Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6615839, 11 pages
https://doi.org/10.1155/2021/6615839

Research Article

Multiparty Homomorphic Machine Learning with Data Security and Model Preservation

Fengtian Kuang,[1] **Bo Mi**,[2] **Yang Li**,[2] **Yuan Weng**,[2] **and Shijie Wu**[3]

[1]*Chongqing Jiaotong University, Mathematics and Statistics, Chongqing 400074, China*
[2]*Chongqing Jiaotong University, Information Science and Engineering, Chongqing 400074, China*
[3]*Unit 78156 of the Chinese People's Liberation Army, Chengdu 610000, China*

Correspondence should be addressed to Bo Mi; mi_bo@163.com

Received 1 November 2020; Revised 29 November 2020; Accepted 22 December 2020; Published 11 January 2021

Academic Editor: Yong Chen

With the widespread application of machine learning (ML), data security has been a serious issue. To eliminate the conflict between data privacy and computability, homomorphism is extensively researched due to its capacity of performing operations over ciphertexts. Considering that the data provided by a single party are not always adequate to derive a competent model via machine learning, we proposed a privacy-preserving training method for the neural network over multiple data providers. Moreover, taking the trainer's intellectual property into account, our scheme also achieved the goal of model parameter protection. Thanks to the hardness of the conjugate search problem (CSP) and discrete logarithm problem (DLP), the confidentiality of training data and system model can be reduced to well-studied security assumptions. In terms of efficiency, since all messages are coded as low-dimensional matrices, the expansion rates with regard to storage and computation overheads are linear compared to plaintext implementation without accuracy loss. In reality, our method can be transplanted to any machine learning system involving multiple parties due to its capacity of fully homomorphic computation.

1. Introduction

With the continuous development of artificial intelligence, data have become precious resources due to their value for mining. Nevertheless, numerous private information is embodied as data, which may be abused to violate personal privacy, business secrets, or even state secrets. For example, once a patient's medical record is exposed to insurance companies, they may never sell him some kind of medical insurance [1]. Similarly, many other machine learning applications have also caught sight of privacy infringements, such as financial analysis, product customization, and public opinion surveillance [2–4]. On the other hand, any data-driving mechanism heavily relies on the quantity and quality of information, which brings about the conflict between data usability and data confidentiality. Fortunately, secure multiparty computation (SMC) [5–7] and homomorphic encryption (HE) [8, 9] provide us powerful tools to process data in a concealed manner. Therefore, the remaining problem to address is how to devise a cryptosystem that is applicable for machine learning in consideration of storage and computation overheads.

As a cryptographic technology orienting decentralized systems, secure multiparty computation aims at data confidentiality for distributed participants. Despite the privacy concern, the involved parties can still figure out a public output as they wish. Based on such cryptosystem, F. Ö. Çatak et al. [10] proposed a privacy-preserving learning protocol for classification in virtue of vertically segmented data from multiple parties. Since the data is just partially shared without concealing, semantic security is unachievable as plain data dose. The first provable secure ML protocol of this kind is presented by R. Devin et al. [11] for text classification. Howbeit, their research only focused on the privacy of data classification and left the learning process unaddressed.

Oriented at centralized systems, homomorphic encryption is another way towards secure machine learning, which is capable of performing specific operations over ciphertexts. Researches of applying FE for data privacy during machine learning have developed rapidly since the significant innovation [12] appeared in 2016. Y. Aono et al. [13] combined the additive homomorphism with deep learning to narrow the gap between system functionality and data security, by applying FH technology to asynchronous stochastic gradient descent algorithm. F. Bourse et al. [14] improved the FHE structure of Chillotti et al. [15] and proposed a homomorphic neural network evaluation framework, namely, FHE-DiNN. Its complexity is strictly linear in network depth, but the model parameters must be proactively predefined. Based on a multikey variant of two HE schemes [16, 17] with ciphertexts packed, H. Chen et al. [18] provided a suite of interfaces for secure machine learning which also exploited bootstrapping for arbitrary circuit evaluation. As matter of fact, almost all existing FHE-based machine learning algorithms are based on the algebraic structure of lattice, such as BGV [19–21], CKKS [22–24], and NTRU [25–27]. These methods suffer from a common defect that decryption may fail due to noise growth. Though bootstrapping can be deemed as an effective tool for noise control, its extra computational burden is hardly acceptable. Surprisingly, J. Li et al. [28] discovered an alternative tool, saying Conjugate Search Problem, to actualize full homomorphism without noise interference. They also applied such cryptosystem for privacy-preserving data training, which achieved the same accuracy as the plaintexts used for learning.

Though more comprehensible and effective than lattice-based secure machine learning, Li's scheme can only be applied to the scenario of a signal data provider. Ordinarily, one party can always provide a small quantity of data which may incur an overfitted model. To ensure the generalization of machine learning, data from diverse sources should be gathered for a specific learning task. In the circumstances of multiparty secure machine learning, each data provider may conceal their information by a dependent key. Therefore, a training framework that operates over heterogeneous (i.e., encrypted by different keys) ciphertext is desiderated. Conversely, the parameters of the system model should be taken as assets held by the trainer as in general business operation. Thus, we should also make sure that the machine is concealed, even when not thoroughly trained.

To preserve the privacy of all participants, this paper presents a complete machine learning mechanism in virtue of CSP and DLP hardness. Our contributions are summarized as follows.

1.1. Contributions

(1) We coded float-type data as low-dimensional upper triangular matrices that are homomorphic under the operations of addition, subtraction, multiplication, division, and comparison. With the help of CSP, the plain matrices can also be projected to semantically secure ciphertexts homomorphically under the same kind of operations. That is to say, our basic cryptosystem is fully homomorphic, since addition and multiplication are simultaneously implemented. Therefore, we can realize secure training and classification/regression once private data are provided under the same key.

(2) We constructed a cyclic group by lifting the plain matrices to a Galois domain. Thereafter, key switching (switch a ciphertext encrypted by one key to another) is made possible via DLP for the purpose of cooperative training.

(3) We combined the two aforementioned technologies and devised a secure machine learning protocol under semihonest model, which preserves the privacy of multiple data providers as well as that of the trainer.

2. System Model

Neural network (NN) is employed as the engineering background and verification model in this paper due to its extensive application. Nevertheless, it is worth mentioning that our scheme can be applied to most machine learning algorithms if privacy is significant to multiple participants.

