Performance Benchmarking of FHE Libraries for Privacy-Preserving Machine Learning

Biswajit M, Arup M, Sohan P

5th July 2025

Motivation and Introduction

- **Secure data processing:** is a critical requirement in ML applications dealing with sensitive data.
- **Machine Learning (ML:)** increasingly requires data privacy, especially in sensitive domains like healthcare and finance.
- **Fully Homomorphic Encryption (FHE):** allows computation directly on encrypted data without decryption.
- Combining ML with FHE enables **privacy-preserving machine learning**, ensuring data confidentiality during computation.
- However, FHE introduces significant **computational overhead** and poses **performance bottlenecks** for real-world applications.
- Benchmarking helps in **understanding performance trade-offs** (accuracy, runtime, resource usage) in privacy-preserving ML.
- Such evaluations guide **efficient deployment strategies** of FHE in real-world privacy-sensitive environments.

What is Fully Homomorphic Encryption?

Fully Homomorphic Encryption (FHE) allows **arbitrary computation on encrypted data** without needing to decrypt it first.

Let $\mathsf{Enc}(m)$ denote the encryption of a message m, and $\mathsf{Dec}(c)$ the decryption of ciphertext c. Then:

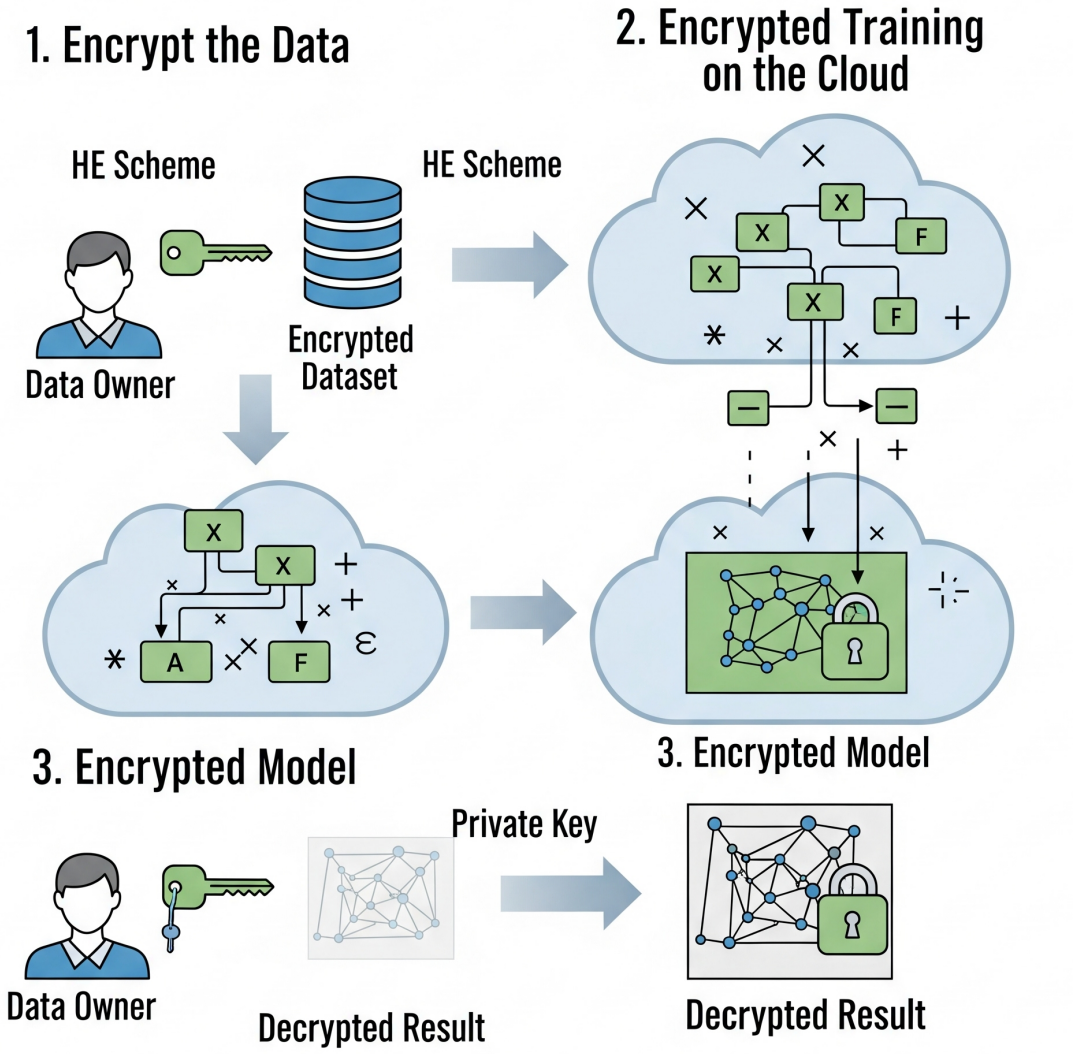
$$\mathsf{Dec}(\mathsf{Eval}(f, \mathsf{Enc}(m))) = f(m)$$

where f is any computable function (e.g., a circuit).

