Proceedings of CONF-SPML 2026 Symposium: The 2nd Neural Computing and Applications Workshop 2025
DOI: 10.54254/2755-2721/2026.TJ30594

The Integration of Fully Homomorphic Encryption and Machine Learning: Technical Paths, Model Adaptation, and Practical Applications

Qi Wang

The University of Manchester, Manchester, United Kingdom
qiwang.aca@gmail.com

Abstract. The growing adoption of machine learning across areas like healthcare and finance has raised serious concerns around data privacy. Traditional techniques such as differential privacy or federated learning often come with limitations—whether in accuracy, communication cost, or reliance on secure protocols. Fully Homomorphic Encryption (FHE) offers a promising alternative, enabling computation directly on encrypted data and making it possible to use data without ever seeing it in raw form. This paper explores how FHE can be integrated with machine learning workflows, from traditional models like linear regression and decision trees to complex deep learning architectures. We review mainstream FHE schemes and core technologies that make private ML feasible. In particular, we analyze the unique challenges of adapting different model types to FHE constraints and highlight real-world applications in medical imaging, financial models, and edge intelligence. However, critical bottlenecks remain. Large models still face efficiency issues, dynamic data settings are poorly supported, and the field lacks standardized benchmarks. Through this review, we outline key future research directions that can help transition FHE-based machine learning from theoretical promise to practical reality.

Keywords: Fully Homomorphic Encryption (FHE), Privacy-Preserving Machine Learning, Encrypted Data Computation

1. Introduction

The widespread use of machine learning (ML) in finance, security, healthcare and other privacy-sensitive domains requires large-scale collection and processing of personal data, which creates substantial leakage risks and has triggered strict regulations such as the GDPR [1]. Traditional privacy-preserving techniques each have inherent limitations: differential privacy inevitably degrades accuracy [2], and federated learning remains vulnerable to inference and poisoning attacks [3]. In contrast, Fully Homomorphic Encryption (FHE) enables ML computations to be performed directly on encrypted data so that raw data and intermediate results never appear in plaintext, offering a compelling solution for high-privacy AI scenarios.

Since Rivest et al.'s early work on privacy homomorphisms [4] and Gentry's first FHE scheme based on bootstrapping [5], lattice-based constructions [6], leveled FHE and techniques such as

DOI: 10.54254/2755-2721/2026.TJ30594

modulus switching [7] and CKKS [8] have greatly improved practicality. On this basis, FHE-friendly ML models and tools have emerged, including CryptoNets for encrypted Convolution Neural Network (CNN) inference [9], optimizing compilers like CHET for automatic parameter selection [10], and hardware acceleration such as Intel HEXL [11]. These advances have enabled applications in areas like medical image classification and financial risk control, yet efficient training of complex models, as well as unified performance and security evaluation frameworks, remain open challenges [12].

The objective of this paper is to systematically summarize the technical paths, application scenarios and open problems in the integration of FHE and ML, with a particular focus on the adaptation mechanisms between different ML models and FHE schemes. The goal is to provide a concise reference framework that links theory and practice and supports the design, implementation and evaluation of privacy-preserving ML systems based on FHE.

2. Basic principles of fully homomorphic encryption and machine learning

2.1. Principles of FHE

The basic idea of FHE is to do operations on encrypted data without decrypting it. So, when the user has an input message and a model that supports FHE, the system will first encode this message to a plaintext, and encrypt the plaintext to a ciphertext with a private secrete key, following the given rules of the encrypting scheme used in the model. There are several public keys that are known for all and have different purposes. Next, this ciphertext is provided to the model, and the model will do homomorphic operations on the ciphertext, producing a ciphertext result to the user. The user can finally use the secrete key to decrypt the ciphertext and decode it to receive the result visibly.

The homomorphic operations are denoted by $\oplus$ (addition) and $\otimes$ (multiplication). Given two plaintexts a and b, we denote $\mathrm{Enc}(\cdot)$ and $\mathrm{Dec}(\cdot)$ to be the encryption and decryption and the homomorphic operations satisfy the following rules:

$$Dec\left(Enc\left(a\right)\oplus \mathrm{Enc}\left(\mathrm{b}\right)\right)=a+b \tag{1}$$

$$Dec\left(Enc\left(a\right)\otimes \mathrm{Enc}\left(\mathrm{b}\right)\right)=a\times b \tag{2}$$

Each ciphertext carries an inherent noise term (quantified as the error in the LWE problems), which will be erased after successfully decrypting if the noise does not exceed a threshold ruled by the safety parameter of the cryptosystem. However, the noise is amplified with each homomorphic operation on ciphertext, particularly with multiplication. If the intrinsic noise exceeds the threshold, the decryption will fail. As a result, bootstrapping plays an important role in refreshing the noise of the ciphertext. It will reduce the noise to a manageable level and thus allow subsequent homomorphic operations while maintaining security and accuracy.