2.1. Neural Network Model. A typical neural network contains three or more layers, which turns into a deep learning model if hidden layers are multiple [29]. The certain principle of NN lies in the fact that numerous neurons can automatically extract features of the inputs layer by layer. Besides the topology of NN, the most important factors that defined it are the weights and bias designated to each link and neuron. As for learning, the essence is how to adjust these parameters in virtue of training data via iterative forward-/back-propagation. Thereafter, to securely implement a neural network model, we should homomorphically evaluate the following functions.

Forward calculation (e.g., sigmoid):

$$f_{fw} = \text{sigmoid}\left(\mathbf{a}_i, \mathbf{w}_i + b_i\right), \tag{1}$$

where $\mathbf{a}_i$ and $\mathbf{w}_i$ are the input and weight vectors corresponding to the proactive links of neuron i, while b_i represents its bias.

Backward calculation:

Loss function (e.g., quadratic loss function):

$$f_{\text{bw\_loss}} = L(Y|ft(X)) = \sum_n \left(t_n - o_n\right)^2, \tag{2}$$

where t_n is the target value and o_n is the actual value.

Parameter adjusting (e.g., gradient descent):

$$f_{\text{bw\_adj}} = \text{old}\, w_{j,k} - \Delta w_{j,k}, \tag{3}$$

$$\Delta w_{j,k} = \alpha \cdot E_k \cdot O_k\left(1 - O_k\right) \cdot O_j^T, \tag{4}$$

where E_k is the error vector between the target value and the actual value, O_j^T is the transpose of the output of the current layer node, and O_k is the output of the node of the next layer.

2.2. System Model and Security Goal. In our system, a powerful trainer expects to acquire a neural network whose topology is predefined. To ensure the completeness of the resultant model, they may request multiple parties for training data. However, the data providers concern about privacy leakage though they have strong wills to cooperate. Meanwhile, the trainer also worries that the system parameters may expose and infringe their intellectual property. Therefore, we should preserve the privacy of all participants and guarantee the functionality of machine learning at the same time. Moreover, taking the trained neural network as a service, a user may not only desire to designate a classification/regression task to the server but also be anxious about data abuse.

2.3. Adversary Model. Suppose that the trainer and all data providers are honest but curious during the whole process. That is to say, they will completely follow the protocol to avoid unnecessary disputes but may be interested in the privacy contained within the data. Furthermore, it is reasonable to assume that both the trainer and data owner are provided with PPT (probabilistic polynomial time) computational power. However, since the trainer is always better equipped than data providers, the hypothesis that they have the accessibility to a quantum machine may also be valid. To define the success of privacy violation, we exploit the concept of symmetric IND-CPA (indistinguishability under chosen-plaintext attack) as below.

2.4. Symmetric IND-CPA [30]. Define an experiment under symmetric cryptosystem SE = (SE.KeyGen, SE.Enc, SE.Dec) as

$$\begin{aligned}
&\mathrm{Exp}_{\mathrm{HE},\mathscr{A}}^{\mathrm{CPA}}(\kappa):\\
&k\leftarrow \mathrm{HE.KeyGen}(\kappa),\\
&(M_0, M_1)\leftarrow \mathscr{A}^{\mathrm{HE.Enc}_k(\cdot)}(\cdot),\ \text{for } |M_0| = |M_1|,\\
&\gamma\leftarrow_R\{0, 1\},\ C^* = \mathrm{HE.Dec}(M_\gamma),\\
&\gamma\prime = \mathscr{A}(C^*),\\
&\text{Output 1, if } \gamma = \gamma\prime, \text{and 0 otherwise,}
\end{aligned} \tag{5}$$

for any PPT adversary $\mathscr{A}$ that queries the oracle $\mathrm{HE.Enc}_k(\cdot)$ polynomial times. Thus, the adversary's advantage can be expressed by

$$\mathrm{Adv}_{\mathrm{HE},\mathscr{A}}^{\mathrm{CPA}}(\kappa) = \left|\Pr\left[\mathrm{Exp}_{\mathrm{HE},\mathscr{A}}^{\mathrm{CPA}}(\lambda) = 1\right] - \frac{1}{2}\right|. \tag{6}$$

Then the cryptosystem SE is IND-CPA-secure if $\mathrm{Adv}_{\mathrm{HE},\mathscr{A}}^{\mathrm{CPA}}(\kappa) < \epsilon(\kappa)$, where $\epsilon(\kappa)$ stands for a negligible function in the security parameter κ.

3. Cryptographic Construction

Focusing on the security goals presented in the system model, we are now ready to construct our cryptographic building blocks. In this part, we first explore the homomorphism of conjugate search problem to underpin the functionality of training over homogeneous (i.e., encrypted by the same key) ciphertexts. Then, we present a key switching technology that can convert a ciphertext encrypted by one key to be decryptable by another.

Conjugate search problem is a special form of group factorization problem (GFP) [31], defined as follows.

3.1. Conjugate Search Problem (CSP) [31]. Given $(C, M) \in \Psi \times \Psi$ over a nonabelian algebraic structure Ψ, it is intractable to solve $H \in \Psi$ such that $C = \mathrm{HMH}^{-1}$.

B. Evgeni [32] proved that the CSP is postquantum secure over the general linear group $\mathrm{GL}_d(R)$ (R means real number field) if $d \geq 4$. Hence, to assure system security, we should code the message as a matrix with degree larger than 4.

To protect the privacy of data providers without affecting the accuracy of training, we resort to homomorphic encryption that is capable of actualizing the forward-/backward-propagation processes covertly. Thereafter, we devised a way that makes CSP semantically secure and homomorphic. It is worth noting that the conjugate search problem is resistant to quantum attacks, which dispels the privacy concern for data providers even if the trainer is extremely equipped.

A typical homomorphic encryption algorithm can be noted as a tetrad HE = (HE.KeyGen, HE.Enc, HE.Dec, HE.Eval), standing for the functions of key generation, encryption, decryption, and evaluation, respectively.

For any data m over the message space R, we first code it as an upper triangular matrix $M \in R^{6\times 6}$ as follows.

3.2. Encoding. Convert the message m into three pairs of random numbers (a_1, a_2), (a_3, a_4), and (a_5, a_6), satisfying $a_1 + a_2 = m$, $a_3 + a_4 = a_5 + a_6 = r$, and $(a_3^2 - a_4^2)(a_5^2 - a_6^2) = 1$, where r is a constant random number of the system. Thus, we can construct the following matrices:

$$\begin{aligned}
M_1 &= \begin{pmatrix} a_1 & a_2 \\ a_2 & a_1 \end{pmatrix},\\
M_2 &= \begin{pmatrix} a_3 & a_4 \\ a_4 & a_3 \end{pmatrix},\\
M_3 &= \begin{pmatrix} a_5 & a_6 \\ a_6 & a_5 \end{pmatrix}.
\end{aligned} \tag{7}$$

Combining the above matrices, the message m is finally coded as

$$M = \begin{pmatrix} M_1 & R_1 & R_2 \\ 0 & M_2 & R_3 \\ 0 & 0 & M_3 \end{pmatrix}, \tag{8}$$

where 0 represents the 2×2 all-zero matrix and R_i $(i = 1, 2, 3)$ stands for random matrices uniformly sampled from $R^{2\times2}$.