- Key Components of FHE
  - **Key Gen:** $\mathsf{KeyGen}(1^\lambda)$: Generates secret and/or public keys.
  - **Encryption:** $\mathsf{Enc}(pk, m)$: Encrypts a plaintext message.
  - **Evaluation:** $\mathsf{Eval}(evk, f, c)$: Applies function f on encrypted data.
  - **Decryption:** $\mathsf{Dec}(sk, c')$: Decrypts the resulting ciphertext.s

Use case of Fully Homomorphic Encryption

Training an ML model with encrypted data



Objective, significance and settings

- Objective of internship:
  - Benchmark and evaluate popular FHE libraries on ML applications.
  - Focus libraries: **TenSEAL** and **OpenFHE**.
  - Focus schemes: **CKKS** (approximate arithmetic) and **BFV** (exact arithmetic).
  - Evaluation of key performance metrics

Objective, significance and settings

- Objective of internship:
  - Benchmark and evaluate popular FHE libraries on ML applications.
  - Focus libraries: **TenSEAL** and **OpenFHE**.
  - Focus schemes: **CKKS** (approximate arithmetic) and **BFV** (exact arithmetic).
  - Evaluation of key performance metrics
- Significance of the benchmarking:
  - **Feasibility**: Evaluate **performance feasibility** of FHE for ML models.
  - Compare **TenSEAL** and **OpenFHE** libraries using **CKKS** and **BFV** schemes.
  - **Privacy vs Accuracy**: Assess trade-offs between **Privacy**, **Accuracy**.
  - **Library Suitability**: Which library and scheme is best suited?
  - **Industry Adoption**: Provides empirical basis for real-world deployment.

Objective, significance and settings

- Objective of internship:
  - Benchmark and evaluate popular FHE libraries on ML applications.
  - Focus libraries: **TenSEAL** and **OpenFHE**.
  - Focus schemes: **CKKS** (approximate arithmetic) and **BFV** (exact arithmetic).
  - Evaluation of key performance metrics
- Significance of the benchmarking:
  - **Feasibility**: Evaluate **performance feasibility** of FHE for ML models.
  - Compare **TenSEAL** and **OpenFHE** libraries using **CKKS** and **BFV** schemes.
  - **Privacy vs Accuracy**: Assess trade-offs between **Privacy**, **Accuracy**.
  - **Library Suitability**: Which library and scheme is best suited?
  - **Industry Adoption**: Provides empirical basis for real-world deployment.
- Benchmark settings:
  - **Hardware**: 13 Gen Intel(R) core(TM) i9- 13900, RAM 64 GB , 32-core, 2.00 GHz, os Windows 11.
  - **Dataset**: Boston Housing Dataset.
  - **Libraries**: TenSEAL, OpenFHE (v1.3.x)
  - **Schemes**: CKKS (Approximate), BFV (Exact)
  - **Models**: Linear Mean Classifier (LMC), Logistic Regression (LRC)

Evaluate key performance metrics

- FHE Library : TenSEAL
- Total Preparation Time
- Total Training Time
- Inference Time
- Accuracy
- Mean Squared Error (MSE)
- Training Mode :
  1. Plain data
  2. Encrypted data
- ML Model:
  1. Linear Mean Classifier (LMC)
  2. Logistic Regression Classifier (LRC)
- Fully Homomorphic Schemes

Evaluate key performance metrics

- FHE Library : TenSEAL
- Total Preparation Time
- Total Training Time
- Inference Time
- Accuracy
- Mean Squared Error (MSE)
- Training Mode :
  1. Plain data
  2. Encrypted data
- ML Model:
  1. Linear Mean Classifier (LMC)
  2. Logistic Regression Classifier (LRC)
- Fully Homomorphic Schemes
- Average Ciphertext Expansion = Average of (Ciphertext size of test vectors / Plaintext size of test vectors)
- Average Noise Growth = Average of (max $|$(Size of decrypted test vector − Size of original test vector)$|$)
- Total CPU usage percentage:
  1. to prepare the data
  2. to train the model
  3. at the time of inference
- Total RAM usage (in MB):
  1. to prepare the data
  2. to train the model
  3. at the time of inference

Docker Architecture

Micro-services Architecture: Three isolated Docker containers

- Client Service: Handles data encryption/decryption, manages secret keys
- Server Service: Performs homomorphic computations on encrypted data
- Benchmark Service: Orchestrates experiments and collects metrics

