TYPE Original Research
PUBLISHED 21 October 2024
DOI 10.3389/fcomp.2024.1465352

Check for updates

OPEN ACCESS

EDITED BY
Hui Dou,
Anhui University, China

REVIEWED BY
Shitharth Selvarajan,
Leeds Beckett University, United Kingdom
Qin Jiang,
Nanjing University of Science and Technology, China

*CORRESPONDENCE
Sean Choi
✉ sean.choi@scu.edu

RECEIVED 16 July 2024
ACCEPTED 27 September 2024
PUBLISHED 21 October 2024

CITATION
Choi S, Patel D, Zad Tootaghaj D, Cao L, Ahmed F and Sharma P (2024) FedNIC: enhancing privacy-preserving federated learning via homomorphic encryption offload on SmartNIC. *Front. Comput. Sci.* 6:1465352. doi: 10.3389/fcomp.2024.1465352

FedNIC: enhancing privacy-preserving federated learning via homomorphic encryption offload on SmartNIC

Sean Choi[1]*, Disha Patel[1], Diman Zad Tootaghaj[2], Lianjie Cao[2], Faraz Ahmed[2] and Puneet Sharma[2]

[1]Cloud Laboratory, Department of Computer Science and Engineering, Santa Clara University, Santa Clara, CA, United States, [2]Networking and Distributed Systems Lab, Hewlett Packard Labs, Hewlett Packard Enterprises, San Jose, CA, United States

Federated learning (FL) has emerged as a promising paradigm for secure distributed machine learning model training across multiple clients or devices, enabling model training without having to share data across the clients. However, recent studies revealed that FL could be vulnerable to data leakage and reconstruction attacks even if the data itself are never shared with another client. Thus, to resolve such vulnerability and improve the privacy of all clients, a class of techniques, called privacy-preserving FL, incorporates encryption techniques, such as homomorphic encryption (HE), to encrypt and fully protect model information from being exposed to other parties. A downside to this approach is that encryption schemes like HE are very compute-intensive, often causing inefficient and excessive use of client CPU resources that can be used for other uses. To alleviate this issue, this study introduces a novel approach by leveraging smart network interface cards (SmartNICs) to offload compute-intensive HE operations of privacy-preserving FL. By employing SmartNICs as hardware accelerators, we enable efficient computation of HE while saving CPU cycles and other server resources for more critical tasks. In addition, by offloading encryption from the host to another device, the details of encryption remain secure even if the host is compromised, ultimately improving the security of the entire FL system. Given such benefits, this paper presents an FL system named FedNIC that implements the above approach, with an in-depth description of the architecture, implementation, and performance evaluations. Our experimental results demonstrate a more secure FL system with no loss in model accuracy and up to 25% in reduced host CPU cycle, but with a roughly 46% increase in total training time, showing the feasibility and tradeoffs of utilizing SmartNICs as an encryption offload device in federated learning scenarios. Finally, we illustrate promising future study and potential optimizations for a more secure and privacy-preserving federated learning system.

KEYWORDS
privacy-preserving machine learning, federated learning, homomorphic encryption, SmartNIC, network offload

1 Introduction

Federated learning (FL) has emerged as a distributed machine learning model training technique that is aimed at preserving the privacy of each client, including privacy in data and model weights, by having decentralized clients train a model on each of their own private data and sending the localized weight to a centralized aggregator for aggregated model weights. However, recent study has shown that despite its initial understanding of the benefits, privacy concerns persist due to potential attacks that malicious parties can perform. Traditional FL can still be vulnerable to various threats that are specially designed for ML models, i.e., inference attacks and data leakage/reconstruction (Geiping et al., 2020; Wang et al., 2018; Zhu et al., 2019; Bhowmick et al., 2019; Hatamizadeh et al., 2022), severely compromising the usefulness of FL in preventing privacy leakage. For example, Zhu et al. (2019) shows how to obtain the private training data from gradients during training, and Geiping et al. (2020) analyze the possibility of reconstructing inputs by inverting gradients in FL systems.

To overcome such issues, researchers have introduced a notion of privacy preservation to ensure that data and model weights are not fully disclosed to another party. To do so, multiple methods have been proposed as follows: multi-party computation (MPC) (Bonawitz et al., 2017; So et al., 2022), which utilizes specialized multi-party protocols to compute a function across multiple private inputs or differential privacy that adds a small amount of statistically insignificant noise to the data (Truex et al., 2019; Choudhury et al., 2019). While the proposed methods are promising in theory, they incur high overheads to the client, often making them infeasible to use in practice. In addition, these methods are designed to work with specific types of adversaries and threat models and may not be generic enough. Furthermore, offloading complex and highly specialized protocols and algorithms like MPC to hardware and custom chips is not well supported, further jeopardizing their uses in production at a large scale.

Unlike the above complex solutions for privacy preservation, there is a classic set of methods that are already widely used for data privacy, which is encryption using cryptography. Encryption is already widely used in modern computing systems at large scale to ensure privacy, especially in communication and storage. For example, most network packets already are encrypted to stop eavesdroppers from capturing and reading network traffic and storage of sensitive data requires encryption of the data to stop the data from being read by unauthorized parties. Due to such high demand for encryption, there is already a sizeable amount of software and hardware support in using highly optimized encryption algorithms, meaning that using encryption for FL is one of the most feasible methods to bring FL to production.

One of the popular methods of encryption used for encryption-based FL is homomorphic encryption (HE), which, at a high level, enables operations on the encrypted data without having to decrypt the data first. The basic assumption behind using this encryption method is that the aggregator can be compromised, thus leaking information regarding the model weights from each client, which, in turn, can be used for model and data reconstruction. By using HE, the aggregator can only see the encrypted weights and perform operations directly on the encrypted weights without decryption, eliminating the need to worry about the raw model information being compromised. However, the major issue with HE is that it is very CPU, disk, and network intensive. Some of the popular HE algorithms increase the size of data to the order of 1,000× (Jin et al., 2023). Therefore, this makes the entire FL with HE process very resource-intensive, making it infeasible for the FL clients, which often are CPU and energy-limited machines, in the real-world FL settings.

To address this problem, this study introduces a system called FedNIC that offloads the resource-intensive portion of FL with HE onto a device called smart network interface cards (SmartNICs). The approach that FedNIC takes does not require clients to provision additional compute resources, but rather the extraneous work that is required by FL with HE, which essentially is the encryption, decryption, and transfer of larger ciphertext, to be performed by separate hardware. One obvious benefit of this approach is saving host compute resources, as HE algorithms are very compute heavy and are known to spend up to 25% CPU cycles and 250 MB of RAM (Reddy et al., 2022). Freeing host resources allows the host to focus on more important tasks that cannot be performed by the NIC, such as ML model updates. Furthermore, by having SmartNICs be the point of all encryption and decryption, we can have a more secure location to store the method and required keys for encryption. This means that even if the host is compromised, the method of encryption and the encryption key are not exposed to the adversary. In addition, SmartNICs can easily change the method of encryption and/or the encryption keys without the host knowing, making it harder to decode the encrypted text. Finally, SmartNICs often are equipped with specialized chips that are optimized for fast cryptography operations, which shows better performance and efficiency of HE vs. when running the same operations on CPUs. To address the challenges of applying homomorphic encryption with FL systems, we propose *offloading homomorphic encryption to SmartNICs to reduce clients' hardware and resource requirements for the FL training process while maintaining robust data privacy*. This main focus of FedNIC is aimed at enhancing the security and feasibility of FL with HE in real-world settings, by greatly reducing client resource requirements. Given this, the following summarizes the set of key contributions of this study.

Key contributions

- A high-level system design that is the first-ever to utilize SmartNICs for *storing and distributing encryption keys and encrypting model weights using Homomorphic Encryption* in a federated learning setting.
- An implementation of the proposed system, called FedNIC, which is a privacy-preserving FL framework that utilizes homomorphic encryption offload onto SmartNICs. FedNIC guarantees higher levels of privacy due to limited attack surface and separate security domain, while also reducing client resource requirements than other FL frameworks that utilize HE.
- Experimental evaluation results of FedNIC that show significant resource overhead reduction with no loss in accuracy or no significant increase in training time, while ensuring privacy against state-of-the-art ML privacy attacks.

Given the high-level introduction, the structure of this paper is as follows: We begin the paper by providing the background (Section 2) of the current state of privacy-preserving federated learning and usage of SmartNICs within cloud data centers. Then, we discuss the overview (Section 3) of the system framework followed by an evaluation (Section 4) of FedNIC performance.

Finally, we discuss some related studies that motivate FedNIC (Section 5) and establish the scope for potential future extension of this study (Section 6).

2 Background

We first delve into the fundamentals of Federated Learning, Homomorphic Encryption in Federated Learning and SmartNICs, which are the topics that are crucial tenets of FedNIC.