For clarity, we denote the space of coded messages as Γ. It is interesting that Γ naturally constitutes a multiplicative cyclic group (excluding the elements whose determinants are zero) and $R \sim \Gamma$ (homomorphic). Furthermore, it is well known that all square matrices with the same dimension compose a ring. Though $\Gamma \subseteq R^{6\times6}$ and its elements are commutative for multiplication, there is an overwhelming probability that a matrix P uniformly sampled from $R^{6\times6}$ is noncommutative with the coded message M. Thereupon, a CSP-based fully homomorphic encryption algorithm can be actualized as below.

3.3. Key Generation. HE.KeyGen(1^κ): uniformly sample a matrix from $R^{9\times9}$, which can also be represented as a combination of nine 6×6 random matrices, namely,

$$P = \begin{pmatrix} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_9 \end{pmatrix}, \quad \text{for } P_i = \begin{pmatrix} p_{i1} & p_{i2} \\ p_{i3} & p_{i4} \end{pmatrix}, (i = 1, 2, \ldots, 9). \tag{9}$$

The probability that P is communitive with elements in Γ should be negligible. Then, the algorithm takes $k = P$ as the symmetric key.

3.4. Encryption. HE.Enc(P, M): output

$$C = \mathrm{PMP}^{-1} = P\begin{pmatrix} M_1 & R_1 & R_2 \\ 0 & M_2 & R_3 \\ 0 & 0 & M_3 \end{pmatrix}P^{-1} \tag{10}$$

as the ciphertext of message m (coded as a matrix M).

3.5. Decryption. HE.Dec(P, C): compute $M = P^{-1}CP$ to obtain

$$M_1 = \begin{pmatrix} a_1 & a_2 \\ a_2 & a_1 \end{pmatrix}. \tag{11}$$

Then, figure out $m = a_1 + a_2$ to recover the plaintext.

3.6. Evaluation. HE.Eval$(f, C_1, \ldots, C_l)$: We describe the very basic operations underpinning formulae (2)-(4) in advance. Suppose that C_1 and C_2 are ciphertexts corresponding to m_1 and m_2 under the same key; the additive and multiplicative arithmetic can be simply carried out by $C_{\text{add}} = C_1 + C_2$ and $C_{\text{mul}} = C_1C_2$. These two operations can be trivially assembled to realize the functions for backward propagation. However, since the exponential operation cannot be implemented directly via homomorphic addition and multiplication, some activation functions of forward propagation such as *sigmoid* should be approximated as the form of polynomials. Thereby, we resort to a specific conversion [32–34],

$$\mathrm{sigmod}(x) = \begin{cases} 0.000734x^4 + 0.014222x^3 + 0.108706x^2 + & \\ 0.392773x + 0.571859, & -\infty < x \le -1.5, \\ 0.002083x^5 + 0.020833x^3 + 0.25x + 0.5, & -1.5 < x \le 1.5, \\ -0.000734x^4 + 0.014222x^3 - 0.108706x^2 + & \\ 0.3922773x + 0.428141, & 1.5 < x < \infty, \end{cases} \tag{12}$$

to replace

$$\mathrm{sigmod}(x) = \frac{1}{1 + e^{-x}}. \tag{13}$$

Noting that the aforementioned formula is expressed as a piecewise function, to homomorphically decide which subfunction should be carried out, we can encrypt the numbers of -1.5 and 1.5 and compare them with x for branching.

To program a piecewise function, J. Li et al. [28] presented a homomorphic algorithm that covertly compares the size between two ciphertexts. Though our scheme is similar to that of [28], we argue that their cryptosystem is not semantically secure because $a_{2i-1} + a_{2i} = m$ and a_{2i-1} is always bigger than a_{2i} for $i = 1, 2, 3$.

3.7. Security Analysis of [28]. By computing

$$\det\left(C^* - TC\prime\right) = \det(P)\det\left(\begin{pmatrix} M_1^* - tM_1' & R_1^* & R_2^* \\ 0 & M_2^* - tM_2' & R_3^* \\ 0 & 0 & M_2^* + tM_2' \end{pmatrix}\right)\det\left(P^{-1}\right), \tag{14}$$

where R_i^* is also a random matrix, for

$$T = \begin{pmatrix} t & R_1 & R_2 \\ 0 & t & R_3 \\ 0 & 0 & t \end{pmatrix}, \tag{15}$$

where

$$t = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{16}$$

and $R_i\,(i = 1, 2, 3)$ is uniformly sampled from $R^{2\times 2}$, the adversary carries out a chosen-plaintext attack such as the following.

$\mathrm{Exp}_{\mathrm{HE},\mathscr{A}}^{\mathrm{CPA}}(\kappa)$: ,
$k \leftarrow \mathrm{HE.KeyGen}(\kappa)$,
$(M_0, M_1, C') \leftarrow \mathscr{A}^{\mathrm{HE.Enc}_k(\cdot)}(\cdot)$,,
for $|M_0| = |M_1|, C' = \mathrm{HE.Enc}_k(M')$, and $M_0 < M' < M_1$,
$\gamma \leftarrow_R \{0, 1\}$, $C^* = \mathrm{HE.Dec}(M_\gamma)$,
$\gamma' = 1$ if $\det(C^* - \mathrm{TC}')$, and $\gamma' = 0$ otherwise,
Output 1, if $\gamma = \gamma'$, and 0 otherwise.
Considering that

$$\begin{aligned} \det(C^* - \mathrm{TC}') = \det(P)\det(M_1^* - tM_1')\det(M_2^* - tM_2') \\ \det(M_3^* - tM_3')\det(P^{-1}), \quad (i = 1, 2, 3), \end{aligned} \tag{17}$$

where

$$\begin{aligned} \det(M_i^* - tM_i') &= (a_{2i-1}^* - a_{2i}')^2 - (a_{2i}^* - a_{2i-1}')^2, \\ &= (m^* - m\prime)((a_{2i-1}^* - a_{2i}^*)t + n(a_{2i-1}' - a_{2i}')), \end{aligned} \tag{18}$$

since $a_{2i-1} > a_{2i}$ is guaranteed throughout Li's scheme [28], $((a_{2i-1}^* - a_{2i}^*)t + n(qa_{2i-1}'h - a\,2i)$ must be positive. It is obvious that $\det(P)\det(P^{-1}) = 1$; hence, the adversary can easily determine whether $m^* = m_0$ or $m^* = m_1$ by checking the sign of $\det(C^* - \mathrm{TC}\prime)$. That is to say,

$$\mathrm{Adv}_{\mathrm{HE},\mathscr{A}}^{\mathrm{CPA}}(\kappa) = \left|\Pr\left[\mathrm{Exp}_{\mathrm{HE},\mathscr{A}}^{\mathrm{CPA}}(\kappa) = 1\right] - \frac{1}{2}\right| = 1. \tag{19}$$