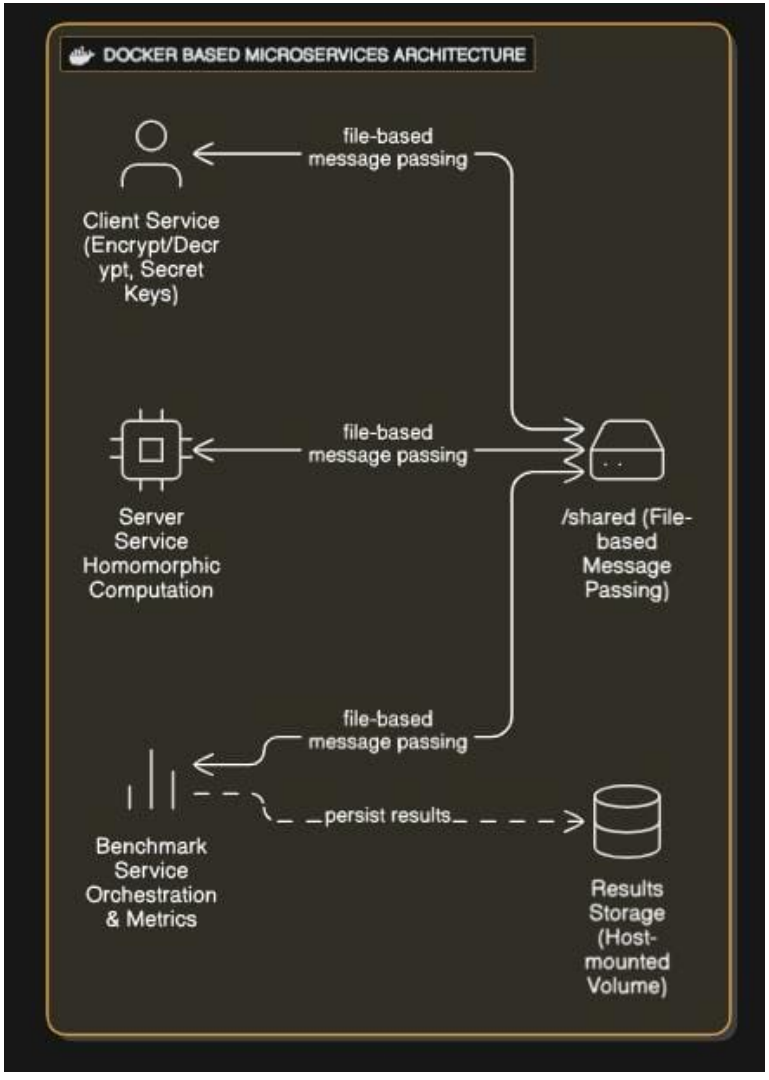
Communication Protocol:

- File-based message passing via shared volumes
- Secure context serialization between containers
- No network overhead for cryptographic material exchange

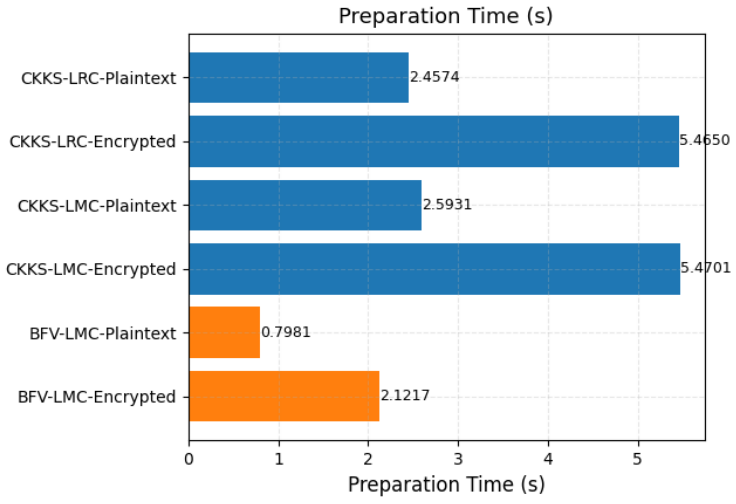
Key Benefits:

- Isolation: Accurate resource measurement without interference
- Reproducibility: Standardized execution environment across systems
- Scalability: Supports single-node to distributed deployments

