Application of Homomorphic Encryption in Machine Learning

Yulliwas Ameur, Samia Bouzefrane, and Vincent Audigier

1 Introduction to Homomorphic Encryption

This new encryption paradigm allows any entity (for example, the cloud provider) to operate on private data in encrypted form without ever decrypting it. For example, one widespread use case is outsourcing healthcare data to cloud computing services for medical studies. The goal of HE is to perform operations on the plain text while manipulating only ciphertexts (Fig. 1). Usually, we must decrypt them and then apply the desired processing to manipulate encrypted data (Fig. 2).

For some cryptosystems with algebraic structures, some operations are possible. For example, two RSA ciphertexts can be multiplied to obtain the multiplication of the two corresponding plain texts. We call this property the multiplicative homomorphic property of the "textbook RSA" cryptosystem. Another operation can also be performed on ciphertexts. For example, in the Paillier cryptosystem [1], we can add two ciphertexts to obtain the addition of the two corresponding plain texts. We call this property the additive property of the "Paillier" cryptosystem. For example, this can be useful when we are interested in e-voting applications to add encrypted votes without knowing the initial vote. Rivest, Adelman, and Dertouzos first introduced the notion of homomorphic encryption in [2]. Building a cryptosystem with both multiplicative and additive properties was a significant problem in cryptography, until the work of Gentry [3]. Gentry proposed a first fully homomorphic encryption based on ideal lattices. The HE is categorized depending on the number of mathematical operations performed on the encrypted message as following:

Y. Ameur (✉) · S. Bouzefrane · V. Audigier
CEDRIC Lab, Cnam, Paris, France
e-mail: yulliwas.ameur@lecnam.net; samia.bouzefrane@lecnam.net; vincent.audigier@lecnam.net

K. Daimi et al. (eds.), *Emerging Trends in Cybersecurity Applications*,
https://doi.org/10.1007/978-3-031-09640-2_18

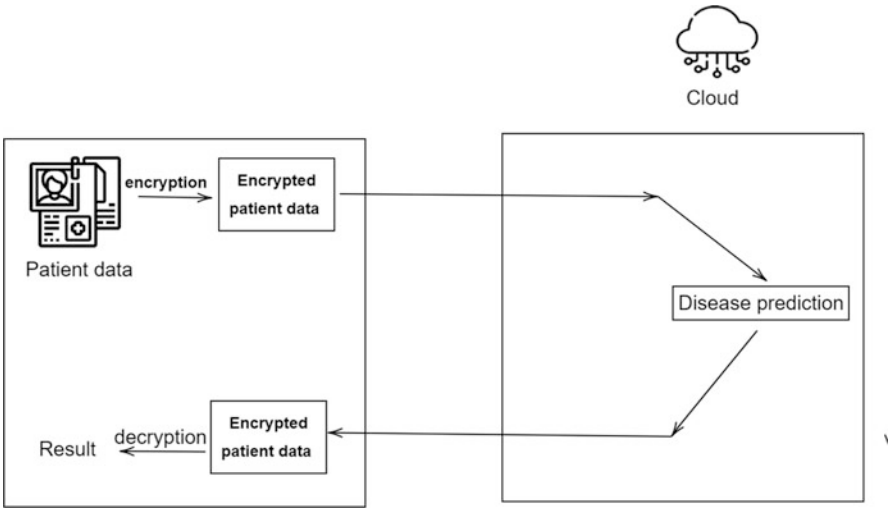

Fig. 1 Diagram showing how to manipulate encrypted data on a cloud. On the left, the user encrypts the data before sending it to the cloud, on the right, the cloud service has to decrypt the data in order to process it

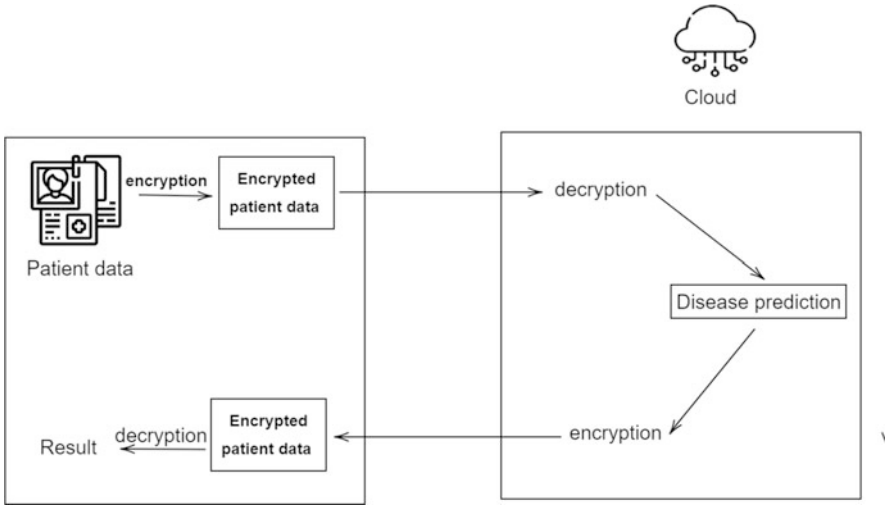

Fig. 2 Diagram showing how to manipulate encrypted data on a cloud by using homomorphic encryption. On the left, the user encrypts the data before sending it to the cloud, on the right, the cloud service can process on data by manipulating only ciphertexts

- **Partially homomorphic encryption (PHE)**: is a cryptosystem that allows a single operation (addition or multiplication) over encrypted data. When a PHE scheme allows for additions over ciphertexts, it is considered an additively

homomorphic scheme [1]. When a PHE scheme allows for multiplications, it is considered multiplicatively homomorphic.