It seems that the conflict between piecewise function evaluation and IND-CPA security is infeasible to address. However, we can introduce a specific form of ciphertext which can be used to encrypt a designated number and compare it with any other normal ciphertext. Our construction is given below.

The data provider randomly chooses a nonzero number $k \in R - \{0\}$ and encrypts $m\prime$ as

$$C\prime = \mathrm{PM}\prime P^{-1} = P\begin{pmatrix} kM_1' & R_1 & R_2 \\ 0 & kM_2' & R_3 \\ 0 & 0 & kM_3' \end{pmatrix}P^{-1}, \tag{20}$$

for

$$M_i' = \begin{pmatrix} a_{i1}' & a_{i2}' \\ a_{i3}' & a_{i4}' \end{pmatrix}, \quad (i = 1, 2, 3), \tag{21}$$

which satisfies

$$\begin{cases} a_{i1}' + a_{i4}' = k \neq 0, & i = 1, 2, 3, \\ a_{i2}' + a_{i3}' = -k \neq 0, & i = 1, 2, 3, \\ a_{i1}'a_{i4}' - a_{i2}'a_{i3}' = m\prime, & i = 1, \\ a_{i1}'a_{i4}' - a_{i2}'a_{i3}' = r, & i = 2, 3. \end{cases} \tag{22}$$

To compare C' with general cipher C^* without decryption, the evaluator computes

$$\Delta = \frac{\det(C^*C')}{\det(C')} - \det(C^* - C') = k^2(m^* - m') \tag{23}$$

and thus achieves

$$C_{\mathrm{comp}} = \begin{cases} m^* > m' & \text{if } \Delta > 0, \\ m^* > m' & \text{if } \Delta = 0, \\ m^* > m' & \text{if } \Delta < 0. \end{cases} \tag{24}$$

3.8. Correctness. The correctness of encryption and decryption algorithms is straightforward, so we only focus on the homomorphism of evaluation.

Homomorphic addition: since

$$\begin{aligned} C_{\mathrm{add}} &= C_1 + C_2, \\ &= P^{-1}\begin{pmatrix} M_{11} & R_{11} & R_{12} \\ 0 & M_{12} & R_{13} \\ 0 & 0 & M_{13} \end{pmatrix}P + P^{-1}\begin{pmatrix} M_{21} & R_{21} & R_{22} \\ 0 & M_{22} & R_{23} \\ 0 & 0 & M_{23} \end{pmatrix}P, \\ &= P^{-1}\begin{pmatrix} M_{11} + M_{21} & R_{11} + R_{21} & R_{12} + R_{22} \\ 0 & M_{12} + M_{22} & R_{13} + R_{23} \\ 0 & 0 & M_{13} + M_{23} \end{pmatrix}P, \end{aligned} \tag{25}$$

we can decrypt it as

$$\begin{aligned} M_{\mathrm{add}} &= P(C_1 + C_2)P^{-1}, \\ &= \begin{pmatrix} M_{11} + M_{21} & R_{11} + R_{21} & R_{12} + R_{22} \\ 0 & M_{12} + M_{22} & R_{13} + R_{23} \\ 0 & 0 & M_{13} + M_{23} \end{pmatrix} \end{aligned} \tag{26}$$

because

$$M_{11} + M_{21} = \begin{pmatrix} a_{11} + a_{21} & a_{12} + a_{22} \\ a_{12} + a_{22} & a_{11} + a_{21} \end{pmatrix}. \tag{27}$$

The addition of m_1 and m_2 can be decoded as

$$m_{\mathrm{add}} = a_{11} + a_{21} + a_{12} + a_{22} = m_1 + m_2. \tag{28}$$

Homomorphic multiplication: because

$$
\begin{aligned}
C_{\text{mul}} &= C_1 C_2, \\
&= P^{-1}\begin{pmatrix} M_{11} & R_{11} & R_{12} \\ 0 & M_{12} & R_{13} \\ 0 & 0 & M_{13} \end{pmatrix}\begin{pmatrix} M_{21} & R_{21} & R_{22} \\ 0 & M_{22} & R_{23} \\ 0 & 0 & M_{23} \end{pmatrix} P, \\
&= P^{-1}\begin{pmatrix} M_{11}M_{21} & R_1^* & R_2^* \\ 0 & M_{12}M_{22} & R_3^* \\ 0 & 0 & M_{13}M_{23} \end{pmatrix} P, \\
M_{11}M_{21} &= \begin{pmatrix} a_{11}a_{21} + a_{12}a_{22} & a_{11}a_{22} + a_{12}a_{21} \\ a_{12}a_{21} + a_{11}a_{22} & a_{12}a_{22} + a_{11}a_{21} \end{pmatrix},
\end{aligned} \tag{29}
$$

we can deduce that

$$
(a_{11}a_{21} + a_{12}a_{22}) + (a_{11}a_{22} + a_{12}a_{21}) = (a_{11} + a_{12})(a_{21} + a_{22}) = m_1 m_2. \tag{30}
$$

Homomorphic comparison: on the premise of $\det(P)\det(P^{-1}) = 1$, it can be seen that

$$
\begin{aligned}
\Delta &= \frac{\det\left(C^* C\prime\right)}{\det(C\prime)} - \det\left(C^* - C\prime\right), \\
&= \frac{\det\left(M^* M\prime\right)}{\det(M\prime)} - \det\left(M^* - M\prime\right)a, \\
&= \prod_{i=1}^{3} \frac{\det\left(M_i^* k M_i'\right)}{\det\left(k M_i'\right)} - \prod_{i=1}^{3} \det\left(M_i^* - k M_i'\right).
\end{aligned} \tag{31}
$$