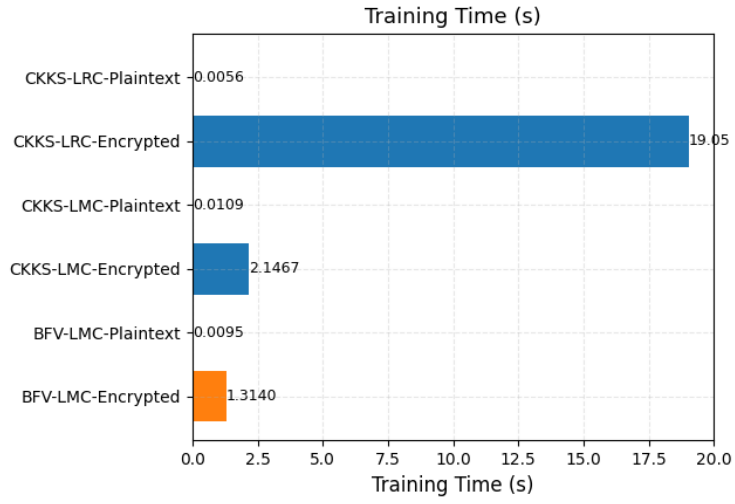
Docker Architecture



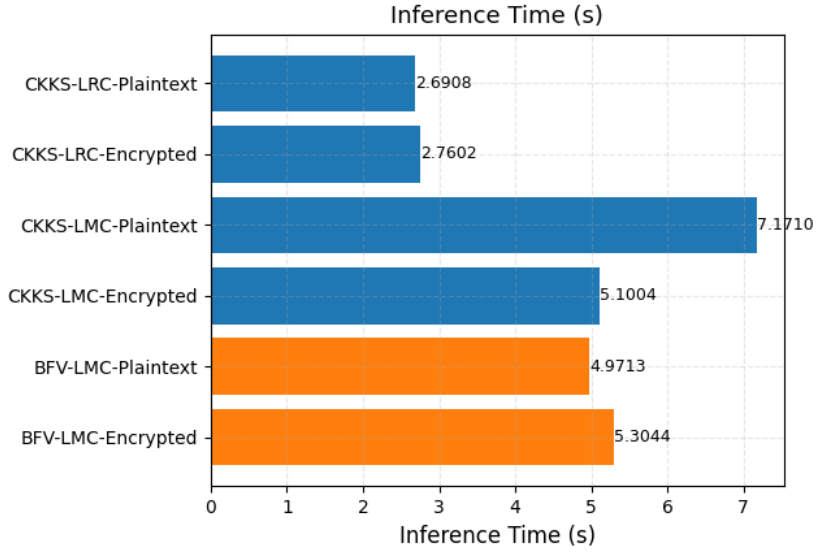
Benchmarking Timings



Preparation Time

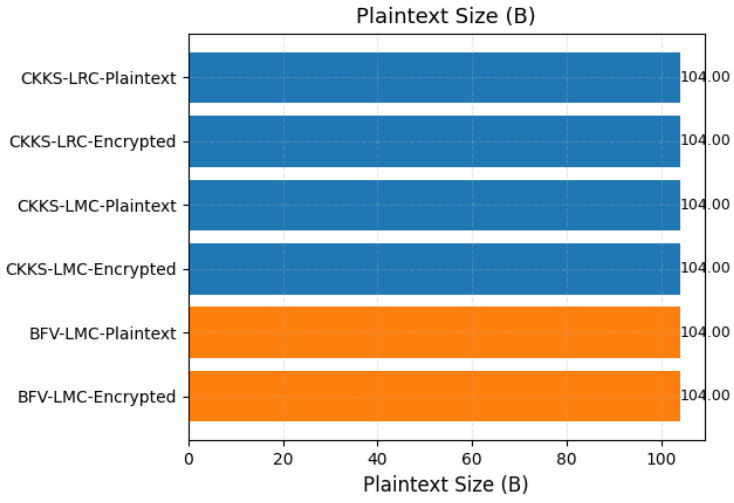


Training Time