- **Somewhat homomorphic encryption (SHE)**: is a cryptosystem that allows us to perform a limited number of both additions and multiplications. SHE cryptosystems typically allow for unlimited additions but only a restricted number of multiplications.
- **Fully homomorphic encryption (FHE)**: FHE schemes are the most powerful HE schemes. They can perform both addition and multiplication, as well as circuits of any depth. The reason HE methods have a restricted circuit depth is that the encryption operation introduces noise to the data, and the decryption step removes that noise. Performing operations on ciphertexts generates more noise, which prevents correct decryption [4].

The **bootstrapping** approach is used by FHE systems to get around this as stated in [3]. Bootstrapping decreases the collected noise, allowing further computation. This procedure can be done as many times as necessary to analyze any particular circuit. However, bootstrapping is computationally costly, so many solutions do not employ it in reality. Therefore, we recommend the reader to refer to [5] for more detailed information on the different homomorphic encryption schemes. We have chosen not to describe the entire functioning of cryptosystems due to the lack of space. Also, selecting secure and efficient instantiations of the underlying cryptographic problem is hard for most of encryption and homomorphic schemes. Therefore, we have chosen to list the most studied schemes by the community of researchers and developers interested in advancing homomorphic encryption.

As usual, new cryptographic proposals need a few years before widespread adoption in the industry, as was the case of elliptic curve cryptography, post-quantum encryption and many other standardization projects. We are waiting for the standardization results and recommendations of the workshops, which include representatives from industry, government organizations and academia. This brief review aims to guide readers fast enough, even if they are not cryptography specialists, to the appropriate HE scheme by directing them to the library(ies) where HE is implementable.

1.1 HE Schemes

Research in the field of FHE may be classified into four major groups. The first family represents the difficulty based on the lattice reduction problem, which mainly comes from Gentry's seminal work [3]. The second category consists of integer-based methods [6], the hardness of which is based on the Approximate of Greatest Common Divisor (A-GCD) problem [7]. Schemes based on learning with error (LWE) [8] and ring learning with error (RLWE) [9], both reducible to lattice problems, constitute the third family. Finally, the Nth-Degree truncated polynomial ring unit (NTRU) family [10].

Table 1 Comparison of HE schemes

Operation	Schemes				
	BFV	BGV	CKKS	FHEW	TFHE
Native Add/Sub	✓	✓	✓	✗	✗
Native Mult	✓	✓	✓	✗	✗
Boolean Logic	✗	✗	✗	✓	✓
SIMD	✓	✓	✓	✓	✓

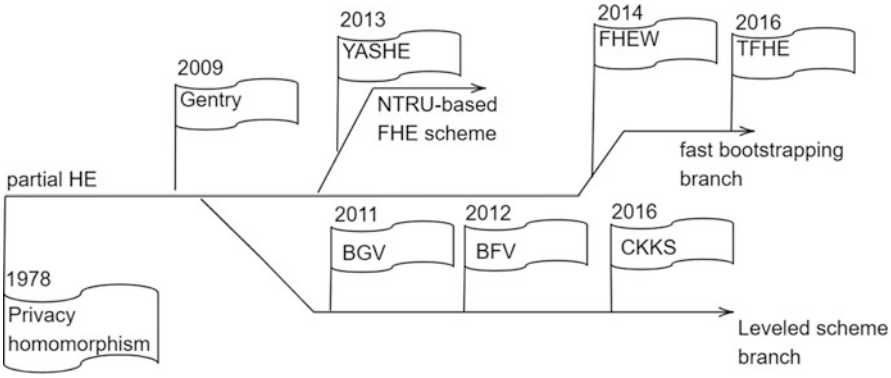

Fig. 3 Homomorphic encryption timeline

All HE schemes have common steps: key generation, encryption, decryption, and homomorphic operations on the ciphertexts. Table 1 summarizes the most implemented and studied schemes by the cryptographic community, and Fig. 3 gives an overview of the homomorphic encryption timeline.

The choice of encryption scheme has a multitude of implications:

- It specifies which operations are possible and, as a result, which types of activation and architectures may be employed.
- It can determine the plaintext space. Messages should be encoded before they may be sent in plaintext. The majority of schemes, including BGV, BFV, only support integers. CKKS can handle real numbers, but TFHE can only handle individual bits.

1.2 HE Libraries

There exist several open-source libraries for the implementation of the HE scheme. They provide key generation, encryption, decryption, and homomorphic operations for each scheme; library APIs frequently include additional features for maintaining and manipulating ciphertexts. Even though there is a lack of technical interoperability, but also a lack of conceptual interoperability; for example, even libraries that use the same scheme can provide surprisingly different results. The ongoing

Table 2 Overview of existing FHE libraries: CPU-targeting (top) and GPU-targeting (bottom)

Name	Input language	Supported schemes				
		BFV	BGV	CKKS	FHEW	TFHE
HE-CPU-TARGETING						
Concrete	Rust	✗	✗	✗	✗	✓
FHEW	C++	✗	✗	✗	✓	✗
FV-NFlib	C++	✓	✗	✗	✗	✗
HEAAN	C++	✗	✗	✓	✗	✗
HElib	C++	✓	✓	✓	✗	✗
lattigo	Go	✓	✗	✓	✗	✗
PALISADE	C++	✓	✓	✓	✓	✓
SEAL	C++, .NET	✓	✓	✓	✗	✗
TFHE	C++	✗	✗	✗	✗	✓
HE-GPU-TARGETING						
cuFHE	C++, Python	✗	✗	✗	✗	✓
nuFHE	C++, Python	✗	✗	✗	✗	✓

standardization efforts attempt to develop a unified view of the most popular schemes (Table 2).

