

BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain

Deepti Sharda¹, Deeksha Gupta¹, Navdeep Kaur¹, Ritika Bansal¹, 

1. Department of Computer Science and Applications, Mehr Chand Mahajan DAV College for Women, Chandigarh, IND

Received: June 18, 2026 | Review began: July 19, 2026 | Review ended: October 07, 2026 | Published: October 08, 2026

© Copyright 2026

This is an open access article distributed under the terms of the Creative Commons Attribution License CC-BY 4.0., which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract

With machine learning being used more and more in sensitive areas like healthcare and finance, protecting the privacy of training data has become a serious concern. Most existing approaches handle encryption, secure computation, and audit logging as separate steps, which creates gaps where data can be exposed. In this paper, we present BlockSecML, a framework that brings together three technologies - Advanced Encryption Standard (AES)-256-Galois/Counter Mode encryption, Paillier partially homomorphic encryption (PHE), and Hyperledger Fabric blockchain - into a single working pipeline for privacy-preserving federated machine learning. The core design is as follows: gradient computation occurs entirely in plaintext at each data owner node; the computed gradients are immediately encrypted under Paillier PHE before any transmission; and the aggregation server performs homomorphic addition on the encrypted gradients without decrypting any individual contribution. Two brief in-memory plaintext windows exist at the data owner node - during local gradient computation and during AES-to-Paillier re-encoding - both of which are confined to the data owner's own device and do not involve network transmission. Every significant pipeline event is logged as an immutable, hash-chained record on a Hyperledger Fabric ledger. Experiments were conducted on an Intel Core i5-10210U laptop with 20 GB RAM. BlockSecML achieves classification accuracy of $97.6 \pm 0.21\%$ on MNIST (Modified National Institute of Standards and Technology), $84.9 \pm 0.58\%$ on MIMIC-III (Medical Information Mart for Intensive Care), and $91.7 \pm 0.27\%$ on a synthetic financial fraud dataset, compared to plaintext baselines of 98.9%, 87.2%, and 93.4%, respectively. Training time overhead ranges from 116% to 179% relative to plaintext training, with PHE computation accounting for approximately 87% of this overhead. All figures represent upper-bound estimates due to the single-machine experimental setup. Blockchain logging adds under 100 ms per round. The code is available at <https://github.com/deeptisharda29/BlockSecML>.

Categories: Blockchain and Cryptocurrency, Security and Privacy, Machine Learning (ML)

Keywords: privacy-preserving machine learning, homomorphic encryption, aes-256, blockchain, hyperledger fabric, federated learning, gradient privacy, audit trail

Introduction

Machine learning systems are increasingly being used to make important decisions in domains such as medical diagnosis, fraud detection, and personalised recommendation systems [1]. In many of these applications, the training data contains sensitive personal information. Aledhari et al. [2] showed in their comprehensive survey that protecting this data during distributed model training is one of the central challenges in applied machine learning. When training data is

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

exposed during training or inference, it can cause serious harm to the individuals whose information is involved. This growing concern has led to the field of privacy-preserving machine learning (PPML), which aims to train and use machine learning models without exposing the underlying data to untrusted parties [3].

Several PPML approaches have been proposed. Differential privacy adds controlled noise to gradients or outputs to prevent individual data points from being identified [1]. Homomorphic encryption, introduced by Gentry [3], allows computation to be performed directly on encrypted data, so that the training server never receives plaintext inputs. However, even without access to raw data, gradient updates can leak sensitive information. Zhu et al. [4] demonstrated this by reconstructing training images directly from gradient values with high fidelity. Blockchain technology has also been applied to machine learning systems to create tamper-proof audit trails [5], ensuring that every significant operation is recorded on an immutable ledger that cannot be retroactively altered. Federated learning, formalised by McMahan et al. [6], keeps data on local devices and shares only model updates with a central server but does not by itself prevent gradient leakage. To address this, differentially private variants have been proposed by Wei et al. [7] and Abadi et al. [8], which add calibrated Gaussian noise to gradients before aggregation, though this reduces model accuracy in proportion to the privacy budget.

This paper presents BlockSecML, which aims to address the fragmentation between these approaches. We integrate Advanced Encryption Standard (AES)-256 in Galois/Counter Mode (GCM) encryption for data at rest and in transit, Paillier partially homomorphic encryption (PHE) for gradient aggregation, and Hyperledger Fabric blockchain for maintaining a tamper-proof event log, all within one coordinated pipeline. It is important to state the precise privacy guarantee upfront: gradient computation happens in plaintext at each data owner node, and only the computed gradients, not the raw data or feature vectors, are encrypted under Paillier before transmission. Two brief plaintext windows exist at the data owner node but do not involve any network transmission. The framework is intentionally designed to run on standard hardware. All experiments were conducted on an Intel Core i5-10210U laptop with 20 GB RAM.

We are transparent about what this work does not claim. The framework has been tested only on logistic regression and a simple convolutional neural network. PHE on a laptop CPU is slow, and training times are long. All components ran on the same physical machine. These limitations are discussed in detail in the 'Limitations' section. The main contributions are as follows: (1) a working three-layer privacy-preserving machine learning pipeline combining AES-256-GCM, Paillier PHE, and Hyperledger Fabric, with results reproducible on standard consumer hardware; (2) evaluation on three datasets across different application domains with honest reporting of overhead; (3) transparent discussion of limitations, including the two plaintext windows, the convolutional gradient limitation of the Convolutional Neural Network (CNN), and the single-machine experimental constraint; and (4) comparison with five related frameworks across six capability dimensions.

Materials And Methods

Federated learning and gradient privacy

Federated learning was proposed as a way to train machine learning models across multiple participants without requiring them to share raw data. Aledhari et al. [2] provide a comprehensive survey of federated learning architectures, protocols, and enabling technologies. McMahan et al. [6] formalised the FedAvg aggregation algorithm, which remains the standard approach. While federated learning protects raw data, Zhu et al. [4] showed that gradient information alone can be sufficient to reconstruct training images with high fidelity, fundamentally changing how gradient privacy must be addressed. Wei et al. [7] proposed a federated learning framework with differential privacy that adds Gaussian noise to local gradients before aggregation, providing both privacy guarantees and convergence analysis. Abadi et al. [8] introduced differentially private stochastic gradient descent, a differentially private training algorithm that provides formal epsilon-delta privacy guarantees. However, both approaches trade accuracy for privacy, and this trade-off is difficult to calibrate for small datasets.

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. *Cureus J Comput Sci* 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

Homomorphic encryption in machine learning

Gentry [3] established the theoretical foundations of fully homomorphic encryption (FHE), which allows arbitrary computation on ciphertext. However, FHE is still too slow for practical machine learning training. PHE, such as the Paillier scheme [9], supports only additive operations, but this is sufficient for gradient aggregation in federated learning. Gradients are computed in plaintext locally and encrypted before being sent to the aggregation server, which sums them homomorphically. Gilad-Bachrach et al. [10] demonstrated neural network inference on encrypted data, achieving competitive accuracy but with significant latency overhead. The SEAL library [11] from Microsoft implements the BFV (Brakerski-Fan-Vercauteren) and CKKS (Cheon-Kim-Kim-Song) schemes and has been widely used in research. Marcolla et al. [12] conducted a comprehensive survey of HE applications in machine learning and concluded that PHE-based approaches for aggregation are the most practically feasible option at present. Phong et al. [13] implemented PHE-based gradient aggregation for logistic regression using the canonical approach: gradients computed locally in plaintext, then encrypted for server-side aggregation using additive homomorphism.

Blockchain for machine learning integrity

Weng et al. [14] proposed DeepChain, which combines blockchain with federated learning. Lu et al. [15] integrated differential privacy with a blockchain-based architecture for industrial IoT federated learning. Kuo et al. [16] explored Hyperledger Fabric for logging model provenance in healthcare machine learning systems, demonstrating manageable blockchain overhead. Awan et al. [17] reported sub-second chaincode execution times in private Fabric networks. Rahman et al. [18] found that public blockchains introduce prohibitive latency for machine learning workflows, reinforcing the appropriateness of permissioned chains.

Combined PPML frameworks

Yang et al. [19] introduced a foundational taxonomy of federated machine learning covering horizontal, vertical, and transfer federation settings, but their work does not address cryptographic gradient protection or audit logging. Phong and Phuong [20] showed that combining AES for transmission with PHE for gradient aggregation reduces communication overhead by approximately 34% compared to transmitting all values under full HE. Fu et al. [21] proposed a verifiable federated learning framework for industrial IoT that uses Lagrange interpolation to verify gradient correctness while keeping gradient values encrypted. A systematic review by Niknam et al. [22] identified that no existing system provides all three capabilities simultaneously: encrypted data storage, encrypted gradient transmission, and tamper-proof audit logging. BlockSecML addresses this gap. Our synthetic financial dataset follows a similar privacy-first design, supporting the goal of secure multi-party model training in finance.

Research gap

The literature review reveals four specific gaps. First, privacy in most existing systems is applied only at certain pipeline stages. Raw data may be encrypted at rest, but gradients are often transmitted without cryptographic protection. BlockSecML encrypts computed gradients under Paillier before any network transmission, though two brief in-memory plaintext windows exist at the data owner node, as noted in the section on 'Workflow and Plaintext Windows'. Second, audit trails in most PPML systems are either absent or stored in centralised logs that can be modified. Blockchain-based logging provides tamper-proof records but has not been integrated with PHE-based gradient aggregation in prior work. Third, homomorphic encryption is often treated as an all-or-nothing solution. PHE for aggregation is feasible but has not been evaluated with blockchain logging on consumer hardware. Fourth, evaluations in the existing literature are almost exclusively conducted on high-performance infrastructure. We address this by running all experiments on an i5-10210U laptop, presenting the results as upper-bound estimates under co-location conditions.