Single Instruction Multiple Data (SIMD) operation is a parallel technique that significantly enhance the efficiency of homomorphic encryption scheme. By enabling the encoding multiple messages into a single ciphertext, SIMD allows homomorphic operations to be applied simultaneously across all messages within that ciphertext. This approach effectively reduces the computational overhead associated with processing large volumes of data, thereby improving throughput and optimizing resource utilization. There are several mainstreaming FHE schemes. Table 1 provides a comparison of these schemes.

DOI: 10.54254/2755-2721/2026.TJ30594

Table 1. Comparison between different FHE schemes

FHE Schemes	Data Type	Advantages	Suitable model	Drawbacks
CKKS	Real number	Can process floating numbers.	Deep learning	Precision loss, noise scale $O(n)$
BGV/BFV	Integer	Controllable accuracy	Traditional ML	Need to preprocess parameters to integer
TFHE	Boolean	Fast bootstrapping	Lightweight models	Cannot evaluate polynomials efficiently

2.2. Principles of ML

There are several models to be discussed. For linear regression, we have

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p \tag{3}$$

to describe the linear relationship between p explanatory variables $\mathrm{X_i}$, $\mathrm{i} = 1, \ldots, \mathrm{p}$ and the response variable Y. A set of parameters β_i, $\mathrm{i} = 0, \ldots, \mathrm{p}$ are needed to be determined by least square estimation in order to minimize the sum of squares of errors between predicted and actual values.

The Logistic Regression is used to classify the events by producing probabilities of the binary results. It computes a linear predictor and then transforms it to obtain the predicted probability by the Sigmoid function. The Sigmoid function is given by

$$\sigma: \ (-\infty, +\infty) \to [0,1], \ \ \sigma(z) = \frac{1}{1+e^{-z}} \tag{4}$$

The Decision Tree is a supervised learning algorithm to do either classification or prediction based on using feature conditions to partition the sample space.

The CNN is composed of several functional layers. We first input our images into input layer, and then do convolution process in convolution layer. The convolution process is outlined in Figure 1, where the convolution kernel slides through the input image with given parameters, such as strides and padding, and does dot product on each convolution window to produce an output. Then we have activation layer, which adds non-linearity to the output by calculating activation function element-wise, and pooling layer, which extract a representative value, such as mean value or maximum value, from each local region in order to reduce the output size and computations. Finally, we have Full Connected layer to linearly transform the output into a vector, which is the final output representing probability or score of each category. A rough diagram of CNN is given in Figure 2.

DOI: 10.54254/2755-2721/2026.TJ30594

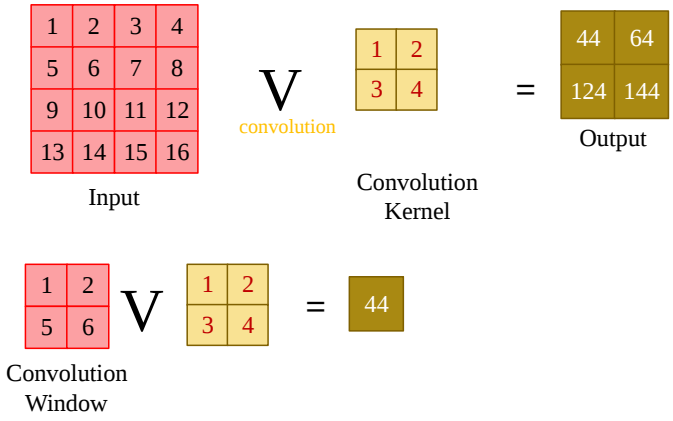

Figure 1. Convolution process with stride 2 in both height and width

2.3. Core requirement in privacy-preserving ML

Normally in privacy-preserving ML, in inference phase, the users may expect that the data is protected during its full lifecycle. It means that not only the original input, but also all the intermediate results computed by the model are required to be encrypted and protected. In addition, the model's parameters should be protected after training phase. However, the safety level should be balanced with the model efficiency. If the encryption parameter is too large, the inference time will be much more delayed compared with plaintext model inference. So, the users should choose acceptable parameters for encryption that can withstand quantum attack, instead of pursuing arbitrarily large parameters.

