Annals of Mathematics and Computer Science ISSN: 2789-7206
Vol 26 (2025) 141-158 https://doi.org/10.56947/amcs.v26.439

SECURING MACHINE LEARNING MODELS: HOMOMORPHIC ENCRYPTION AND ITS IMPACT ON CLASSIFIERS

RISHITHA ADAMSETTY[1], ASWANI KUMAR CHERUKURI[2*], ANNAPURNA JONNALAGADDA[3]

Abstract. Homomorphic encryption (HME) enables encrypted computations and provides secure data analysis while addressing key concerns around data privacy and regulatory compliance. It has significant implications for machine learning (ML) models, particularly in enhancing the privacy, security, and usability of ML models. Training ML models on homomorphically encrypted data is a growing area of research. While HME offers several security-related benefits to ML models, there are concerns about the ML model's performance, when applied over homomorphically encrypted data. Research efforts are needed to analyze the potential impact of HME on the performance of ML models. This paper deals with our work in this direction, presenting the analysis of various ML classifiers on homomorphically encrypted data. We discuss the performance of classifiers such as logistic regression, Support Vector Machines, Neural networks, and Decision trees on data encrypted using HME. We follow a systematic approach to analyze the performance in terms of accuracy and efficiency. Our results indicate that the performance of classifiers is nearly identical to the results obtained when they were applied to unencrypted data.

1. Introduction

Data privacy has become critical in the age of digital transformation, particularly in cloud models [1] and collaborative computing environments [20]. Cloud computing technologies are highly vulnerable to cyberattacks because they hold enormous volumes of sensitive data. For instance, the controversial 2019 Capital One data breach [2] brought to light the weaknesses in cloud security by revealing the personal data of over 100 million consumers. The advent of collaborative computing environments has complicated data security management since they allow for real-time data sharing and cross-border teamwork. Because of a misconfigured database, a major data breach in 2020 in Microsoft's cloud-based business exposed 250 million client records [3], underscoring the importance of strict security protocols. To protect business and personal data, according to legal requirements, and preserve stakeholder and user trust, it is necessary that these

Date: Received: Dec 11, 2024; Accepted: Jan 10, 2025.
* Corresponding author.
2010 *Mathematics Subject Classification.* Primary 46L55; Secondary 44B20.
Key words and phrases. Decision Trees, Homomorphic Encryption, Logistic Regression, Machine Learning, Neural Networks, Support Vector Machine.

configurations have strong data privacy measures. Organizations suffer a possibility of suffering significant financial losses, legal consequences, and reputational harm if they don't take these precautions.

Data privacy has also become more and more important in the quickly developing fields of artificial intelligence and machine learning (ML) during model training and inference. To learn and produce correct predictions, machine learning models typically need enormous volumes of data, which often include private or confidential information. For instance, [23] the 2017 Equifax hack exposed the private information of over 147 million individuals, triggering worries regarding the security of data utilized in ML training and other computational operations. Privacy issues sometimes arise during inference, the stage of machine learning models where decisions or predictions are made on newly collected information. Sensitive customer data is continuously injected into the model in real-time applications like autonomous systems or personalized recommendations. This data may be intercepted or improperly accessed in the absence of adequate measures. Sensitive speech recordings that were used to train Amazon's Alexa were made public in 2019, a noteworthy incident [22] that brought to light possible risks related to the inference phase. To safeguard individual rights and uphold corporate secrecy, it is important to ensure the privacy of this data [4].

Privacy preserving techniques like Differential Privacy (DP), Federated Learning (FL), secure Multi-Party Computation (MPC) and Homomorphic Encryption (HME) are helpful in safeguarding private information during training and inference of ML models [10]. DP ensures that individual data points cannot be easily re-identified by adding statistical noise to datasets. By utilizing FL, sensitive data can remain localized as models can be trained on a variety of decentralized devices or servers without requiring the transfer of raw data. Secure MPC allows for cooperative data analysis without compromising the privacy of each party's data. As data generated by conventional cryptographic methods cannot be used for computing until data is decrypted, which leads to the risk of system security. HME permit the execution of mathematical operations on encrypted data there by avoiding potential security breaches an unauthorized access during data processing [5].

The potential applications of HME includes: analyzing the patient data such as denomic sequences / clinical records in healthcare industry, protecting client privacy & facilitating advanced analytics on encrypted financial data, reducing the danger of data disclosure & illegal process in cloud environments etc. These uses cases demonstrate how HME responds to the increasing need in industries depending on sensitive data and computational power for reliable data privacy solutions. HME is advantageous over other standard privacy preserving mechanism such as DP, FL and MPC because of the following features:

(1) It guarantees the security of sensitive data during whole computational process by enabling direct computations on encrypted data. This feature ensure the security of data during data processing by outside parties or in cloud environments.

(2) It reduces the danger of data breaches and unauthorized access during computation by guaranteeing end-to-end encryption.

TFHE is the Fast fully homomorphic encryption over Torus. It provides faster homomorphic computations. This study specifically employs the Fast Fully Homomorphic Encryption over the Torus [6] (TFHE) scheme, highlighting its effectiveness and strong cryptographic features. With a focus on classifiers in the context of machine learning, the main objective of this study is to evaluate the viability and efficiency of homomorphic encryption in maintaining the integrity and confidentiality of classifier models during the training and inference stages. This study addresses the complexity involved in applying homomorphic encryption algorithms to classifiers throughout a range of dataset sizes, giving a detailed analysis of the impact of homomorphic encryption on Machine learning classifiers. Performance measures like accuracy, precision, and recall are vital indicators of a model's ability to predict and generalize to new data in typical machine learning workflows. Nevertheless, these metrics need to be assessed in context with the limitations and computational difficulties posed by encryption when working with HME-encrypted data. When performing performance analysis on HME data, it is important to evaluate how encryption affects model efficiency and accuracy, as well as any inherent trade-offs between privacy and predictive capacity. The compute overhead of homomorphic operations is also measured, as this can impact the times needed for inference and model training.

The rest of the paper is organized as follows. Section 2 provides related work and background. Section 3 provides the methodology we have adopted to analyze the impact of HME techniques on different ML classifiers. Section 4 provides the details on the experiments and results we have obtained. Section 5 provides conclusions and future work.