2.1 Federated learning

Federated learning is a learning task that is solved by a loose federation of participating devices (also referred to as clients) that are coordinated by a central coordinator (also referred to as the aggregator) (McMahan et al., 2017). More specifically, federated learning assumes that each client has a local training data set, which is assumed to be private and is not shared with any other party including the aggregator, that each client uses to train a set of local model weights. Once the local training is completed, it is shared with the central aggregator. Once receiving the set of model weights from multiple clients, the aggregator updates the global model weights by aggregating all of the weights it has received, and then, the aggregator passes back the updated global weight to the clients for the clients to perform the next iteration.

There are two main advantages of FL that arise from not having to share the training data. First is that the communication and energy overhead is reduced due to not having to transfer data from the clients to a centralized model training framework, which allows the clients to be small and energy-efficient devices. Second and the most important aspect is that the privacy of data is preserved as it never leaves the client.

There are many popular frameworks that implement federated learning:

- FedML (He et al., 2020): FedML is a framework that aims to be an open research library and benchmark to facilitate FL algorithm development and fair performance comparison. It offers a machine learning toolkit featuring APIs that facilitate federated learning and distributed training across varying scales. The toolkit supports cross-silo and cross-device federated learning, along with simulated federated learning. It incorporates diverse communication backends, including MPI, gRPC, and PyTorch RPC, for efficient distributed computing.
- IBM Federated Learning (Ludwig et al., 2020): IBM Federated Learning is a Python framework that aims to provide infrastructure and coordination for federated learning. It is particularly well-suited for enterprise and hybrid-Cloud settings. It has broad machine-learning model support including but not limited to neural networks, decision tree, linear regression, etc.
- TensorFlow Federated (Inc., 2020): Tensorflow Federated is an open-source framework based on Tensorflow, a popular machine learning library, for performing machine learning, simulations and other computations on decentralized data.

The framework of choice for FedNIC is FedML due to its open-source nature, allowing flexibility in modifying the underlying code as needed, and also due to its benchmarks widely available in the research community. Yet, even with these frameworks, the main issue with traditional FL settings is that it is possible to breach the privacy of data via the attacks mentioned in Section 1, such as inference attacks and data leakage/reconstruction.

To avoid such issues, multiple techniques have been used to improve privacy preservation in FL settings. Table 1 lists the potential techniques for privacy preservation. Out of multiple techniques, many researchers are focused on utilizing encryption-based methods, mainly due to the simplicity of implementation along with a multitude of hardware support for encryption. However, as mentioned in Table 1, encryption-based methods require a secure key exchange or have to cope with high computational and network overheads, making it infeasible for compute and energy-limited devices often used for FL. Therefore, FedNIC is a work that complements such efforts to further increase their efficiencies.

2.2 Homomorphic encryption

Homomorphic encryption (HE) is a form of encryption that allows mathematical operations to be performed on the encrypted data without needing to decrypt it first, leaving the outcome of the operation in encrypted format. A highly desirable property of HE is that the resulting output between performing the operations on encrypted data is identical to the output had the operations been performed on the unencrypted data. Thus, HE has become popular in systems where privacy-preserving properties must be ensured for in aggregation operations across multiple entities, such as aggregating model weight across multiple clients on a centralized server.

Most HE algorithms generally consist of four functions:

- $KeyGen(\lambda) \rightarrow (pk, sk)$: Given a security parameter λ, this function generates a pair of public and secret key (pk, sk).
- $Encrypt(pk, m_i) \rightarrow c_i$: This function takes the public key and a message m_n and encrypts the message to generate the encrypted ciphertext c_n.
- $Evaluate(c_i, c_j, f) \rightarrow c'_{i,j} = Encrypt(pk, f(m_i, m_j))$: The evaluate function takes two ciphertext and applies a target function f. f is generally either addition or multiplication. To ensure that the homomorphic properties are preserved, the output of *Evaluate* on two encrypted ciphertext c_i, c_j generated from two messages m_i, m_j are guaranteed to be the same as the result of *Encrypt* applied on the result of f on m_i, m_j.
- $Decrypt(sk, c_i) \rightarrow m_i$: Finally, the decrypt function allows the parties with the secret key to decrypt the ciphertext back to the original message.

Out of these functions, FedNIC focuses on utilizing SmartNICs to participate mainly in encryption and decryption of the data.

There are four classes of HE algorithms: partially HE, somewhat HE and fully HE. Partially HE algorithms support evaluation with one type of operation, either addition or multiplication,

TABLE 1 Overview of privacy-preserving techniques for FL.

Technique type	Method	Advantage	Disadvantages
Encryption	Cryptographic method that encrypt model weights to disable any parties from compromising the weights	Able to offload to hardware. Strong privacy, even when aggregator is compromised	High compute overhead, especially for homomorphic encryption. Secure key exchange often needed
Multi-party computation (MPC)	Protocol to require two or more clients to perform joint computation when viewing any data	Single party has no control	High network overhead. Prone to client dropouts. Hard to offload
Differential privacy (DP)	Adding small noise to data	Privacy is achieved without background knowledge	Unwanted noise can jeopardize the DP and/or model quality

where as somewhat HE algorithms support both addition and multiplication, but for a limited number of equations, finally fully HE algorithms allow for an infinite number of evaluations for both types of operations. Each class of HE algorithm has benefits and disadvantages, which we compare in this study. Particularly, FedNIC focuses on the following set of HE algorithms:

- Paillier (Paillier, 1999): Paillier is the partially homomorphic encryption scheme in which encrypted numbers demonstrate the ability to undergo multiplication with non-encrypted scalars, addition among themselves, and addition with non-encrypted scalars.
- TenSEAL (Benaissa et al., 2021): TenSEAL, a fully homomorphic open-source encryption library based on Microsoft SEAL, is specifically designed for conducting homomorphic encryption operations on tensors. This library delves into vector encryption/decryption methodologies utilizing both the BFV (Brakerski, 2012) and CKKS (Cheon et al., 2017) schemes. The spectrum of operations includes element-wise addition, subtraction, and multiplication for both encrypted–encrypted and encrypted–plain vectors, incorporating functionalities like dot product and vector–matrix multiplication.
- Palisade (Badawi et al., 2022): Palisade, currently part of OpenFHE, is a cryptography library proficient in implementing fully homomorphic encryption and multi-party extensions of fully homomorphic encryption. It offers support for a diverse range of schemes such as BFV, BGV (Brakerski et al., 2012), CKKS and FHEW (Ducas and Micciancio, 2015). This library also facilitates seamless integration into hardware accelerators.
- Pyfhel (Ibarrondo and Viand, 2021): Pyfhel, an acronym for Python for Homomorphic Encryption Libraries, serves as a framework supporting SEAL and Palisade as backends, accommodating BFV, BGV, and CKKS schemes for various operations including addition, subtraction, multiplication, and scalar product.

FedNIC provides some insights into the performance of the algorithms by comparing Paillier and TenSEAL in Section 4.

There are few FL frameworks that attempt to incorporate HE into FL. Some notable studies include:

- Python-Paillier (Lao et al., 2021): This library is a python3 implementation of the paillier homomorphic encryption library. It also provides a federated learning simulation framework that utilizes the paillier algorithm.

TABLE 2 A comparison of various types of SmartNICs.

	FPGA-based	ASIC-based	SoC-based
Programmability	Hard	Limited	Easy
# of cores	10+ cores	200+ cores	50+ cores
Accelerator support	Varies	Low	High
Secure key storage	Varies	Hard	Easy
Hardware cost	High	Low	Medium

- FedML-HE (Jin et al., 2023): FedML-HE is a research effort to extend FedML's privacy-preserving capabilities by adding methods for homomorphic encryption on clients.

FedNIC compares the performance of both of these frameworks to provide deeper insights into the benefits that FedNIC can provide.

2.3 SmartNICs

SmartNICs are a new class of network interface cards (NIC) that are built to run tasks that the CPU normally handles (e.g., checksum computation, TCP offload, and more) in addition to handling basic networking tasks. At the core of the SmartNICs are the main processing units that are tasked with processing the ingress packets and emitting them out on the egress. These processing units are often programmable to execute custom programs directly on the data plane, which ensures fast execution of the custom programs at the packet level. In addition to the processing unit, most SmartNICs are equipped with various accelerators that can be leveraged to further expedite widely used computing operations, such as encryption operations. For example, NVIDIA's popular BlueField (NVIDIA, 2024) SmartNIC incorporates accelerators for hardware root-of-trust for Secure boot, True random number generator (TRNG), compression and decompression acceleration, and more. They also allow for features like RDMA access to GPUs, showing huge potential for future uses in applications over multiple domains.