Proposed model/system architecture

Design Goals

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

The primary design goal of BlockSecML is to ensure that gradient values are never transmitted between nodes in plaintext. A secondary goal is to maintain a verifiable, immutable record of every significant operation. A third goal is to keep the system implementable on standard hardware. BlockSecML is a prototype that demonstrates that the integration is technically possible and gives an honest account of the associated costs. It is explicitly not designed for large-scale production deployment.

Overview of the Four Layers

BlockSecML is organised into four layers, as shown in Figure 7. Data flows vertically from the Data Owner Node at the bottom through the Encryption and Transmission Layer and the Machine Learning Layer, while audit records branch to the Blockchain Audit Layer at each key event. No raw data, feature vectors, or model weights ever reach the blockchain; it receives only SHA-256 hashes of event payloads and gradient norms.

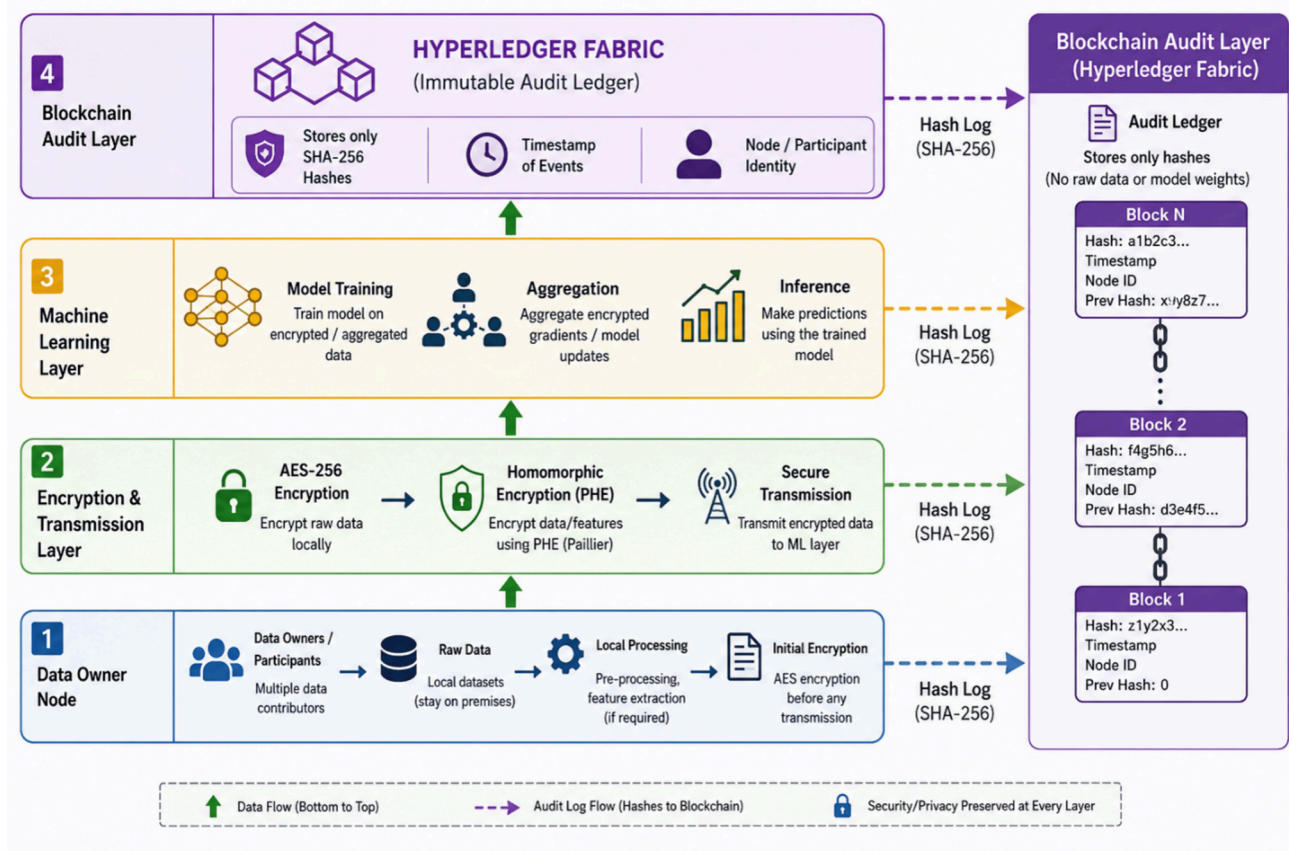


FIGURE 1: BlockSecML System Architecture—Four-Layer Stack

AES-256, Advanced Encryption Standard 256-bit; PHE, Partially Homomorphic Encryption; SHA-256, Secure Hash Algorithm 256-bit

Layer-by-Layer Description

Layer 1 - Data Owner Node: The Data Owner Node is where all gradient computation occurs. It runs a Python client that handles data ingestion, pre-processing, local model training, and gradient encryption. Raw records are normalised and stripped of non-essential columns. AES-256-GCM encryption is applied to the full data batch for storage. For the federated training step, the data owner uses its locally held training data to compute model gradients entirely in plaintext using standard backpropagation. This is the canonical approach established by Phong et al. [13] and is the only cryptographically correct way to use Paillier PHE for gradient aggregation, since Paillier supports only additive operations and cannot evaluate sigmoid or other non-linear functions on ciphertexts. Immediately after computation, the plaintext

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

gradient vector is encrypted element-wise under the Paillier public key. The plaintext gradients exist only transiently in the data owner's local memory during this step and are overwritten immediately after Paillier encryption. The data owner also computes the L2 norm of the plaintext gradient before encryption. This norm is transmitted alongside the encrypted gradient to the blockchain for anomaly detection purposes, as described in the section on 'Gradient Norm Anomaly Detection'. AES session keys are retrieved from the Key Management Service hosted within the Fabric network and stored only in process memory.

Layer 2 - Encryption and Transmission: This layer manages two parallel data paths. The first path carries Paillier-encrypted gradient vectors to the machine learning layer. These are the gradients that were computed in plaintext at the data owner node and immediately encrypted. No feature vectors are transmitted to the machine learning layer; the machine learning layer receives only encrypted gradient values. The second path carries AES-encrypted metadata, including participant pseudonyms, batch sequence numbers, timestamps, and the plaintext gradient norm value, directly to the blockchain chaincode. The gradient norm is transmitted in plaintext to the chaincode because it carries no sensitive information about the gradient direction or training data and is needed by the chaincode for anomaly detection. The orchestration script enforces a coupling constraint: a gradient batch is forwarded to the machine learning layer only after the corresponding metadata has been successfully committed to the Fabric ledger.

Layer 3 - Machine Learning: This layer uses Flower version 1.6 as the federated learning framework. For tabular datasets (Medical Information Mart for Intensive Care (MIMIC-III) and financial fraud), scikit-learn logistic regression is used. For Modified National Institute of Standards and Technology (MNIST) database, a two-layer CNN is used in TensorFlow. The aggregation server receives Paillier-encrypted gradient vectors from all data owner nodes. It exploits the additive homomorphism of the Paillier scheme: $\text{Enc}(a)$ multiplied by $\text{Enc}(b)$ modulo n^2 equals $\text{Enc}(a+b)$, to sum all participants' encrypted gradients without decrypting any individual contribution. Only the summed aggregate ciphertext is decrypted, yielding the mean gradient across all participants. For logistic regression, all gradient values are encrypted under Paillier before transmission. For the CNN, only the fully connected layer gradients are encrypted under Paillier; convolutional layer gradients are transmitted under AES-256-GCM and decrypted at the aggregation server before summation. This is a deliberate practical compromise to keep training time manageable on the laptop, following the approach of Gilad-Bachrach et al. [10], who identified that fully connected layer gradients are most directly linked to individual training data. This limitation means that convolutional layer gradients are potentially susceptible to gradient inversion attacks if the aggregation server is compromised, as discussed in the section on 'Resistance to Gradient Inversion Attacks'. After each round, the SHA-256 hash of the updated model parameters is sent to the blockchain before the model is distributed to participants.

Layer 4 - Blockchain Audit Layer: Hyperledger Fabric 2.5 with three peer organisations, one Raft ordering service, and one application channel provides the audit infrastructure. The chaincode is written in Go and exposes five functions: `RegisterData()`, `InitTraining()`, `SubmitGradient()`, `AggregateModel()`, and `LogInference()`. Each invocation writes a JSON record containing the event type, participant pseudonym, SHA-256 hash of the payload, hash of the previous event, a sequence number, and a timestamp. High-stakes operations require unanimous endorsement from all three peer organisations. Table 7 summarises the chaincode functions. It should be noted that `SubmitGradient()` and `AggregateModel()` require all three organisations (Org1, Org2, and Org3). `InitTraining()` requires two organisations (Org1 and Org2), as it records only round parameters, not cryptographic artefacts.