1.3 FHE Restrictions

Current HE methods have a significant restriction. They cannot support division operations and comparisons easily, such as the equality/inequality test. Number comparison and sign determination are critical processes for MLaaS

2 Privacy-Preserving in Machine Learning (PPML): HE Solutions

Using a third-party infrastructure reduces the problems of resources and complexity but introduces privacy issues of sensitive information. To construct a privacy-preserving framework for machine learning techniques, it must first identify the most important privacy requirements:

- Input privacy: Only the real data owner should have access to the input.
- Output privacy: The output/result of the ML methods' assessment is not permitted to be known by the cloud server.
- Model privacy: A private machine learning model that is also an asset should not be shared with anyone except its owner.

Another approach is to look if the privacy-preserving framework targets the learning phase or the inference phase of the machine learning algorithm. Depending on the framework we want to design, we have to use privacy-preserving technologies. The leading privacy-preserving machine learning techniques are

- **Multi-party computation**: These methods involve one or more trusted parties that can be used to outsource specific computations by the algorithm owner.
- **Differential privacy**: These methods rely on data randomization and perturbation. Because it affects the information, this method has the drawback of negatively influencing the model's performance.
- **Federated learning**: Federated learning is a machine learning setting where many clients collaboratively train a model under the administration of a central server while keeping the training data local.
- **Garbled circuit**: Garbled circuit, also known as Yao's garbled circuit, is an underlying technology of secure two-party computation initially proposed by Andrew Yao. GC provides an interactive protocol for two parties (a garbler and an evaluator) obliviously evaluate an arbitrary function represented as a Boolean circuit.
- **Homomorphic encryption**: An encryption that allows performing operations over encrypted data. See Sect. 1 for more detail.
- **Hybrid PPML techniques**: In addition to the above-mentioned single-protocol PPML, some commonly used frameworks typically use hybrid protocols, which combine two or more protocols by making use of the advantages and avoiding the problems of each. For example, the basic idea behind the mixed protocol that combines HE and GC is to calculate operations that have an efficient representation as Arithmetic circuits (e.g., additions and multiplications) using HE and operations that have an efficient representation as Boolean circuits (e.g., comparisons) using GC. However, converting between share systems is not simple, and the charges are very high. Furthermore, various frameworks integrate MPC with differential privacy.

We are interested therein machine learning research Using homomorphic encryption "HEML." According to this bibliometrics [11], the number of papers on HEML has constantly been rising since 2009. Each year from 2005 to 2015, fewer than 100 HEML papers were published. However, the number of publications per year increased significantly after 2015, reaching between 200 and 500 in recent years. This section summarizes recent works and gives many practical applications of homomorphic encryption for privacy-preserving purposes for each machine learning algorithm. We end the section with a summary of the work to ease the reading of this synthesis (Fig. 4).

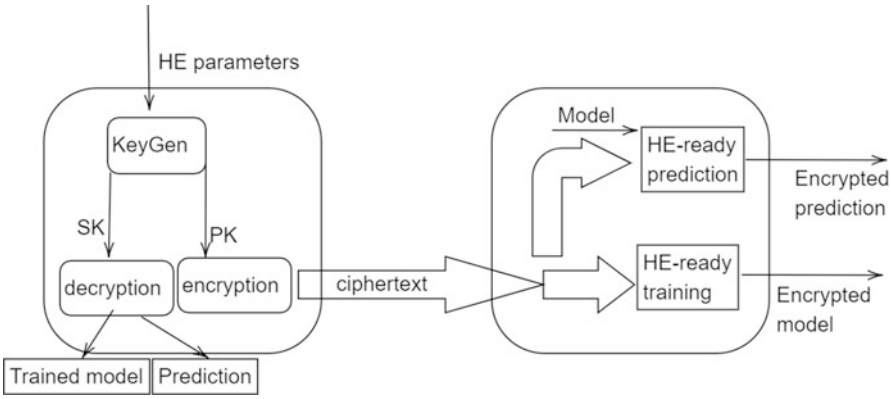

Fig. 4 HEML scenario

2.1 Logistic Regression

Logistic regression is a powerful machine learning approach that uses a logistic function to model two or more variables. Logistic models are commonly used in the medical community to predict binary outcomes, such as whether a patient requires treatment or whether a disease appears [12]. It has been utilized in applications such as evaluating diabetes patients' medications [13], and social sciences [14].

iDASH is an annual competition that attempts to deploy novel cryptographic methods in a biological environment. Since 2014, genomics and biomedical privacy have been incorporated into iDASH. Both the third track of the 2017 iDASH competition [15] and the second track of the 2018 iDASH competition driven the development of homomorphic encryption-based solutions for building a logistic regression model over encrypted data. The performance of LR training based on homomorphic encryption (HE) has improved significantly as a result of these two competitions. Homomorphic encryption has been used in much research on training logistic regression models.

Wu et al. [16] trained a privacy-preserving logistic regression model using HE; however, the time complexity of linear HE increases exponentially with the number of parameters. Aono et al. [17] used an additive HE scheme and delegated particular challenging HE computations to a trusted client, the authors in this work introduced an approximation to convert the likelihood function into a low-degree polynomial.

The issue of doing LR training in an encrypted environment was discussed by Kim et al. [18]. They used complete batch gradient descent in the training phase, using the least-squares approach to approximate the logistic regression. They also employed the CKKS scheme, which allows for a homomorphic approximation of the sigmoid function.

There is no closed-form solution to logistic regressions, so we must use non-linear optimization methods to find the maximum likelihood estimators of the regression parameters. During training, gradient descent and Newton-Raphson are the most commonly used methods. The Newton-Raphson method requires matrix inversion, and most HE schemes do not natively support division and matrix inversion. On the other hand, Gradient descent does not require the division operation and so is a better candidate for homomorphic logistic regression.