SmartNICs can be categorized into three different types based on the architecture of the processing unit and its processing capabilities: FPGA-, ASIC-, and SoC-based (Firestone et al., 2018). Table 2 highlights the difference between the types of SmartNICs. The first choice of SmartNIC for FedNIC is SoC-based SmartNICs, as shaded by gray in Table 2, mainly due to ease of programming and extensive support for cryptographic operations. In addition, given that SoC-based SmartNICs are built to run a complete operating system, it is able to perform secure boot, enabling capabilities to distribute and store encryption keys securely. Some examples of SoC-based SmartNICs are NVIDIA Bluefield series (NVIDIA, 2024) and AMD 400-G (Dastidar et al., 2023). This feature allows FedNIC's computation, such as the HE calculation, and the threat model to easily be incorporated as a system; thus, this study mainly utilizes SoC-based SmartNIC for evaluation. Yet, the design of FedNIC is not limited just to SoC-based SmartNIC, as other types of SmartNICs can be built to support the minimum set of functionalities that are needed for FedNIC.

3 FedNIC overview

In this section, we discuss a high-level overview of the components that make up FedNIC and the threat model that it assumes and the workflow.

3.1 FedNIC design overview

3.1.1 Adversary definition with threat model

We currently define the set of clients and roles required in FedNIC, the assumptions FedNIC makes about the threat model. Figure 1 illustrates the clients, interactions between clients, and the overall threat model of our proposed solution.

First, the main assumption is that both the clients, who are responsible for running the FL agent to train the local model with local private data, and the aggregator, who is responsible for collecting all local model weights and aggregating them, can be compromised. FedNIC assumes a semi-honest adversary $\mathcal{A}$, where semi-honest means that the adversary cannot deviate from the protocol, that can corrupt the aggregation server or any subset of local clients with local data. To elaborate, when $\mathcal{A}$ corrupts a client, the private information in local models and data are compromised by $\mathcal{A}$, but when $\mathcal{A}$ corrupts the aggregation server, $\mathcal{A}$ can try to, but cannot compromise private information from local models nor global models due to encryption. In other words, we assume a threat model where the participating clients are honest but curious (HBC), whereas the model aggregator can be compromised and dishonest.

Second, each client communicates with either a local or remote encryption engine to transfer the model weights, which then are encrypted by the encryption engine. The communication between the clients and the encryption engine is assumed to be encrypted. While there can be attacks like man-in-the-middle to compromise the local weight between the client and the encryption engine, the assumption is that it has the same effect as having a compromised client. The threat model assumes that the encryption engine is in a set of devices that are in a different security domain, booted securely, and thus can be trusted. The main difference of FedNIC is that it utilizes SmartNICs as the encryption engine, thereby enabling this assumption to hold due to their hardware capabilities.

Along with the clients, SmartNICs, and the aggregator, FedNIC assumes the existence of a trusted authenticator that cannot be compromised. The authenticator is an independent party that is trusted by participating devices and the aggregator. The main role of the authenticator is to generate and propagate a set of secure encryption keys for the encryption engine to use. We assume that communication channels between devices and the aggregator may be compromised; hence, attacks like man-in-the-middle and snooping can happen. However, the key provisioning and key distribution procedures, as well as the connections between the authenticator and the SmartNICs of the authenticator are considered secure.

3.1.2 Client placement

The participating client devices are assumed to be located at edge locations and can be any device that can run an FL agent on the host CPU. The SmartNICs can be local to clients or can be installed on edge servers that directly talk to the clients. The aggregator can either be situated at a specific edge location or deployed within the cloud infrastructure. Finally, the authenticator is situated at a predetermined secure server that can be deployed in a private or public cloud setting.

3.1.3 Workflow

Devices may join and leave the system at any given time. Upon joining, the HE Engine on a device needs to authenticate with the authenticator first before receiving the current key for homomorphic encryption and decryption. The FL agent that runs on the host CPU/GPU is a lightweight wrapper that works with a set of existing machine learning or deep learning frameworks [e.g., Scikit-learn (Pedregosa et al., 2011), TensorFlow (Abadi et al., 2015), and PyTorch (Paszke et al., 2019)], and no additional modifications are required to the existing model training code. Once the FL Agent completes one training iteration, the updated model parameters are sent to the encryption engine for encryption. Once encrypted, the encrypted model parameters are then sent to the aggregator. After receiving the encrypted model parameters from all participating devices, the model aggregator merges all updated parameters without decrypting them. Under this assumption, we require that the encryption be performed using one of the homomorphic encryption methods and to clarify the aggregator doesn't own the key for homomorphic encryption/decryption as it can also be compromised. For this study, the workflow does not consider the straggling clients and the client dropout problems in this abstract as there are several proposals in the existing literature to handle this issue (Park et al., 2021; Chai et al., 2020). After the model parameters are merged, the aggregated model parameters are returned to the encryption engine for decryption and then sent back to the FL agent on each client for the next set of training iterations. Finally, to further improve the security of FedNIC, the authenticator periodically examines the identity of each device and issues a new encryption key for homomorphic encryption/decryption. The overall algorithm for the workflow can be found in Algorithm 1.

4 Evaluation

In this section, we thoroughly evaluate performance of FedNIC in terms of communication time, encryption/decryption time, and

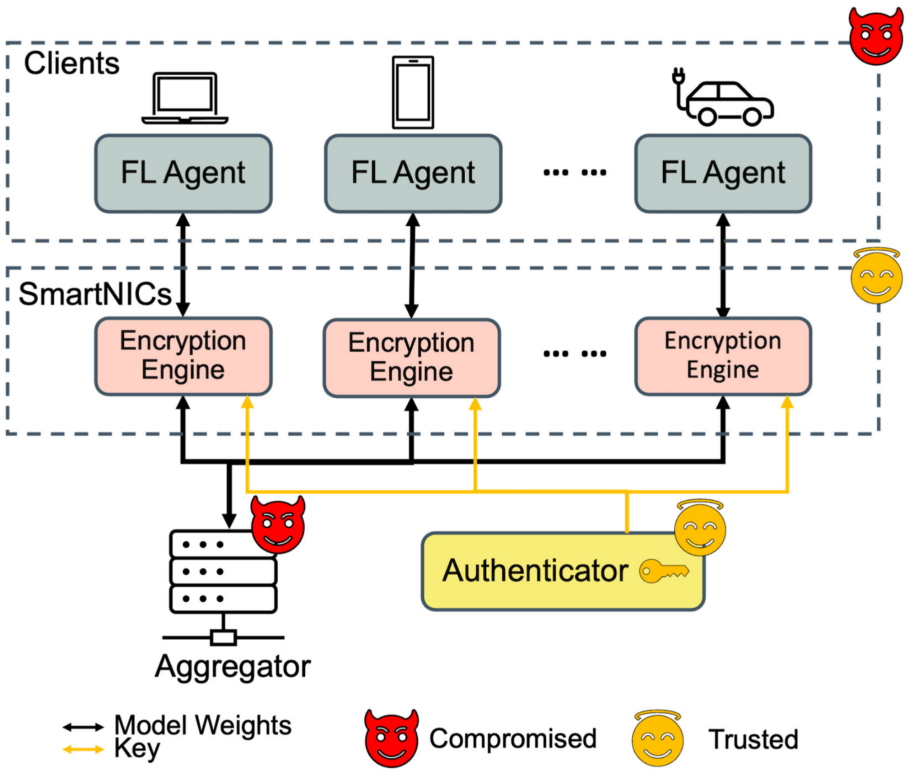

FIGURE 1
Clients and threat model of FedNIC.

aggregation latency with respect to the non-homomorphic solution (Raw), and without utilizing the SmartNIC (FedML-HE). We start by discussing the experimental methodology, implementation details, the testbed setup, and the results of the evaluation.

4.1 Evaluation methodology

4.1.1 Choice of federated learning framework

Different federated learning frameworks like Python-Paillier and IBM-FL were considered before going ahead with FedML-HE. Python-Paillier was unsatisfactorily slow, as the underlying FE algorithm, Paillier, is implemented in Python without extensive optimization for speed. The purpose of Python-Paillier is more focused on the simulation and testing of the Paillier algorithm, rather than having a production-ready FL system, thus Python-Paillier was not chosen as the base framework for FedNIC Another framework in consideration was the FL framework by IBM called IBMFL. This framework, even though it supported fully homomorphic encryption algorithms, it supported only ×86 and IBM Z architecture. This was a blocker for FedNIC, since Bluefields that are used in our experiments are equipped with ARM cores, thus runs the aarch64 version of Linux (NVIDIA, 2024). A future study can be done to utilize IBM FL with SmartNICs or networking devices that are ×86 or IBM Z based. The final framework that was evaluated was the FedML and the FedML-HE framework, which is a research study to enable HE on FedML. This framework integrates nicely with Palisade and TenSEAL HE algorithms and is built for scale to multiple clients. It also provided numerous communication protocols like gRPC, MPI, and MQTT that give FedNIC more flexibility on which protocols to use. Finally, FedML is open-sourced and written in Python, which makes it quite easy to add and modify modules of interest. Therefore, given the broad sets of benefits, we chose FedML framework as the base framework for FedNIC.

