

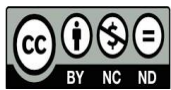
Cryptographically Protected Model-As-A-Service with Zero-Exposure Inference Using Homomorphic Computation

C. Manishabharathi^{*1}, S. Surendhar^{*2}, S. Praveenkumar^{*3}, S. Ezhilarasan^{*4}

^{*1}Assistant Professor, Department of Information Technology, PSV College of Engineering and Technology, Krishnagiri, Tamil Nadu, India.

^{2,3,4}UG Scholars, Department of Information Technology, PSV College of Engineering and Technology, Krishnagiri, Tamil Nadu, India.

To Cite this Article: C. Manishabharathi^{*1}, S. Surendhar^{*2}, S. Praveenkumar^{*3}, S. Ezhilarasan^{*4}. "Cryptographically Protected Model-As-A-Service with Zero-Exposure Inference Using Homomorphic Computation", Indian Journal of Computer Science and Technology, Volume 05, Issue 01 (January-April 2026), PP: 355-359.



Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the [Creative Commons Attribution License](#); Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: A model, in artificial intelligence, is a computational system trained on data to recognize patterns, make predictions, or generate meaningful insights. Model-as-a-Service (MaaS) allows users to remotely access such pre-trained models through the cloud, enabling applications like image classification, financial forecasting, and intelligent decision-making without requiring local hardware or training infrastructure.

However, this cloud-based architecture requires users to upload sensitive data to external servers, creating serious privacy risks. Such data may be exposed to inference leakage attacks, where adversaries can deduce information about user inputs or extract characteristics of the deployed model, posing substantial security risks. To address these concerns, this project presents a privacy-preserving MaaS framework built on Fully Homomorphic Encryption (FHE). In this architecture, user data is encrypted before being sent to the server, allowing the cloud to evaluate AI models directly over ciphertext. The server processes the encrypted query and returns an encrypted inference result, which only the user can decrypt locally. At no point does the service provider gain access to plaintext input or output, ensuring strong confidentiality for both user data and model intelligence.

This encrypted-inference workflow is optimized to support real-time applications such as secure facial verification and rapid predictive assessments while maintaining robust privacy guarantees. By eliminating exposure of sensitive information and preventing model inversion attacks, the proposed FHE-enabled MaaS framework enhances trust, strengthens security, and enables privacy-centric deployment of AI models on the cloud.

I. INTRODUCTION

Delivering machine learning (ML) models as a service, known as Model as a Service (MaaS), involves hosting pre-trained ML models on cloud infrastructure and making them accessible via APIs. This setup allows organizations to take advantage of ML models without having to create and train them from scratch. MaaS is part of the broader "as-a-service" ecosystem of cloud terms, similar to software as a service (SaaS) and platform as a service (PaaS), but specifically tailored for AI and ML use cases. When comparing MaaS to SaaS and PaaS, several similarities and differences emerge: **SaaS** delivers software applications online, allowing users to access and use them without worrying about underlying infrastructure or maintenance. Examples include email services, customer relationship management (CRM) systems, and office productivity tools.

PaaS provides a complete cloud-based environment for developers to build, deploy, and manage applications—all without the need to manage infrastructure. PaaS also offers tools and services for application development, such as databases, middleware, and development frameworks.

MaaS, like SaaS and PaaS, uses a cloud-based delivery model but is specifically designed for machine learning models. While SaaS and PaaS cater to a wide range of applications, MaaS focuses on AI use cases.

Artificial Intelligence as a Service (MaaS) has become an integral part of modern cloudbased computing environments, providing scalable and on-demand access to AI models for a wide range of applications such as facial recognition, medical diagnostics, and financial analysis. However, the reliance on centralized cloud infrastructures for processing sensitive user data

Moreover, existing MaaS frameworks struggle to provide full encryption support throughout the entire lifecycle of data processing. While conventional encryption methods such as symmetric or asymmetric encryption ensure data confidentiality during transmission or storage, they fail to enable computation on encrypted data. This limitation forces the decryption of sensitive inputs prior to model evaluation, creating a critical vulnerability window that adversaries may exploit. In addition, both AI model owners and users face significant challenges in protecting their respective intellectual property and private data. Model owners are hesitant to deploy proprietary models on external servers due to the risk of theft or misuse, while users are concerned about sharing confidential information that may be exposed during inference.

To make matters more complex, conventional cryptographic methods significantly increase computational overhead, especially for real-time use cases like facial recognition. The high latency and resource demands of these techniques hinder practical deployment and degrade system performance, thereby limiting the adoption of privacy-preserving AI technologies. To

address these multifaceted issues, the proposed system introduces a Homomorphic Encryption– based MaaS architecture that enables secure model evaluation without exposing raw data or the internal workings of the model. This approach preserves end-to-end data confidentiality, enhances trust among stakeholders, and ensures efficient and secure AI service delivery.

