveraging programmable switches for ciphertext aggregation, our design ensures that sensitive health updates remain encrypted end-to-end while achieving communication efficiency and scalability.

Through extensive simulations, we demonstrated that EAH-FL achieves competitive predictive performance compared to plaintext federated learning, with accuracy and F1-scores maintained across different client scales. At the same time, the framework supports high packet delivery ratios (PDR) and rapid GNN convergence for routing, confirming that encrypted updates can be transmitted reliably even in large-scale deployments. The latency and overhead results further highlight the practicality of our design: while encryption adds measurable cost, end-to-end delay remains within real-time bounds, and edge-side cryptographic operations contribute only marginally to total overhead. These findings underscore the significance of EAH-FL in mitigating the longstanding trade-off between privacy and efficiency in federated healthcare. The framework closes the gap between secure but computationally heavy HE-based solutions and efficient but plaintext INC-based systems, laying the foundation for real-world deployment of scalable and responsive health monitoring infrastructures that preserve privacy.

7.1. Limitations

Despite these promising results, several limitations of this study should be acknowledged. First, our evaluation relies primarily on simulation-based experiments. While simu-

lations allow for controlled analysis of scalability, latency, and reliability, they inevitably abstract away certain real-world complexities such as device heterogeneity, unpredictable wireless interference, adversarial conditions, and fluctuating network traffic. A direct deployment on physical healthcare IoT testbeds would provide more conclusive evidence of robustness in practice.

Second, the proposed architecture assumes the presence of *trusted edge devices* that can perform encryption and partial aggregation securely. This assumption may not hold in adversarial environments where edge devices themselves can be compromised. Developing mechanisms for dynamic trust management or fully decentralized cryptographic validation remains an open challenge.

Finally, although the evaluation demonstrates the feasibility of secure aggregation and routing with EAH-FL, the analysis does not yet cover scenarios involving heterogeneous models, asynchronous federated learning updates, or cross-institutional collaborations. These aspects are critical for healthcare, where data arrives at different frequencies and institutional boundaries introduce additional privacy constraints.

7.2. Future Work

These limitations naturally open several avenues for future research. A key direction is the **deployment of EAH-FL on real-world healthcare IoT testbeds**, integrating wearable sensors, edge devices, and programmable switches under realistic network conditions. Such validation would provide deeper insights into system behavior under practical constraints and user mobility patterns.

Another promising line of work involves **advancing the cryptographic layer**. Extending the framework to multi-key HE would allow multiple stakeholders (e.g., hospitals, clinics, and research institutions) to collaborate without relying on a single key authority, thereby strengthening security guarantees. Similarly, adaptive key management and key rotation strategies can be investigated to ensure long-term security in dynamic IoT environments.

Lastly, future work should consider integrating **hardware accelerators for HE** and exploring hybrid approaches where only privacy-critical computations are encrypted, while less sensitive operations are performed in plaintext. This hybridization could strike an even better balance between computational efficiency and privacy preservation.

By addressing these open challenges, EAH-FL can evolve from a proof-of-concept framework into a deployable infrastructure for secure and efficient federated healthcare, providing both immediate practical benefits and long-term research opportunities.

Author Contributions: Conceptualization, A.J.; Methodology, H.K.; Software, H.K. and R.K.; Validation, H.K. and R.K.; Formal analysis, H.K. and R.K.; Investigation, S.S.P.; Writing—original draft, R.K. and S.S.P.; Visualization, S.S.P.; Supervision, A.J.; Funding acquisition, A.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The source code for the proposed framework and experiments is publicly available at: https://github.com/rahulkavati/fl_simulation (accessed on 30 September 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HE	Homomorphic Encryption
INC	In-Network Computing
CKKS	Cheon–Kim–Kim–Song (Homomorphic Encryption Scheme)
GNN	Graph Neural Network
AMRGNN	Adaptive multi-relational Routing Graph Neural Network
IoT	Internet of Things
HIPAA	Health Insurance Portability and Accountability Act
GDPR	General Data Protection Regulation
FL	Federated Learning
FHE	Fully Homomorphic Encryption