4.1.2 Choice of HE libraries

In the selection of homomorphic encryption algorithms/libraries, we considered a diverse set of tools to ensure a comprehensive evaluation of cryptographic techniques. The set of candidates, namely, Paillier, TenSEAL, Pyfhel, and Palisade, were chosen and evaluated to address the requirements of our FedNIC objectives. Each algorithm had its advantages and disadvantages, but the final choice of algorithm was Palisade due to the following reasons. While Paillier encryption is known for its additive homomorphic properties, particularly suited for scenarios requiring aggregated computations, the encryption performance was very low, often resulting in large weights and 40× slower encryption/decryption times. Pyfhel is recognized for its user-friendly interface and robust support for arithmetic operations in homomorphic encryption, but the encryption/decryption performance was 3× slower than Palisade. TenSEAL is designed to support polynomial-based encryption schemes, offering a flexible framework for polynomial

```
• $S$: Aggregator
• $C_i$: Client
• $E_{i'}$: Encryption engine paired with $C_i$
• $N$: Number of clients
• $M$: Number of encryption agent
• $D_i$: Local private data for Client $C_i$
• $W_i$: Unencrypted model weights
• $[W_i]$: Encrypted model weights

// Authenticator Generates Key
$(pk, sk) \leftarrow HE.KeyGen()$
for each $j \in [M]$ do
    Send $(pk, sk)$ to $E_j$
end

// Local FL Training on the Client
for t = 1,2,..., T do in parallel
    for each Client $i \in [N]$ do
        if $t = 1$ then
            $W_i \leftarrow Init(W)$
        end
        if $t > 1$ then
            Receive $W_{glob}$ from $E_{i'}$
            $W_i \leftarrow W_{glob}$
        end
        $W_i \leftarrow Train(Wi, D_i)$
        Send $W_i$ to $E_{i'}$
    end
end

// SmartNIC Operation
for each SmartNIC $i' \in [M]$ do in parallel
    // Encryption Operation
    if $W_i$ received from $C_i$ then
        $[W_i] \leftarrow HE.Enc(pk, W_i)$
        Send $[W_i]$ to $S$
    end
    // Decryption Operation
    if $[W_{glob}]$ received from $S$ then
        $W_i \leftarrow HE.Dec(sk, [W_{glob}])$
        Send $W_i$ to $C_i$
    end
end

// Aggregation on $S$
$[W_{glob}] \leftarrow \sum_{i=1}^{n} [W_i]$
Send $W_{glob}$ to all $E_j$ where $j \in M$
```

Algorithm 1. HE-based federated machine learning.

evaluation and manipulation, but the encryption/decryption performance was 2× slower than Palisade. Palisade is known for its scalability and comprehensive support for lattice-based homomorphic encryption schemes and showed the best encryption/decryption performance in our evaluation. Given all the choices, the final choice of algorithm is Palisade due to its superior performance. The results of this evaluation can be found in Section 4.4.1

4.1.3 Dataset and model

The dataset used for the evaluation is the FEMNIST dataset (Caldas et al., 2019). It is a variation of the popular MNIST dataset, a widely used dataset for image recognition tasks, which is built by partitioning the MNIST data based on the writer of the digit/character. The FEMNIST dataset consists of 805,263 samples of images of size 28 by 28 pixels, obtained from 3,550 users. The images are categorized into one of 62 classes (10 numbers, 26 lowercase, and 26 uppercase letters); thus, it is widely used to train image classification models that classify the handwritten image to one of the 62 classes. For easier comparison between clients, the training dataset has been sampled to 1,600 images for each client, and for each FL round, each client trains on 32 images before sending the weights for encryption.

The machine learning model that was built as part of the evaluation is convolutional neural network (CNN) with two layers (Krizhevsky et al., 2012) and dropout. CNN is also a widely used model for many classification task and is often coupled with the MNIST dataset to compare model training performance on different model and dataset configurations (Reddi et al., 2021). This model has 1,206,590 parameters trained in total, and each client trains the same number of parameters but on each of its local train dataset. For the evaluation, the client trains the given model for 100 epochs per FL training round, and the evaluation is run across 50 FL rounds. Given that the hyperparameters of the models and the type of HE algorithm stayed constant across different privacy-preserving FL systems (FedML vs. FedNIC), the model performance stayed constant across these systems. Table 3 provides an overview of the set of evaluations.

4.2 Implementation

The implementation of FedNIC is as follows. FedNIC is implemented as an extension to FedML-HE, where the homomorphic encryption part of the implementation has been replaced with FedNIC's encryption and decryption library. The FedNIC's encryption and decryption library works as follows. The encryption and decryption are performed via the Palisade homomorphic encryption library written in C++. The library uses gRPC as the communication mechanism between the client, SmartNICs, and the aggregation server to pass the encrypted weights. The CNN training logic that runs on each of the clients is written in Python using the PyTorch library. In summary, the process inside the SmartNIC, which is responsible for the encryption and decryption of model weights, utilizes gRPC to retrieve model weights from the clients and the aggregation server and calls the C++ Palisade library to perform the necessary crypto operations. The clients receive decrypted weights from the SmartNIC to continue on to the next iteration of the FL training.

TABLE 3 Overview of experimental methodology.

Type	Encry. alg.	Num params.	Weights (B)	Encry. weights (B)	FL rounds	Epochs
Raw	N/A	1,206,590	28,958,160	N/A	50	100
FedML	Palisade	1,206,590	28,958,160	78,764,304	50	100
FedNIC	Palisade	1,206,590	28,958,160	78,764,304	50	100

4.3 Testbed setup

The evaluation testbed setup shown in Figure 2 consists of two clients and one aggregator server. Both the clients and the aggregator are deployed on 3× HPE ProLiant DL385 servers with dual AMD EYPC 7F52 16-core, 32 threads CPUs running at 3.5 GHz. One of the clients is equipped with NVIDIA BlueField-2 (NVIDIA, 2024), which consists of 8× ARMv8 A72 running at 2.5 GHz, and 16 GB of RAM and the other is equipped with BlueField-3 (NVIDIA, 2024) SmartNIC, which consists of 16× ARMv8.2 A78 CPU running at 3.3GHz and 16 GB of RAM. The authenticator is running alongside the aggregator. Both the clients and the aggregator are configured with the Ubuntu 22.04 operating system and all Bluefields were configured with Ubuntu 22.04 Linux operating system.

4.4 Evaluation results

4.4.1 Comparison of different HE libraries

To understand the performance of homomorphic operations, we compared the averaged encryption and decryption time on two SoC-based SmartNICs, BlueField-2 (NVIDIA, 2024) and Bluefield-3 (NVIDIA, 2024), and AMD EPYC 7F52 CPU (AMD, 2024) using the four most popular homomorphic libraries: Paillier, TenSEAL, Pyfhel, and Palisade. Figure 3 shows our experimental results. First of all, regardless of the algorithm, homomorphic encryption is very time-consuming; thus, the majority of the end-to-end processing time on FedNIC is spent on this specific operation. It is reasonable to predict that this overhead will be much larger when training bigger language or computer vision models. This further highlights our motivation for offloading the homomorphic operations from the CPU to SmartNICs is to save the CPU for other critical business operations. Second, Paillier shows a much higher overhead than TenSEAL in all scenarios, demonstrating that the implementation of TenSEAL is more efficient. TenSEAL can be further fine-tuned by tweaking the values of several parameters including `poly_modulus_degree`, `coeff_mod_bit_sizes`, and `global_scale`. With fine-tuning, the results show that the decryption and encryption time of TenSEAL on BlueField-3 slightly outperforms the AMD EYPC 7F52 CPUs, indicating a very promising potential. Third, although in most cases the performance of SmartNICs yields a longer execution time of homomorphic operations, the improvement of BlueField-3 over BlueField-2 is significant. Lastly, Palisade encryption and decryption on Bluefield-3 provides huge improvements over AMD CPU. This result is exciting as this result does not take crypto accelerators into consideration and we expect the gap between SmartNICs and CPU to be wider when crypto accelerators are in place.

4.4.2 Communication time latency

In this set of experiments shown in Figure 4, we compared the communication time of the two clients and Bluefield-2 (if using FedNIC) on each client host when using non-homomorphic encryption (Raw), running homomorphic encryption on the clients (FedML-HE), and running homomorphic encryption on the Bluefield2 nodes (FedNIC). The communication time is obtained by summing all of the time spent on the network between the clients, SmartNICs (if exists), and the aggregator, across all FL rounds. Given that FedNIC adds an additional layer of communication between the client and the SmartNIC, whereas traditional FL with HE implementations do not need such communication, Figure 4 shows the impact of the overhead of adding such a layer. The total communication time added on FedNIC is about 54% higher vs. FedML-HE, due to the large overhead caused by increased ciphertext size. Further optimizations need to be done to improve this time, such as a better networking stack for each participant, using different transfer protocols, selective parameter encryption, and more.