2. Related Work

In order to improve data security and accuracy in machine learning, B. S. Rao et al. [7] developed a privacy-aware model that combines homomorphic encryption (HE) with a learning automata-based artificial neural network (LA-ANN). The model performs better than current models in terms of accuracy, time delay, key length, cost-effectiveness, and transmission speed. With an emphasis on linear regression, S. Behera and J. R. Prathuri [21] showcased the use of the Pailler cryptosystem, an additive homomorphic encryption technique, to enhance privacy in machine learning models. The ability of the encryption method to secure linear regression by executing operations on encrypted data without requiring decryption is highlighted in the study. FedHEONN, a Federated Learning (FL) technique [19] for one-layer neural networks employing homomorphic encryption (HE), was introduced by Fontenla-Romero et al. Enhancing resistance to model inversion assaults in a federated system is the goal of the technique and demonstrates FedHEONN's effectiveness in communication. A privacy-preserving Federated Learning (FL) technique using homomorphic encryption for healthcare data was proposed by Wang et al. [18], that protects patient privacy and security

by ensuring client identification, quickly eliminating participants who withdraw, and encrypting model parameters using Paillier Homomorphic encryption.

TrustFedHealth [18] is a platform for heart disease prediction in smart healthcare that integrates blockchain, homomorphic encryption (HE), mobile edge computing (MEC), and federated learning (FL). This study discusses concerns about the security, privacy & transparency of data as well as the possibility of making data unhackable with Fully Homomorphic Encryption (FHE). In order to get over FHE processing limitations, the US Defense Advanced Research Projects Agency (DARPA) [17] is funding the DPRIVE project, which attempts to build hardware accelerator processors. A Partial Homomorphic Encryption (PHE) model for precisely forecasting brain tumors is presented in [16], notably utilizing the Paillier Cryptosystem. The encryption model demonstrates efficiency and security in tumor location prediction by employing a single random integer throughout the image to preserve structural information. Fully Homomorphic Encryption (FHE) was utilized by Mittal and Singh [15] in the context of machine learning, notably logistic regression. The study shows promise for safe data analysis, with the logistic regression model performing similarly to encrypted data, demonstrating that FHE can guarantee data privacy during computation.

A performance impact investigation of homomorphic encryption with an emphasis on linear regression was carried out by Prantl et al. [14] After analyzing the online and offline designs, it concludes that although homomorphic encryption is possible for linear regression, it comes with a significant time overhead. Orion [8] presents an automated Fully Homomorphic Encryption (FHE) compiler designed specifically for neural network inference. With an emphasis on convolutional neural networks (CNNs), Orion translates neural networks specified by PyTorch to FHE and shows notable speedups over state-of-the-art alternatives. HE-based privacy-preserving deep learning is by Falcetta and Roveri [9], demonstrates the efficacy of HE-based convolutional neural networks (CNNs) for image classification tasks and provides a comprehensive guide for creating privacy-preserving CNNs. The possible applications of FHE in finance, AI development, and medical research are also discussed.

Overall, the literature survey includes a wide range of findings that demonstrate the changing landscape of homomorphic encryption (HE) applications in machine learning and data security. Researchers have proved the success of privacy-aware models that combine HE with machine learning techniques. From improving privacy in linear regression models to using federated learning techniques with HE for secure model updates, the research highlights the growing importance of HE in protecting sensitive data in a variety of domains, including healthcare and finance. Furthermore, programs such as DARPA's DPRIVE project attempt to reduce the computational limits of fully homomorphic encryption (FHE), paving the way for more widespread acceptance and implementation in real-world circumstances. The literature analysis demonstrates the significant advancements made in using HE to ensure data security.

Machine learning enables computers to learn from data and carry out tasks without the need for explicit programming. Algorithms that are trained on labeled datasets associate input data with matching output labels in supervised

learning. Classifiers are essential to supervised learning because they categorize input data into predetermined groups. Neural networks, support vector machines, decision trees, and logistic regression are a few examples. On the other hand, unsupervised learning involves algorithms looking over unlabelled datasets on their own to find underlying structures or patterns. For example, Dimensionality reduction and clustering are common tasks.

Some existing solutions to secure machine learning models like differential privacy, federated learning (FL), and secure enclaves etc have disadvantages. DP is useful in safeguarding individual data points, can contribute significant noise, affecting model accuracy. While federated learning protects data privacy by training models on decentralized data, it still requires a trusted aggregator, which poses security vulnerabilities. Secure areas offer hardware-based protection, but they may not be appropriate for all deployment circumstances.

Secure multi-party computations use intricate cryptographic protocols to allow several people to collaborate on the computation of a function while maintaining the privacy of their individual inputs. MPC can be difficult to scale and computationally demanding, particularly for large-scale datasets and complicated calculations. The Watermarking Model incorporates unique IDs to monitor ownership and identify unlawful usage in machine learning models. It might not, however, be able to stop model inversion attacks, in which malicious parties use the model to reassemble private training data, compromising the security of intellectual property and privacy. Robustness techniques focus on strengthening the resistance of models against adversarial assaults, which can compromise data privacy by taking advantage of model flaws. When applied to trivial data, these methods could potentially decrease model accuracy and increase computational complexity.

HME performs the computations on encrypted data itself which avoids data breaches during communication & collaboration across various models as well as ML projects. Homomorphic encryption comprises partial, somewhat, and fully homomorphic schemes, each delineated by their computational capabilities [13]. Partial homomorphic encryption allows a singular mathematical operation (addition or multiplication) on encrypted data. Examples include schemes like RSA, ElGamal, Paillier, Goldberg-McCarthy, and DGHV. Somewhat homomorphic encryption schemes (SWHE), for example BGN expands this scope, allowing both addition and multiplication operations, yet constraints still exist on the complexity or number of operations. In contrast, fully homomorphic encryption (FHE) enables arbitrary combinations of addition and multiplication operations on encrypted data, mirroring the full range of computations possible on plaintext. BFV, CKKS and TFHE are some examples for FHE schemes. In this study, TFHE scheme is used. The detailed study on various features of HME and TFHE are discussed in Appendix A.

One distinctive feature of TFHE and also the specialty of TFHE is its ability to perform functional bootstrapping during the rescaling and rounding steps. This means that, in addition to reducing noise, the bootstrapping process can evaluate a function on the encrypted message. The function to be evaluated is represented as a lookup table encoded in the ciphertext. By modifying the initial

vector v used during the blind rotation to represent a different function, one can programmatically perform computations on the encrypted data. This means that one can reduce noise and evaluate a function simultaneously in case of TFHE. So, this is faster and efficient because of its programmable bootstrapping technique [12] compared to other FHE schemes.

3. Proposed Methodology

This research aims at analyzing the impact of HME on various ML classifiers. It also evaluates the performance of ML models on un-encrypted data as well as encrypted data. The schematic flow of the procedure adopted in this study is depicted figure 1.