3. The technical pathways of the integration of FHE and ML

3.1. Optimization and adaptation when encryption

Before encrypting, the users can divide the high-dimensional input into several components in order to fit the polynomial degree of FHE. For example, the user can divide input images with size of 256×256 into several 64×64 components when $N = 4096$ in FHE to reduce the complexity of calculation. Additionally, the users can map their data into the range that FHE supports (the range of the module) by normalizing to prevent precision loss caused by overflow.

For activation layer, the users may choose simple functions, such as quadratic function x^2, to replace the computationally complex activation function, such as ReLU, to do operations on cipher text efficiently. The work from JOON-WOO LEE, HYUNGCHUL KANG et al allows the users to sacrifice some efficiency to use minimax approximate polynomials, approximating ReLU to achieve higher accuracy [13]. Furthermore, the computation order of the model is crucial for the efficiency. For example, in CNN, we normally have Batch Normalize (BN) layer after Convolution process. However, to be ciphertext-friendly, because BN layer just does linear transformation on data, we can infuse this transformation on the parameters of convolution to do convolution and BN layer simultaneously, which can reduce computation complexity.

In addition, the users can also combine other methods into the system. For instance, the development of GAZELLE library provides another perspective of integration by providing a mixed crypto scheme. This library divides CNN into two parts-linear layers and non-linear layers. The linear layers, such as Full Connection layer and Convolution Layer, are simple to calculate but have

DOI: 10.54254/2755-2721/2026.TJ30594

huge computations. The non-linear layers, such as activation functions and MaxPool, have less computations but the logic is very complex. Thereby, this library does normal FHE operations in linear layers and uses garbled circuits to do non-linear layers to improve efficiency. The key result that can support such connection between FHE and garbled circuit is an optimized encryption switching protocols presented by GAZELLE. Thus, this library finds a way to specialize the CNN processes by combining two types of operations. Figure 2 gives an illustration on how layers are classified in GAZELLE [14].

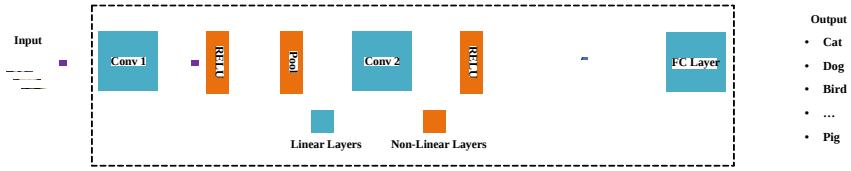

Figure 2. A structure of CNN, where layers are classified by linearity utilized in GAZELLE

3.2. Key techniques for homomorphic training and inference

Privacy-preserving ML needs specialized optimizations across both the training and inference phases. During training, executing backpropagation on encrypted data is a core challenge. Gradients must be homomorphically computed using the chain rule (e.g., $\partial L/\partial w \approx Enc(\partial L/\partial y) \otimes Enc(\partial y/\partial w)$), which requires careful noise management to prevent ciphertext corruption over successive operations. For inference, we can store the ciphertext of fixed parameters, such as model parameters, offline, and load it immediately during inference. Also, we can use Single Instruction Multiple Data (SIMD) property of FHE to manipulate multiple input data at once.

3.3. Dynamic management of noise and accuracy

Every cipher text contains an intrinsic noise term, and the modulus chain is used to determine the level of the cipher text, where the level will be reduced after multiplication. Thus, modulus switch can reduce the noise level and the size of the cipher text to control the noise and allow more multiplication to be done on the cipher text. However, due to finite modulus chain, bootstrapping could refresh the cipher text to reset the level to a higher level. For some approximating FHE schemes, such as CKKS, the cipher text is not precise after decryption since an error term is also included. The scale parameter controls the error by determining the precision of the encoding and the multiplication between cipher texts can produce inflated scales. So, the rescale process can reduce the inflated scale.

Therefore, in integrating FHE and ML, we can adopt dynamic managements on these parameters. First, it is computationally expensive to do bootstrapping frequently. So, automatic bootstrapping compiler can insert bootstrapping flexibly in the system once it is needed. For example, DaCapo from Seonyoung Cheon et al. can automatically determine when to place bootstrapping by analyzing live-out cipher texts to minimize the latency [15]. Secondly, Error-Latency-Aware Scale Management (ELASM) presented by Yongwoo Lee et al. can actively manage scale and minimize the error-latency cost function [16].