4.4.3 Training time

Figure 5 shows the total training time taken in seconds on the clients using non-homomorphic encryption, running homomorphic encryption on clients and when running homomorphic encryption on the SmartNIC, averaged across 50 different experiment runs. Our experiments show a negligible increase in the total training time when running the FedNIC, when compared to no encryption solution or encryption directly on the clients. This is expected as FedNIC is designed to make near-minimal changes to the training algorithm on the clients.

4.4.4 Aggregation time

In the set of evaluations shown in Figure 6, we evaluate the aggregation time taken on the aggregation server using non-homomorphic encryption, running homomorphic encryption on the clients, and when running homomorphic encryption on the SmartNIC. Notice that the aggregation time increases greatly on both systems using homomorphic encryption due to the increased size of the ciphertext and the complexity of adding larger sets of model weights. In addition, given that the effect of an increase in ciphertext is seen on each client and is aggregated across every client, so as more clients are added to the total workflow, we expect the difference to increase. However, FedNIC does not exhibit any increase in aggregation time when comparing between running homomorphic encryption on the client or on the SmartNIC. This result is as expected as FedNIC is designed to make minimal changes to the aggregation server as well.

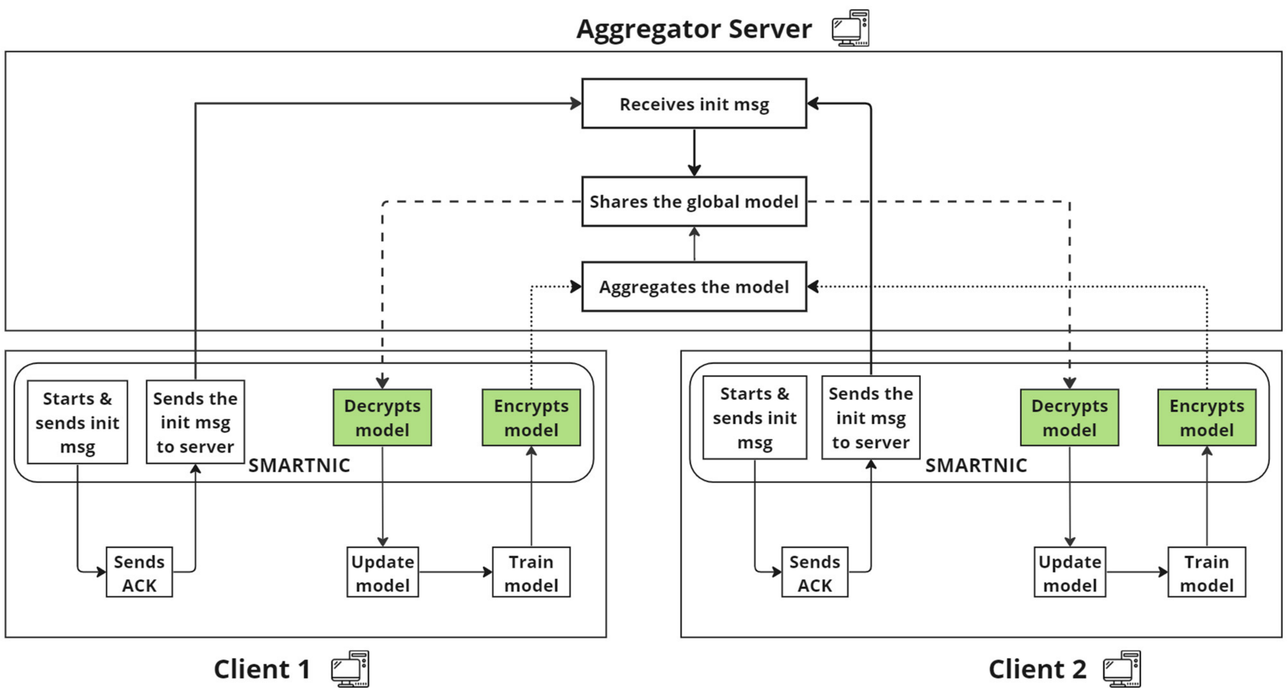

FIGURE 2
Overall experimental platform architecture.

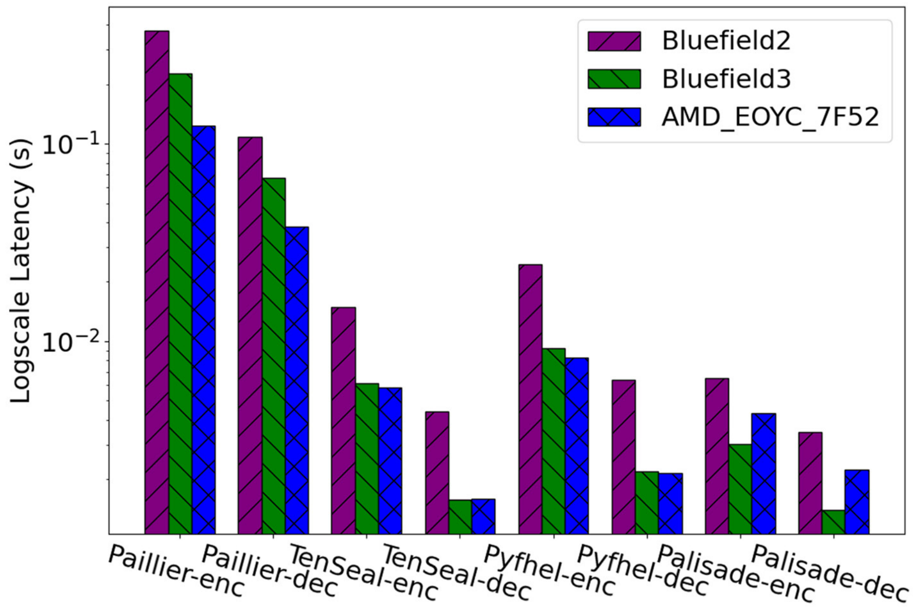

FIGURE 3
Encryption and decryption time of model weights on BlueField-2, BlueField-3, and AMD CPU using Paillier, TenSEAL, Pyfhel, and Palisade.

4.4.5 Total workflow time

We then compare the total time taken on each client, Bluefield-2 and aggregation server when running non-homomorphic, running homomorphic on clients, and running homomorphic encryption on the Bluefild-2, averaged across 50 different experiment runs. Figure 7 shows that the total time using the FedNIC-HE approach is 46% higher than the FedML-HE and 76% higher than the case where we do not use any encryption at all. The main reason for performance degradation is due to the added latency in sending the ciphertext to the SmartNIC, which adds a communication overhead between the host and the SmartNIC. Although, this added latency may not

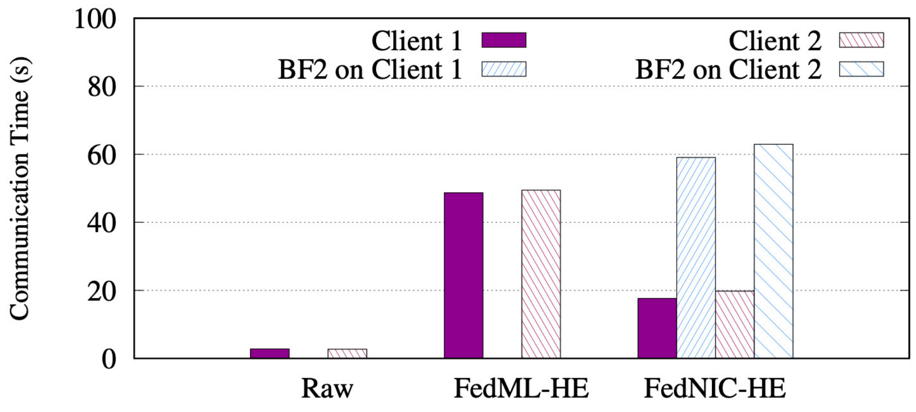

FIGURE 4
Communication time on the clients and BlueField-2, using non-homomorphic (Raw), running homomorphic encryption on clients (FedML-HE), and running homomorphic encryption on the Bluefield-2 (FedNIC).

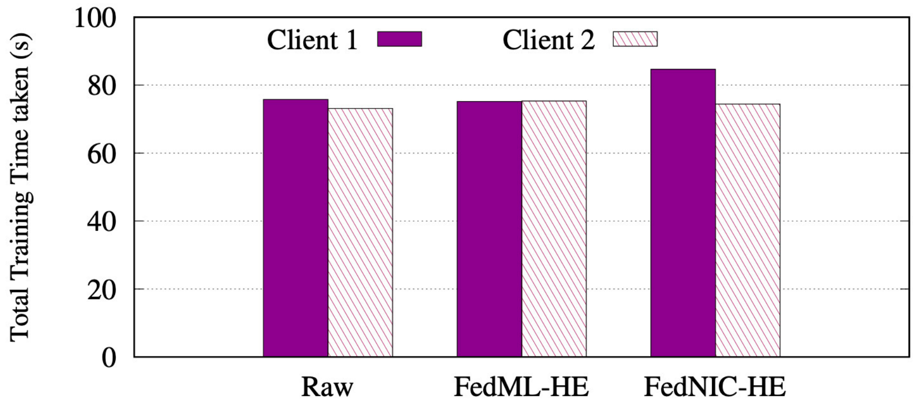

FIGURE 5
Total training time taken on the clients, using non-homomorphic, running homomorphic encryption on clients, and running homomorphic encryption on the Bluefield-2.

be desirable, we argue that such added latency allows a more secure federated machine learning framework to be achieved. As mentioned earlier, we believe that the optimizations on the networking stack and the aggregation methods can bring this gap even smaller. Furthermore, if the crypto accelerator is used, it is possible to offset the added communication latency with faster encryption/decryption operations.