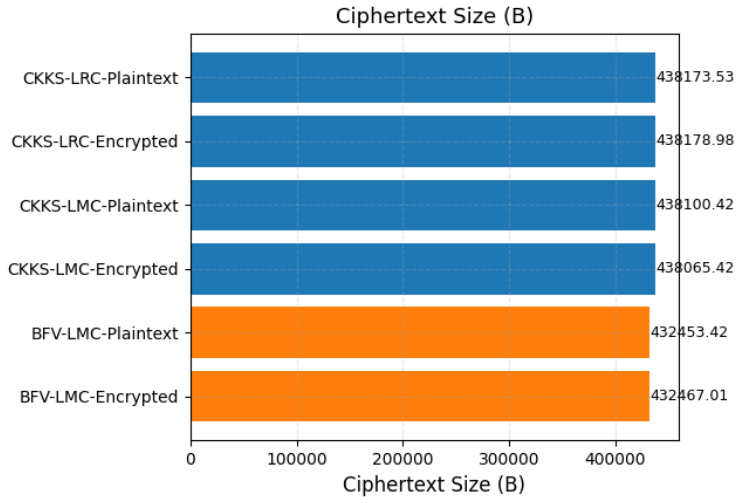


Inference Time

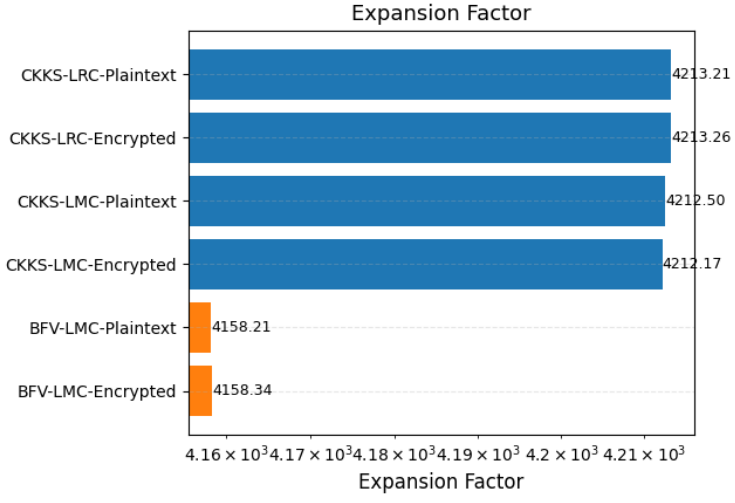
Ciphertext Expansion and Noise Growth



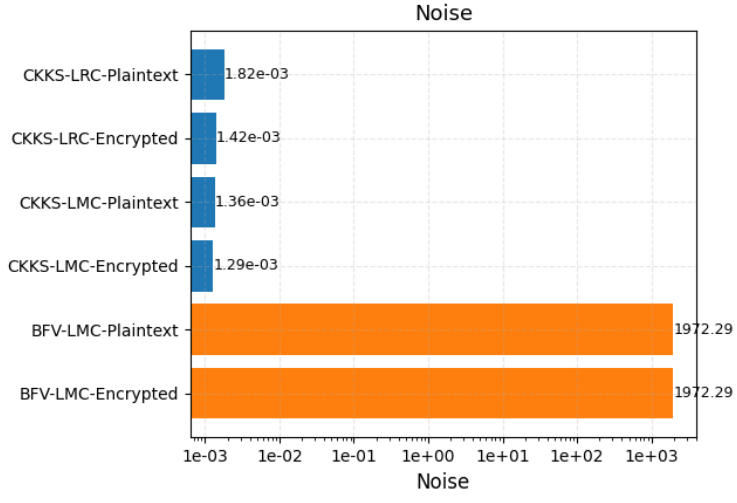
Avg. Plaintext Size (bytes)



Avg. Ciphertext Size (bytes)