Although the gradient descent approach appears to be better suited for homomorphic evaluation than other training methods, some technical issues remain for implementation. The sigmoid function is the most challenging to evaluate since existing homomorphic encryption techniques only allow the evaluation of polynomial functions, so Taylor polynomials have been widely employed for sigmoid function approximation [19, 20]

For implementation and performance of private logistic regression (He-based solutions), see Table 3.

2.2 Naive Bayes and Decision Trees

Naive Bayesian classification is a simple probabilistic Bayesian classification based on Bayes' theorem. It uses a naive Bayesian classifier, or naive bayes classifier, belonging to the family of linear classifiers. Bost et al. [21] propose a privacy-preserving naive bayes classification algorithm. A client learns the classification of her data point X in their model without knowing the classification model or disclosing any information about her input. The model's estimated parameters are encrypted and transferred to a cloud server. The authors use two partially homomorphic encryption schemes, quadratic reciprocity [22] and Paillier [1], in the same work [21] implements a privacy-preserving strategy for three algorithms, and one of those is decision trees. This work has shown that polynomials may be utilized to express decision trees. The decision tree node values must be compared to the evaluation data, and the outputs must be used to construct the polynomial, yielding the evaluation results. For implementation and performance of private naive bayes and decision tree (He-based solutions), see Table 4.

2.3 K-Nearest Neighbors

The k-Nearest Neighbors (k-NN) a simple method that can handle continuous, categorical, and mixed data. Furthermore, as a non-parametric method, k-NN can be relevant for many data structures as long as the number of observations is sufficiently large. In addition, for a predefined number of neighbors k, the model does not require any training step since the prediction for a new observation is obtained by:

Table 3 Summary of main works on private prediction for logistic regression: "-" means that the information has not been disclosed

Ref.	HE scheme/Type	Platform	Evaluation time	Accuracy	Datasets
Logistic regression					
[16]	[1] /LHE	-	-	82.89%	Dataset SPECT—267 instances—23 features
[17]	[1] /LHE	2.60 GHz × 2 CPU, 128 GB RAM	-	73.7%	SPECTF heart dataset—267 instances—44 features)
[17]	[1] /LHE	2.60 GHz × 2 CPU, 128 GB RAM	-	80.7%	Pima diabetes dataset—768 instances—8 features
[18]	CKKS/LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM	131min	86.03%	Edinburgh—1253 instances—10 features
[18]	CKKS/LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM	101min	69.30%	Lbw—189 instances—10 features
[18]	CKKS/LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM	265min	79.23%	Nhanes3—15649 instances—16 features
[18]	CKKS/LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM	119min	68.85%	Pcs—379 instances—10 features
[18]	CKKS/LHE	intel Xeon 2.3 GHz processor with 16 cores and 64GB of RAM	109min	74.43%	Uis—575 instances,—9 features
[19]	YASHE/LHE	Intel Core i7- 3520M at 2893.484 MHz	-	-	Heart Disease Framingham—4000 instances—15 features
[20]	Linearly homomorphic encryption	Amazon EC2 c4.8xlarge machines running Linux, with 60GB of RAM each.	149.7sec	98.62%	MNIST dataset—60 000 instances—784 features

Table 4 Summary of main works on private prediction for naive bayes and decision tree "-" means that the information has not been disclosed

Ref.	HE scheme/Type	Platform	Evaluation time	Accuracy	Datasets
Naive bayes					
[21]	[1] + [22]/LHE	Two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM	479 ms	-	Breast Cancer—2 classes—9 features
[21]	[1] + [22]/LHE	Two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM	1415 ms	-	Nursery—9 classes—5 features
[21]	[1] + [22]/LHE	Two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM	3810 ms	-	Audiology—14 classes—70 features
Decision tree					
[21]	[1] + [22]/LHE	Two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM	239 ms	-	Nursery
[21]	[1] + [22]/LHE	Two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM	899 ms	-	ECG

- Identifying the k nearest neighbors (according to a given distance)
- Computing the majority class among them (for a classification problem) or by averaging values (for a regression problem).

Homomorphic encryption has already been investigated by various authors for k-NN [23–26]. The authors of [23] suggested a homomorphic additive encryption scheme [1]. They investigated the privacy preservation in an outsourced k-NN system with various data owners. The untrusted entity securely computes the compu-

Table 5 Summary of main works on private K-nearest neighbors: "-" means that the information has not been disclosed

REF	HE scheme/Type	Platform	Evaluation time	Accuracy	Datasets
K-nearest neighbors					
[23]	[1]/LHE	-	-	97.85%	Cancer 1 (9 features)
[23]	[1]/LHE	-	-	96.49%	Cancer 2—569 instances,—30 features)
[23]	[1]/LHE	-	-	81.82%	Diabetes
[23]	[1]/LHE	-	-	97%	MNIST dataset—60 000 instances—784 features
[26]	[27]/FHE	Intel Core i7-6600U CPU	11.6 min	94.8	MNIST dataset—60,000 instances—784 features

tations of distances by using HE. However, the comparison and classification phases require interactions. Given that the computational and communication difficulties scale linearly, they admit that the method may not be practical for massive data volumes. The cost of communications between the entities is also a limitation in the deployment of this work [24].

Recently, [26] proposed a secure k-NN algorithm in quadratic complexity concerning the size of the database completely non-interactively by using a fully homomorphic encryption [27]. However, they assume that the majority vote is done on a clear-text domain, which is a significant security flaw that we will address here. Doing a majority vote on a clear-text domain imposes interaction between entities, which causes information leakage. For implementation and performance of private k nearest neighbors (He-based solutions), see Table 5

2.4 Neural Networks and Deep Learning

Deep learning is one of the most sophisticated techniques in machine learning, and it has received a lot of attention in recent years. It is presently employed in a variety of areas and applications, including pattern recognition, medical prediction, and speech recognition. Deep learning experiences are enhanced significantly by utilizing strong infrastructures such as clouds and implementing collaborative learning for model training. However, this compromises privacy, particularly when sensitive data is processed during the training and prediction stages, as well as when the training

model is disseminated. In this section, we discuss known privacy-preserving deep learning algorithms based on homomorphic encryption, we present recent challenges concerning the intersection of HE cryptosystems and neural networks models, as well as methods to overcome limitations.