5 Related studies

We review related research that explores privacy-enhancing techniques in FL and the utilization of hardware accelerators like SmartNICs.

5.1 Privacy-preserving methods

Existing privacy-preserving solutions for FL are mainly multi-party computation (MPC) secure aggregation protocols (Bonawitz et al., 2017; So et al., 2022), noise-based differential privacy (DP) solutions (Truex et al., 2019; Choudhury et al., 2019), and federated averaging with local randomization (McMahan et al., 2017). MPC protocols require extra steps to mask private inputs, and it does not work well with client dropouts. DP solutions add privacy noise to original inputs to prevent the reconstruction of individual data points. This may cause model performance degradation or convergence problems. In Federated Averaging with a local randomization approach, participants add random noise to their local model updates before sharing them with the central server. The noise addition introduces an element of privacy without relying on homomorphic encryption. The introduction of local randomization by adding noise to local model updates can adversely affect the convergence and accuracy of the federated learning model. The random perturbations may hinder the learning process, leading to slower convergence or reduced model performance, especially when the noise added is significant. There are works that employ a mixture of two or more of these methods. For example, FedML-HE (Jin et al., 2023) employ DP on top of HE that add noise to the encrypted weights for additional privacy. FedNIC can also implement such methods as encryption/decryption and model weight modification happens

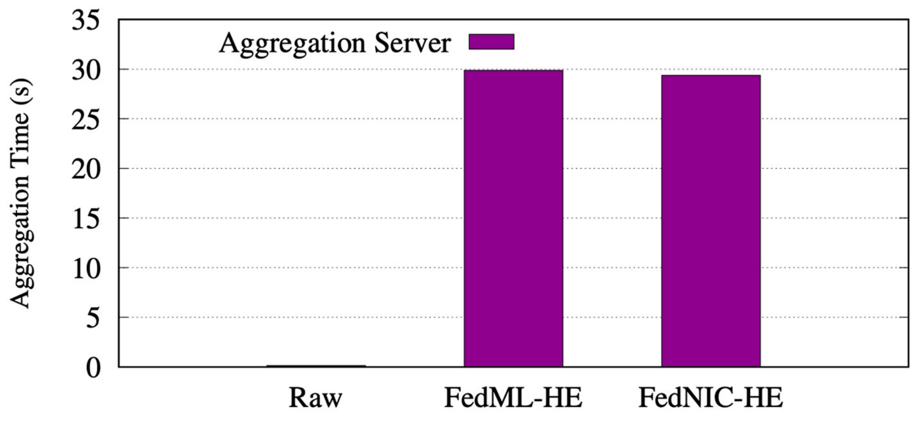

FIGURE 6
Aggregation time taken on the aggregation server, using non-homomorphic, running homomorphic encryption on clients, and running homomorphic encryption on the Bluefield-2.

entirely on the SmartNIC. The benefit of performing these privacy-preserving operation is that the entire operation can be completely hidden from the client, increasing the total security of the system.

Finally, there are more recent studies that are relevant to privacy preservation that can show the feasibility of FedNIC being deployed to real-world scenarios. First of all, Shitharth et al. (2023) utilize blockchain to enhance security for federated learning, which increases privacy for IoT use cases. A small change in FedNIC's architecture can easily support Shitharth et al. (2023) to offload the blockchain operations onto SmartNIC, which can be seen in studies by Patel and Choi (2023) and Kapoor et al. (2023). Another notable work is Yoosuf et al. (2022), which utilizes HE for data deduplication in cloud workloads. Yoosuf et al. (2022) shows that HE can be used in a real-world manner that supports the feasibility of FedNIC.

5.2 Homomorphic-based FL solutions

Existing homomorphic-based solutions adopt existing homomorphic libraries directly on the host leading to significant resource utilization overhead and may become the performance bottleneck of the FL system (Fang and Qian, 2021; Jiang et al., 2021; Zhang et al., 2020; Jin et al., 2023). BatchCrypt (Zhang et al., 2020) discusses the significant computation and communication costs incurred by homomorphic encryption and proposes a solution using batch encryption, for cross-silo federated learning to reduce the encryption and communication overhead of homomorphic encryption. FedNIC complements these efforts by providing a method to offload the HE operations that are currently run on the clients.

5.3 ML acceleration with programmable network devices

SmartNICs have been adopted to assist the FL system in aggregating ML model updates and mitigating communication overheads. A DPDK-based, lightweight communication protocol is introduced for the FL aggregation server in Shibahara et al. (2023) to accelerate model aggregation by leveraging the multi-core ARM processor on BlueField-2. Similarly, a new aggregation solution is proposed by Zang et al. (2022), achieving better performance and privacy of FL systems by offloading the key functions to FPGA-based SmartNICs.

More generally, recent studies have endorsed the utilization of in-network aggregation as a strategy to enhance the efficiency of distributed machine learning training (Sapio et al., 2021; Lao et al., 2021; Gebara et al., 2021). ATP delves into the concept of distributing aggregation functionality between a switch for enhanced performance and a server for increased capacity, aiming to seamlessly accommodate multi-job scenarios (Lao et al., 2021). However, the practical feasibility of these approaches is constrained by the absence of effective aggregation hardware, making them impractical for shared federated learning environments. To the best of our knowledge, none of the prior studies leverage SmartNICs/DPUs to offload and accelerate homomorphic operations to fortify the privacy-preserving aspect of federated learning systems.

5.4 Methods for efficiency improvement

A similar line of related studies is to enhance the efficiency of security-enhancing methods. A notable example is Gajarla et al. (2021), which enhances encryption efficiency by creating a sanitizer that is used to sanitize the block that contains the sensitive information of the file, instead of having to encrypt the entire file. This technique can be considered to be part of FedNIC to increase the efficiency of HE, since we can reduce the amount of data to be encrypted. Another example is the study by Karthikeyan et al. (2023) where it aims to improve the energy efficiency of cloud data center workloads by providing intelligent placement of workers. This is quite relevant to FedNIC, as it would allow FedNIC to reduce energy usage by placing the aggregation servers where it can provide the most energy efficiency.

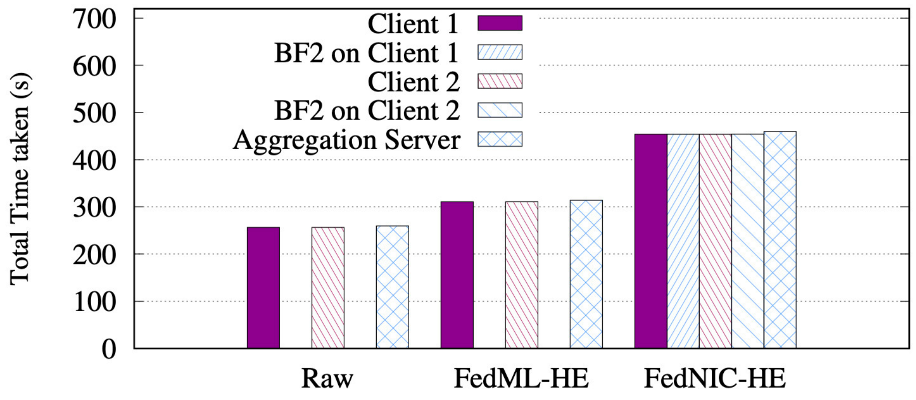

FIGURE 7
Total time taken on the clients, BlueField-2, and the aggregation server using non-homomorphic (Raw), running homomorphic encryption on clients (FedML-HE), and running homomorphic encryption on the Bluefield-2 (FedNIC).

6 Discussion and future studies

6.1 Future studies

6.1.1 Utilizing hardware crypto accelerators

The surge in SmartNIC adoption within large-scale data centers has led to the integration of various hardware acceleration engines. Notable examples include the utilization of hardware crypto engines in NVIDIA BlueField (NVIDIA, 2024) devices and dedicated compression/decompression engines. However, the current DOCA (NVIDIA, 2023) acceleration engine has limited support for a handful number of cryptographic algorithms such as RSA, DH, DSA, ECDSA, and ECDH; thus, it does not currently fully support homomorphic encryption. Thus, the once-homomorphic encryption is integrated into the DOCA framework, FedNIC can easily be enhanced by the hardware accelerator support, providing a significant potential for enhancing the overall efficiency, cost, and security. In addition, there is a series of Bluefield cards called Bluefield X that houses both a GPU and an NPU. While these cards are very limited in supply and hard to obtain, their architecture is very promising for accelerating FedNIC, since there is already active research around utilizing GPU to accelerate HE algorithms (Morshed et al., 2020; Özcan et al., 2023). Thus, FedNIC can easily utilize these methods to accelerate HE on SmartNIC.