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. *Cureus J Comput Sci* 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

Function	Trigger	Fields Recorded	Endorsement
RegisterData()	Dataset uploaded	dataset_hash, owner_id, timestamp	Org1 OR Org2
InitTraining()	Round starts	round_id, participant_count, model_version	Org1 AND Org2
SubmitGradient()	Gradient received	gradient_hash, participant_id, round_id, sequence_id	Org1 AND Org2 AND Org3
AggregateModel()	Model updated	model_hash, round_id, gradient_norm	Org1 AND Org2 AND Org3
LogInference()	Inference served	query_hash, requester_id, model_version, response_hash	Org1 OR Org2

TABLE 1: Chaincode Functions and Endorsement Policies

Note: SubmitGradient() and AggregateModel() require unanimous endorsement from all three peer organisations (Org1 AND Org2 AND Org3) because they record cryptographic artefacts (gradient hashes and model hashes). InitTraining() requires only two-organisation endorsement (Org1 AND Org2) because it records only round parameters. This distinction is fully consistent with the Layer 4 description in section 'Proposed Model/System Architecture'.
Org, Organisation

Threat Model

The system is designed primarily under a semi-honest (honest-but-curious) threat model. Participants follow the protocol correctly and do not deviate from the prescribed gradient submission procedure but may attempt to infer information from what they legitimately receive. The aggregation server may try to deduce individual gradient values from the encrypted aggregate but will not modify the aggregation protocol. The gradient norm anomaly detection mechanism described in the section on 'Gradient Norm Anomaly Detection' is an additional defensive layer that extends partial protection beyond the semi-honest model toward detecting one simple class of malicious behaviour: submissions with abnormal gradient magnitudes. This does not constitute full Byzantine fault tolerance. A sophisticated malicious participant who keeps gradient norms within the expected range while still poisoning the model would not be detected by this mechanism, and full Byzantine robustness is identified as future work. External network adversaries are mitigated through AES-256-GCM encryption on all inter-node channels and mutual transport layer security authentication for all gRPC connections within the Fabric network.

Workflow and Plaintext Windows

Figure 2 shows the complete step-by-step workflow. Two plaintext windows at the data owner node are explicitly shown. The first is during local gradient computation (Step 2), where the data owner runs standard backpropagation on its local training data in plaintext. The second is during the AES-to-Paillier re-encoding step that occurs between AES-encrypted storage and Paillier-encrypted transmission. Both windows are confined entirely to the data owner's local device memory and do not involve any network transmission. These windows are a known limitation of the current implementation and are acknowledged in the abstract.

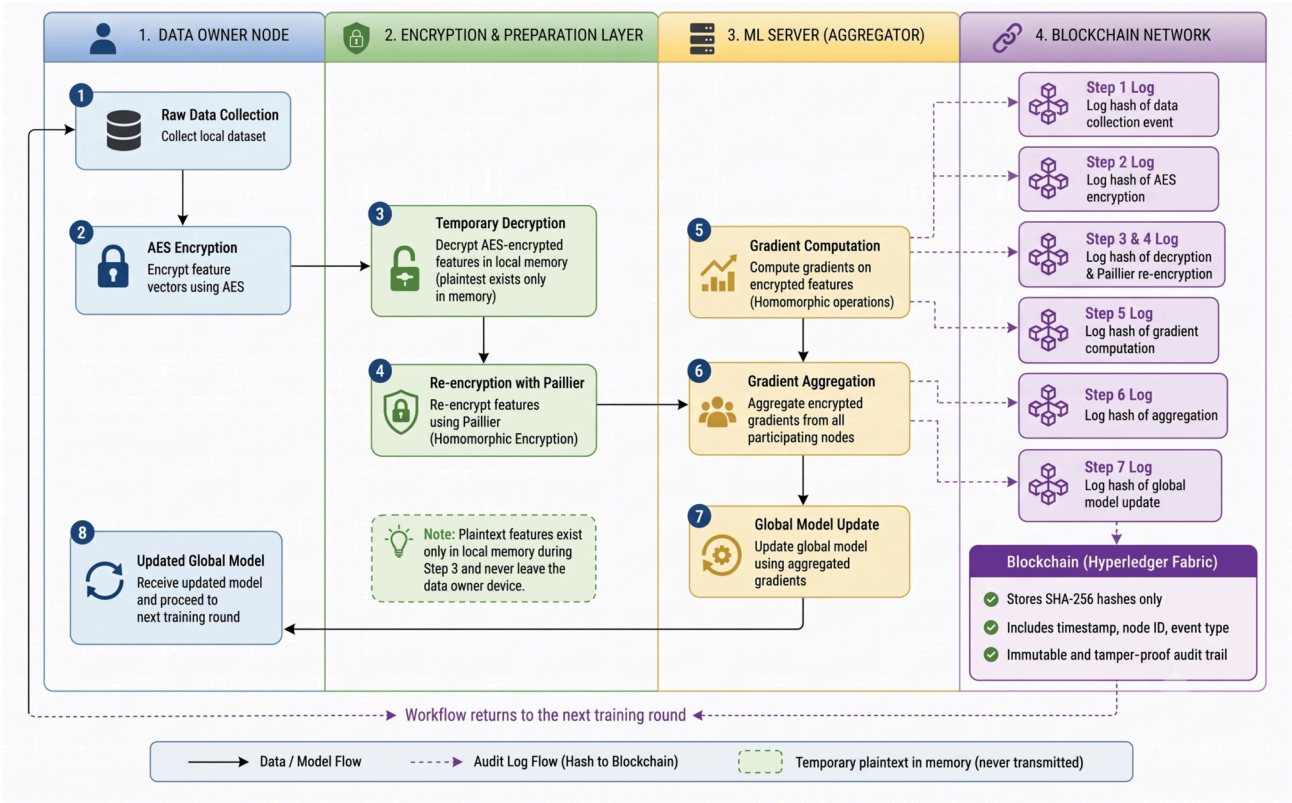


FIGURE 2: BlockSecML Operational Workflow—Swimlane Diagram
AES, Advanced Encryption Standard; SHA-256, Secure Hash Algorithm 256-bit

Comparison With Related Frameworks

Table 2 compares BlockSecML with five related frameworks across six capability dimensions. The gradient inversion resistance entry for BlockSecML is marked as Partial because full PHE protection applies only to logistic regression gradients and CNN fully connected layer gradients; convolutional layer gradients are decrypted at the aggregation server before summation, as discussed in the section on 'Resistance to Gradient Inversion Attacks'. The multi-domain validation entry reflects that two domains use real datasets and one uses a Conditional Tabular Generative Adversarial Network (CTGAN)-generated synthetic dataset.

Capability	BlockSecML	CryptoNets [10]	DP-SGD [8]	DeepChain [14]	Yang et al. [19]	Lu et al. [15]
Encrypted data at rest	Yes (AES)	No	No	Partial	No	No
Encrypted gradient compute	Yes (PHE)	Yes (LHE)	No	No	No	No
Blockchain audit trail	Yes (Fabric)	No	No	Yes	No	Partial
Gradient inversion resistant	Partial (FC Layer only)	Yes (inference)	Partial	No	No	No
Feasible on consumer CPU	Partial	No	Yes	Yes	Yes	Yes
Multi-domain validation	Yes (2 Real + 1 Synthetic)	No	Partial	No	Partial	No

TABLE 2: Comparison of BlockSecML With Related Frameworks

AES, Advanced Encryption Standard; DP-SGD, Differentially Private Stochastic Gradient Descent; FC, Fully Connected; LHE, Leveled Homomorphic Encryption; PHE, Partially Homomorphic Encryption

Methodology

Datasets

Three datasets were used to test BlockSecML across different application domains. The first is the MIMIC-III clinical database [23], developed by the MIT Laboratory for Computational Physiology. We used a de-identified subset of 4,800 ICU patient records with 18 clinical features for 30-day hospital readmission prediction. Access to MIMIC-III requires completion of the CITI Data or Specimens Only Research training course and acceptance of a data use agreement through PhysioNet (<https://physionet.org/content/mimiciii/1.4/>). No additional ethics approval was required for use of this publicly available de-identified dataset, consistent with PhysioNet data use agreement terms. The second is a synthetic financial fraud dataset generated using CTGAN [24], consisting of 100,000 records with 20 features. It is acknowledged that CTGAN-generated data may not fully capture the distributional complexity of real financial fraud data, which limits the generalisability of results in this domain. Evaluation on a real financial dataset is identified as future work, subject to data availability. It is worth noting that more hardware-efficient homomorphic encryption libraries such as OpenFHE [25] exist and are expected to reduce PHE overhead significantly compared to python-paillier; however, python-paillier was selected for this prototype due to its straightforward Python integration. For image classification, the MNIST handwritten digit dataset introduced by LeCun et al. [26] was used, consisting of 60,000 training images and 10,000 test images of size 28 × 28 pixels in greyscale. All datasets were normalised using min-max scaling. Missing values in MIMIC-III were filled using median imputation. Class imbalance was addressed using Synthetic Minority Over-sampling Technique on the training partition only, to prevent information leakage from the test set [27]. All code is available

How to cite this article:

at <https://github.com/deeptisharda29/BlockSecML>. The authors completed the CITI Data or Specimens Only Research training course and accepted the PhysioNet data use agreement prior to accessing this dataset. Only the publicly available de-identified subset was used; no additional institutional ethics approval was required.