HE cannot be used naively in neural networks algorithms. There are a lot of challenges and restrictions that must be overcome. The constraints differ according to the scheme, however, several common issues emerge in most systems. The learning and inference phases of the deep learning algorithm can be distinguished.

2.4.1 Privacy-Preserving Deep Learning: Private Training

Several techniques have been proposed; they consider **collaborative training**, in which the training is performed collaboratively between different participants, or **individual training**, in which the training is performed by a single participant, such as a client who wants to use a cloud to train its model.

Aono et al. [28] proposed a solution in **collaborative learning** mode, where participants send the calculated encrypted local gradients to the cloud after each iteration of local training, starting with the initial weights obtained from the cloud. To ensure homomorphic ciphertext integrity, each participant uses a unique TLS/SSL secure channel. The cloud then updates the encrypted global weights vector, which the participants download. The approach theoretically achieves the same accuracy as standard asynchronous SGD, whereas evaluations show that MLP and CNN reach 97% and 99% accuracy, respectively. In terms of efficiency, an overhead in communication and calculation was seen; however, the authors considered it negligible. However, the accuracy/privacy trade-off might be adjusted to efficiency/Privacy, allowing the precision to be preserved while maintaining Privacy.

The privacy-preserving back-propagation technique described in [29] is based on BGV fully homomorphic encryption and Maclaurin polynomial approximation of the sigmoid activation function. The client encrypts input data and configured parameters before uploading them to the cloud, which executes one loop. The client downloads and decrypts the findings before updating its local model. It then encrypts and sends the updated parameters back to the cloud, which repeats the process. This method is repeated until the maximum error threshold or number of iterations is achieved. Although BGV encryption provides for the protection of private data throughout the learning process, it does need the approximation of the activation function. This might lead to a drop in accuracy. In terms of efficiency, while the solution achieved a two times greater efficiency, i.e., 45% of the training time of the standard model, it experienced compute and communication costs due to the encryption-related overhead.

Zhang et al. [30] employs the Taylor theorem to estimate the sigmoid activation function polynomially. The evaluation findings revealed a reduction in accuracy for both classification and prediction tasks. However, the authors proposed adding additional Taylor series terms to decrease classification loss, raising the BGV encryption level, resulting in poor performance. The method could achieve 2.5 times

greater classification efficiency and two times higher overall efficiency in learning time.

Zhang et al. [31] described a more recent solution based on encryption. A client who wants to participate to the model's training transmits its data encrypted using the Paillier scheme [1] to the server, which performs all possible neural network calculations except non-linear activation functions. To continue execution, the encrypted weighted sums before each activation function are provided to the client, who will be in charge of performing the calculation. The result is then encrypted again and sent back to the server.

2.4.2 Privacy-Preserving Deep Learning: Private Inference

Fully homomorphic encryption is deployed in a line of research that performs private classification of encrypted data using a neural network that has been trained using plain data.

Gilad-Bachrach et al. [32] is the first solution for privacy-preserving deep learning for inference, developed by Microsoft Research. The approach is based on the YASHE a (leveled homomorphic encryption) LHE scheme was proposed. The user encrypts their data and sends it to the cloud, which runs the model and returns an encrypted prediction. It has since been demonstrated that the YASHE scheme is vulnerable to subfield lattice attack [33]. To make the network compatible with homomorphic encryption, max-pooling is replaced by a scaled-mean pool function, and activation functions are approximated by the square function, which is the lowest-degree non-linear polynomial function. According to the authors, these adjustments should preferably be considered during training on unencrypted data. For example, the solution could achieve 99% accuracy and 59,000 predictions per hour on a single PC for the MNIST dataset.

To overcome the heavy computation cost of HE, a dual cloud model was proposed, in which two clouds, A and B, collaborate to generate classification results in a secure environment [34]. Cloud A operates the neural network on private data encrypted by the client with Paillier cryptosystem [1], but delegates activation function computations to the cloud B since they share a key. The technique is repeated until the final layer is reached. Client A then protects the final output with a random salt from cloud B, which uses the softmax function and sends the final encrypted result to the client. A theoretical scenarios-based security and accuracy study demonstrated how the approach successfully defends against potential threats.

Chabanne et al. [35] suggested a method for classification problems over the CNN model based on BGV an FHE scheme. The combination of polynomial approximation and batch normalization is the major technological breakthrough. During the training phase, a batch normalization layer is introduced before each ReLU layer to avoid excessive accuracy deterioration, and max-pooling is replaced by average-pooling, which is more FHE-friendly and has a small overhead.

Prior to each ReLU, a batch normalization layer is introduced with a low-degree (2) polynomial approximation. When the model is complete, the user encrypts

its private data and sends it to the model, carrying out the specified analysis. The evaluation findings revealed that the solution has a short running time, with comparable performance, as if there was no privacy.

Attempt to use HE for deep learning problems. They provide methods by using low-degree polynomials to approximate the most generally used neural network activation functions (ReLU, Sigmoid, and Tanh) [36]. This is a critical step in the development of effective homomorphic encryption methods. They then train convolutional neural networks using those approximation polynomial functions before implementing the models over encrypted data and evaluating its performance.