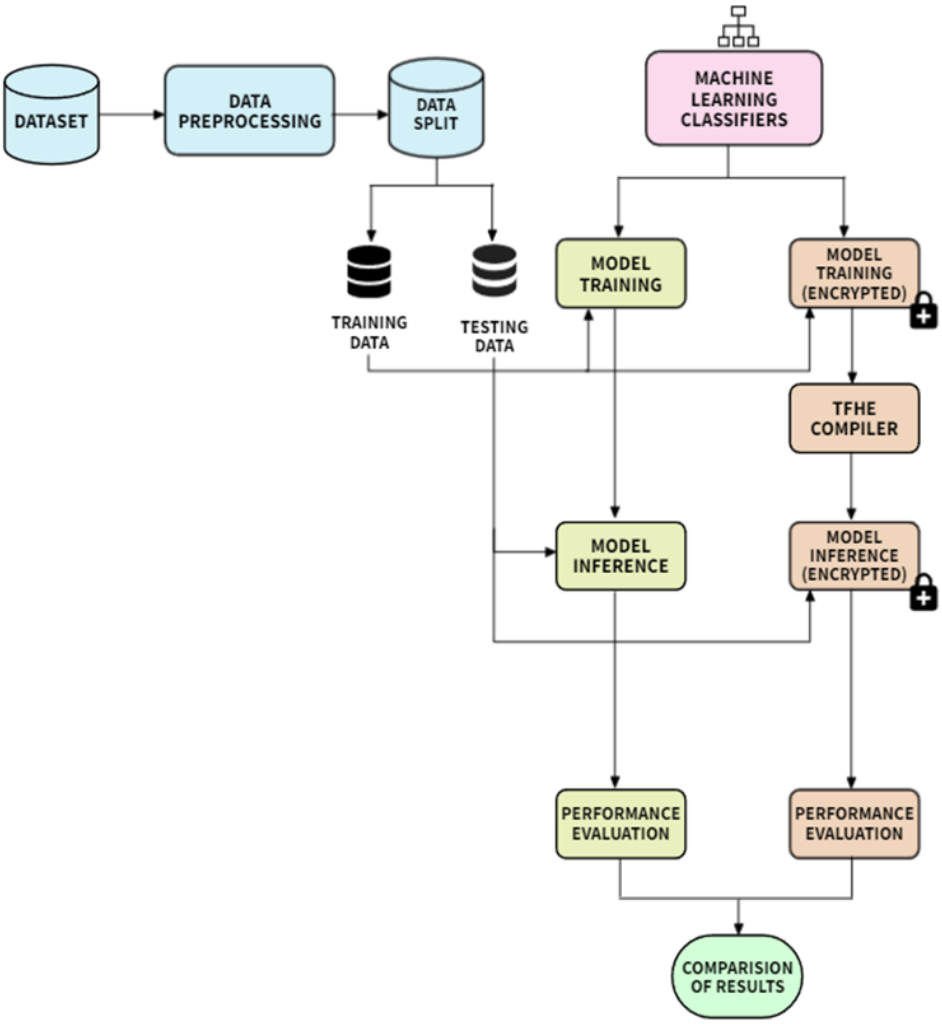


Figure 1. Architectural Diagram

The following are various steps involved in accomplishing the task:

(1) **Pre-processing:** This step is necessary to address the missing values, scaling features, encoding categorical variables etc. This guarantees uniformity in the format and quality of data.
(2) **Splitting the data:** The chosen data set is split into training data and testing data in the ratio 80:20 respectively. The training data is used to train the ML classifier where as the testing data is used for classifier inference.
(3) **ML classifiers:** This study considers various ML classifiers such as Logistic Regression(LR), Support Vector Machine(SVM), Neural Networks(NN) and Decision Trees(DT). The classifier models are implemented in two different versions: 1. Trained with Un-encrypted Data(MTUD), 2. Trained on Encrypted Data (MTED). During the process of training, the model undergoes quantization where all values are converted into integers to facilitate compatibility with FHE constraints. In this way the model can be trained in an encrypted manner. After training, the model is compiled using Concrete's FHE Compiler, producing an equivalent FHE circuit represented as an MLIR program. This serves as an abstraction layer that enables efficient transformation, optimization, and translation of the machine learning model into a form suitable for execution in a privacy-preserving manner using FHE with low-level Cryptographic operations. This step ensures that the model is suitable for secure computations in an encrypted environment. Finally, the compiled model can perform private inference on encrypted inputs, utilizing the generated keys. Following this the model is checked for its performance using the same metrics used for first version. This resultant model is a secure and privacy-preserving machine learning model capable of processing encrypted data without revealing sensitive information throughout the process.
(4) **Model Inference:** In order the check the predictions of the classifiers, testing data set is used by the trained ML classifier models. The inferences are further validated using various performance metrics discussed below.
(5) **Performance Evaluation:** The performance of various ML classifiers using the datasets described by [24, 25]. The following are different metrics used for evaluation: Accuracy, Precision, Recall, F1-score and computational time.

3.1. **DataSets.** This study uses two datasets of different sizes related to the health industry, collected from Kaggle are used in this study. One is a smaller dataset, Heart-Attack dataset [24] of size 12 KB consisting of 14 columns and 304 rows. And the other one is a larger dataset, Diabetes dataset [25] of size 22,206 KB consisting of 22 columns and 2,53,680 rows.

3.2. **Technical Specifications.** Google Collab is used for testing and performance metrics analysis because of its user-friendliness, collaborative features, and easy interaction with Python libraries such as scikit-learn for machine learning applications. The models are trained and tested using scikit learn and pytorch libraries and concrete ML library to implement TFHE.

TABLE 1. Performance Analysis of classifiers on Heart Attack dataset

Classifier	Logistic Regression		Support Vector Machine		Neural Networks		Decision Trees	
Type	MTUD	MTED	MTUD	MTED	MTUD	MTED	MTUD	MTED
Accuracy	0.8524	0.8524	0.8688	0.8688	0.8524	0.8524	0.8360	0.8360
Precision	0.8709	0.8709	0.875	0.875	0.8709	0.8709	0.8437	0.8509
Recall	0.8437	0.8437	0.875	0.875	0.8437	0.8437	0.8437	0.8360
F1-score	0.8571	0.8571	0.875	0.875	0.8571	0.8571	0.8437	0.8352
Time	0.0223 s	1.0427 s	0.0349 s	1.4467 s	120 s	860 s	60 s	690 s

TABLE 2. Performance Analysis of classifiers for Diabetes dataset

Classifier	Logistic Regression		Support Vector Machine		Neural Networks		Decision Trees	
Type	MTUD	MTED	MTUD	MTED	MTUD	MTED	MTUD	MTED
Accuracy	0.8658	0.8658	0.8631	0.8629	0.8642	0.8642	0.8661	0.8661
Precision	0.8337	0.8337	0.8249	0.8213	0.8343	0.8343	0.8333	0.8333
Recall	0.8658	0.8658	0.8631	0.8629	0.8642	0.8642	0.8661	0.8661
F1-score	0.8336	0.8336	0.8236	0.8107	0.8385	0.8385	0.8225	0.8225
Time	0.4278 s	419.819 s	10.117 s	440.757 s	120 s	2500 s	100 s	890 s