Encryption Setup

For AES encryption, PyCryptodome version 3.18 was used with AES-256 in GCM, a mode whose security and performance properties were formally established by McGrew and Viega [28], who demonstrated its resistance to forgery and its suitability for high-throughput authenticated encryption. A fresh 96-bit initialisation vector was generated for each data batch using Python's `os.urandom()` function. The GCM authentication tag was stored as part of the encrypted bundle, providing both confidentiality and integrity verification on decryption. For Paillier PHE, the `python-paillier` library version 1.5.0 was used with a 2,048-bit key pair. A 2,048-bit RSA-equivalent modulus provides approximately 112 bits of symmetric security strength, which Barker and Roginsky [29] identify as the minimum acceptable level for long-term cryptographic operations. The public key is shared with all data owner nodes at the start of training. The private key stays with the aggregation server and is never transmitted over the network. Gradient values were scaled to integers by multiplying by 10,000 and rounding before Paillier encryption. This integer quantisation introduces a small approximation error, which contributes to the accuracy drop observed in Results and Discussion. Encrypting the gradient values for one MIMIC-III participant per round takes approximately 11 seconds on the i5-10210U. For MNIST, encrypting the fully connected layer gradient values per participant per round takes approximately 28-32 seconds.

Machine Learning Models

For MIMIC-III and the financial fraud dataset, scikit-learn logistic regression with the saga solver and L2 regularisation was used. For MNIST, a CNN with two convolutional layers (32 and 64 filters, 3×3 kernels with ReLU activation), max pooling after each convolutional layer, and one fully connected layer of 128 units followed by a softmax output layer was implemented in TensorFlow 2.12. Training used mini-batch stochastic gradient descent with a learning rate of 0.01 and a batch size of 64. Federated learning was orchestrated using Flower version 1.6 with three simulated participants, each holding a disjoint partition of the training data. For logistic regression, all gradient values are encrypted under Paillier before transmission, following the canonical approach of Phong et al. [13]. For the CNN, only the fully connected layer gradients are encrypted under Paillier; convolutional layer gradients are transmitted under AES-256-GCM. This difference is discussed further in the security analysis.

Blockchain Setup

Hyperledger Fabric 2.5 [30] was deployed locally using Docker Compose on the same laptop used for machine learning training. The network comprised three peer nodes (Org1, Org2, Org3), one Raft ordering node, and one application channel. The chaincode was written in Go 1.21. All three peers and the ordering node ran simultaneously with the machine learning training process, sharing the four CPU cores of the i5-10210U. This arrangement is not recommended for production but reflects the constraints of a single-machine prototype. Recent work has shown that federated learning can detect financial fraud while keeping raw transaction data local [31]. Our synthetic dataset follows a similar privacy-first design for fraud patterns. This approach supports the broader goal of secure multi-party model training in finance, where regulatory constraints often prevent raw transaction data from leaving the institution that generated it.

Evaluation Protocol

Classification accuracy was measured on the held-out test partition after 50 training rounds were completed. All metrics were averaged across five independent runs with different random seeds, and standard deviations are reported. Per-round training time was measured as the wall-clock time from gradient computation start to receipt of updated model parameters by all participants. Inference latency was measured end-to-end, including AES decryption of the query, model prediction, and `LogInference()` chaincode invocation, under varying concurrent user loads using Python's `concurrent.futures` module. Blockchain logging overhead was measured as the time from sending a chaincode invocation to receiving the transaction commit event.

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. *Cureus J Comput Sci* 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

Implementation

Experimental Setup

Table 3 provides a complete summary of the hardware and software environment. All components of BlockSecML ran on this single machine. No cloud resources, GPU, or additional hardware were used. This setup was chosen deliberately to produce results that are reproducible by other researchers working with standard consumer hardware.

Component	Details	Purpose
Processor	Intel Core i5-10210U @ 1.60GHz (boost 2.11 GHz), 4 cores	All computation
RAM	20 GB DDR4	Model + Fabric containers
Operating System	Ubuntu 22.04 LTS	Host environment
Python	3.10	ML and orchestration code
ML Framework	TensorFlow 2.12, scikit-learn 1.3	Model training
Federated Learning	Flower (flwr) 1.6	FL orchestration
HE Library	python-paillier 1.5.0	PHE gradient encryption
Symmetric Encryption	PyCryptodome 3.18 (AES-256-GCM)	Data storage and transit
Blockchain	Hyperledger Fabric 2.5 (Docker Compose)	Audit trail
Chaincode Language	Go 1.21	Smart contracts

TABLE 3: Implementation Environment

AES-256, Advanced Encryption Standard 256-bit; FL, Federated Learning; GCM, Galois/Counter Mode; HE, Homomorphic Encryption; ML, Machine Learning; PHE, Partially Homomorphic Encryption

Single-Machine Constraints and Upper-Bound Framing

All reported training times and latency figures should be interpreted as upper-bound estimates under co-location conditions rather than as isolated cryptographic costs. In this single-machine setup, the machine learning training process, three Hyperledger Fabric peer containers, the ordering node, and the inference server shared the four cores of the i5-10210U. CPU scheduling contention between these processes contributes to the reported overhead and cannot be separated from the cryptographic cost of PHE operations in the current experimental setup. In a real multi-machine deployment, the Fabric peers would run on separate infrastructure, and the machine learning training process would have exclusive access to the host CPU. Training times and latency figures would be lower under such conditions, with the difference attributable primarily to CPU scheduling overhead rather than PHE cryptographic cost. The reported PHE overhead of approximately 87% of total overhead should therefore be treated as an upper-bound estimate. We present these numbers as worst-case figures appropriate to a single-machine prototype and identify multi-machine evaluation as the primary direction for future experimental work.

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

In the Flower simulation framework, participant training runs sequentially rather than in true parallel, as all processes share the same four CPU cores. The per-round training time of 61.8 seconds for MNIST reflects this sequential execution. PHE encryption per participant takes approximately 28-32 seconds. Three participants executing sequentially contribute 84-96 seconds of PHE time, but the reported 61.8 seconds reflects overlapping idle periods during aggregation and blockchain logging steps within each round. In a true parallel deployment across three separate machines, each participant's PHE time would run concurrently, and the total round time would be closer to 35-45 seconds.

Implementation Notes

All Paillier encryption operations ran on CPU. The python-paillier library does not support GPU acceleration. The chaincode invocation latency averaged 87 milliseconds per round, consistent with Awan et al. [17] for similar Fabric configurations. Peak memory usage during training was approximately 14 GB, within the 20 GB available. Several simplifications were made in this implementation. All three simulated participant nodes ran on the same physical machine. The Key Management Service was implemented as a Python script rather than a hardware security module. The anomaly detection mechanism was tested by manually injecting outlier gradients during development and observing their rejection by the norm-threshold check. This was an implementation-level development observation and not a formal security experiment with defined metrics for the detection rate or false positive rate.

Results And Discussion

Experimental setup summary

All results were obtained on the Intel Core i5-10210U laptop described in Table 3. Each experiment was run five times with different random seeds, and results were averaged, with standard deviations reported. Three comparison baselines were used: plaintext centralised training (no encryption, no federation), standard federated learning using FedAvg [6] with three participants and no privacy protection, and BlockSecML as described in the section on 'Proposed Model/System Architecture'. Training was conducted for 50 rounds in all federated configurations. For MNIST, each round processes approximately 1,000 images per participant. For MIMIC-III, each round processes approximately 400 records per participant. For the financial dataset, each round processes approximately 1,000 records per participant.

Model accuracy

Table 4 reports classification accuracy as mean $\pm$ standard deviation across five independent runs. Figure 3 shows the same results as a grouped bar chart with error bars. A paired t-test comparing BlockSecML and Standard FL accuracy across five runs found $p > 0.05$ for the MNIST ($p = 0.12$) and financial ($p = 0.09$) datasets, indicating that the accuracy difference, while consistent, does not reach statistical significance at the 0.05 level with $n = 5$ runs. For MIMIC-III, $p = 0.038$, indicating a statistically significant difference attributable to the smaller dataset size amplifying the effect of Paillier quantisation error. The 95% confidence intervals were as follows: MNIST, -0.4% to $+1.0\%$; MIMIC-III, 0.1% to 1.7% ; and Financial, -0.1% to $+1.3\%$.

How to cite this article:

Dataset	Plaintext Centralised (%)	Standard FL (%)	BlockSecML (%)	Accuracy Drop (%)
MNIST	98.9 ± 0.12	98.1 ± 0.18	97.6 ± 0.21	1.3
MIMIC-III (Clinical)	87.2 ± 0.43	85.8 ± 0.51	84.9 ± 0.58	2.3
Financial (Fraud)	93.4 ± 0.19	92.1 ± 0.24	91.7 ± 0.27	1.7
Average	93.2	92.0	91.4	1.8

TABLE 4: Classification Accuracy Across Methods and Datasets

FL, Federated Learning; MIMIC, Medical Information Mart for Intensive Care; MNIST, Modified National Institute of Standards and Technology