The other possible future study direction is to leverage the existing compression/decompression engine on SmartNICs [e.g., BlueField-2 (NVIDIA, 2024)] to accelerate certain parts of the federated machine learning process. For example, previous study, by Bitar (Liu et al., 2022), shows promising results by leveraging the compression engine on NVIDIA BlueField-2 to accelerate the partitioning process of the datasets. Utilizing hardware acceleration offload method, Bitar is able to achieve $4.6 - 8.6\times$ higher throughput than software-based solutions for serialization. Given such success, a future study planned is to explore the possibility of leveraging the compression/decompression engine to accelerate corresponding steps of HE operations.

6.1.2 Data plane-assisted federated machine learning

Aside from processing the encryption and decryption using an SoC-based SmartNIC like Bluefield, there is potential to simply offload the entire operation onto a programmable data plane device like FPGA or ASIC-based SmartNICs or switches. Given that FPGA-based SmartNICs can be programmed to perform HE (Agrawal et al., 2023), a research area of interest is to build an FPGA or ASIC-based SmartNICs that already have HE acceleration features built into the data plane. With such features, it is possible to easily encrypt weights that are sent to the aggregation server, which can improve the performance of FedNIC greatly.

6.1.3 Selective encryption

One of the major issues with HE is the significant increase in the size of ciphertext. In order to reduce this impact, there are studies that employ selective encryption, which encrypts a subset of the model weights (Jin et al., 2023) to reduce the computational overhead in encryption. Employing this technique will allow the discrepancy between the total time taken for FedNIC vs raw to reduce significantly, further increasing the feasibility of FedNIC in production.

6.2 Discussion

6.2.1 Industry standards and shortcomings

Current industry standards in attacks and defenses for federated ML system is well discussed in Han et al. (2024). There are multiple types of attacks that are present, such as data/model poisoning attacks, backdoor attacks, and Byzantine attacks, and it is an active ongoing research to provide defenses for each type of attack. While algorithms such as Krum (Blanchard et al., 2017) can provide a way for trusted aggregation servers to perform trusted model updates in the presences of compromised clients, these algorithms are not effective when the aggregation server is compromised. FedNIC is an orthogonal method that allows for a compromised network and aggregation server to have zero information about what is sent by the client. In addition, FedNIC addresses short-comings when a client is compromised, as the SmartNIC is in a different security domain. The design of FedNIC prevents the client from knowing what happens to their data after it is sent. Thus, FedNIC can easily verify the validity of the data, as well as the current state of the client by investigating the data before encrypting it and sending it to the aggregation server. In summary, FedNIC enables a trusted authenticator that can improve the security of the entire system, while easily allowing the integration of algorithmic techniques as defined by Han et al. (2024).

6.2.2 Cost analysis

While it is difficult to perform a detailed cost analysis to understand what it costs to offset HE using SmartNICs, we provide a rough cost analysis based on the amount of CPU used vs. the cost of SmartNICs. Given that HE operations alone can use up to 25% of the CPU, this implies that adding HE to an existing federated ML system requires provisioning of 33% more CPUs. The AMD EPYC CPU that we have utilized for the evaluation testbed costs ∼$3,000 each, which means that adding HE capabilities alone would cost roughly ∼$1,000. A Bluefield-3 SmartNIC cost highly depends on the configuration and the speed of the port, but on average it is ∼$2,500 for a 100G version. A typical data center 100G NIC can cost roughly $1,000; thus, the added cost for adding a Bluefield would be $1,500 per server. We can see that adding a Bluefield costs ∼$500 more than just increasing the CPU, but with this is when ignoring added benefits by installing the Bluefield over a typical NIC. For example, Bluefields come with more RAM than a typical NIC and can improve the security of the system as mentioned by FedNIC. In addition, HE operations utilize ∼50% of the Bluefield's CPU; thus, there is processing power left for other operations. Therefore, we can see that the cost between adding more CPU to support HE operations vs. adding a Bluefield SmartNIC instead of a typical NIC is comparable.

7 Conclusion

In this paper, we present FedNIC, a privacy-preserving FL framework that offloads HE to SmartNICs to improve the security and feasibility of privacy-preserving FL via HE. Acknowledging the compute-intensive nature of HE and its potential impact on client CPU resources, leveraging the SmartNICs as hardware accelerators, to effectively offload HE operations, significantly improving computational efficiency and frees up valuable host resources. In addition, the FedNIC system design allows for a more secure exchange of keys and encryption, stopping adversaries from obtaining any information about the encryption process.

Experimental results show no model accuracy with a small increase in total training time, while conserving valuable host CPU cycles used for encryption and decryption. This research shows the feasibility and advantages of employing SmartNICs for HE in federated learning scenarios, contributing to the realization of more secure and privacy-preserving AI models. Furthermore, future studies show potential in improving the performance further by utilizing newer hardware that can perform HE computation far more efficiently with dedicated ASICs. As mentioned, FedNIC design is not only limited to HE but also can be applied to other, more efficient and well-supported, types of encryption algorithms. Thus, as the field of privacy-preserving FL continues to evolve, FedNIC emerges as a promising solution to address computational challenges and foster the widespread adoption of secure machine learning practices.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

SC: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. DP: Writing – review & editing, Writing – original draft, Visualization, Software, Data curation. DZ: Writing – review & editing, Writing – original draft, Visualization, Resources, Project administration, Investigation, Data curation, Conceptualization. LC: Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Investigation, Conceptualization. FA: Writing – review & editing, Writing – original draft, Resources, Conceptualization. PS: Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This project has been funded with the support Hewlett-Packard Enterprises and NSF CRII 2245352. The funder was not involved in the study design, collection, analysis, interpretation of data, the writing of this article, or the decision to submit it for publication.

Conflict of interest

DZ, LC, FA, and PS were employed at Hewlett Packard Enterprises.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Available at: tensorflow.org (accessed January 13, 2023).

Agrawal, R., de Castro, L., Yang, G., Juvekar, C., Yazicigil, R., Chandrakasan, V., et al. (2023). "Fab: an fpga-based accelerator for bootstrappable fully homomorphic encryption," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (Los Alamitos, CA: IEEE Computer Society), 882–895. doi: 10.1109/HPCA56546.2023.10070953

AMD (2024). *Server Processor Specifications.* Available at: https://www.amd.com/en/products/specifications/server-processor.html (accessed January 13, 2023).

Badawi, A. A., Bates, J., Bergamaschi, F., Cousins, D. B., Erabelli, S., Genise, N., et al. (2022). *Openfhe: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915.* Available at: https://eprint.iacr.org/2022/915 (accessed January 13, 2023).

Benaissa, A., Retiat, B., Cebere, B., and Belfedhal, A. E. (2021). Tenseal: a library for encrypted tensor operations using homomorphic encryption. *arXiv* [Preprint]. arXiv:2104.0315. doi: 10.48550/arXiv.2104.0315

Bhowmick, A., Duchi, J., Freudiger, J., Kapoor, G., and Rogers, R. (2019). Protection against reconstruction and its applications in private federated learning. *ArXiv.*

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. (2017). "Machine learning with adversaries: byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems, Vol. 30*, eds. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et al. (Red Hook, NY: Curran Associates, Inc).

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., et al. (2017). "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY: ACM), 1175–1191. doi: 10.1145/3133956.3133982

Brakerski, Z. (2012). "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology – CRYPTO 2012 - Vol. 7417* (Berlin, Heidelberg: Springer-Verlag), 868–886. doi: 10.1007/978-3-642-32009-5_50

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2012). "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12* (New York, NY: Association for Computing Machinery), 309–325. doi: 10.1145/2090236.2090262

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konecný, J., McMahan, H. B., et al. (2019). Leaf: A benchmark for federated settings. *ArXiv.*

Chai, Z., Ali, A., Zawad, S., Truex, S., Anwar, A., Baracaldo, N., et al. (2020). "Tifl: a tier-based federated learning system," in *Proceedings of the 29th international symposium on high-performance parallel and distributed computing* (New York, NY: ACM), 125–136. doi: 10.1145/3369583.3392686

Cheon, J. H., Kim, A., Kim, M., and Song, Y. (2017). "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology-ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23* (Cham: Springer), 409–437. doi: 10.1007/978-3-319-70694-8_15

Choudhury, O., Gkoulalas-Divanis, A., Salonidis, T., Sylla, I., Park, Y., Hsu, G., et al. (2019). Differential privacy-enabled federated learning for sensitive health data. *arXiv* [Preprint]. arXiv:1910.02578. doi: 10.48550/arXiv.1910.02578

Dastidar, J., Riddoch, D., Moore, J., Pope, S., and Wesselkamper, J. (2023). The amd 400-g adaptive smartnic system on chip: a technology preview. *IEEE Micro* 43, 40–49. doi: 10.1109/MM.2023.3260186

Ducas, L., and Micciancio, D. (2015). "Fhew: bootstrapping homomorphic encryption in less than a second," in Advances *in Cryptology-EUROCRYPT 2015*, eds. E. Oswald, and M. Fischlin (Berlin, Heidelberg: Springer Berlin Heidelberg), 617–640. doi: 10.1007/978-3-662-46800-5_24

Fang, H., and Qian, Q. (2021). Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet* 13:94. doi: 10.3390/fi13040094

Firestone, D., Putnam, A., Mundkur, S., Chiou, D., Dabagh, A., Andrewartha, M., et al. (2018). "Azure accelerated networking: smartnics in the public cloud," in *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation, NSDI'18* (Berkeley, CA: USENIX Association), 51–64.