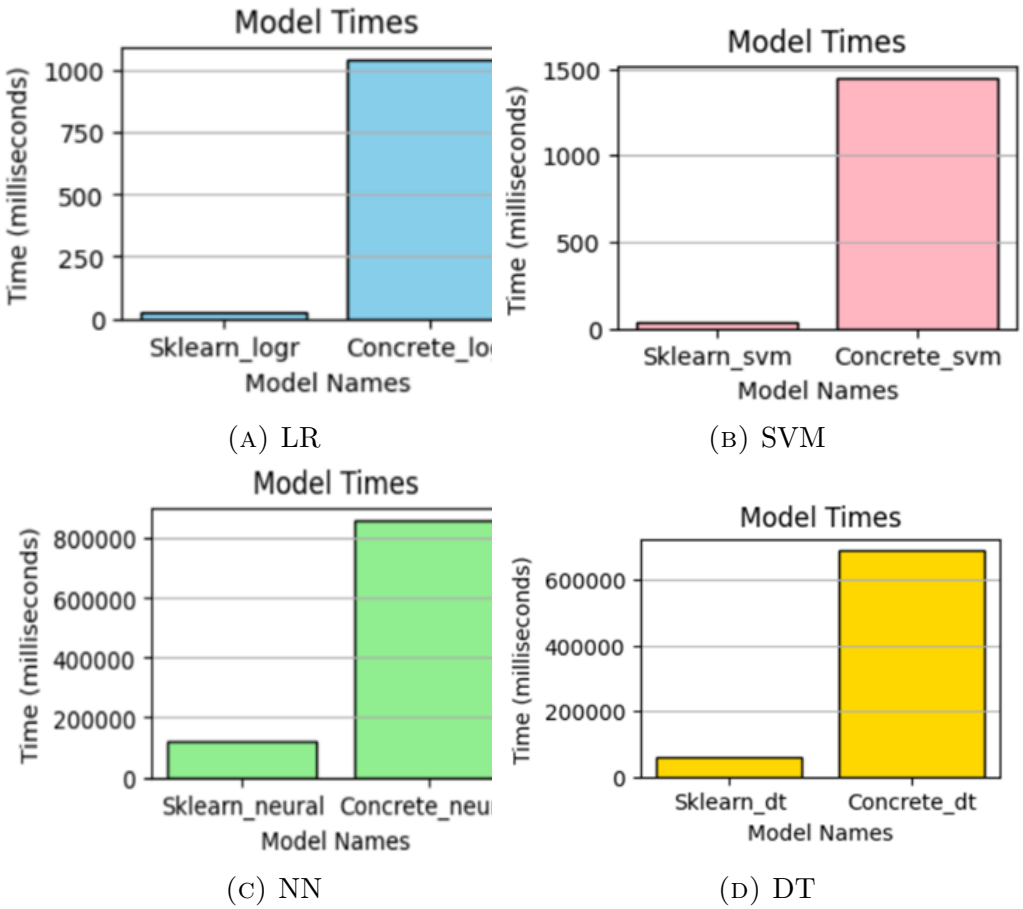
The following section discusses the performance of the ML classifiers using various evaluation metrics.

4. Results and Discussion

Table 1 illustrates the performance of MTUD and MTED versions of LR, SVM, NN & NN on Heart Attack set dataset [24]. It is evident from these results that there is consistency in the performance metrics of MTUD and MTED versions of all four classifiers. Even though a slight difference is seen in the case of DT classifier in which Recall & F1-score of MTED version is little higher than the MTED version and precision is seen more for MTUD version, but overall the difference is negligible.

we further analyzed the computational time taken by MTUD and MTED versions of the classifiers. It is evident from the results depicted in Fig 2 that there is a significant increase in time when using homomorphically encrypted classifiers.

The analysis is further carried out on a larger Diabetes dataset [?]. Table 2 demonstrates the same. one can observe that there is a very slight decrease in metrics like Accuracy, Precision, Recall, and F1 score of encrypted versions of classifiers. This can be clearly observed in the case of the Support Vector Machine. For example, accuracy for the unencrypted version is 0.8249, and the encrypted version is 0.8213. This might be due to the complexity and computational overhead raised by using a larger dataset. The metrics of other homomorphically encrypted classifiers when rounded off to three nearest digits give the same results