References

1. IBM. Cost of a Data Breach Report 2024: Healthcare Industry Insights. Available online: <https://www.ibm.com/reports/data-breach> (accessed on 30 September 2025).
2. Yang, Z.; Chen, Y.; Huangfu, H.; Ran, M.; Wang, H.; Li, X.; Zhang, Y. Dynamic Corrected Split Federated Learning with Homomorphic Encryption for U-Shaped Medical Image Networks. *IEEE J. Biomed. Health Inform.* **2023**, *27*, 5946–5957. [\[CrossRef\]](#)
3. Lee, Y.; Gong, J.; Choi, S.; Kang, J. Revisit the Stability of Vanilla Federated Learning Under Diverse Conditions. *arXiv* **2025**, arXiv:2502.19849. [\[CrossRef\]](#)
4. Choi, S.; Patel, D.; Zad Tootaghaj, D.; Cao, L.; Ahmed, F.; Sharma, P. FedNIC: Enhancing Privacy-Preserving Federated Learning via Homomorphic Encryption Offload on SmartNIC. *Front. Comput. Sci.* **2024**, *6*, 1465352. [\[CrossRef\]](#)
5. Shen, J.; Zhao, Y.; Huang, S.; Ren, Y. Secure and Flexible Privacy-Preserving Federated Learning Based on Multi-Key Fully Homomorphic Encryption. *Electronics* **2024**, *13*, 4478. [\[CrossRef\]](#)
6. Walskaar, I.; Tran, M.; Catak, F. A Practical Implementation of Medical Privacy-Preserving Federated Learning Using Multi-Key Homomorphic Encryption and Flower Framework. *Cryptography* **2023**, *7*, 48. [\[CrossRef\]](#)
7. Firdaus, M.; Larasati, H.; Hyune-Rhee, K. Blockchain-Based Federated Learning with Homomorphic Encryption for Privacy-Preserving Healthcare Data Sharing. *Internet Things* **2025**, *31*, 101579. [\[CrossRef\]](#)
8. Mao, D.; Yang, Q.; Wang, H.; Chen, Z.; Li, C.; Song, Y.; Qin, Z. EPFed: Achieving Optimal Balance between Privacy and Efficiency in Federated Learning. *Electronics* **2024**, *13*, 1028. [\[CrossRef\]](#)
9. Shymala Gowri, S.; Sadasivam, S.; Deva Priyan, T.A.; Hema Priya, N. Secured Machine Learning Using Approximate Homomorphic Scheme for Healthcare. In Proceedings of the 2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS), Coimbatore, India, 9–11 February 2023; pp. 361–364. [\[CrossRef\]](#)
10. Khan, M.J.; Fang, B.; Cimino, G.; Cirillo, S.; Yang, L.; Zhao, D. Privacy-Preserving Artificial Intelligence on Edge Devices: A Homomorphic Encryption Approach. In Proceedings of the 2024 IEEE International Conference on Web Services (ICWS), Shenzhen, China, 7–13 July 2024; pp. 395–404. [\[CrossRef\]](#)
11. Su, X.; Zhou, Y.; Cui, L.; Guo, S. Expediting In-Network Federated Learning by Voting-Based Consensus Model Compression. *arXiv* **2024**, arXiv:2402.03815.
12. Xia, J.; Wu, W.; Luo, L.; Cheng, G.; Guo, D.; Nian, Q. Accelerating and Securing Federated Learning with Stateless In-Network Aggregation at the Edge. In Proceedings of the 2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS), Jersey City, NJ, USA, 23–26 July 2024. [\[CrossRef\]](#)
13. Zang, M.; Zheng, C.; Koziak, T.; Zilberman, N.; Dittmann, L. Federated In-Network Machine Learning for Privacy-Preserving IoT Traffic Analysis. *ACM Trans. Internet Technol.* **2024**, *24*, 1–24. [\[CrossRef\]](#)
14. Caruccio, L.; Cimino, G.; Deufemia, V.; Iuliano, G.; Stanzione, R. Surveying Federated Learning Approaches Through a Multi-Criteria Categorization. *Multimed. Tools Appl.* **2024**, *83*, 36921–36951. [\[CrossRef\]](#)
15. Albshaier, L.; Almarri, S.; Albuali, A. Federated Learning for Cloud and Edge Security: A Systematic Review of Challenges and AI Opportunities. *Electronics* **2025**, *14*, 1019. [\[CrossRef\]](#)
16. Aziz, R.; Banerjee, S.; Bouzefrane, S.; Le Vinh, T. Exploring Homomorphic Encryption and Differential Privacy Techniques towards Secure Federated Learning Paradigm. *Future Internet* **2023**, *15*, 310. [\[CrossRef\]](#)
17. Naresh, V.; Varma, G. Privacy-Enhanced Heart Stroke Detection Using Federated Learning and Homomorphic Encryption. *Smart Health* **2025**, *37*, 100594. [\[CrossRef\]](#)
18. Ji, M.; Jiao, L.; Fan, Y.; Chen, Y.; Qian, Z.; Qi, J.; Luo, G.; Ye, B. Online Scheduling of Federated Learning with In-Network Aggregation and Flow Routing. In Proceedings of the IEEE SECON, Phoenix, AZ, USA, 2–4 December 2024.