Gajarla, B., Rebba, A., Kakathota, K., Kummari, M., and Shitharth, S. (2021). "Handling tactful data in cloud using pkg encryption technique," in *4th Smart Cities Symposium (SCS 2021), Vol.* 2021 (Bahrain), 338–343. doi: 10.1049/icp.2022.0366

Gebara, N., Ghobadi, M., and Costa, P. (2021). In-network aggregation for shared machine learning clusters. *Proc. Mach. Learn. Syst.* 3, 829–844.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. (2020). "Inverting gradients - how easy is it to break privacy in federated learning?" in *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20* (Red Hook, NY: Curran Associates Inc), 16937–16947.

Han, S., Buyukates, B., Hu, Z., Jin, H., Jin, W., Sun, L., et al. (2024). Fedsecurity: Benchmarking attacks and defenses in federated learning and federated llms. *ArXiv.* doi: 10.1145/3637528.3671545

Hatamizadeh, A., Yin, H., Roth, H., Li, W., Kautz, J., Xu, D., et al. (2022). "Gradvit: gradient inversion of vision transformersm," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (New Orleans, LA, IEEE), 10011–10020. doi: 10.1109/CVPR52688.2022.00978

He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., et al. (2020). Fedml: A research library and benchmark for federated machine learning. *ArXiv.*

Ibarrondo, A., and Viand, A. (2021). "Pyfhel: python for homomorphic encryption libraries," in *Proceedings of the 9th on Workshop on Encrypted Computing Applied Homomorphic Cryptography, WAHC '21* (New York, NY: Association for Computing Machinery), 11–16. doi: 10.1145/3474366.3486923

Inc., G. (2020). *Tensorflow federated.* Available at: https://www.tensorflow.org/federated (accessed January 13, 2023).

Jiang, Z., Wang, W., and Liu, Y. (2021). Flashe: additively symmetric homomorphic encryption for cross-silo federated learning. *arXiv* [Preprint]. arXiv:2109.00675. doi: 10.48550/arXiv.2109.00675

Jin, W., Yao, Y., Han, S., Joe-Wong, C., Ravi, S., Avestimehr, S., et al. (2023). Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system. *ArXiv.*

Kapoor, E., Jampani, G., and Choi, S. (2023). "Blocknic: smartnic assisted blockchain," in *2023 Silicon Valley Cybersecurity Conference (SVCC)* (San Jose, CA: IEEE), 1–8. doi: 10.1109/SVCC56964.2023.10165427

Karthikeyan, R., Sundaravadivazhagan, B., Cyriac, R., Balachandran, P. K., and Shitharth, S. (2023). Preserving resource handiness and exigency-based migration algorithm (PRH-EM) for energy efficient federated cloud management systems. *Mob. Inf. Syst.* 2023:7754765. doi: 10.1155/2023/7754765

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems, Vol. 25*, eds. F. Pereira, C. Burges, L. Bottou, and K. Weinberger (Red Hook, NY: Curran Associates, Inc), 1097–1105.

Lao, C., Le, Y., Mahajan, K., Chen, Y., Wu, W., Akella, A., et al. (2021). "ATP: in-network aggregation for multi-tenant learning," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)* (USENIX Association), 741–761.

Liu, J., Maltzahn, C., Curry, M. L., and Ulmer, C. (2022). "Processing particle data flows with smartnics," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)* (Waltham, MA: IEEE), 1–8. doi: 10.1109/HPEC55821.2022.9926325

Ludwig, H., Baracaldo, N., Thomas, G., Zhou, Y., Anwar, A., Rajamoni, S., et al. (2020). Ibm federated learning: an enterprise framework white paper v0.1. *ArXiv.*

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. (2017). "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics* (Proceedings of Machine Learning Research (PMLR)), 1273–1282.

Morshed, T., Aziz, M. M. A., and Mohammed, N. (2020). "CPU and GPU accelerated fully homomorphic encryption," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (San Jose, CA: IEEE), 142–153. doi: 10.1109/HOST45689.2020.9300288

NVIDIA (2023). *Nvidia doca software framework.* Available at: https://developer.nvidia.com/networking/doca (accessed January 13, 2023).

NVIDIA (2024). *NVDIA Bluefield Networking Platform.* Available at: https://www.nvidia.com/en-us/networking/products/data-processing-unit/ (accessed January 13, 2023).

Özcan, A. A., Ayduman, C., Türkoğlu, E. R., and Savaş, E. (2023). Homomorphic encryption on gpu. *IEEE Access* 11, 84168–84186. doi: 10.1109/ACCESS.2023.3265583

Paillier, P. (1999). "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology – EUROCRYPT '99*, ed. J. Stern (Berlin, Heidelberg: Springer Berlin Heidelberg), 223–238. doi: 10.1007/3-540-48910-X_16

Park, J., Han, D.-J., Choi, M., and Moon, J. (2021). Sageflow: robust federated learning against both stragglers and adversaries. *Adv. Neural Inf. Process. Syst.* 34, 840–851. doi: 10.5555/3540261.3540326

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *ArXiv.*

Patel, D., and Choi, S. (2023). "Smartnic-powered multi-threaded proof of work," in *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)* (Kuwait: IEEE), 200–207. doi: 10.1109/BCCA58897.2023.10338942

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* 12, 2825–2830.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., et al. (2021). Adaptive federated optimization. *ArXiv*.

Reddy, H. M., Sajimon, P. C., and Sankaran, S. (2022). "On the feasibility of homomorphic encryption for internet of things," in *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)* (Yokohama: IEEE), 1–6. doi: 10.1109/WF-IoT54382.2022.10152214

Sapio, A., Canini, M., Ho, C.-Y., Nelson, J., Kalnis, P., Kim, C., et al. (2021). "Scaling distributed machine learning with {In-Network} aggregation," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)* (USENIX Association), 785–808.

Shibahara, N., Koibuchi, M., and Matsutani, H. (2023). Performance improvement of federated learning server using smart NIC. *ArXiv*. doi: 10.1109/CANDARW60564.2023.00035

Shitharth, S., Manoharan, H., Shankar, A., Alsowail, R. A., Pandiaraj, S., Edalatpanah, S. A., et al. (2023). Federated learning optimization: a computational blockchain process with offloading analysis to enhance security. *Egypt. Inf. J.* 24:100406. doi: 10.1016/j.eij.2023.100406

So, J., He, C., Yang, C.-S., Li, S., Yu, Q., Ali, E., et al. (2022). Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. *Proc. Mach. Learn. Syst*. 4, 694–720. doi: 10.48550/arXiv.2109.14236

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., et al. (2019). "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM workshop on artificial intelligence and security* (New York, NY: ACM), 1–11. doi: 10.1145/3338501.3357370

Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., Qi, H., et al. (2018). Beyond inferring class representatives: user-level privacy leakage from federated learning. *arXiv* [Preprint]. arXiv:1812.00535. doi: 10.48550/arXiv.1812.00535

Yoosuf, M. S., Muralidharan, C., Shitharth, S., Alghamdi, M., Maray, M., and Rabie, O. B. J. (2022). Fogdedupe: a fog-centric deduplication approach using multi-key homomorphic encryption technique. *J. Sens*. 2022, 1–16. doi: 10.1155/2022/6759875

Zang, S., Fei, J., Ren, X., Wang, Y., Cao, Z., Wu, J., et al. (2022). "A smartnic-based secure aggregation scheme for federated learning," in *The 3rd International Conference on Computer Engineering and Intelligent Control* (CEUR-WS), 81–89.

Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., Liu, Y., et al. (2020). "{BatchCrypt}: efficient homomorphic encryption for {Cross-Silo} federated learning," in *2020 USENIX annual technical conference (USENIX ATC 20)* (IEEE), 493–506.

Zhu, L., Liu, Z., and Han, S. (2019). "Deep leakage from gradients," in *Advances in Neural Information Processing Systems, Vol. 32*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (Berkely, CA: Curran Associates, Inc), 14774–14784.