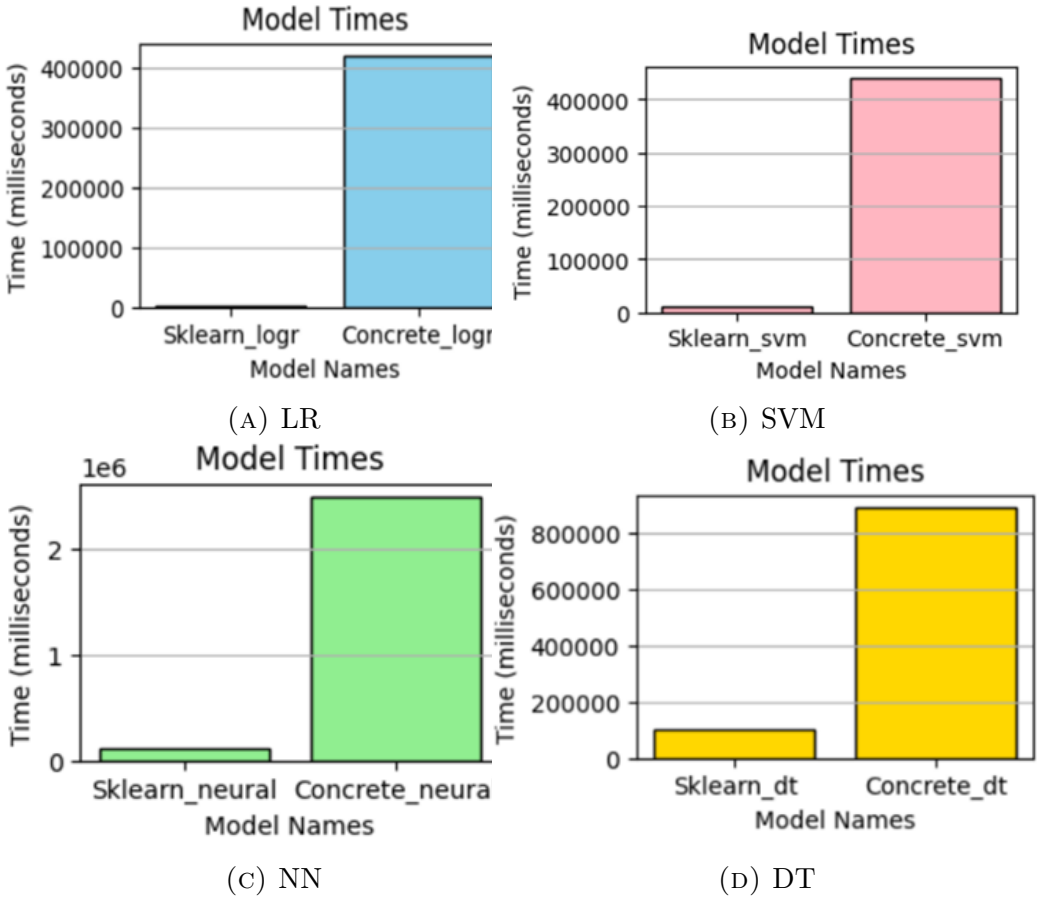
(A) LR (B) SVM

(C) NN (D) DT

FIGURE 2. Comparison of Computational time of MTUD and MTED models of LR, SVM, NN and DT over Heart Attack Dataset

as those of unencrypted classifiers. When implementing more complex classifiers like Neural networks and decision trees for this larger dataset the computational time is so high as shown in Fig 3. It may also be due to insufficient resources and selecting different hardware accelerators to continue execution.

Overall analysis shows that all algorithms performed similarly on both unencrypted and encrypted datasets. This shows that homomorphically encrypted machine learning models can perform similarly to their unencrypted counterparts. Furthermore, the insights provided into the computation time for each case show that, while the encryption process may take longer in the encrypted dataset, the overall computational efficiency and overhead associated with performing operations on encrypted data are still acceptable. The findings have major significance for this project, demonstrating that using homomorphic encryption does not impair the performance of machine learning models. This underlines the feasibility of using homomorphic encryption approaches to improve data privacy and security while maintaining model efficiency. Furthermore, while processing encrypted datasets results in an increase in computing time, this slight discomfort diminishes in comparison to the important data privacy and security benefits provided



(A) LR (B) SVM

(C) NN (D) DT

FIGURE 3. Comparison of Computational time of MTUD and MTED models of LR, SVM, NN and DT over Diabetes Dataset

by homomorphic encryption. Homomorphic encryption enables enterprises to utilize the full potential of machine learning while maintaining confidentiality by protecting sensitive data throughout the whole machine learning pipeline.

Simple Encrypted Arithmetic Library (Microsoft SEAL), Homomorphic Encryption Library (Helib), and Privacy-Preserving Advanced Library Implementations and Standards Definition (PALISADE) are some more alternative libraries [11] that can be used to implement FHE schemes. This study uses the health sector, but Homomorphic Encryption for privacy preservation can be implemented in various real-case scenarios that include sensitive data. For example, in financial data security for encrypted financial transactions or privacy-preserving credit scoring, or privacy-preserving machine learning for regression, clustering, natural language processing tasks, security in the cloud etc.

5. CONCLUSION AND FUTURE WORK

In conclusion, comparing performance metrics between unencrypted and encrypted datasets across different classifiers shows that homomorphic encryption integrates smoothly with machine learning operations while maintaining model

performance. This encryption adaptability ensures that sensitive data can be secured without compromising prediction model accuracy and dependability.

These results have significance in building safer and privacy-preserving machine learning applications in industry and academia. Business organizations can leverage HME techniques to utilize sensitive data confidentially while maintaining privacy as per data privacy regulations. Future research directions in this topic would include handling the complexity of HME models and making HME compatible with all ML models.

References

[1] Can, O., Thabit, F., Aljahdali, A. O., Al-Homdy, S., Alkhzaimi, H. A. (2023). A comprehensive literature of genetics cryptographic algorithms for data security in cloud computing. Cybernetics and Systems, 1-35. https://doi.org/10.1080/01969722.2023.2175117

[2] Kafi, M. A., Akter, N. (2023). Securing financial information in the digital realm: case studies in cybersecurity for accounting data protection. American Journal of Trade and Policy, 10(1), 15-26. https://doi.org/10.18034/ajtp.v10i1.659

[3] Hemanth, D. J., Gupta, B. B., Elhoseny, M., Shinde, S. V. (Eds.). (2023). Intelligent Edge Computing for Cyber Physical Applications. 10.1016/C2021-0-02211-2

[4] Su, G., Wang, J., Xu, X., Wang, Y., Wang, C. (2024). The Utilization of Homomorphic Encryption Technology Grounded on Artificial Intelligence for Privacy Preservation. International Journal of Computer Science and Information Technology, 2(1), 52-58. https://doi.org/10.62051/ijcsit.v2n1.07

[5] Tekin, E. N. (2023). Homomorphic Encryption: A Comprehensive Study of Types, Techniques, and Real-world Applications (Master's thesis, Middle East Technical University). https://hdl.handle.net/11511/105506

[6] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M. (2020). TFHE: fast fully homomorphic encryption over the torus. Journal of Cryptology, 33(1), 34-91. https://doi.org/10.1007/s00145-019-09319-x

[7] Rao, B. S., Chattopadhyay, S., Singh, P., Hazela, B., Sabarinathan, G., Yamini, K. (2023, June). Privacy-Aware Artificial Intelligence with Homomorphic Encryption using Machine Learning. In 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS) (pp. 259-265). IEEE. 10.1109/ICSCSS57650.2023.10169776

[8] Ebel, A., Garimella, K., Reagen, B. (2023). Orion: A Fully Homomorphic Encryption Compiler for Private Deep Neural Network Inference. arXiv preprint arXiv:2311.03470. https://dblp.org/rec/journals/corr/abs-2311-03470

[9] Falcetta, A., Roveri, M. (2022). Privacy-preserving deep learning with homomorphic encryption: An introduction. IEEE Computational Intelligence Magazine, 17(3), 14-25. https://doi.org/10.1109/MCI.2022.3180883

[10] Naresh, V. S., Thamarai, M. (2023). Privacy-preserving data mining and machine learning in healthcare: Applications, challenges, and solutions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 13(2), e1490. https://doi.org/10.1002/widm.1490

[11] Doan, T. V. T., Messai, M. L., Gavin, G., Darmont, J. (2023). A survey on implementations of homomorphic encryption schemes. The Journal of Supercomputing, 79(13), 15098-15139. https://doi.org/10.1007/s11227-023-05233-z

[12] Chillotti, I., Gama, N., Georgieva, M., Izabachene, M. (2016). Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22 (pp. 3-33). Springer Berlin Heidelberg. https://eprint.iacr.org/2016/870