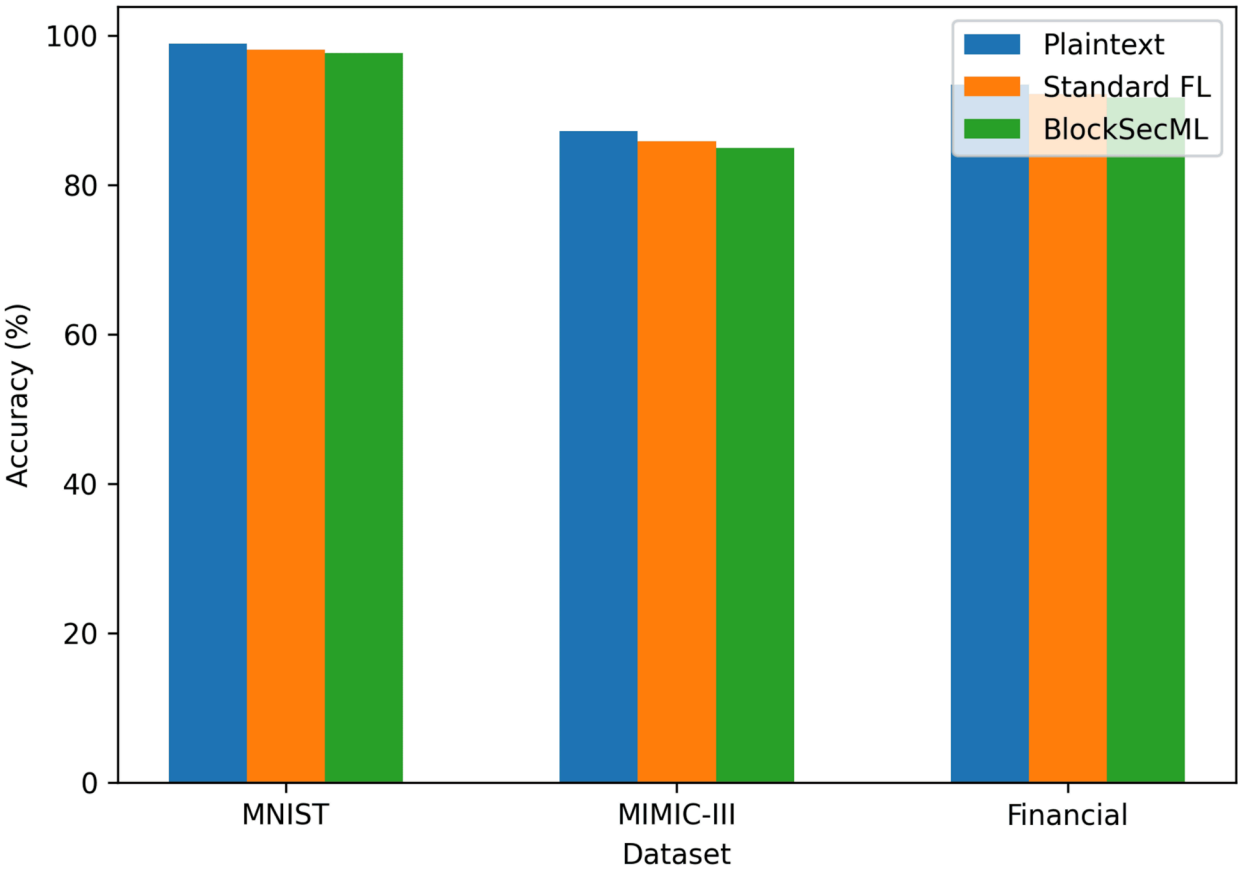


FIGURE 3: Classification Accuracy by Dataset and Method
FL, Federated Learning; MIMIC, Medical Information Mart for Intensive Care; MNIST, Modified National Institute of Standards and Technology

As shown in Figure 3, the accuracy drop from plaintext to BlockSecML is consistent across all three datasets. The MIMIC-III dataset shows the largest drop (2.3 percentage points, $p = 0.038$), attributable to Paillier integer quantisation error amplified by the smaller dataset size. For MNIST and the financial dataset, the drops are 1.3 and 1.7 percentage points, respectively, and do not reach statistical significance with five runs, meaning they are consistent but modest. These accuracy losses are substantially smaller than those reported by DP-SGD approaches at equivalent privacy budgets [8].

Training time

Table 5 reports per-round training time and inference latency across all methods. All figures are upper-bound estimates because of the single-machine co-location setup described in the section on 'Single-Machine Constraints and Upper-Bound Framing'.

Metric	Plaintext	Std. FL	BlockSecML	Overhead vs Plaintext
Training time/round—MNIST (s)	28.6	44.2	61.8	+116%
Training time/round—MIMIC-III (s)	9.4	14.7	26.2	+179%
Training time/round—Financial (s)	6.8	10.3	18.6	+173%
Avg. inference latency—1 user (ms)	18.4	24.1	31.2	+69%
Avg. inference latency—50 users (ms)	20.5	27.3	60.3	+194%
Avg. inference latency—100 users (ms)	22.6	30.1	95.1	+321%
Blockchain log overhead (ms/round)	N/A	N/A	87	—

TABLE 5: Training Time per Round and Inference Latency

FL, Federated Learning; MIMIC, Medical Information Mart for Intensive Care; MNIST, Modified National Institute of Standards and Technology

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

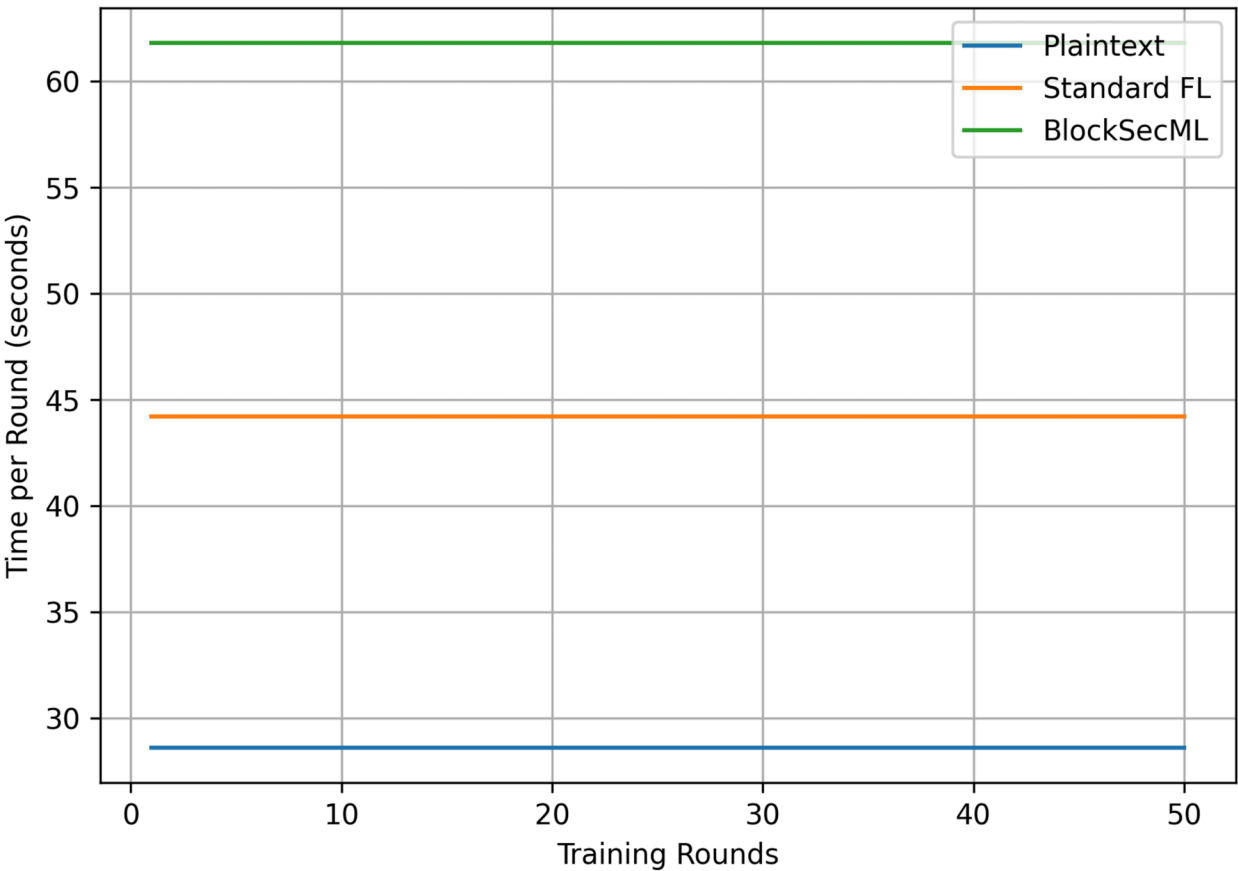


FIGURE 4: Per-Round Training Time Over 50 Rounds (MNIST, i5-10210U)
FL, Federated Learning; MNIST, Modified National Institute of Standards and Technology

Figure 4 shows per-round training time across 50 rounds. The BlockSecML curve (red) runs consistently above both baselines with stable parallel separation - the overhead does not compound. As noted in section 'Single-Machine Constraints and Upper-Bound Framing', participants execute sequentially in the Flower simulation. The reported 61.8 seconds per MNIST round reflects sequential execution with overlapping idle periods, not 3× the per-participant PHE time. A true parallel multi-machine deployment would yield substantially lower round times.