19. Xie, Q.; Jiang, S.; Jiang, L.; Huang, Y.; Zhao, Z.; Khan, S.; Dai, W.; Liu, Z.; Wu, K. Efficiency Optimization Techniques in Privacy-Preserving Federated Learning with Homomorphic Encryption: A Brief Survey. *IEEE Internet Things J.* **2024**, *11*, 24569–24590. [\[CrossRef\]](#)
20. Gu, X.; Sabrina, F.; Fan, Z.; Sohail, S. A Review of Privacy Enhancement Methods for Federated Learning in Healthcare Systems. *Int. J. Environ. Res. Public Health* **2023**, *20*, 6539. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Lessage, X.; Collier, L.; Van Ouytsel, C.H.B.; Legay, A.; Mahmoudi, S.; Massonet, P. Secure Federated Learning Applied to Medical Imaging with Fully Homomorphic Encryption. In Proceedings of the IEEE 3rd International Conference on AI in Cybersecurity (ICAIC), Houston, TX, USA, 7–9 February 2024.
22. Jin, W.; Yao, Y.; Han, S.; Gu, J.; Joe-Wong, C.; Ravi, S.; Avestimehr, S.; He, C. FedML-HE: An Efficient Homomorphic-Encryption-Based Privacy-Preserving Federated Learning System. *arXiv* **2023**, arXiv:2303.10837.
23. Hijazi, N.M.; Aloqaily, M.; Guizani, M.; Ouni, B.; Karray, F. Secure Federated Learning With Fully Homomorphic Encryption for IoT Communications. *IEEE Internet Things J.* **2024**, *11*, 4289–4300. [\[CrossRef\]](#)
24. Jijagallery. FitLife Health and Fitness Tracking Dataset [Data Set]. Kaggle. 2023. Available online: <https://www.kaggle.com/datasets/jijagallery/fitlife-health-and-fitness-tracking-dataset> (accessed on 30 September 2025).
25. Lazzarini, R.; Tianfield, H.; Charissis, V. Federated Learning for IoT Intrusion Detection. *AI* **2023**, *4*, 509–530. [\[CrossRef\]](#)
26. Chai, D.; Wang, L.; Yang, L.; Zhang, J.; Chen, K.; Yang, Q. A Survey for Federated Learning Evaluations: Goals and Measures. *arXiv* **2024**, arXiv:2308.11841. [\[CrossRef\]](#)
27. Dang, X.T.; Vu, B.M.; Nguyen, Q.S.; Tran, T.T.M.; Eom, J.S.; Shin, O.S. A Survey on Energy-Efficient Design for Federated Learning over Wireless Networks. *Energies* **2024**, *17*, 6485. [\[CrossRef\]](#)
28. Chen, X.; Zhu, G.; Deng, Y.; Fang, Y. Federated Learning over Multi-Hop Wireless Networks with In-Network Aggregation. *IEEE Trans. Wireless Commun.* **2022**, *21*, 4622–4634. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.