DOI: 10.54254/2755-2721/2026.TJ30594

4. FHE and adaptation solutions for different ML models

4.1. Traditional ML models

Linear models can directly be manipulated from homomorphic addition and multiplication benefitting from its linear form. The noises would increase slowly.

For Logistic Regression, the sigmoid function can be approximated as a polynomial, which enables more efficient implementation. Noticeably, due to unclosed domain of logistic function, using least squares approximation by minimizing the mean square error (MSE) in a range rather than using Taylor polynomial approximation provides a better result, as Taylor polynomial only results a local approximation near a certain point [17].

Tree models, such as decision tree and random forest, may rely on the comparison between the result and threshold which can be solved by encoding the tree's decision logic into a series of arithmetic circuits or polynomials. We can use TFHE to accelerate Boolean computation inference as TFHE supports fast bootstrapping to enable deep, sequential comparisons without excessive noise growth [18].

4.2. Deep learning models

CNN has several crucial optimizations in order to integrate FHE to its model. We can expand the convolution kernel into a recursive vector, transforming the two-dimensional convolution process to a one-dimensional polynomial multiplication. Furthermore, as the convolution layers and FC layers can be transformed to matrix multiplication on vector, we can use diagonal method to optimize the multiplication by storing all the diagonals of the matrix and encoding these diagonals. Then, this approach uses diagonal matrix vector multiplication to rotate and multiply, as displayed in Figure 2 [19]. The Baby-step-Giant-step (BSGS) algorithm can optimize this matrix multiplication process. Furthermore, the SoftMax function that is used to evaluate the probability of classification result was first approximated and implemented on encrypted status. This would allow the encrypted model to return the classification result with the highest probability calculated from SoftMax, instead of returning the primitive logits of each classification result and let the users to calculate SoftMax by themselves, which makes attacking and extracting model parameters more difficult [13].

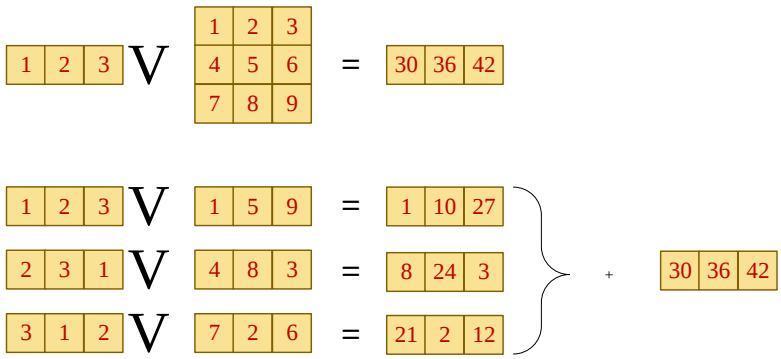

Figure 3. Vector-matrix multiplication by rotating diagonals, where we take three diagonals from the matrix to dot multiply each rotation of the vector and then add the result

DOI: 10.54254/2755-2721/2026.TJ30594

Applying recurrent neural networks (RNNs/LSTMs) to FHE faces challenges due to their sequential nature. Long chains of encrypted operations lead to significant noise accumulation, making them difficult to compute reliably. Mentioned by Sinem et al. in Rhode system, a common solution is to use collective bootstrapping to periodically manage noise levels, and optimized operations to slow its growth, allowing the full sequential chain to be computed [20]. Robert Podschwadt et al. redesign the model into Parallel RNN Blocks to break the long sequential chain into shorter, independent ones, drastically reducing multiplicative depth and noise growth [21]. Also, a lot of non-linear activation functions are manipulated in RNN, which can be approximated by polynomials mentioned before.

Transformer models, especially their self-attention mechanism, present even greater difficulties. The attention module requires matrix multiplications with quadratic complexity $O(n^2)$, which becomes prohibitively expensive when performed homomorphically. Recent research explores using sparse attention patterns and low-rank approximations to reduce the number of encrypted operations, for example, by computing only top-k attention values [22].

5. Application scenarios and practical cases of integration technologies

FHE enables practical privacy-preserving ML across various industries by allowing computations on encrypted data.