According to formula (21), we have

$$
\begin{aligned}
\frac{\det\left(M_i^* k M_i'\right)}{\det\left(k M_i'\right)} &= a_{2i-1}^* 2 - a_{2i}^* 2, \\
\det\left(M_i^* - M_i'\right) &= \left(a_{2i-1}^* 2 - a_{2i}^* 2\right) - k^2 2i \\
&\quad \left(a_{2i-1}^* + a_{2i}^* - \det\left(M_i'\right)\right), \quad i = 1, 2, 3.
\end{aligned} \tag{32}
$$

Recall that $a_{2i-1}^* + a_{2i}^* = m^*$ and $a_{i1}' a_{i4}' - a_{i2}' a_{i3}' = m\prime$ when $i = 1$, while $a_{2i-1}^* + a_{2i}^* = a_{i1}' a_{i4}' - a_{i2}' a_{i3}' = r$ when $i = 2, 3$. In terms of the condition that $(a_3^2 - a_4^2)(a_5^2 - a_6^2) = 1$, we can reduce formula (30) to

$$
\Delta = k^2\left(m^* - m\prime\right). \tag{33}
$$

It is obvious that the signs of Δ and $m^* - m\prime$ are exactly the same, since $k^2 > 0$, which determines the relationship between m^* and m' without decryption.

3.9. Security. Thanks to the hardness of Conjugate Search Problem, an adversary must find P such that $P^{-1}\text{CP} = M$ to recover the plaintext. As for the semantic security of our scheme, it can be seen that $((a_{2i-1}^* - a_{2i}^*) + (a_{2i-1}' - a_{2i}'))$ in formula (17) is not always positive due to arbitrary relationship between a_{2i-1} and a_{2i}. Therefore, when an adversary executes a chosen-plaintext attack as mentioned before, their advantage is negligible. Noting that any normal ciphertext can just be compared with specifically encrypted messages without decryption, the data provider has full control over their privacy and permits exact comparisons only if necessary.

After each training, the neural network coefficients are concealed by the key of the data provider. When multiple data providers take part in the training process, those semimanufactured parameters should also be re-encrypted under the key of subsequent data holder for homomorphic computation. Therefore, we devised a way to decrypt and re-encrypt the machine coefficients without exposing them to data providers, in consideration of the trainer's property right. Our key switching scheme is based on the hardness of Discrete Logarithm Problem (DLP).

3.10. Discrete Logarithm Problem. Given a cyclic group G, a generator $g \in G$, and a random element $h \in G$, it is difficult to find the discrete logarithm a such that $g^a = h$.

Accordingly, if an adversary has obtained a ciphertext $y = h^b = g^{ab} \in G$, it is hard for them to recover h because of the confidentiality on ab [35]. However, in light of the Lagrange theorem [36], we can exploit a trapdoor to reverse y back to h.

3.11. Lagrange Theorem. Denote H as a subgroup of finite G; then, $|H|||G|$, for $|H|$ and $|G|$ are the orders of groups H and G.

Since any $h \in G$ generates a subgroup $H \subseteq G$ via $H = \{h^a | a \in Z\}$, we can conclude that $h^{|G|} = e$ in terms of the Lagrange theorem, where e is the identity of group G.

Based on the aforementioned mathematical tools, we are now ready to construct our key switching scheme as a triad KS = (KS.KeyGen, KS.CSPtoDLP, KS.DLPtoCSP). Without loss of generality, we denote $k_t = (b, s)$, $k_A = P_A$, and $k_B = P_B$ as secret keys belonging to the trainer T and two data providers A and B, respectively. Then KS.KeyGen can be used to generate the encryption/decryption key pair for the trainer, while KS.CSPtoDLP is used to convert a ciphertext C_A encrypted by k_A to be decryptable by k_t and KS.CSPtoDLP is utilized to modify C_t (encrypted under k_t) as C_B whose corresponding key is k_B.

3.12. Key Generation. KS.KeyGen(1^κ): as mentioned before, we denote the space of coded messages as Γ. Suppose that the precision of matrix elements in HE is l-bits whose integer part is m-bits and the decimal part is n-bits. We can multiply any coded plaintext M by 2^n to lift it over $Z_{2^l}^{6\times 6}$. Accordingly, the message space is changed to a cyclic group $\Gamma\prime$ for $|\Gamma\prime| = 2^{12l}(2^l - 1)^3$. Moreover, for each $2^n M_i$, it composes a group Γ_i' satisfying $|\Gamma_i'| = 2^l - 1$. Thereby, we uniformly sample an odd number $b \in Z_{2^l-1}$ and compute $s \in Z_{2^l-1}$ such that $s \cdot b = 1 \bmod (2^l - 1)$. Output $k_t = (b, s)$ as the key to the trainer.

3.13. Switching $\mathbf{C_A}$ *to* $\mathbf{C_t}$ *KS.CSPtoDLP*(C_A). The trainer T changes the encrypted model parameters C_A as $C_A' = 2^n C_A \bmod 2^l \in Z_{2^l}^{6\times 6}$ and sends $C_{At} = (C_A')^b \bmod 2^l$ to data provider A. On receiving C_{At}, A computes $C_t = P_A^{-1} C_{At} P_A \bmod 2^l$ as their response.

3.14. Switching $\mathbf{C_t}$ *to* $\mathbf{C_B}$ *KS.DLPtoCSP*(C_t). On receiving C_t from the trainer T, the data provider B computes their response as $C_{tB} = P_B C_t P_B^{-1} \bmod 2^l$. Therefore, the trainer T can reverse C_{tB} back to a ciphertext $C_B = P_B M P_B^{-1}$ purely encrypted under k_B via $C_B' = (C_{tB})^s \bmod 2^l$ and then right-shift its elements by n-bits.

3.15. Correctness. Since $C_A' = P_A(2^n M)P_A^{-1} \bmod 2^l$, we have

$$C_{At} = \underbrace{P_A(2^n M)P_A^{-1} P_A(2^n M)P_A^{-1}, \ldots, P_A(2^n M)P_A^{-1}}_{b \text{ times}} = P_A(2^n M)^b P_A^{-1} \bmod 2^l. \tag{34}$$

Thus, $C_t = P_A^{-1} P_A (2^n M)^b P_A^{-1} P_A = (2^n M)^b \bmod 2^l$.
Similarly, because $C_{tB} = P_B (2^n M)^b P_B^{-1} \bmod 2^l$,

$$C_B' = P_B (2^n M)^{b\cdot s} P_B^{-1} \bmod 2^l, = P_B \begin{pmatrix} (2^n M_1)^{1+k_1(2^l-1)} & R_1^* & R_2^* \\ 0 & (2^n M_2)^{1+k_3(2^l-1)} & R_3^* \\ 0 & 0 & (2^n M_3)^{1+k_3(2^l-1)} \end{pmatrix} P_B^{-1}, \tag{35}$$

where k_i are integers for $i = 1, 2, 3$.