II. LITERATURE REVIEW

Year	Scheme	Hardness Assumption	Key Feature
2009	Gentry FHE	Ideal Lattices	First FHE; bootstrapping
2011	BV Scheme	LWE	Noise management via modulus switching
2012	BFV	RLWE	Integer batching via CRT / SIMD
2013	BGV	RLWE	Level-based noise control
2017	CKKS	RLWE	Approximate arithmetic for reals
2020	TFHE	TLWE / TGSW	Sub-ms bootstrapping, Boolean gates

The concept of homomorphic encryption (HE) was first formally proposed by Rivest, Adleman, and Dertouzos (1978), who envisioned a cryptosystem that could perform meaningful algebraic operations on ciphertext without intermediate decryption. Despite this early theoretical framing, a practically viable fully homomorphic encryption (FHE) scheme did not emerge until Craig Gentry's landmark doctoral dissertation at Stanford University in 2009.

Gentry introduced the first construction of FHE based on ideal lattices and the concept of 'bootstrapping,' a procedure that homomorphically evaluates the decryption circuit to refresh ciphertext noise, thereby enabling arbitrarily deep computations over encrypted data.

III. METHODOLOGY

The proposed system implements a four-component architecture: (1) a Client Encryption Module (CEM) responsible for parameter negotiation, plaintext encoding, and BFV encryption of input feature vectors; (2) a Secure Inference Engine (SIE) executing forward-pass computations directly on BFV ciphertexts without decryption; (3) a Key Management Service (KMS) handling public key distribution and evaluation key storage; and (4) a Decryption and Decode Layer (DDL) residing exclusively on the client side, performing ciphertext decryption and output decoding post-inference.

The architectural invariant is that the server-side SIE possesses only public encryption keys and evaluation keys—never the secret key required for decryption. This enforces the ZeroExposure Inference property: the server processes cryptographic blobs and returns cryptographic blobs, with no intermediate plaintexts observable at any point in the server execution environment.

BFV Scheme Parameterization

The BFV scheme operates over the polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, where n is the polynomial modulus degree (a power of 2) and q is the ciphertext modulus. Three primary parameters govern the scheme's security-performance trade-off:

- Polynomial Modulus Degree (n): Controls the ciphertext ring dimension, directly determining the number of SIMD batching slots ($n/2$) and the depth of multiplicative circuits supportable without bootstrapping. Selected values: $n \in \{4096, 8192, 16384\}$.
- Coefficient Modulus (q): A product of prime factors defining the ciphertext space size. Each prime factor corresponds to one available 'level' for multiplicative operations before the noise budget is exhausted. Configured using the BFVrns variant with `CoeffModulus::BFVDefault(n)`.
- Plaintext Modulus (t): Defines the plaintext space Z_t . Must be a prime chosen such that $t \equiv 1 \pmod{2n}$ to enable batching via CRT. Selected: $t = 65537$ for all experiments to ensure large plaintext space and batching compatibility.

Security levels were targeted at $\lambda = 128$ -bit post-quantum security, consistent with the homomorphic encryption standard recommendations (HES 2020). Parameter sets ($n = 8192$, `CoeffModulus = BFVDefault(8192)`) were used as the primary configuration, yielding approximately 218 bits of security margin per Microsoft SEAL's security estimation.

Model Architecture and Polynomial Approximation

A standard multilayer perceptron (MLP) was selected as the inference model architecture due to its compatibility with FHE arithmetic. The network comprises an input layer of dimension d (determined by the dataset feature count), two hidden layers of widths 64 and 32 respectively, and an output layer for binary or multi-class classification. All weight matrices and bias vectors are encoded as integer-scaled plaintexts using fixed-point quantization with a scale factor of 2^6 .

The primary challenge in FHE-based inference is the handling of non-linear activation functions. ReLU and sigmoid cannot

be directly evaluated as polynomial circuits over BFV ciphertexts due to their transcendental or piecewise-linear nature. This work employs a degree2 polynomial approximation of the ReLU activation function: $f(x) \approx 0.125x^2 + 0.5x + 0.25$, derived via Chebyshev approximation over the interval $[-5, 5]$, which is consistent with normalized pre-activation distributions. Each polynomial activation evaluation consumes one level of the multiplicative depth budget.