[13] Jain, N., Cherukuri, A. K. A., Kamalov, F. (2024). Revisiting Fully Homomorphic Encryption Schemes for Privacy-Preserving Computing. In Emerging Technologies and Security in Cloud Computing (pp. 276-294). IGI Global. https://doi-org.ezp.cud.ac.ae/10.4018/979-8-3693-2081-5.ch012

[14] Prantl, T., Engel, S., Horn, L., Kaiser, D., Iffländer, L., Bauer, A., ... Kounev, S. (2023, August). Performance Impact Analysis of Homomorphic Encryption: A Case Study Using Linear Regression as an Example. In International Conference on Information Security Practice and Experience (pp. 284-298). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-99-7032-2_17

[15] Mittal, S., Singh, S. (2023, August). Data Analytics over Encrypted Data from Fully Homomorphic Encryption. In 2023 IEEE 4th Annual Flagship India Council International Subsections Conference (INDISCON) (pp. 1-5). IEEE. https://content.iospress.com/articles/journal-of-computer-security/jcs219001

[16] Sarwar, A., Hossain, M. S., Bhuiyan, R. A., Mahmud, T., Zaman, S., Hossin, M. I. (2023, June). Homomorphic Encryption on Deep learning in accurate prediction of Brain Tumour. In 2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM) (pp. 1-6). IEEE. 10.1109/NCIM59001.2023.10212646

[17] Moore, S. K. (2024). Chips to Compute with Encrypted Data are Coming: Fully homomorphic encryption could make data unhackable. IEEE Spectrum, 61(01), 38-40. https://doi.org/10.1109/MSPEC.2024.1038046

[18] Wang, B., Li, H., Guo, Y., Wang, J. (2023). PPFLHE: A privacy-preserving federated learning scheme with homomorphic encryption for healthcare data. Applied Soft Computing, 146, 110677. https://doi.org/10.1016/j.asoc.2023.11067

[19] Fontenla-Romero, O., Guijarro-Berdiñas, B., Hernández-Pereira, E., Pérez-Sánchez, B. (2023). FedHEONN: Federated and homomorphically encrypted learning method for one-layer neural networks. Future Generation Computer Systems, 149, 200-211. https://doi.org/10.1016/j.future.2023.07.018

[20] Farayola, O. A., Olorunfemi, O. L., Shoetan, P. O. (2024). Data privacy and security in it: a review of techniques and challenges. Computer Science IT Research Journal, 5(3), 606-615. https://doi.org/10.51594/csitrj.v5i3.909

[21] Behera, S., Prathuri, J. R. (2020, November). Application of homomorphic encryption in machine learning. In 2020 2nd PhD Colloquium on Ethically Driven Innovation and Technology for Society (PhD EDITS) (pp. 1-2). IEEE. 10.1109/PhDEDITS51180.2020.9315305

[22] Maccario, G., Naldi, M. (2023). Alexa, is my data safe? The (ir) relevance of privacy in smart speakers reviews. International Journal of Human–Computer Interaction, 39(6), 1244-1256. https://doi.org/10.1080/10447318.2022.2058780

[23] Erickson, S. L., Stone, M., Serdar, G., Pfeffer, B. (2023). When Crisis Victims Are Not Customers: SCCT and the Equifax Data Breach. Journal of Managerial Issues, 35(2). info:doi/

[24] Szymon Idziniak, (2021). Heart Attack Analysis & Prediction Dataset: A dataset for heart attack classificationhttps://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset

[25] ALEX TEBOUL, (2015), Diabetes Health Indicators Dataset Notebook https://www.kaggle.com/code/alexteboul/diabetes-health-indicators-dataset-notebook

Appendix A. APPENDIX

A.1. **Homomorphic Encryption.** The Greek words "homo," which means "same," and "morph," which means "shape," are the origins of the word "homomorphic." In mathematics, a homomorphism is a structure-preserving map between two algebraic systems with comparable or identical operations. Unlike traditional encryption, which involves encrypting and decrypting messages, Homomorphic Encryption (HE) stands out by allowing computations over encrypted data. This unique capability enables operations like addition and multiplication on ciphertext without disclosing the underlying information. HE in this way offers end to end encryption that makes it useful in scenarios involving sensitive information like healthcare, finance etc

Let us take two messages m1 and m2 and their encrypted cipher texts c1 = E(m1) and c2 = E(m2). If function E is homomorphic, then one can obtain the value of E(m1 + m2) or E(m1 x m2) by using c1 and c2 without knowing the values of m1 and m2

$$E(m1 + m2) = c1 + c2 = E(m1) + E(m2) \tag{A.1}$$

$$E(m_1 \times m_2) = c_1 \times c_2 = E(m_1) \times E(m_2) \tag{A.2}$$

A.2. **Types of Homomorphic encryption Schemes.** Homomorphic encryption comprises partial, somewhat, and fully homomorphic schemes, each delineated by their computational capabilities [13]. Partial homomorphic encryption allows a singular mathematical operation (addition or multiplication) on encrypted data. Examples include schemes like RSA, ElGamal, Paillier, Goldberg-McCarthy, and DGHV. Somewhat homomorphic encryption schemes (SWHE), for example BGN expands this scope, allowing both addition and multiplication operations, yet constraints still exist on the complexity or number of operations. In contrast, fully homomorphic encryption (FHE) enables arbitrary combinations of addition and multiplication operations on encrypted data, mirroring the full range of computations possible on plaintext. BFV, CKKS and TFHE are some examples for FHE schemes. In this study, TFHE scheme is used.

A.3. **Fully Homomorphic encryption scheme over Torus (TFHE).** TFHE is a Fully Homomorphic encryption scheme over Torus (T = R/Z). The elements in T are actually are in its submodule,

$$T_q = \left\{ \frac{i}{q} \;\middle|\; i \in \mathbb{Z}_q \right\} = \left\{ 0, \frac{1}{q}, \ldots, \frac{q-1}{q} \right\} \tag{A.3}$$

where $q = 2^r$and r is either 32 or 64 and Z_q is the group of integers modulo q. TFHE can perform homomorphic addition, homomorphic multiplication, rotation etc TFHE uses various types of ciphertexts, including LWE (Learning with Errors), Ring LWE, and Ring GSW, to perform these homomorphic operations.