During the encoding process in HE, it is easy to choose a_{2i-1} such that $2^n a_{2i-1} \neq 0 \bmod 2^l$. Considering that $a_1 + a_2 = m$ and $a_{2i-1} + a_{2i} = r$ for $i = 1, 2$, the space of $2^n M_i$ must be a cyclic group Γ_i' for $|\Gamma_i'| = 2^l - 1$. According to the Lagrange Theorem, it can be seen that $(2^n M_i)^{1+k_i(2^l-1)} = 2^n M_i \bmod 2^l$; thus $C_B' = P_B(2^n M)P_B^{-1} \bmod 2^l$. By right-shifting n-bits on C_B', we obtain $C_B = P_B M P_B^{-1}$.

3.16. Security. Note that, after receiving C_{At}, the trainer can trivially compute $M = (C_t^s \bmod 2^l)/2^n$ to recover the message. Nevertheless, since the model parameters are of their intellectual property, such operation does not conflict with our security goal.

As for data providers, they can just witness an exponential form of the plaintext (i.e., $C_t = (2^n M)^b \bmod 2^l$). According to the hardness of DLP, the information about message M will not be exposed.

4. Privacy-Preserving Machine Learning with Multiple Data Providers

To preserve privacy for machine learning, many cryptographic training and classification/regression methods have been proposed in the scene of a single data provider. In most cases, data should be sourced from multiple providers to guarantee the generality of training. Therefore, we present a secure machine learning mechanism with the capacity of training as well as classification/regression in consideration of data and parameter privacy.

As for training, the cloud is supposed to obtain model parameters with the help of labeled data. During the initialization phase, the trainer T computes $k_t \leftarrow \text{KS.KeyGen}(1^\kappa)$ for key switching and each data provider i generates $k_i \leftarrow \text{HE.KeyGen}(1^\kappa)$ for homomorphic training.

Denote the encoded training data owned by provider i as M_i and the system parameters as M_t. The server primarily encrypts the initialized system coefficients (may contain some private intellectual property information) as $C_t = (2^n M_t)^b \bmod 2^l$ to the first data provider who executes $C_1 \leftarrow \text{HE.Enc}(k_1, M_1)$ and $\widetilde{C}_1 \leftarrow \text{KS.DLPtoCSP}(C_t)$ as their response. On encrypted data C_1 and $\widetilde{C}_1$ corresponding to the same key k_1, the cloud can thus achieve $\overline{C}_1 \leftarrow \text{HE.Eval}(f_{\text{training}}, \widetilde{C}_1)$ which are updated system parameters decryptable by k_1.

For clarity, we describe the above processes as shown in Table 1.

Note that KS.DLPtoCSP$(\cdot)$ is a protocol that should be carried out by both the data provider and the cloud.

To make the updated coefficients homomorphically computable with data encrypted by the following providers, we can exploit the key switching scheme to re-encrypt it. Without loss of generality, the updated parameters under key k_i will be represented as $\overline{C}_i$. By means of $C_t \leftarrow \text{KS.CSPtoDLP}(\overline{C}_i)$ and $\widetilde{C}_{i+1} \leftarrow \text{KS.DLPtoCSP}(C_t)$, the cloud can obtain the re-encrypted coefficients $\widetilde{C}_{i+1}$ with the help of successive providers. After receiving C_{i+1} from the next provider, they can compute $\overline{C}_{i+1} \leftarrow \text{HE.Eval}(f_{\text{training}}, C_{i+1}, \widetilde{C}_{i+1})$ since both ciphertexts are encrypted by k_{i+1}. In consideration of the final parameters $\overline{C}_N$, the cloud needs to execute $C_t \leftarrow \text{KS.CSPtoDLP}(\overline{C}_N)$ with the last provider and then computes $M = (C_t^s \bmod 2^l)/2^n$ to restore the plain parameters.

The subsequent training and recovering processes are presented in Table 2.

The classification/regression process is straightforward that, on encrypted data $C_u \leftarrow \text{HE.Enc}(k_u, M_u)$ and system parameters $\widetilde{C}_u \leftarrow \text{KS.DLPtoCSP}(C_t)$ for $C_t = (2^n M_t)^b \bmod 2^l$, the cloud can homomorphically compute $\overline{C}_u \leftarrow \text{HE.Eval}(f_{\text{cla/reg}}, C_u, \widetilde{C}_u)$. By decrypting the received $\overline{C}_u$, the user obtains the classification/regression result such that $M_{\text{cla/reg}} \leftarrow \text{HE.Dec}(k_u, \overline{C}_u)$. This process can be found in Table 3.

5. Experiment Analysis

We drew support from the power load data of Chongqing Tongnan Electric Power Co., Ltd., dating from May 4 to May 10 in 2015, to verify the effectiveness of our training method. A short-term electrical load prediction model is also testified in virtue of 96 historical data pieces sampled during 4 consecutive days. The original machine learning model is exactly the same as that of [29], which has considered nothing about privacy. Our experiment environment is shown in Table 4.

Table 1: Initialization and first training.

Key generation		
Data providers		Cloud
$k_i \leftarrow \text{HE.KeyGen}(1^\kappa)$		$k_t \leftarrow \text{KS.KeyGen}(1^\kappa)$
First training		
Data provider 1		Cloud
Receives C_t	$\Leftarrow$	$C_t = (2^n M_t)^b \bmod 2^l$
$\widetilde{C}_1 \leftarrow \text{KS.DLPtoCSP}(C_t)$	$\Rightarrow$	Receives $\widetilde{C}_1$
$C_1 \leftarrow \text{HE.Enc}(k_1, M_1)$	$\Rightarrow$	Receives C_1
		$\overline{C}_1 \leftarrow \text{HE.Eval}(f_{training}, C_1, \widetilde{C}_1)$

Table 2: Subsequent training and recovering.