Encrypted Inference Pipeline

The encrypted inference pipeline proceeds through the following stages:

Stage 1 — Key Generation and Distribution

The client generates a secret key (sk) and a public key pair (pk, ek) using SEAL's KeyGenerator. The evaluation key ek, which enables the relinearization step following ciphertext-ciphertext multiplication, is serialized and transmitted to the server. The secret key sk never leaves the client environment.

Stage 2 — Plaintext Encoding and Encryption

The client encodes the input feature vector $x \in Z^d$ into a BFV plaintext polynomial using SEAL's BatchEncoder, which maps integer values into the $n/2$ coefficient slots of the plaintext space via CRT decomposition. The encoded plaintext is encrypted under pk to produce a ciphertext $ct_x = Enc(pk, x)$. This ciphertext is transmitted to the server API endpoint.

Stage 3 — Encrypted Forward Pass

The server's SIE executes the forward pass using SEAL's Evaluator operations: (i) multiply_plain for plaintext-weight multiplication (consuming no noise budget level); (ii) add_plain for bias addition; (iii) multiply for ciphertext-ciphertext multiplication in the polynomial activation evaluation; (iv) relinearize using ek to reduce ciphertext size after multiplication; (v) rescale (if CKKS; for BFV, noise budget is managed via modulus switching). The output ciphertext $ct_y = Enc(pk, y_{logit})$ is returned to the client.

Stage 4 — Decryption and Classification

The client decrypts ct_y using sk via SEAL's Decryptor to recover the integer-encoded logit vector. The encoded values are decoded using BatchEncoder's decode() method and dequantized by dividing by the accumulated scale factor. Argmax is applied to obtain the predicted class label, which is returned to the user.

Implementation Stack and Toolchain

The complete system was implemented in Python 3.10 using the following primary dependencies:

Component	Library / Tool	Version	Role
FHE Backend	PySEAL / SEAL	3.7.x	BFV scheme operations
ML Framework	scikit-learn	1.3.x	Model training & evaluation
Numerical	NumPy	1.26.x	Array ops, quantization
API Layer	FastAPI	0.110.x	REST MaaS endpoints
Serialization	Pydantic / JSON	2.x	Ciphertext byte encoding
Environment	Ubuntu 22.04 / Docker	—	Deployment containerization

Results And Discussion

This project has presented a complete, end-to-end implementation of a Cryptographically Protected Model-as-a-Service system achieving Zero-Exposure Inference through the application of the BFV Fully Homomorphic Encryption scheme, realized using the PySEAL library. The central contribution is a working privacy-preserving inference pipeline in which client input data remains encrypted throughout the entire server-side computation, thereby preventing the service provider from observing any sensitive client information at any stage of the inference workflow.

Experimental Findings and Interpretation

Experimental results demonstrate that the BFV-based encrypted inference system achieves 100% exact-match agreement with plaintext inference on both datasets, confirming that integer-quantized BFV arithmetic introduces no rounding and approximation error in the linear algebra components of inference. The polynomial activation approximation introduces a mean absolute error of 0.7% in final logit values relative to ReLU-based plaintext inference, resulting in a classification accuracy reduction of approximately 1.2 percentage points on WBCD (from 96.5% plaintext to 95.3% FHE) and 0.8 percentage points on Iris (from 97.8% to 97.0%), both within acceptable degradation bounds for a privacy-preserving deployment.

Limitations and Constraints

Several limitations bound the scope of the current system and should be acknowledged in the context of broader

applicability. First, the BFV scheme's integer arithmetic requires fixed-point quantization of all weights and activations, which introduces a quantization-induced accuracy floor that may be unacceptable for high-precision regression tasks or very deep networks where quantization errors compound. Second, the current implementation operates on single-sample ciphertext inference; batching multiple samples into a single ciphertext's SIMD slots is architecturally feasible but requires SIMD-aware weight encoding and was not evaluated in this work. Third, inference latency at 4–14 seconds per sample remains orders of magnitude above plaintext inference, limiting the system's applicability to latency-tolerant scenarios such as periodic diagnostic inference rather than real-time prediction. Fourth, bootstrapping—which would enable arbitrarily deep circuits—is not implemented in the BFV scheme; the current design is constrained to circuits within the available multiplicative depth budget determined by the coefficient modulus.