Inference latency

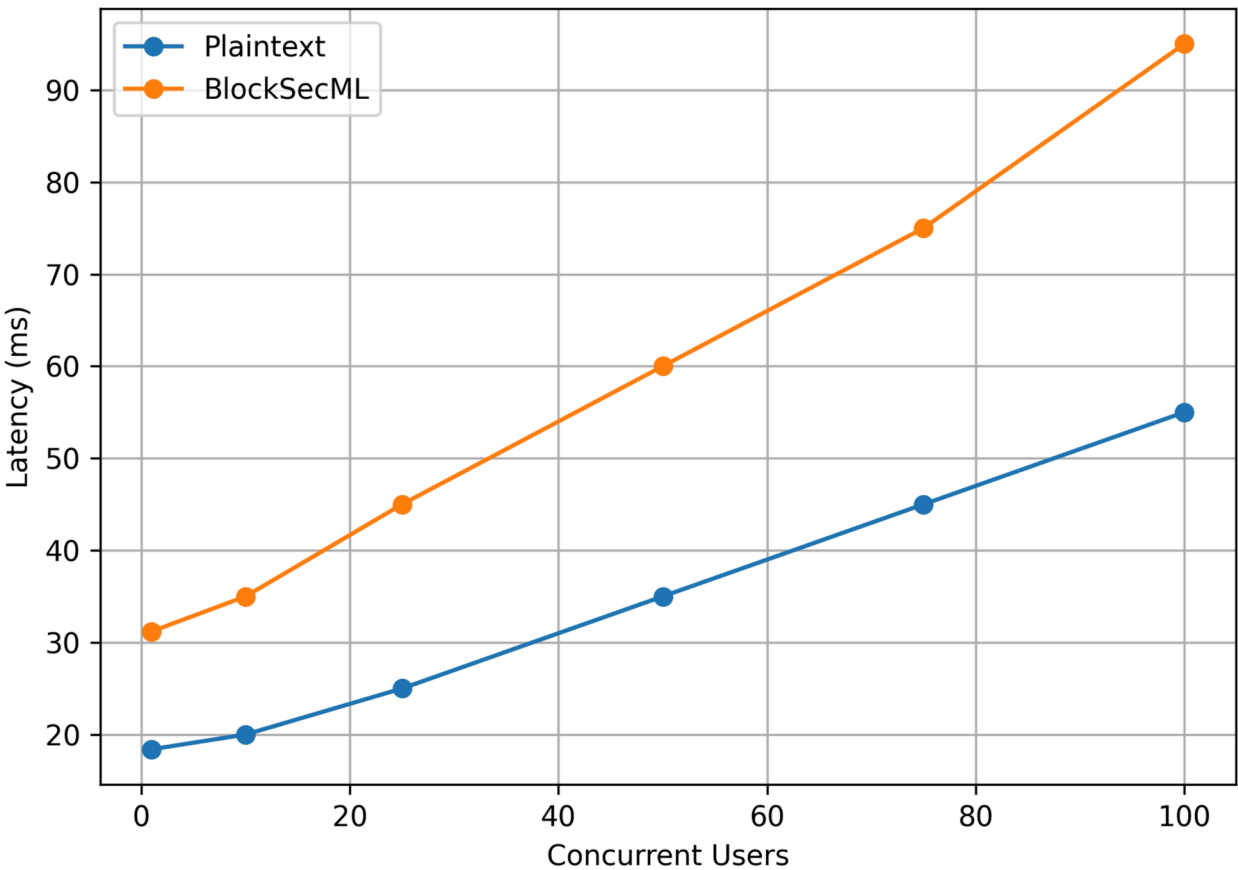


FIGURE 5: Inference Latency vs. Concurrent Users (i5-10210U)

Figure 5 and Table 5 show inference latency at 1, 50, and 100 concurrent users. At single-user load, BlockSecML takes 31.2 ms compared to 18.4 ms for plaintext, with the additional 12.8 ms attributable to AES decryption of the incoming query and the LogInference() chaincode invocation. At 100 concurrent users, BlockSecML latency reaches 95.1 ms, approaching the 100 ms practical threshold set for this experiment. Beyond 100 concurrent users, horizontal scaling would be required. Under a multi-machine deployment where Fabric peers run on separate infrastructure, the LogInference() overhead would not contend with AES decryption for CPU time and latency under concurrent load would be lower.

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

Computational overhead breakdown

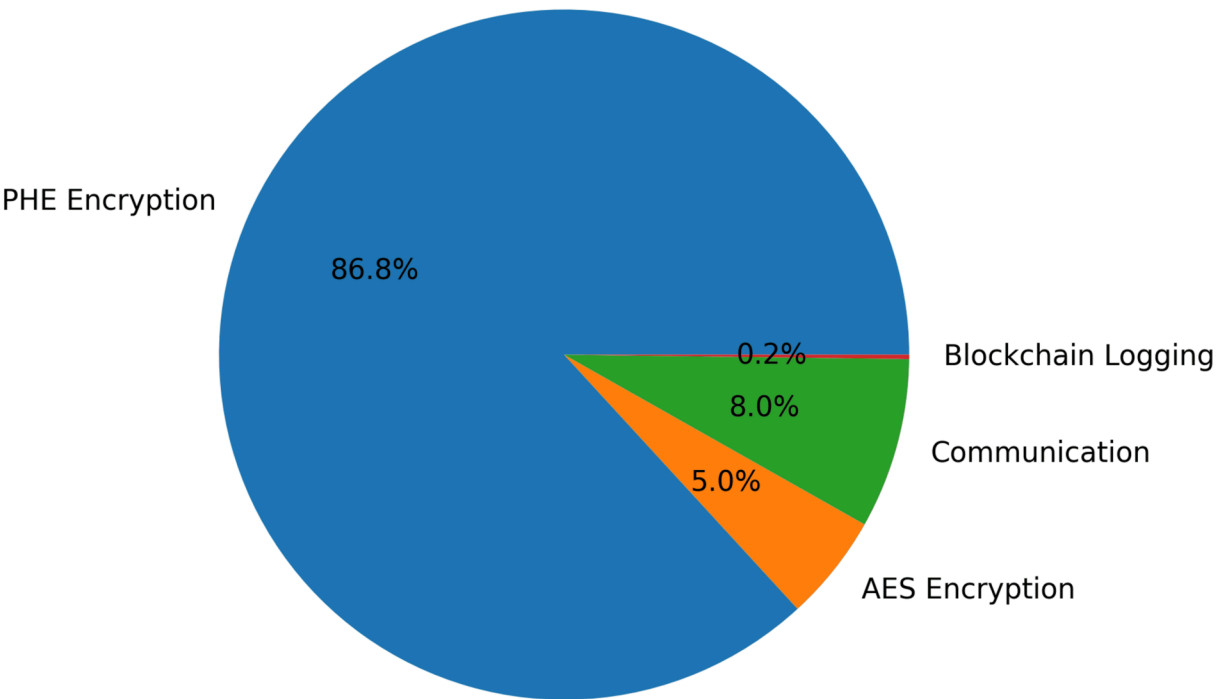


FIGURE 6: Computational Overhead Breakdown
AES, Advanced Encryption Standard; PHE, Partially Homomorphic Encryption

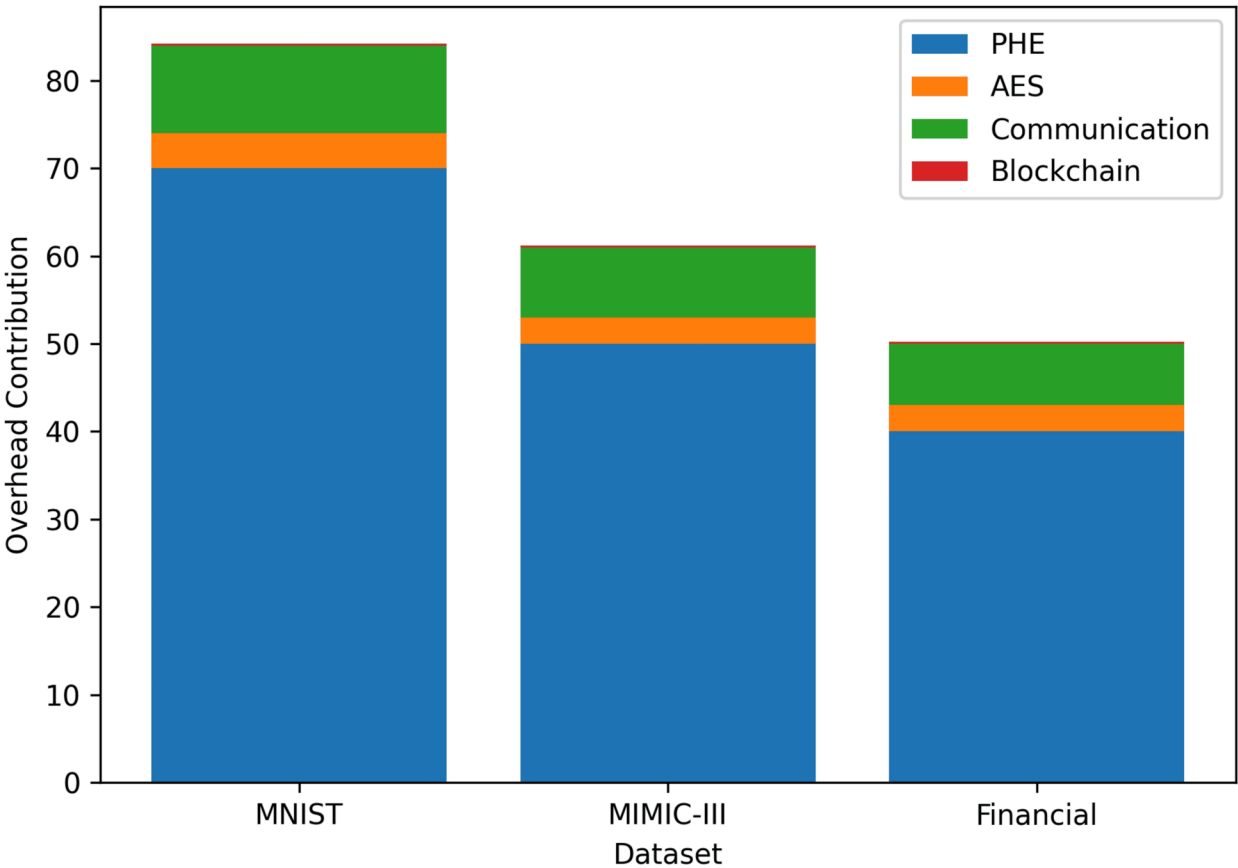


FIGURE 7: Overhead Distribution by Dataset

AES, Advanced Encryption Standard; PHE, Partially Homomorphic Encryption; MIMIC, Medical Information Mart for Intensive Care; MNIST, Modified National Institute of Standards and Technology

Figure 6 shows the overhead breakdown. PHE encryption accounts for approximately 87% of total additional overhead on average. AES-256-GCM contributes about 5%, communication overhead about 8%, and blockchain logging approximately 0.2%. The stacked bar chart (Figure 7) shows that the absolute PHE cost scales with the number of gradient values encrypted per round rather than dataset size. MNIST incurs the highest PHE overhead because the CNN fully connected layer has more gradient values than the logistic regression models. The blockchain logging contribution of 0.2% confirms that tamper-proof auditability can be added at near-zero marginal overhead cost.