Subsequent training		
Data providers i		Cloud
Receives $\overline{C}_i$	$\Leftarrow$	$\overline{C}_i$
$C_t \leftarrow \text{KS.CSPtoDLP}(\overline{C}_i)$	$\Rightarrow$	Receives C_t
Data providers $i+1$		Cloud
Receives C_t	$\Leftarrow$	C_t
$\widetilde{C}_{i+1} \leftarrow \text{KS.DLPtoCSP}(C_t)$,	$\Rightarrow$	Receives $\widetilde{C}_{i+1}$
$C_{i+1} \leftarrow \text{HE.Enc}(k_{i+1}, M_{i+1})$	$\Rightarrow$	Receives C_{i+1}
		$\overline{C}_{i+1} \leftarrow \text{HE.Eval}(f_{\text{training}}, C_{i+1}, \widetilde{C}_{i+1})$
Recovering		
Data provider N		Cloud
Receives $\overline{C}_N$	$\Leftarrow$	$\overline{C}_N$
$C_t \leftarrow \text{KS.CSPtoDLP}(\overline{C}_N)$	$\Rightarrow$	Receives C_t
		$M_t \leftarrow (C_t^s \bmod 2^l)/2^n$

Table 3: Classification and regression.

Classification/regression		
User u	Cloud	
Receives C_t	$\Leftarrow$	$C_t = (2^n M_t)^b \bmod 2^l$
$\widetilde{C}_u \leftarrow \text{KS.DLPtoCSP}(C_t)$	$\Rightarrow$	Receives $\widetilde{C}_u$
$C_u \leftarrow \text{HE.Enc}(k_u, M_u)$	$\Rightarrow$	Receives C_u
Receives $\overline{C}_u$	$\Leftarrow$	$\overline{C}_u \leftarrow \text{HE.Eval}(f_{\text{cla/reg}}, C_u, \widetilde{C}_u)$
$M_{\text{cla/reg}} \leftarrow \text{HE.Dec}(k_u, \overline{C}_u)$		

Table 4: Experiment environment.

CPU	OS	RAM (GB)	Programming language
i5-10210 U 1.60 GHz	Win10 64-bit	16	Python

To simulate the scenario of multiparty machine learning, we divide the data into three parts and realize the training process corresponding to 3 different keys in HE. To prove that our method is not harmful to the accuracy of the trained network, as is shown in Figure 1, we compared the prediction result directly achieved via original model (without privacy-preserving) with that of ours (privacy-preserving scheme). Figure 1 illustrates that the two results are completely consistent.

The experimental results are shown in Table 5; our scheme can perform encryption training and prediction for multiple data providers in general machine learning. As for the efficiency of training and prediction, our scheme is 73578 and 12000 times slower than its plain version. Nevertheless, since the server is always powerful on computational capacity and the data providers only have to carry out trivial multiplications over $R^{6\times6}$, our scheme is practical in cloud environments. Moreover, if the accuracy is tolerable, we can shorten the ciphertext to make it more efficient.

In terms of communication overheads, encrypted data for training or prediction are 18 times larger than plain messages. In each iteration, the cloud should also exchange the ciphertexts of system parameters with two successive data providers, which are also 18 times of original

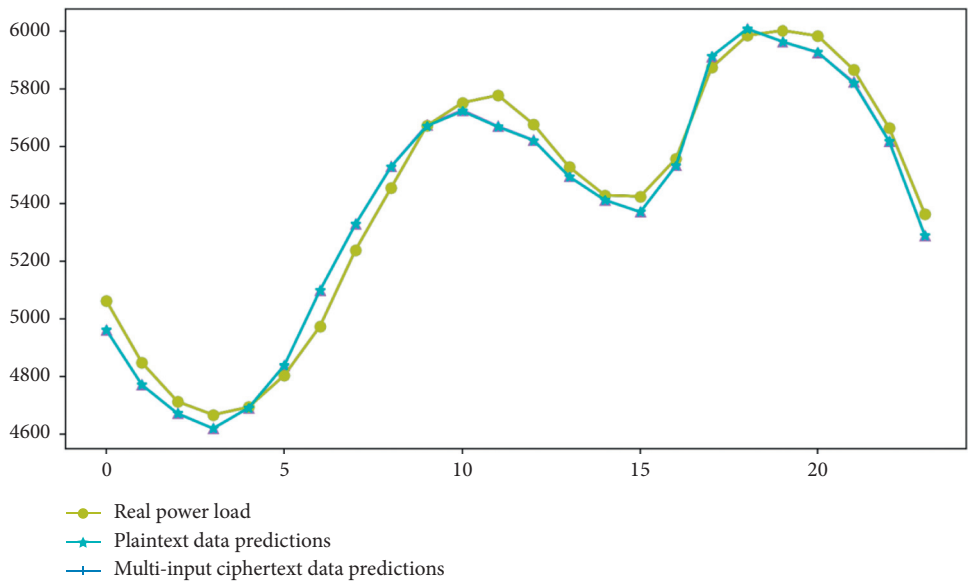

Figure 1: Comparison of prediction results.

Table 5: Computational efficiency.

Model	Training time	Predicted time	Epochs
Original model	0.19 s	4 ms	400
Our model	233.33 min	48 s	400

coefficients. Considering that the expansion rate is not big and system parameters are quite limited, the communication burden causes just little performance degradation.

6. Conclusions

We presented a privacy-preserving machine learning method that works over multiple data providers in this paper. Thanks to the hardness of the conjugate search problem, data can be homomorphically processed for training or classification/regression under the same key. It is worth mentioning that we solved the intrinsic conflict between IND-CPA security and homomorphic comparision (without decryption), by specifically encoding the data which is allowed to be compared. To support training among multiple data providers, a key switching technology is also proposed based on the difficulty of the discrete logarithm problem and Lagrange theorem, which evaded the necessity of multikey homomorphic computation. Experiment illustrated that the accuracy of machine learning cannot be affected by the privacy capability of our scheme. The expansion rate of computation/communication complexity is small enough, which makes the scheme practical in cloud environments.

Data Availability

Our dataset comes from Chongqing Tongnan Electric Power Co., Ltd. (telephone: 023-44559308; official website: http://www.12398.gov.cn/html/information/753078881/753078881201200006.shtml).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank Darong Huang and Yang Liu for their comments and suggestions. This work was supported in part by the National Natural Science Foundation of P.R. China under Grants 61573076, 61703063, and 61903053, the Science and Technology Research Project of the Chongqing Municipal Education Commission of P.R. China under Grants KJZD-K201800701, KJ1705121, and KJ1705139, and the Program of Chongqing Innovation and Entrepreneurship for Returned Overseas Scholars of P.R. China under Grant cx2018110.

Supplementary Materials

data.txt： contains the data used for training and prediction in this research; it is from the power load data of Chongqing Tongnan Electric Power Co., Ltd., dating from May 4 to May 10 in 2015. (*Supplementary Materials*)

References

[1] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *Journal of Big Data*, vol. 6, no. 1, pp. 1–18, 2019.