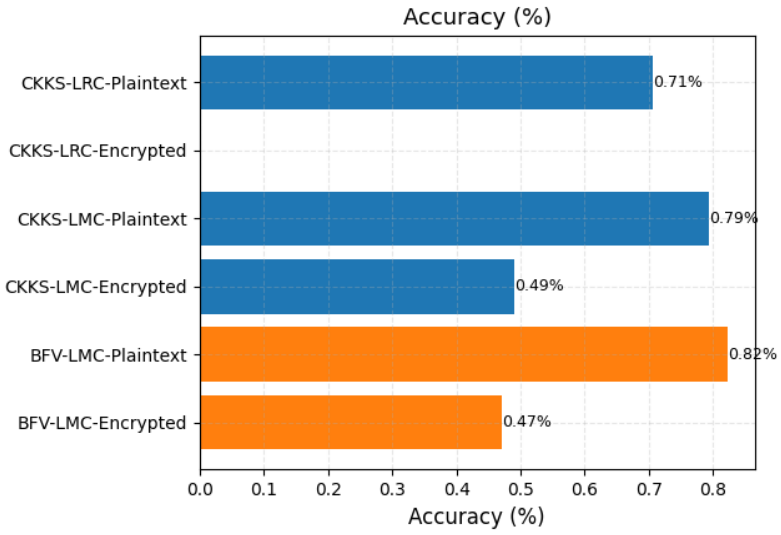


Avg. Expansion Factor

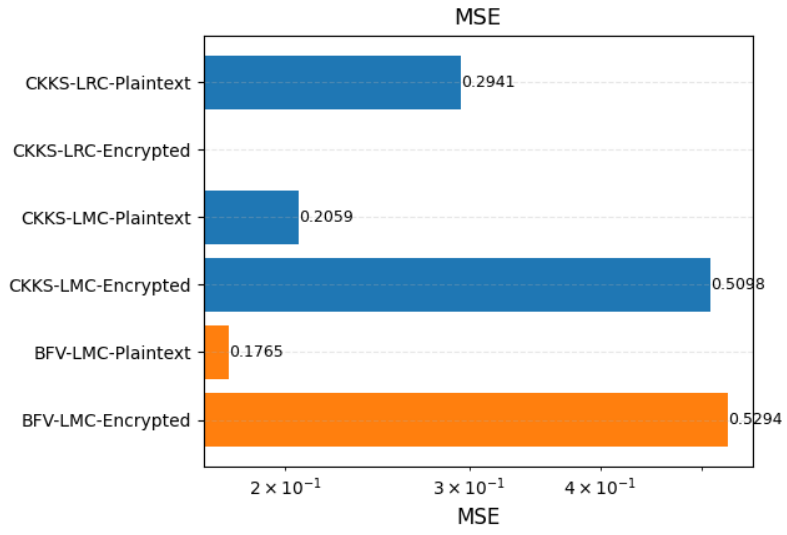


Avg. Noise Growth

Accuracy and MSE

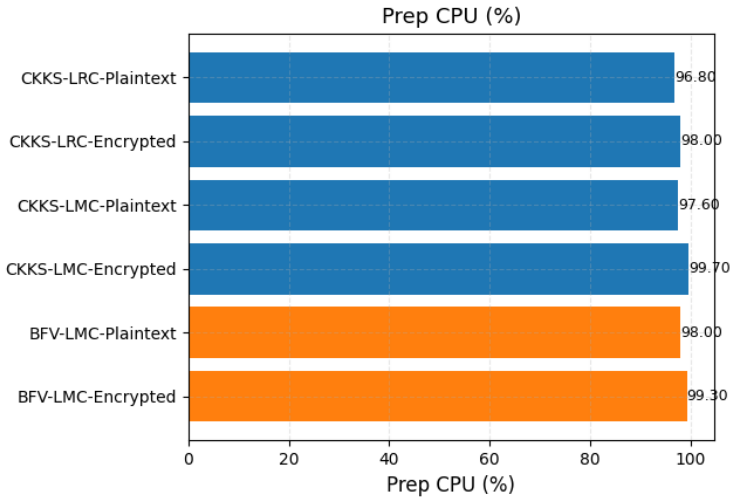


Accuracy

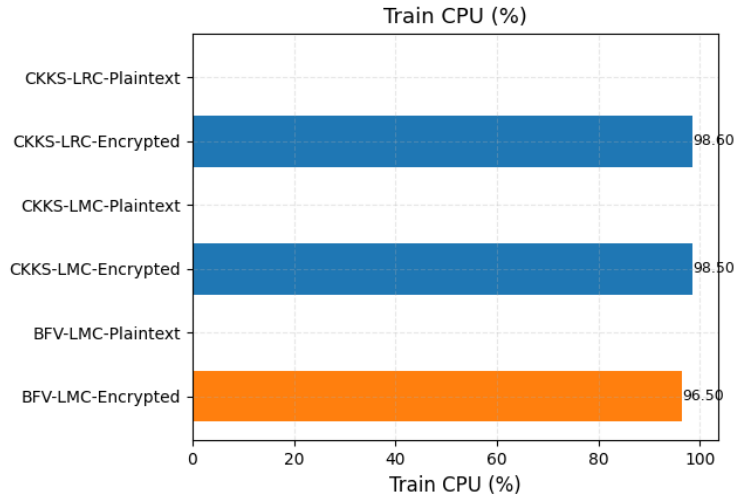


Mean Squared Error (MSE)

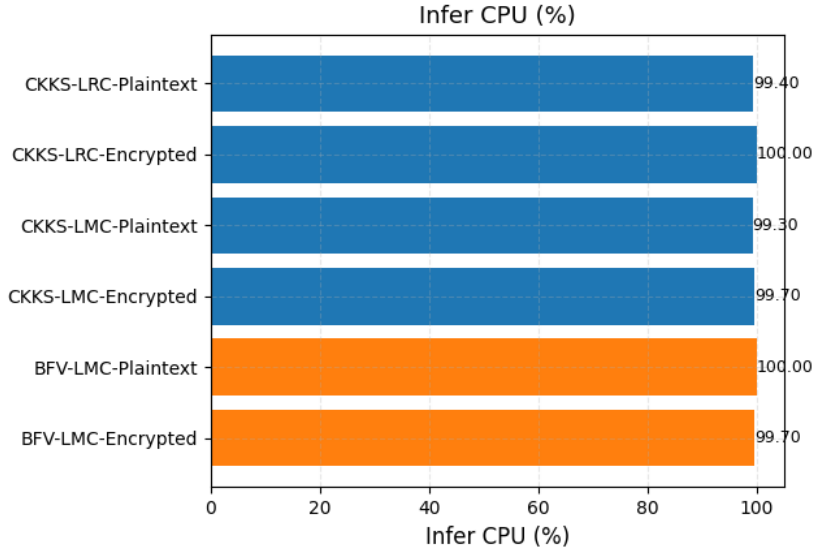
Total CPU Usage % (Preparation, Training, Inference)