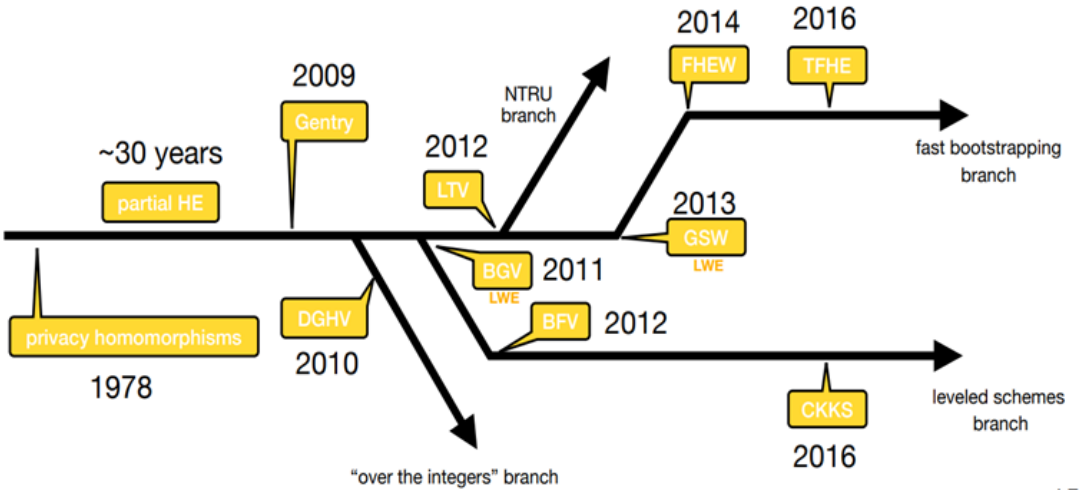


FIGURE 4. Timeline chart of Homomorphic Encryption Schemes [?]

A.3.1. *Learning with Errors (LWE).* Learning With Errors (LWE) is introduced as a hard problem on lattices proposed by Oded Regev in 2005.The problem involves a secret vector, a random set of elements (mask), and a little Gaussian error, resulting in a product called an LWE sample.

$$\text{Message } m \in \mathbb{Z}_p \Rightarrow \text{Ciphertext in } \mathbb{Z}_q^{(n+1)}$$

Say we have a message M and secret be $(s_0, \ldots, s_{n-1}) \in \mathbb{Z}^n$ and random elements be $(a_0, \ldots, a_{n-1}) \in \mathbb{Z}_q^n$ and $e \in \mathbb{Z}_q$ denotes a gaussian error. When we encrypt the message we get,

$$b = \sum_{i=0}^{n-1} a_i \cdot s_i + e + \Delta m \in \mathbb{Z}_q \tag{A.4}$$

So, we can represent LWE ciphertext as $(a_0, \ldots, a_{n-1}, b) \in \mathbb{Z}_q^{(n+1)}$. Now, for decryption first we perform,

$$b - \mathbf{a} \cdot \mathbf{s} = \Delta m + e \quad \text{and} \quad \left(\frac{\Delta m + e}{\Delta} \right) \to m \tag{A.5}$$

LWE is shown to be additive homomorphic, allowing the addition of two ciphertexts under the same secret key. It also has limited multiplicative homomorphic property, where ciphertexts can be multiplied to a small constant.

$$\text{When, } C_1 = (a^1, b_1) \text{ and } C_2 = (a^2, b_2), C_{\text{add}} = C_1 + C_2 = (a^1 + a^2, b_1 + b_2) \tag{A.6}$$

$$\text{When, } C = (a, b) \text{ and } \gamma \text{ is a small integer, } C_{\text{mul}} = \gamma \cdot C = (\gamma \cdot a, \gamma \cdot b) \tag{A.7}$$

A.3.2. *Learning with Errors Over Rings (RLWE).* In RLWE, Messages are polynomials modulo p, reduced to the modulo xn+1 where n is the cyclotomic polynomial. The ciphertext consists of two polynomials, with the secret key being a polynomial modulo p.

$$\text{Message } M \in \mathbb{Z}_p[X]/(X^n+1) \Rightarrow \text{Ciphertext in } (Z_q[X]/(X^n+1))^2$$

Say we have a message M and secret be $(S_0 + S_1X + \cdots + S_{N-1}X^{(N-1)}) \in \mathbb{Z}^n$ and random elements be $(A_0 + A_1X + \cdots + A_{N-1}X^{(N-1)}) \in \mathbb{Z}_q^n$ and $(E_0 + E_1X + \cdots + E_{N-1}X^{(N-1)}) \in \mathbb{Z}_q$ denotes a gaussian error. When we encrypt the message we get,

$$B = A \cdot S + E + \Delta M \in \mathbb{Z}_q[X]/(X^n+1))2 \tag{A.8}$$

So, we can represent RLWE ciphertext as $(A_0 + A_1X + \cdots + A_{N-1}X^{(N-1)}, B) \in \mathbb{Z}_q[X]/(X^n+1))2$. Now, for decryption first we perform,

$$B - A \cdot S = \Delta M + E \quad \text{and} \quad (\Delta M + E)/\Delta \to M \tag{A.9}$$

Homomorphic properties include addition and small constant polynomial multiplication and a large constant multiplication. To multiply with larger constants, a decomposition solution is introduced where the large constant is decomposed using a base and the decomposition elements are multiplied by the encrypted message. The decomposition base is embedded in the ciphertext at the encryption phase, controlling noise growth during multiplication.

$$\text{When } C_1 = (A^1, B_1) \text{ and } C_2 = (A^2, B_2), C_{\text{add}} = C_1 + C_2 = (A^1 + A^2, B_1 + B_2) \tag{A.10}$$

$$\text{When } C = (A, B) \text{ and } \gamma \text{ is a small integer}, C_{\text{mul}} = \gamma \cdot C = (\gamma \cdot A, \gamma \cdot B) \tag{A.11}$$

When $C = (A, B)$ and γ is a large integer, we decompose with respect to a small base like $\beta = 2$ so,

$$\gamma = \gamma_1 \frac{q}{b} + \gamma_2 \frac{q}{b^2} + \ldots + \gamma_l \frac{q}{b^l} \text{ and } C_{\text{L-mul}} = \gamma \cdot C = (\gamma \cdot A, \gamma \cdot B) \tag{A.12}$$

A.3.3. *Gentry-Sahai-Waters over Rings (RGSW).* LWE and RLWE allow only constant multiplication. So, for multiplication between ciphertexts, GSW (Gentry-Sahai-Waters) approach is introduced.

$$M \in \mathbb{Z}_p[X]/(X^N+1) \to \text{Ciphertext in } \left(\mathbb{Z}_q[X]/(X^N+1)\right)^{(2l \times 2)}$$

RSGW allows addition and small constant polynomial multiplication just like in RLWE but RGSW also allows Multiplication between two ciphertexts. When $C_1 = (A^1, B_1)$ and $C_2 = (A^2, B_2)$ the

$$C_{\text{mul}} = C_1 \times C_2 = (A^1 \times A^2 + B_1 \times B_2, A^1 \times B^2 + B_1 \times A_2) \tag{A.13}$$

A.4. **Operations in TFHE.**

A.4.1. *Encoding and Decoding.* The Cipher texts are further encoded after encryption which involves representing messages as polynomials in the ring of integers modulo q. As shown in Figure 5, Message is encoded into Most Significant bits (MSB) and error is encoded into Least Significant Bits (LSB). This makes it easier for result of the operations such as addition and multiplication to stay in MSB. And similarly decoding takes place before decryption.