In healthcare, the patient's wearable device continuously uploads encrypted health data, such as heart rate and blood sugar, to the cloud. The cloud service providers can calculate statistics such as mean and logistic regression to predict health risks and issue encrypted alerts. During this process, the model may be public known, but the input and output data are all private. FHE also facilitates multi-center collaborative training without sharing raw patient data. For example, hospitals can encrypt and contribute CT imaging features to train a lung cancer detection model while preserving patient confidentiality. Similarly, hospitals can encrypt and upload pathological images to a cloud ML model for encrypted inference [23].

In terms of finance, an investment company may have an independently developed algorithm model and the real-time data of market. This company can upload both of them to the cloud in encrypted states to get investment advices from the encrypted result while the cloud will do calculations by keeping both model and data invisible [23].

In edge computing and IoT, FHE supports secure local analysis. Encrypted sensor data in industrial settings can be analyzed for predictive maintenance without leaking sensitive operational information. In addition, neural networks of automatic steering can use the real-time input from camara of current circumstances around the vehicle to do analyzation and prediction in the cloud without exposing the vehicle drivers' privacy [24].

6. Challenges and directions of future development

6.1. Core challenges

The core challenges of the integration of FHE and ML happens in various ways. On the one hand, due to different hardware setup for current solutions, the standard of evaluating a model is often not intuitive. So, a more meaningful and unified evaluation criteria should be proposed. Also, because there is no unified standard for ciphertext, encryption keys and encryption parameters of different FHE libraries, developers might be limited on a certain library [12]. On the other hand, the technical issues on certain complex models need to be solved. For example, plain transformers originally are

DOI: 10.54254/2755-2721/2026.TJ30594

large-scale models in terms of number of parameters and computations. After encryption, the models become much more complex and harder to store. Training a model under encryption would be time-consuming. Therefore, more efficient and compact system should be presented for transformers in order to reduce the training time and cost.

6.2. Directions of future development

The future of the integration of ML and FHE would have several directions to the development. First, more unified evaluation standards and more generalized standards for ciphertext on FHE libraries as mentioned above enable users to compare different systems and transfer from one system to another. Secondly, more specialized architecture and accelerators can be developed to improve the efficiency. Third, designing new FHE schemes which may have better properties, such as faster computation and lower noises, can increase more practicability for the integration. Lastly, with compilers to automatically choose encryption parameters and optimize the model procedures, users can find it easier to get started with the system even if they do not have much cryptography background.

7. Conclusion

In conclusion, FHE presents a promising path for privacy-preserving ML by enabling computation on encrypted data. While the potential is significant—from securing traditional models like linear regression to complex deep learning architectures—there are still core challenges needed to be investigated.

The main obstacles remain the computational cost of large-scale models, the ability to adapt to dynamic data environments, and the lack of standardized benchmarks for fair comparison.

Looking ahead, the key lies in creating more efficient hardware-aware algorithms, developing adaptive training techniques, and building unified security frameworks. Ultimately, bridging the gap between theoretical advancements and practical applications will be crucial for making “data usable but invisible” a reality in fields such as healthcare, fintech, and the IoT.

References

[1] GDPR, “General Data Protection Regulation (GDPR), ” GDPR, 2018. https: //gdpr-info.eu/

[2] H. Jiang, Y. Gao, S. S. M, L. GarzaPerez, and M. Robin, “Differential Privacy in Privacy-Preserving Big Data and Learning: Challenge and Opportunity, ” arXiv (Cornell University), Jan. 2021, doi: https: //doi.org/10.48550/arxiv.2112.01704.

[3] Betul Yurdem, Murat Kuzlu, Mehmet Kemal Gullu, Ferhat Ozgur Catak, and M. Tabassum, “Federated Learning: Overview, Strategies, Applications, Tools and Future Directions, ” Heliyon, vol. 10, no. 19, pp. e38137–e38137, Sep. 2024, doi: https: //doi.org/10.1016/j.heliyon.2024.e38137.

[4] R. L. Rivest, L. Adleman, and M. L. Dertouzos, “On data banks and privacy homomorphisms, ” in Foundations of Secure Computation, R. A. DeMillo, Ed. New York, NY, USA: Academic Press, 1978, pp. 169–179.

[5] C. Gentry, “A fully homomorphic encryption scheme, ” Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 2009.

[6] O. Regev, “On lattices, learning with errors, random linear codes, and cryptography, ” Journal of the ACM, vol. 56, no. 6, pp. 1–40, Sep. 2009, doi: https: //doi.org/10.1145/1568318.1568324.

[7] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(Leveled) fully homomorphic encryption without bootstrapping, ” Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on - ITCS ’12, 2012, doi: https: //doi.org/10.1145/2090236.2090262.