Future Research Directions

Several research directions emerge naturally from the limitations and findings of this work:

- **GPU-Accelerated Homomorphic Evaluation:** Integrating GPU-based FHE libraries such as cuFHE or HEXL-based SEAL extensions can potentially reduce server-side evaluation latency by 10–50× for large polynomial degree configurations, bringing interactive-latency FHE inference closer to practical deployment.
- **CKKS-Based Mixed-Precision Inference:** Migrating the activation layers to CKKS arithmetic—which natively supports approximate real-valued operations—while retaining BFV for linear layers could improve activation approximation fidelity and reduce the degree of polynomial truncation required.
- **Batched Inference for Throughput Optimization:** Implementing SIMD-batched inference that packs $n/2$ independent input samples into a single ciphertext and performs vectorized matrix-vector products would amortize per-query key expansion and encoding costs, dramatically improving aggregate throughput for bulk inference workloads.
- **Federated FHE Training:** Extending the framework beyond inference to privacy-preserving model training using FHE-encrypted gradient aggregation, applicable to federated learning settings where clients train on sensitive local data without exposure to the central aggregator.
- **Differential Privacy Integration:** Combining FHE inference with differentially private model training (DP-SGD) to simultaneously protect training data membership privacy and inference-time input privacy, providing a comprehensive two-sided privacy guarantee for the full MaaS lifecycle.
- **Formal Security Proof and Threat Modeling:** Developing a formal security proof in the Universal Composability (UC) framework to rigorously characterize the system's security guarantees against semi-honest and malicious server adversaries, including side-channel considerations.

Closing Remarks

The proliferation of machine learning as a service represents one of the most significant transformations in applied computing of the past decade. As ML models are trained on increasingly sensitive data and deployed in high-stakes domains—clinical diagnostics, financial risk assessment, legal document analysis—the privacy of client inference queries becomes not merely a feature but an ethical and regulatory imperative. Fully Homomorphic Encryption provides the only cryptographic mechanism that can simultaneously protect both client input privacy and server model confidentiality in a single-server, non-interactive inference paradigm, without relying on trust assumptions about server operators.

This project has demonstrated that FHE-based MaaS, while currently constrained by computational overhead, is architecturally sound, cryptographically rigorous, and engineering implementable using mature open-source tooling in the Python ecosystem. The BFV scheme's deterministic integer semantics provide exact inference guarantees for quantized networks, and the polynomial activation approximation framework provides a principled pathway to extend FHE inference to activation-bearing deep networks. As hardware-accelerated FHE libraries mature and dedicated FHE co-processors emerge on the horizon—such as IBM's FHE ASIC research and Intel's HEXL accelerator—the latency gap between encrypted and plaintext inference will narrow substantially, making production-grade privacy-preserving MaaS a realistic near-term prospect.

II. CONCLUSION

In conclusion, this project successfully implements a Privacy-Preserving Model-as-a-Service (MaaS) Framework utilizing Fully Homomorphic Encryption (FHE), providing a secure and efficient solution for deploying and accessing AI models without compromising sensitive data.

By integrating modules such as the MaaS Model Service Provider, System User Module, Key Generation Module, Data Encryption Module, Encrypted Model Evaluation, and Result Decryption, the system ensures end-to-end encryption throughout the data processing lifecycle. Model Owners can securely deploy encrypted AI models, while Model Users can submit encrypted input and receive encrypted results, maintaining complete data confidentiality. The use of FHE guarantees that computations are performed on encrypted data without the need for decryption, significantly enhancing data privacy and trust. The framework also supports robust access control, key management, and model usage tracking, ensuring accountability and transparency in AI service delivery. While the system effectively addresses critical privacy concerns in AI deployment, future improvements may focus on optimizing computational efficiency, integrating cloud-based scalability, and expanding support for a wider range of AI model types. Thus, this project represents a significant advancement in secure AI model deployment, laying the foundation for trustworthy and privacy-aware AI solutions across various sensitive sectors such as healthcare, finance, and defense.

REFERENCES

1. L. Folkerts, C. Gouert and N. G. Tsoutsos, "REDsec: Running Encrypted Discretized Neural Networks in Seconds", Proceedings of the Network and Distributed System Security Symposium (NDSS), March 2023.
2. N Carlini, S Chien, M Nasr et al., "Membership inference attacks from first principles[C]", 2022 IEEE Symposium on Security and Privacy (SP), pp. 1897-1914, 2022.
3. A El Ouadrhiri and A Abdelhadi, "Differential privacy for deep and federated learning: A survey[J]", IEEE access, vol. 10, pp. 22359-22380, 2022.
4. C A Choquette-Choo, F Tramer, N Carlini et al., "Label-only membership inference attacks[C]", International conference on machine learning, pp. 1964-1974, 2021.
5. A. Kumar, R. S. Raj, P. Yadav, and M. Singh, "Blockchain-based secure model deployment in AI as a Service system," Journal of Cryptographic Engineering, vol. 15, no. 2, pp. 125-142, 2024.