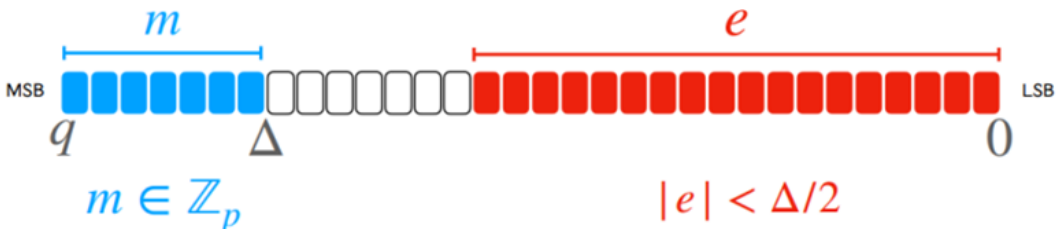


FIGURE 5. Encoding of Cipher Texts [**?**]

A.4.2. *External Product.* In TFHE, external product (RLWE X RGSW) means we can apply product between messages encrypted in RLWE and messages encrypted in RGSW. So, this involves decomposition of RLWE and further multiplying those components with RGSW. This is one of the most important operations in TFHE as it involves C-mux (Controlled mux) which is a fundamental operation in TFHE, functioning as a multiplexer or selector gate. Homomorphic evaluation of C-mux is achieved through linear operations and external products.

$$C_m \otimes C_\mu(x) = \text{RLWE}(X)(m \cdot \mu(x), a(x) + m \cdot \bar{a}(x), e(x) + m \cdot \bar{e}(x)) \quad \text{(A.14)}$$

where C_m is RGSW and $C_\mu(x)$ is RLWE

A.4.3. *Rotation and Blind Rotation.* Rotation is one more operation in TFHE that involves moving the coefficients of a polynomial by a specified number of positions. This can be achieved by multiplying the polynomial by x^{-p} effectively shifting the coefficients.

$$\text{If } M(X) = M_0 + M_1(X) + \cdots + M_p X^p + \cdots + M_{(N-1)} X^{(N-1)}$$
$$\text{now we need to multiply it with } X^{(-p)} \text{ we get}$$
$$M_p + M_{(p+1)}(X) + \cdots + M_{(N-1)} X^{(N-p-1)} - M_0 X^{(N-p)} - \cdots - M_{(p-1)} X^{(N-1)}$$

Whereas Blind rotation involves encrypting both the message and the number of positions, enabling rotations without revealing the actual positions which is nothing but rotating an encrypted polynomial M by p encrypted positions where the number of positions shifted is kept secret and is achieved using the C-mux gate. For Blind Rotation,

$$p \text{ is represented as } p = p_0 \cdot 2^0 + \cdots + p_j \cdot 2^j + \cdots + p_k \cdot 2^k$$

now we need to multiply it with X^{-p} we get

$$M \cdot X^{-p_0 \cdot 2^0} \cdot \ldots \cdot X^{-p_j \cdot 2^j} \cdot \ldots \cdot M^{-p_k \cdot 2^k}$$

Where $M \cdot X^{-p_j \cdot 2^j} = \begin{cases} M & \text{if } p_j = 0 \\ M \cdot X^{-2j} & \text{if } p_j = 1 \end{cases}$ which is done using Cmux Rotation carries as $M_0 = M \cdot X^{-p_0}$ and $M_1 = M_0 \cdot X^{-(p_1)^2}$ and so on.

A.4.4. *Sample Extraction.* Now, another interesting operation is Sample Extraction which involves extraction of LWE from RLWE. So basically, RLWE to LWE involves rearranging coefficients and does not increase the noise in the ciphertext. The extracted key can be used in subsequent computations without revealing the full secret key. All coefficients are rearranged, for example $a_0 = A_0$, $b = B_0$, $s_0 = S_0$ and $a_1 = -A_{N-1}$, etc.

A.4.5. *Key Switching.* Key Switching is a fundamental operation in TFHE that changes the encryption key associated with a ciphertext. It is used to switch between different sets of keys and parameters, enabling various homomorphic operations. Key switching can be performed between LWE ciphertexts, between ring-LWE and LWE-RLWE ciphertexts.

$$C_{\text{new}} = \text{Keysswitch}(c, s^{\text{new}}) \tag{A.15}$$

For each component of the ciphertext, multiply the corresponding coefficients of the target secret key(new) with the coefficients of the original ciphertext and sum these products for each component.

A.4.6. *Bootstrapping.* After every homomorphic operation certain noise is generated. To remove this noise Bootstrapping is performed. This operation involves Blind rotation, rescaling, key switching etc. First, we perform a blind rotation on the ciphertext. The rotation is essentially a circular shift of the coefficients, and it is accomplished by multiplying the ciphertext by a power of the polynomial variable (x) to achieve the desired rotation. After the blind rotation, the ciphertext now represents the rotated message plus a new noise term. To extract the original message and mitigate the effects of noise, rescaling and rounding operations are applied. The rescaling operation ensures that the noise term is small enough to be manageable. The rounding operation converts the rotated and rescaled message back into an integer value, essentially mapping it to a smaller set of possible values. Since TFHE operates in the ring, a modulus switching operation is performed to reduce the ciphertext from a larger modulus q to a smaller modulus 2n where n is the bit size of the original plaintext space. This reduction helps manage computation in a more efficient manner. Finally, a key-switching operation is used to return the ciphertext to the original encryption key.

One distinctive feature of TFHE and also the specialty of TFHE is its ability to perform functional bootstrapping during the rescaling and rounding steps. This means that, in addition to reducing noise, the bootstrapping process can evaluate a function on the encrypted message. The function to be evaluated is represented as a lookup table encoded in the ciphertext. By modifying the initial vector v used during the blind rotation to represent a different function, one can programmatically perform computations on the encrypted data. This means that one can reduce noise and evaluate a function simultaneously in the case of

TFHE. So, this is faster and more efficient because of its PROGRAMMABLE BOOTSRAPPING technique [12] compared to other FHE schemes.

[1,2] School of Computer Science Engineering and Information Systems, Vellore Institute of Technology, Vellore-632014, India
Email address: adamsetty.rishitha2020@vitstudent.ac.in, chrukuri@acm.org

[3] School of Computer Science and Engineering, Vellore Institute of Technology, Vellore-632014, India
Email address: jannapurna@gmail.com