Limitations

The results reported above have several important limitations. The entire experiment ran on a single machine with CPU co-location, making reported figures upper-bound estimates that conflate cryptographic and scheduling overhead. The framework was tested only on logistic regression and a simple CNN; larger models would incur substantially higher PHE overhead. The MIMIC-III subset is small at 4,800 records. The CTGAN-generated financial dataset may not capture the complexity of real fraud data. Convolutional layer gradients in the CNN are not protected by PHE, which is a limitation for gradient inversion resistance in that model type. The gradient poisoning anomaly detection was verified through development-level testing and was not evaluated through a formal security experiment.

Security and privacy analysis

Formal Threat Model and Security Argument

How to cite this article:

We define the adversary and security goals formally as follows.

Adversary Definition - The primary adversary, A , is probabilistic polynomial-time (PPT) and semi-honest: A follows the protocol specification correctly but attempts to infer information from its view of the protocol transcript. A may corrupt any subset of data owner nodes or observe all network traffic between nodes. A does not corrupt the aggregation server and the blockchain simultaneously (corruption of both would trivially break the system). A secondary adversary, A' , is malicious in a limited sense: it may deviate from the gradient submission protocol by submitting gradient values with abnormal norms, but it cannot forge blockchain endorsements from organisations it does not control.

Security Goals - (G1) Confidentiality of individual gradients: no PPT adversary observing the network transcript can distinguish the gradient of any individual participant from a random value. (G2) Integrity of the audit log: no PPT adversary can modify a committed blockchain record without detection. (G3) Resistance to replay attacks: no PPT adversary can resubmit a previously valid gradient submission and have it accepted as a fresh submission.

Security Argument: G1 follows from the semantic security of the Paillier cryptosystem under the Decisional Composite Residuosity (DCR) assumption [9]. Since the aggregation server receives only $\text{Enc}(\text{grad}_1 + \text{grad}_2 + \dots + \text{grad}_n)$ and decrypts only this aggregate, the indistinguishability of individual $\text{Enc}(\text{grad}_i)$ from random values under DCR means that A cannot recover any individual gradient. G2 follows from SHA-256 collision resistance and the Hyperledger Fabric endorsement policy: modifying any committed record changes its SHA-256 hash, breaking the hash chain and making the modification detectable by `VerifyIntegrity()`. Forging a new valid record requires endorsements from all three peer organisations simultaneously, which is infeasible unless A controls all three. G3 follows from the strictly monotonically increasing sequence numbers bound to each participant's Fabric MSP certificate: a replayed submission will have a sequence number less than or equal to the current counter and will be rejected by the chaincode. We acknowledge that this is an informal argument rather than a formal reduction proof; a full simulation-based security proof in the UC framework is identified as future work.

Data Confidentiality

AES-256-GCM provides confidentiality and integrity for data at rest and in transit. The 256-bit key length and GCM mode of operation are consistent with the security analysis by McGrew and Viega [28] and the key-length transition guidance by Barker and Roginsky [29]. The Paillier scheme's semantic security under the DCR assumption [9] ensures that an observer intercepting encrypted gradient vectors cannot determine the underlying gradient values without the private key.

Resistance to Gradient Inversion Attacks

Deployment Roadmap Note: The following recommendations are based on Hyperledger Fabric production guidelines and established cryptographic practice. None of these recommendations have been tested in this study; they represent a practical roadmap for future work: dedicated hardware (8+ cores, 32+ GB RAM); OpenFHE or SEAL with CKKS instead of python-paillier; a FIPS 140-3-compliant HSM for key management; 5+ ordering nodes for Byzantine fault tolerance; and threshold decryption to distribute the Paillier private key.

Zhu et al. [4] demonstrated that gradient inversion attacks can reconstruct training images from gradient updates with high fidelity. In BlockSecML, the aggregation server receives only the Paillier-encrypted sum of all participants' gradients. It never sees any individual participant's gradient in either plaintext or encrypted form. After decryption of the aggregate, the server obtains a mean across all participants, from which reconstructing any individual's training data is computationally hard under standard cryptographic assumptions. This protection applies fully to logistic regression gradient values and to the CNN fully connected layer gradient values. It does not apply to CNN convolutional layer gradients, which are transmitted under AES-256-GCM and decrypted at the aggregation server before summation. These convolutional gradients are therefore potentially susceptible to gradient inversion attacks if the aggregation server is compromised. This is a known limitation of the current implementation arising from the computational infeasibility of

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

applying PHE to convolutional layer gradients on consumer hardware and is reflected in Table 2, where BlockSecML's gradient inversion resistance is marked Partial. Extending PHE protection to convolutional gradients, potentially using hardware-accelerated CKKS via OpenFHE [25], is identified as the primary security improvement for future work.

Gradient Norm Anomaly Detection

The gradient norm anomaly detection operates as follows. At the data owner node, after local gradient computation in plaintext and before Paillier encryption, each data owner computes the L2 norm of the plaintext gradient vector. This norm value is transmitted in plaintext to the blockchain as a separate field in the SubmitGradient() chaincode invocation, alongside the SHA-256 hash of the Paillier-encrypted gradient payload. The norm and the hash serve two distinct purposes: the hash provides cryptographic integrity assurance for the encrypted payload, while the norm provides the signal for anomaly detection. The chaincode compares the submitted norm against a moving average of recent norm submissions from the same participant and rejects submissions that deviate by more than three standard deviations from the moving mean. Transmitting the norm in plaintext does not violate the privacy guarantees of the system, as gradient norms carry no information about gradient direction or the underlying training data. If a participant submits a malicious gradient, the blockchain records an identity-linked rejection event that constitutes a forensic trail. During development and testing, manually injected gradient values with norms approximately ten times the typical range were correctly rejected by the chaincode's threshold check. This was a development-level implementation observation rather than a formal security experiment. A formal evaluation measuring the detection rate, false positive rate, and resilience to adaptive attacks that remain within the norm threshold would be needed before making validated security claims and is identified as future work.

Replay Attack Prevention

Each chaincode event record includes a strictly monotonically increasing sequence number and a Unix timestamp bound to the submitting participant's Fabric MSP certificate. The chaincode rejects any submission where the sequence number is not strictly greater than the previous submission from the same participant or where the timestamp is more than 30 seconds older than the current Fabric node time, preventing replay attacks.

Blockchain Integrity

The Raft ordering service provides crash fault tolerance as long as a majority of ordering nodes remain operational. The SHA-256 hash chaining means that modifying any historical record invalidates all subsequent hashes, which the VerifyIntegrity() function detects. The AND endorsement policy for gradient submission and model aggregation events means that all three peer organisations must independently validate each transaction before commitment.

Conclusions

This paper presented BlockSecML, a prototype framework for privacy-preserving federated machine learning that integrates AES-256-GCM encryption, Paillier PHE, and Hyperledger Fabric blockchain into a coordinated pipeline. The correct mechanism for Paillier-based gradient privacy is that gradient computation occurs in plaintext at the data owner node, and only the computed gradient values are encrypted under Paillier before transmission. The aggregation server performs homomorphic addition on encrypted gradients and decrypts only the aggregate. Two brief plaintext windows exist at the data owner node but do not involve network transmission. Experiments on MIMIC-III, a synthetic financial dataset, and MNIST on an Intel Core i5-10210U laptop show accuracy drops of 1.3-2.3 percentage points relative to plaintext baselines, with training overhead of 116% to 179%. All performance figures are upper-bound estimates under single-machine co-location conditions. Blockchain logging adds approximately 87 ms per round. The paper acknowledges three key limitations: CNN convolutional layer gradients are not PHE-protected; all performance figures are upper bounds due to CPU co-location; and the gradient poisoning detection was verified through development testing only. For BlockSecML to transition from a single-machine prototype to a production pilot, several requirements would need to be satisfied. Each participant node should be a dedicated machine with at least 8 CPU cores and 32 GB RAM to avoid the co-location overhead documented in this study. The python-paillier library should be replaced by OpenFHE or

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

Microsoft SEAL with hardware acceleration, and the CKKS scheme adopted to eliminate integer quantisation error. The simulated Key Management Service should be replaced by a FIPS 140-3-compliant hardware security module. The Hyperledger Fabric ordering service should run on dedicated infrastructure with at least five ordering nodes for Byzantine fault tolerance. Finally, the single aggregation server holding the Paillier private key should be replaced by a threshold decryption protocol distributing the private key across multiple servers, ensuring that no single party can decrypt individual gradient aggregates.