Zhu and Lv [37] suggest a recent homomorphic encryption-based approach. The user encrypts their personal information and transmits it to the server for prediction. The Paillier scheme accelerates linear, convolutional, and pooling transformations. The authors chose ReLU as the activation function and suggested, rather than utilizing polynomial approximation, an interactive protocol between the client and the server for its computation. The user gets the ReLU input data, decrypts it, and communicates the positivity or negativity of this input to the server, enabling the server to calculate the output. The evaluation findings revealed that the solution could reach near model accuracy in plain text and was similar to Cryptonet[32]. In terms of efficiency, the approach saves a significant amount of time.

Recently, authors in [38] have resulted in considerable improvements by using the scheme TFHE [27]. FHE methods permit unrestricted encrypted operations and give accurate polynomial approximations to non-polynomial activation functions using a programmable bootstrapping technique, an extension of the bootstrapping technique that allows resetting the noise in ciphertext to a fixed level while—at the same time—evaluating a function for free.

2.5 Clustering

Clustering is an unsupervised machine learning problem that automatically identifies natural grouping (clusters) in data. A clustering algorithm can be collaborative or individual. In both cases, a model can be based on a server, and the calculations are exclusively performed on the server or assisted by a server. In this case, some calculations are delegated to the server. Three models can be found in the literature:

1. data comes from several parties, and these parties collaborate to train a clustering model.
2. single party that holds the data but not the computational resources needed to perform the calculations. The data is outsourced to perform clustering.
3. data comes from multiple parties and is paired to build a shared database. Then the data is outsourced to perform clustering.

Cases 2 and 3 are similar; this case is called “outsourced clustering.” The first case is called “distributed clustering.” Plenty of clustering algorithms have been

seen in the privacy-preserving framework: k-means, k-medoids, GMM, Meanshift, DBSCAN, baseline agglomerative HC BIRCH and Affinity Propagation. Among them, k-means has been intensively studied. In what follows, we focus on works that use homomorphic encryption.

2.5.1 Collaborative Clustering

In the case of collaborative clustering, several parties own data and want to collaborate to get good quality clustering without disclosing the information contained in the data. This case has been extensively studied in two parts. Liu et al. [39] interested in the case where two parties with limited computational resources would like to run k-means by outsourcing the computations to the cloud. Both parties will have a result based on both datasets. In this case, one party's data should be kept confidential from the cloud and the other party. The authors based two schemes to propose a solution: the Liu encryption and the Pallier encryption. Each party encrypts the data and sends it to the cloud. The cloud performs calculations and comparisons based on additional information about both parties. To recalculate the centers, the cloud sends the sum of all vectors to both parties, and the parties use a protocol to calculate the new centers. The authors [40] propose a protocol to perform secure k-means in the semi-honest model. In this work, the Pallier scheme has been used. The computation of the Euclidean distance requires interaction with the data owner to perform the multiplications. The comparison is performed using bit-by-bit encryption.

The authors [41] studied clustering using the k-medoids algorithm applied to intrusion detection. Multiple organizations collaborate to perform clustering and have better results without sharing the content of this information in the clear. In addition, the system relies on a semi-honest party to perform clustering using Pallier encryption. The k-medoid algorithm requires more complex operations than addition. This requires interactions between collaborators to decrypt this data at runtime and thus perform the operations.

2.5.2 Individual Clustering

A clustering is individual if only one person has data and he wants to have the results of the clustering of this data. Most of the works interested in this kind of clustering require an intermediate decryption step. The authors [42] demonstrate a solution to perform k-means using a collaboration between the client and a server. They used the BV scheme [43]. In this work, they proposed three variant solutions. Each solution takes as input a dataset of dimension nxd, an integer k which denotes a the number of clusters and a threshold of iterations. The algorithm returns a matrix of dimension kxd that indicates the cluster centers. In the first variant, the computation of the centers and the assignment are done at the client level, which implies that the client performs a lot of computations (only the distances are computed at the server level). In the second variant, the client performs the comparisons and the division. At the

same time, the server calculates the distances and the assignment of the points, then the sum to calculate the new centers. This variant induces an information leakage on how the points are distributed on the clusters. Finally, a third variant tries to solve the information leakage problem by returning an encrypted assignment vector of a point instead of the clear assignment. The authors [44] propose a method for k-means that limits interaction with the data owner using the concept of "Updatable Distance Matrix (UDM)." The latter is a 3D matrix whose first two dimensions equal the number of data in the dataset, and the third dimension equals the number of attributes. Each cell in the matrix is initialized to the difference between the attributes of the data vectors. The idea is to save the encrypted data and the UDM matrix to a third party. This matrix is updated at each iteration of k-means using an offset matrix obtained by calculating the difference between the new and current centers. This method is expensive in terms of time and memory to store the UDM matrix.

The authors [45] have tried an exact implementation of k-means that requires no intermediate decryption. Instead, the method relies on building a logic circuit to perform k-means using TFHE. From a theoretical point of view, the method gives equivalent results to the plaintext version. However, this method is not feasible; with two dimensions and 400 points, the execution time has been estimated at 25 days.

The authors[46] also propose a solution that focuses on k-means. In this solution, the BGV scheme [47] is used. The authors remark that deciphering the intermediate steps at the client level is a costly operation. The proposed solution relies on using a third party as a trusted entity to decrypt the intermediate results. A private key equivalent (but different) to the owner's and a switch matrix are generated to be used by the trusted server. The proposed solution is considered secure in a semi-honest model but not in the malicious case.