Preparation CPU Usage

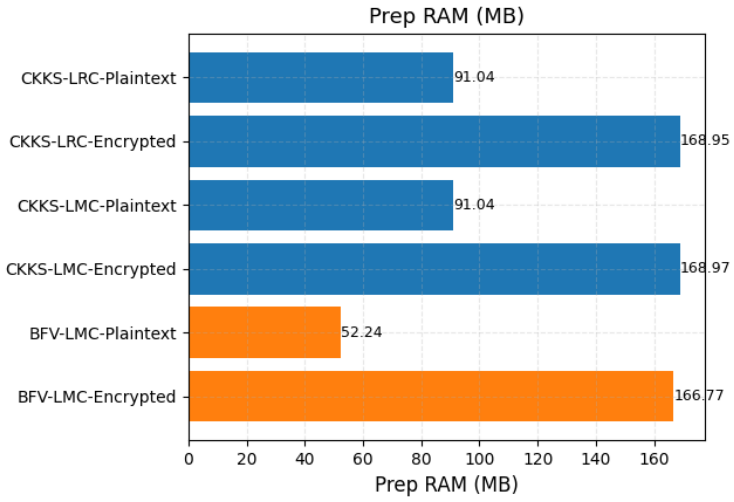


Training CPU Usage

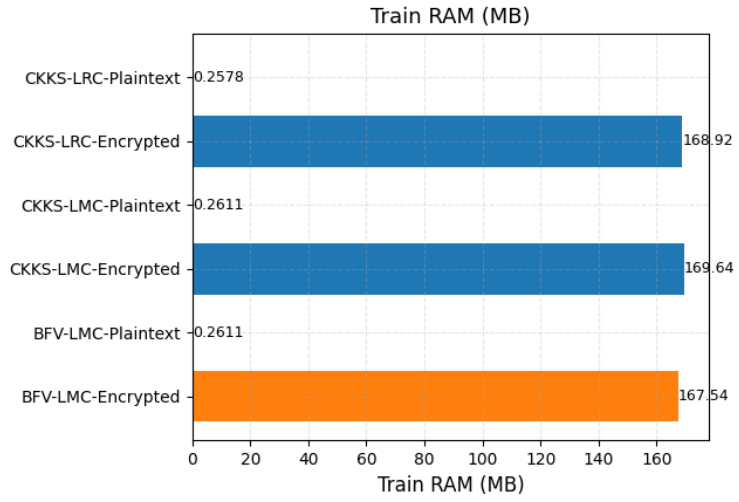


Inference CPU Usage

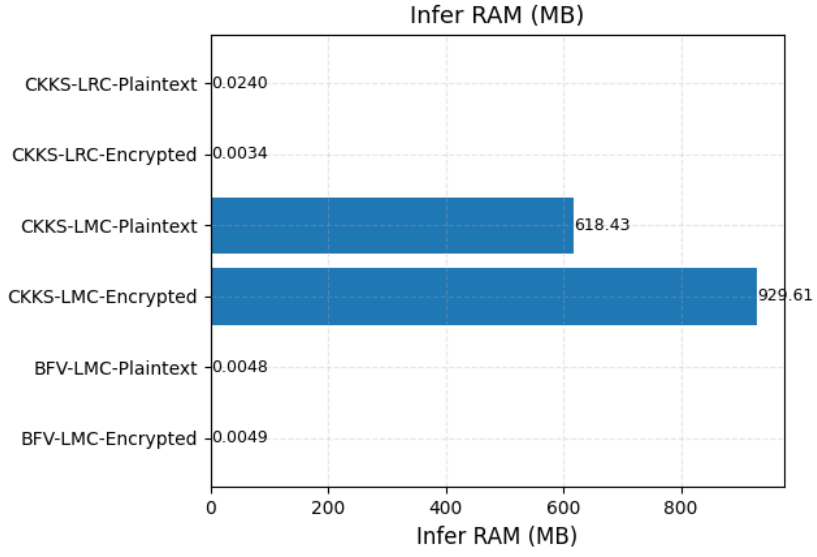
Total RAM Usage (MB)