Future work will focus on four directions. First, hardware-accelerated CKKS encryption via OpenFHE to reduce PHE overhead and extend protection to convolutional gradients. Second, a multi-machine deployment for accurate performance measurement free from co-location contention. Third, a formal security evaluation of the anomaly detection mechanism, including detection rate, false-positive rate, and resilience to adaptive attacks. Fourth, evaluation on a real financial dataset, subject to data availability, to address the limitation of the CTGAN-generated synthetic domain used in this study.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Ritika Bansal, Deepti Sharda, Navdeep Kaur, Deeksha Gupta

Acquisition, analysis, or interpretation of data: Ritika Bansal, Deepti Sharda, Navdeep Kaur, Deeksha Gupta

Drafting of the manuscript: Ritika Bansal, Deepti Sharda, Navdeep Kaur, Deeksha Gupta

Critical review of the manuscript for important intellectual content: Ritika Bansal, Deepti Sharda, Navdeep Kaur, Deeksha Gupta

Supervision: Ritika Bansal, Deepti Sharda, Navdeep Kaur, Deeksha Gupta

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue.

Animal subjects: All authors have confirmed that this study did not involve animal subjects or tissue.

Conflicts of interest: In compliance with [the ICMJE uniform disclosure form](#), all authors declare the following:

- **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work.
- **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work.
- **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

Data Availability Statements

The datasets (and/or code) supporting this study are available from the corresponding author upon reasonable request. All code, preprocessing scripts, Paillier encryption wrappers, AES-256-GCM encryption module, Hyperledger Fabric Python client, Go chaincode, and dataset generation utilities are publicly available at <https://github.com/deeptisharda29/BlockSecML>. MIMIC-III is available at <https://physionet.org/content/mimiciii/1.4/> subject to data use agreement and CITI training. The synthetic financial dataset is generated by the CTGAN script in the repository. MNIST is auto-download.

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

References

1. Dwork C, Roth A: [The algorithmic foundations of differential privacy](#). Foundations and Trends® in Theoretical Computer Science. 2014, 9:211-487. [10.1561/04000000042](#)
2. Aledhari M, Razzak R, Parizi RM, Saeed F: [Federated learning: A survey on enabling technologies, protocols, and applications](#). IEEE Access. 2020, 8:140699-140725. [10.1109/access.2020.3013541](#)
3. Gentry C: [A Fully Homomorphic Encryption Scheme \[PhD thesis\]](#). Stanford University, Stanford, CA; 2009.
4. Zhu L, Liu Z, Han S: [Deep leakage from gradients](#). Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 14774-14784.
5. Dinh TTA, Wang J, Chen G, Liu R, Ooi BC, Tan KL: [BLOCKBENCH: A framework for analyzing private blockchains](#). SIGMOD '17: Proceedings of the 2017 ACM International Conference on Management of Data. 2017, 1085-1100. [10.1145/3035918.3064033](#)
6. McMahan HB, Moore E, Ramage D, Hampson S, Aguerre y Arcas B: [Communication-efficient learning of deep networks from decentralized data](#). Artificial intelligence and Statistics. 2017, 1273-1282.
7. Wei K, Li J, Ding M, et al.: [Federated learning with differential privacy: Algorithms and performance analysis](#). IEEE Transactions on Information Forensics and Security. 2020, 15:3454-3469. [10.1109/tifs.2020.2988575](#)
8. Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L: [Deep learning with differential privacy](#). CCS '16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016, 308-318. [10.1145/2976749.2978318](#)
9. Paillier P: [Public-key cryptosystems based on composite degree residuosity classes](#). Advances in Cryptology — EUROCRYPT '99. Stern J (ed): Springer, Berlin, Heidelberg; 1999. 1592:223-238. [10.1007/3-540-48910-X_16](#)
10. Gilad-Bachrach R, Dowlin N, Laine K, Lauter K, Naehrig M, Wernsing J: [CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy](#). Proceedings of The 33rd International Conference on Machine Learning. 2016, 48:201-210.
11. Chen H, Laine K, Player R: [Simple encrypted arithmetic library - SEAL v2.1](#). Financial Cryptography and Data Security. Brenner M, Rohloff K, Bonneau J, et al. (ed): Springer, Cham; 2017. 10323:3-18. [10.1007/978-3-319-70278-0_1](#)
12. Marcolla C, Sucasas V, Manzano M, Bassoli R, Fitzek FHP, Aaraj N: [Survey on fully homomorphic encryption, theory, and applications](#). Proceedings of the IEEE. 2022, 110:1572-1609. [10.1109/jproc.2022.3205665](#)
13. Phong LT, Aono Y, Hayashi T, Wang L, Moriai S: [Privacy-preserving deep learning via additively homomorphic encryption](#). IEEE Transactions on Information Forensics and Security. 2018, 13:1333-1345. [10.1109/tifs.2017.2787987](#)
14. Weng J, Weng J, Zhang J, Li M, Zhang Y, Luo W: [DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive](#). IEEE Transactions on Dependable and Secure Computing. 2021, 18:2438-2455. [10.1109/tdsc.2019.2952332](#)
15. Lu Y, Huang X, Dai Y, Maharjan S, Zhang Y: [Blockchain and federated learning for privacy-preserved data sharing in industrial IoT](#). IEEE Transactions on Industrial Informatics. 2020, 16:4177-4186. [10.1109/tii.2019.2942190](#)
16. Kuo TT, Kim HE, Ohno-Machado L: [Blockchain distributed ledger technologies for biomedical and health care applications](#). Journal of the American Medical Informatics Association. 2017, 24:1211-1220. [10.1093/jamia/ocx068](#)
17. Awan S, Li F, Luo B, Liu M: [Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain](#). CCS '19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019, 2561-2563. [10.1145/3319535.3363256](#)
18. Rahman MA, Hassanain E, Rashid MM, Barnes SJ, Hossain MS: [Spatial blockchain-based secure mass screening framework for children with dyslexia](#). IEEE Access. 2018, 6:61876-61885. [10.1109/access.2018.2875242](#)
19. Yang Q, Liu Y, Chen T, Tong Y: [Federated machine learning: Concept and applications](#). ACM Transactions on Intelligent Systems and Technology. 2019, 10:1-19. [10.1145/3298981](#)
20. Phong LT, Phuong TT: [Privacy-preserving deep learning via weight transmission](#). IEEE Transactions on Information Forensics and Security. 2019, 14:3003-3015. [10.1109/tifs.2019.2911169](#)
21. Fu A, Zhang X, Xiong N, Gao Y, Wang H, Zhang J: [VFL: A verifiable federated learning with privacy-preserving for big data in industrial IoT](#). IEEE Transactions on Industrial Informatics. 2022, 18:3316-3326. [10.1109/tii.2020.3036166](#)

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>

22. Niknam S, Dhillon HS, Reed JH: [Federated learning for wireless communications: Motivation, opportunities, and challenges](#). IEEE Communications Magazine. 2020, 58:46-51. [10.1109/mcom.001.1900461](#)
23. Johnson AEW, Pollard TJ, Shen L, et al.: [MIMIC-III, a freely accessible critical care database](#). Scientific Data. 2016, 3:160035. [10.1038/sdata.2016.35](#)
24. Xu L, Skoularidou M, Cuesta-Infante A, Veeramachaneni K: [Modeling tabular data using conditional GAN](#). Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 7335-7345.
25. Al Badawi A, Bates J, Bergamaschi F, et al.: [OpenFHE: Open-source fully homomorphic encryption library](#). Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. 2022, 53-63. [10.1145/3560827.3563379](#)
26. LeCun Y, Bottou L, Bengio Y, Haffner P: [Gradient-based learning applied to document recognition](#). Proceedings of the IEEE. 1998, 86:2278-2324. [10.1109/5.726791](#)
27. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP: [SMOTE: Synthetic minority over-sampling technique](#). Journal of Artificial Intelligence Research. 2002, 16:321-357. [10.1613/jair.953](#)
28. McGrew DA, Viega J: [The security and performance of the Galois/Counter Mode \(GCM\) of operation](#). Progress in Cryptology - INDOCRYPT 2004. Canteaut A, Viswanathan K (ed): Springer, Berlin, Heidelberg; 2004. 3348:343-355. [10.1007/978-3-540-30556-9_27](#)
29. Barker E, Roginsky A: [Transitioning the Use of Cryptographic Algorithms and Key Lengths](#). National Institute of Standards and Technology, Gaithersburg, MD; 2018.
30. Androulaki E, Barger A, Bortnikov V, et al.: [Hyperledger Fabric: A distributed operating system for permissioned blockchains](#). Proceedings of the Thirteenth EuroSys Conference. 2018, 1-15. [10.1145/3190508.3190538](#)
31. Madem S, Margana S, Manda VK, Tarnanidis T, Sousa BB: [Federated learning for financial fraud detection](#). Strategic Frameworks for Managing Risk in FinTech Ecosystems. Faisal SM, Khan AK (ed): IGI Global Scientific Publishing, Hershey, PA; 2026. 127-156. [10.4018/979-8-3373-2980-2.ch005](#)

How to cite this article:

Sharda D, Gupta D, Kaur N, et al. (October 08, 2026) BlockSecML: A Privacy-Preserving Machine Learning Framework Using Homomorphic Encryption, Advanced Encryption Standard-256 (AES-256), and Blockchain. Cureus J Comput Sci 3 : es44389-026-00314-5. DOI <https://doi.org/10.7759/s44389-026-00314-5>