3 Discussion and Challenges

Many challenges need to be tackled to apply privacy-preserving machine learning in real-world applications. Although the HE standards, platforms, and implementations described in this chapter contribute to the advancement of HEML, there are still specific remaining challenges to be tackled, including overhead, performance, interoperability, bootstrapping bottlenecks, sign determination, and common frameworks:

- **Overhead**: Compared to its unencrypted counterpart, HEML has significant overhead, making it unsuitable for many applications. However, for non-HE models, the training phase of ML comprises a computationally intensive effort. However, even with modern techniques, it becomes increasingly difficult with HE. A recent trend is to bypass the training step by employing pre-trained models to find a balance between complexity and accuracy.
- **Hardware Acceleration and parallelization:** Incorporating well-known and new leading to many algorithms is one approach to deal with the computational

overhead. High-performance computers, distributed systems, and specialized resources can all be used in HEML models. Multi-core processing units (GPUs, FPGAs, etc.) and customized chips (ASICs) provide more friendly and efficient HEML environments. Another approach to improving overall efficiency is batching and parallelizing numerous bootstrapping operations. To accelerate FHE programs, one of the research directions is to develop the ability to support multiple hardware acceleration, this is one of the projects under development in the PALISADE library[4].

- **Comparison and min/max function:** We need new methods to compare numbers which are encrypted by Homomorphic Encryption (HE). Actually, comparison and min/max functions are evaluated using Boolean functions where input numbers are encrypted bit-wise. However, the bit-wise encryption methods require relatively expensive computations for basic arithmetic operations such as addition and multiplication.
- **PPML tools:** For the deployment of these technologies, it is practically difficult to design a high-performing and secure PPML solution without a thorough HE understanding. However, PPML developers must be knowledgeable in both machine learning and security. PPML, which uses HE, has not been extensively accepted by the ML community due to HE's high barrier to entry and the absence of user-friendly tools. In terms of model accuracy, how we can ensure that the PPML Homomorphic encryption (HE), we need to develop metric for evaluating models in encrypted domain.
- **Hybrid protocols:** Adopting hybrid protocols, which combine two or more protocols to use the advantages and avoid the disadvantages of each, is a promising direction for performance improvements.
- **Homomorphic encryption (HE) with missing data:** Missing data are a significant problem, as the information available is incomplete and, therefore, less accurate. To solve this problem, we often use suppression of observations with missing data and imputation of missing data. The actual challenge is how to do these methods in the context of encrypted data by using homomorphic encryption.

References

1. P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in *International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 1999), pp. 223–238
2. R.L. Rivest, L. Adleman, M.L. Dertouzos, On data banks and privacy homomorphisms, in *Foundations of Secure Computation* (Academia Press, 1978), pp. 169–179
3. C. Gentry, Fully homomorphic encryption using ideal lattices, in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 June 2, 2009*, ed. by M. Mitzenmacher (ACM, 2009), pp. 169–178. https://doi.org/10.1145/1536414.1536440
4. PALISADE Lattice Cryptography Library (release 1.11.5) (2021). https://palisade-crypto.org/

5. A. Acar et al., A survey on homomorphic encryption schemes: theory and implementation. ACM Comput. Surv. **51**(4) (2018). ISSN:0360-0300. https://doi.org/10.1145/3214303
6. M. van Dijk et al., Fully homomorphic encryption over the integers, in *Advances in Cryptology – EUROCRYPT 2010*, ed. by H. Gilbert (Springer, Berlin, Heidelberg, 2010), pp. 24–43. ISBN:978-3-642-13190-5
7. C. Gentry, Computing arbitrary functions of encrypted data. Commun. ACM **53**(3), 97–105 (2010). ISSN:0001-0782. https://doi.org/10.1145/1666420.1666444
8. O. Regev, On lattices, learning with errors, random linear codes, and cryptography, in *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05 (Association for Computing Machinery, Baltimore, MD, USA, 2005), pp. 84–93. ISBN:1581139608. https://doi.org/10.1145/1060590.1060603
9. V. Lyubashevsky, C. Peikert, O. Regev, On ideal lattices and learning with errors over rings. J. ACM **60**(6) (2013). ISSN:0004-5411. https://doi.org/10.1145/2535925
10. K.R. Rohloff, D. Cousins, A scalable implementation of fully homomorphic encryption built on NTRU, in *Financial Cryptography Workshops* (2014)
11. Z. Chen et al., Bibliometrics of machine learning research using homomorphic encryption. Mathematics **9**, 2792 (2021). https://doi.org/10.3390/math9212792
12. T. Hastie, R. Tibshirani, J. Friedman, Unsupervised learning. *The Elements of Statistical Learning* (Springer, 2009), pp. 485–585
13. R. Bender, U. Grouven, Ordinal logistic regression in medical research. J. R. Coll. Physicians Lond. **31**(5), 546 (1997)
14. V. Gayle, P. Lambert, R.B. Davies, Logistic regression models in sociological research, in *University of Stirling, Technical Paper,* 1 (2009)
15. X. Wang et al., *iDASH secure genome analysis competition 2017* (2018)
16. S. Wu et al., Privacy-preservation for stochastic gradient descent application to secure logistic regression, in *The 27th Annual Conference of the Japanese Society for Artificial Intelligence,* vol. 27 (2013), pp. 1–4
17. Y. Aono et al., Scalable and secure logistic regression via homomorphic encryption, in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy* (2016), pp. 142–144
18. M. Kim et al., Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR Med. Inf. **6**(2), e8805 (2018)
19. J.W. Bos, K. Lauter, M. Naehrig, Private predictive analysis on encrypted medical data. J. Biomed. Inf. **50**, 234–243 (2014)
20. P. Mohassel, Y. Zhang, Secureml: A system for scalable privacy-preserving machine learning, in *2017 IEEE Symposium on Security and Privacy (SP)* (IEEE, 2017), pp. 19–38
21. R. Bost et al., Machine learning classification over encrypted data. IACR Cryptol. ePrint Arch. **2014**, 331 (2015)
22. S. Goldwasser, S. Micali, Probabilistic encryption & how to play mental poker keeping secret all partial information, in *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82 (Association for Computing Machinery, San Francisco, California, USA, 1982), pp. 365–377. ISBN:0897910702. https://doi.org/10.1145/800070.802212
23. F. Li, R. Shin, V. Paxson, Exploring privacy preservation in outsourced K-nearest neighbors with multiple data owners, in *Proceedings of the 2015 ACM Workshop on Cloud Computing Security Workshop*, CCSW '15 (Association for Computing Machinery, Denver, Colorado, USA, 2015), pp. 53–64. ISBN:9781450338257. https://doi.org/10.1145/2808425.2808430
24. B.K. Samanthula, Y. Elmehdwi, W. Jiang, k-Nearest neighbor classification over semantically secure encrypted relational data. IEEE Trans. Knowl. Data Eng. **27**(5), 1261–1273 (2015). https://doi.org/10.1109/TKDE.2014.2364027
25. W.K. Wong et al., Secure KNN computation on encrypted databases, in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09 (Association for Computing Machinery, Providence, Rhode Island, USA, 2009), pp. 139–152. ISBN:9781605585512. https://doi.org/10.1145/1559845.1559862