Preparation RAM Usage

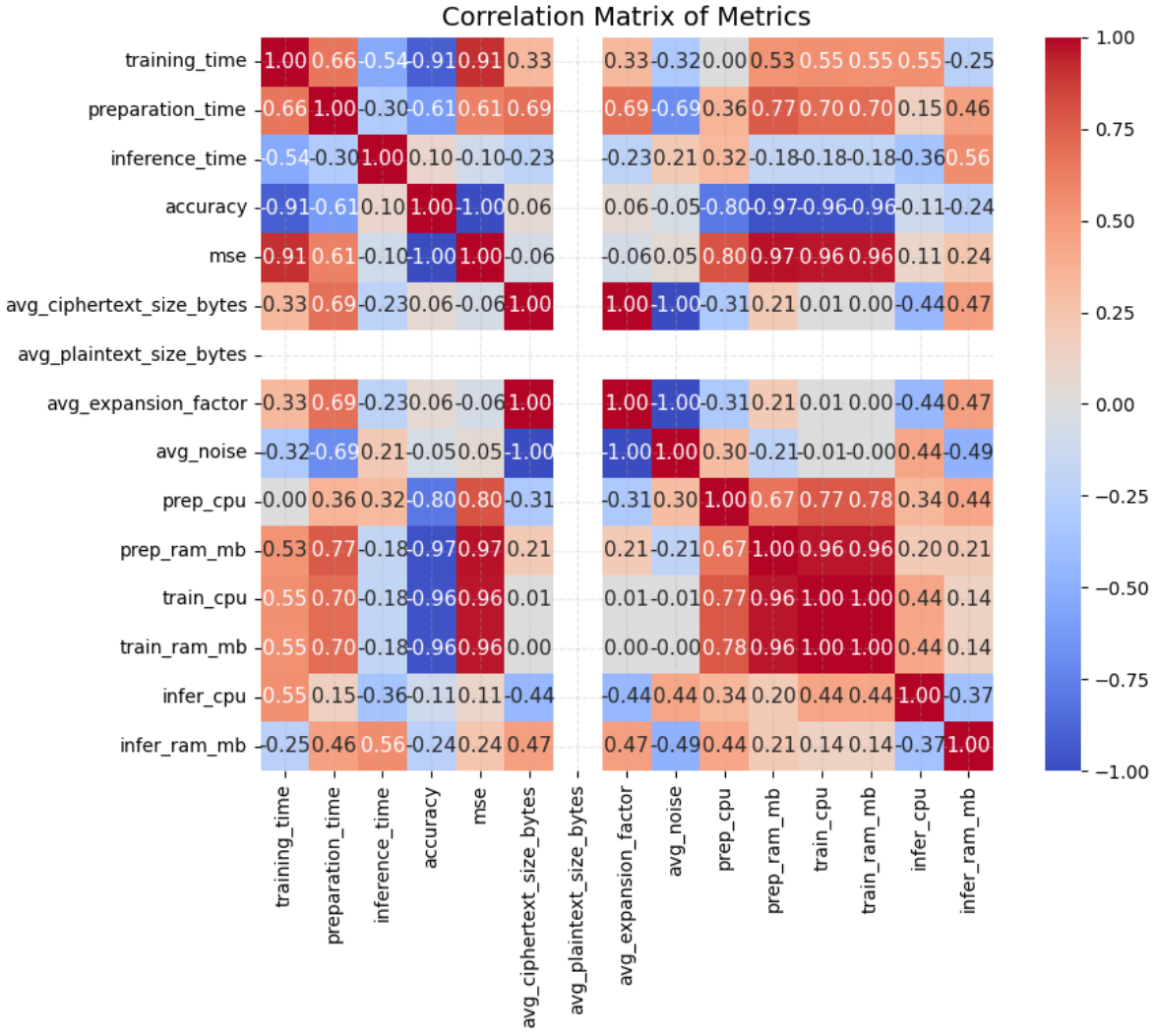


Training RAM Usage



Inference RAM Usage

Correlation Matrix of Metrics



Conclusion and Future Work

- Observations:
  1. Noise growth (training on encrypted data): for BFV is 1972.29 relatively higher than ckks. As a result: very poor accuracy: 47% and MSE: 52%.
     - It would be nice to investigate the reason behind poor performance.
  2. Training time : for “ckks-LRC-encrypted” is higher than “ckks-LMC-encrypted” as expected. But surprisingly, inference time for “ckks-LRC-encrypted” is much lower than “ckks-LMC-encrypted”.
  3. RAM Usage: During inference for the case “ckks-LRC-encrypted” RAM : 0.0034 MB which is very much lower than the case “ckks-LMC-encrypted” (929.61 MB).

Conclusion and Future Work

- Observations:
  1. Noise growth (training on encrypted data): for BFV is 1972.29 relatively higher than ckks. As a result: very poor accuracy: 47% and MSE: 52%.
     - It would be nice to investigate the reason behind poor performance.
  2. Training time : for "ckks-LRC-encrypted" is higher than "ckks-LMC-encrypted" as expected. But surprisingly, inference time for "ckks-LRC-encrypted" is much lower than "ckks-LMC-encrypted".
  3. RAM Usage: During inference for the case "ckks-LRC-encrypted" RAM : 0.0034 MB which is very much lower than the case "ckks-LMC-encrypted" (929.61 MB).

- Assume in the dataset, there are some crucial features with both data types *float* and *int*.
  - But, schemes ckks works for floating point arithmetic and BFV works for integer arithmetic.
  - How do we tackle such scenario?
  - It is important for the model accuracy perspective.

Conclusion and Future Work

- Observations:
  1. Noise growth (training on encrypted data): for BFV is 1972.29 relatively higher than ckks. As a result: very poor accuracy: 47% and MSE: 52%.
     - It would be nice to investigate the reason behind poor performance.
  2. Training time : for "ckks-LRC-encrypted" is higher than "ckks-LMC-encrypted" as expected. But surprisingly, inference time for "ckks-LRC-encrypted" is much lower than "ckks-LMC-encrypted".
  3. RAM Usage: During inference for the case "ckks-LRC-encrypted" RAM : 0.0034 MB which is very much lower than the case "ckks-LMC-encrypted" (929.61 MB).

- Assume in the dataset, there are some crucial features with both data types *float* and *int*.
  - But, schemes ckks works for floating point arithmetic and BFV works for integer arithmetic.
  - How do we tackle such scenario?
  - It is important for the model accuracy perspective.

- For Logistic Regression, polynomial approximation of sigmoid function is necessary.
  - A degree 3 polynomial from https://eprint.iacr.org/2018/462.pdf, which approximate the sigmoid function in the range $[-5, 5]$.
  - $\sigma(x) = 0.5 + 0.197x - 0.004x^3$.
  - How can one train Logistic Regression by BFV?
  - Also in our study, we defined own training method, which is not working properly. It is subject to being thoroughly investigated.

Thank You!