[8] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic Encryption for Arithmetic of Approximate Numbers, ” Advances in Cryptology – ASIACRYPT 2017, pp. 409–437, 2017, doi: https: //doi.org/10.1007/978-3-319-70694-8_15.

DOI: 10.54254/2755-2721/2026.TJ30594

[9] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy, " in Proc. 33rd Int. Conf. Mach. Learn. (ICML), New York, NY, USA, Jun. 20–22, 2016, vol. 48, pp. 201–210.
[10] Roshan Dathathri et al., "CHET: an optimizing compiler for fully-homomorphic neural-network inferencing, " Jun. 2019, doi: https: //doi.org/10.1145/3314221.3314628.
[11] F. Boemer, S. Kim, G. Seifu, F. D. M. de Souza, and V. Gopal, "Intel HEXL: Accelerating homomorphic encryption with Intel AVX512-IFMA52, " Cryptology ePrint Archive, Paper 2021/420, 2021. [Online]. Available: https: //eprint.iacr.org/2021/420
[12] R. Podschwadt, D. Takabi, and P. Hu, "SoK: Privacy-preserving Deep Learning with Homomorphic Encryption, " arXiv (Cornell University), Feb. 2022, doi: https: //doi.org/10.48550/arxiv.2112.12855.
[13] J.-W. Lee et al., "Privacy-Preserving Machine Learning With Fully Homomorphic Encryption for Deep Neural Network, " IEEE Access, vol. 10, pp. 30039–30054, 2022, doi: https: //doi.org/10.1109/ACCESS.2022.3159694.
[14] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "Gazelle: A Low Latency Framework for Secure Neural Network Inference, " arXiv.org, Jan. 16, 2018. https: //arxiv.org/abs/1801.05507 (accessed Jul. 18, 2023).
[15] S. Cheon, Y. Lee, D. Kim, J. M. Lee, S. Jung, T. Kim, D. Lee, and H. Kim, "DaCapo: Automatic Bootstrapping Management for Efficient Fully Homomorphic Encryption, " in Proc. 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, Aug. 2024, pp. 6993–7010.
[16] Y. Lee, S. Cheon, D. Kim, D. Lee, and H. Kim, "ELASM: Error-Latency-Aware Scale Management for Fully Homomorphic Encryption, " in Proc. 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, Aug. 2023, pp. 4697–4714.
[17] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure Logistic Regression Based on Homomorphic Encryption: Design and Evaluation, " JMIR Medical Informatics, vol. 6, no. 2, p. e19, Apr. 2018, doi: https: //doi.org/10.2196/medinform.8805.
[18] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast Fully Homomorphic Encryption Over the Torus, " Journal of Cryptology, Apr. 2019, doi: https: //doi.org/10.1007/s00145-019-09319-x.
[19] D. Shanks, "Class number, a theory of factorization and genera, " in Proc. Symp. Pure Math., vol. 20, 1969, pp. 415–440.
[20] S. Sav, A. Diaa, A. Pyrgelis, J.-P. Bossuat, and J.-P. Hubaux, "Privacy-Preserving Federated Recurrent Neural Networks, " arXiv (Cornell University), Jul. 2022, doi: https: //doi.org/10.48550/arxiv.2207.13947.
[21] R. Podschwadt and D. Takabi, "Non-interactive Privacy Preserving Recurrent Neural Network Prediction with Homomorphic Encryption, " pp. 65–70, Sep. 2021, doi: https: //doi.org/10.1109/cloud53861.2021.00019.
[22] S. Lu, R. Li, W. Liu, C. Guan, and X. Yang, "Top-k sparsification with secure aggregation for privacy-preserving federated learning, " Computers & Security, vol. 124, p. 102993, Jan. 2023, doi: https: //doi.org/10.1016/j.cose.2022.102993.
[23] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?, " Proceedings of the 3rd ACM workshop on Cloud computing security workshop - CCSW '11, 2011, doi: https: //doi.org/10.1145/2046660.2046682.
[24] X. Li, H. Gao, J. Zhang, S. Yang, X. Jin, and K.-K. R. Choo, "GPU Accelerated Full Homomorphic Encryption Cryptosystem, Library, and Applications for IoT Systems, " IEEE Internet of Things Journal, vol. 11, no. 4, pp. 6893–6903, Feb. 2024, doi: https: //doi.org/10.1109/jiot.2023.3313443.