26. M. Zuber, R. Sirdey, Efficient homomorphic evaluation of k-NN classifiers. Proc. Privacy Enhanc. Technol. **2021**, 111–129 (2021)
27. I. Chillotti et al., TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. **33**(1), 34–91 (2020)
28. Y. Aono et al., Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. **13**(5), 1333–1345 (2017)
29. F. Bu et al., Privacy preserving back-propagation based on BGV on cloud, in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems* (IEEE, 2015), pp. 1791–1795
30. Q. Zhang, L.T. Yang, Z. Chen, Privacy preserving deep computation model on cloud for big data feature learning. IEEE Trans. Comput. **65**(5), 1351–1362 (2016). https://doi.org/10.1109/TC.2015.2470255
31. Q. Zhang et al., GELU-Net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning, in *IJCAI* (2018), pp. 3933–3939
32. R. Gilad-Bachrach et al., Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy, in *International Conference on Machine Learning* (PMLR, 2016), pp. 201–210
33. M. Albrecht, S. Bai, L. Ducas, A subfield lattice attack on overstretched NTRU assumptions, in *Proceedings, Part I, of the 36th Annual International Cryptol- ogy Conference on Advances in Cryptology—CRYPTO 2016—Volume 9814* (Springer, Berlin, Heidelberg, 2016), pp. 153–178. ISBN:978-3-662-53017-7. https://doi.org/10.1007/978-3-662-53018-4_6
34. M. Baryalai, J. Jang-Jaccard, D. Liu, Towards privacy-preserving classification in neural networks, in *2016 14th Annual Conference on Privacy, Security and Trust (PST)* (2016), pp. 392–399. https://doi.org/10.1109/PST.2016.7906962
35. H. Chabanne et al., Privacy-preserving classification on deep neural network. Cryptology ePrint Arch. (2017)
36. E. Hesamifard, H. Takabi, M. Ghasemi, Deep neural networks classification over encrypted data, in *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy* (2019), pp. 97–108
37. Q. Zhu, X. Lv, 2P-DNN: Privacy-preserving deep neural networks based on Homomorphic cryptosystem. Preprint (2018). arXiv:1807.08459
38. I. Chillotti, M. Joye, P. Paillier, Programmable bootstrapping enables efficient homomorphic inference of deep neural networks, in *Cyber Security Cryptography and Machine Learning*, ed. by S. Dolev et al. (Springer International Publishing, Cham, 2021), pp. 1–19. ISBN:978-3-030-78086-9
39. X. Liu et al., Outsourcing two-party privacy preserving K-means clustering protocol in wireless sensor networks, in *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)* (2015), pp. 124–133. https://doi.org/1.1109/MSN.2015.42
40. Z.L. Jiang et al., Efficient two-party privacy preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing. Information Sciences **518**, 168–180 (2020). ISSN:0020-0255. https://doi.org/10.1016/j.ins.2019.12.051. https://www.sciencedirect.com/science/article/pii/S0020025519311624
41. G. Spathoulas, G. Theodoridis, G.-P. Damiris, Using homomorphic encryption for privacy-preserving clustering of intrusion detection alerts. Int. J. Inf. Secur. **20**, 347–370 (2021). https://doi.org/10.1007/s10207-020-00506-7
42. A. Theodouli, K.A. Draziotis, A. Gounaris, Implementing private k-means clustering using a LWE-based cryptosystem, in *2017 IEEE Symposium on Computers and Communications (ISCC)* (2017), pp. 88–93
43. Z. Brakerski, V. Vaikuntanathan, C. Gentry, Fully homomorphic encryption without bootstrapping, in *In Innovations in Theoretical Computer Science* (2012)
44. N. Almutairi, F. Coenen, K. Dures, K-means clustering using homomorphic encryption and an updatable distance matrix: secure third party data clustering with limited data owner interaction, in *DaWaK* (2017)

45. A. Jäschke, F. Armknecht, Unsupervised machine learning on encrypted data. IACR Cryptol. ePrint Arch. **2018**, 411 (2018)
46. G. Sakellariou, A. Gounaris, Homomorphically encrypted K-means on cloud-hosted servers with low client-side load. Computing **101**(12), 1813–1836 (2019). ISSN:0010-485X. https://doi.org/10.1007/s00607-019-00711-w
47. Z. Brakerski, C. Gentry, V. Vaikuntanathan, (Leveled) Fully homomorphic encryption without bootstrapping, in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12 (Association for Computing Machinery, Cambridge, MA, 2012), pp. 309–325. ISBN:9781450311151. https://doi.org/10.1145/2090236.2090262