[2] F. F. Ting, Y. J. Tan, and K. S. Sim, "Convolutional neural network improvement for breast cancer classification," *Expert Systems with Applications*, vol. 120, pp. 103–115, 2019.

[3] B. Lutnick, B. Ginley, D. Govind et al., "An integrated iterative annotation technique for easing neural network training in medical image analysis," *Nature Machine Intelligence*, vol. 1, no. 2, pp. 112–119, 2019.

[4] L. Zhao, Q. Wang, Q. Zou et al., "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1486–1500, 2019.

[5] Q. Feng, D. He, Z. Liu, H. Wang, and K.-K. R. Choo, "SecureNLP: a system for multi-party privacy-preserving natural language processing," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3709–3721, 2020.

[6] N. Agrawal, S. A. Shahin, M. J. Kusner et al., "QUOTIENT: two-party secure neural network training and prediction," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1231–1247, London, UK, November 2019.

[7] S. Sayyad, "Privacy Preserving Deep Learning Using Secure Multiparty Computation," in *Proceedings of the 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 139–142, Coimbatore, September 2020.

[8] M. Izabachène, R. Sirdey, and M. Zuber, "Practical Fully Homomorphic Encryption for Fully Masked Neural Networks," in *Proceedings of the International Conference on Cryptology and Network Security*, pp. 24–36, Fuzhou, China, October 2019.

[9] T. N. Yelina, S. V. Bezzateev, and V. A. Mylnikov, "The Homomorphic Encryption in Pipelines Accident Prediction by Using Cloud-Based Neural Network," in *Proceedings of the 2019 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF)*, pp. 1–4, IEEE, Saint Petersburg, Russia, June 2019.

[10] F. Ö. Çatak, "Secure multi-party computation-based privacy preserving extreme learning machine algorithm over vertically distributed data," in *Proceedings of the International Conference on Neural Information Processing*, pp. 337–345, Springer, Istanbul, Turkey, November 2015.

[11] D. Reich, A. Todoki, R. Dowsley et al., "Privacy-preserving classification of personal text messages with secure multiparty computation," *Advances in Neural Information Processing Systems*, pp. 3757–3769, 2019.

[12] R. Gilad-Bachrach, N. Dowlin, K. Lain et al., "Cryptonets: applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the International Conference on Machine Learning*, pp. 201–210, New York, NY, USA, June 2016.

[13] Y. Aono, T. Hayashi, L. Wang et al., "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[14] F. Bourse, M. Minelli, M. Minihold et al., "Fast Homomorphic Evaluation of Deep Discretized Neural Networks," in *Proceedings of the Annual International Cryptology Conference*, pp. 483–512, Springer, Santa Barbara, CA, USA, August 2018.

[15] I. Chillotti, N. Gama, M. Georgieva et al., "Faster Fully Homomorphic Encryption: Bootstrapping in Less than 0.1 Seconds," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 3–33, Springer, Hanoi, Vietnam, December 2016.

[16] Z. Brakerski, "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP," in *Proceedings of the Annual Cryptology Conference*, pp. 868–886, Springer, Santa Barbara, CA, USA, August 2012.

[17] J. H. Cheon, A. Kim, M. Kim et al., "Homomorphic Encryption for Arithmetic of Approximate Numbers," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 409–437, Springer, Hong Kong, China, December 2017.

[18] H. Chen, W. Dai, M. Kim et al., "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," in *Proceedings of the Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 395–412, London, UK, November 2019.

[19] F. Bu, Y. Ma, Z. Chen et al., "Privacy Preserving Back-Propagation Based on BGV on Cloud," in *Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications*, pp. 1791–1795, IEEE, Munich, Germany, September 2015.

[20] K. Sarpatwar, N. K. Ratha, K. Nandakumar et al., "Privacy Enhanced Decision Tree Inference," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 34-35, 2020.

[21] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory*, vol. 6, no. 3, pp. 1–36, 2014.

[22] F. Boemer, R. Cammarota, D. Demmler et al., "MP2ML: a mixed-protocol machine learning framework for private inference," in *Proceedings of the 15th International Conference on Availability*, pp. 1–10, Boras, Sweden, March 2020.

[23] F. Boemer, A. Costache, R. Cammarota et al., "nGraph-HE2: a high-throughput framework for neural network inference on encrypted data," in *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pp. 45–56, London, UK, November 2019.

[24] M. Azraoui, M. Bahram, B. Bozdemir et al., "SoK: Cryptography for Neural Networks," in *Proceedings of the IFIP International Summer School on Privacy and Identity Management*, pp. 63–81, Springer, 2019.

[25] E. Shishniashvili, L. Mamisashvili, and L. Mirtskhulava, "Enhancing IoT security using multi-layer feedforward neural network with tree parity machine elements," *International Journal of Simulation--Systems, Science & Technology*, vol. 21, no. 2, pp. 371–375, 2020.

[26] M. S. Sruthi and A. A. TV, "Protected entry design for data encryption and decryption using big data in cloud," *International Journal of Emerging Technology and Innovative Engineering*, vol. 5, no. 3, pp. 1–7, 2019.

[27] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, pp. 1219–1234, NY, New York, May 2012.

[28] J. Li, X. Kuang, S. Lin, X. Ma, and Y. Tang, "Privacy preservation for machine learning training and classification based on homomorphic encryption schemes," *Information Sciences*, vol. 526, pp. 166–179, 2020.

[29] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991.

[30] S. F. Sun, X. Yuan, J. K. Liu et al., "Practical backward-secure searchable encryption from symmetric puncturable encryption," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 763–780, Toronto, Canada, October 2018.

[31] L. Gu and S. Zheng, "Conjugacy systems based on nonabelian factorization problems and their applications in cryptography," *Journal of Applied Mathematics*, vol. 2014, pp. 1–10, Article ID 630607, 2014.

[32] O. Çetin, F. Temurtaş, and Ş. Gülgönül, "An application of multilayer neural network on hepatitis disease diagnosis using approximations of sigmoid activation function," *Dicle Medical Journal/Dicle Tip Dergisi*, vol. 42, no. 2, pp. 150–157, 2015.

[33] K. Zhang, K. Peng, S. X. Ding, Z. Chen, and X. Yang, "A correlation-based distributed fault detection method and its application to a hot tandem rolling mill process," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 3, pp. 2380–2390, 2020.

[34] K. Zhang, H. Hao, Z. Chen, S. X. Ding, and K. Peng, "A comparison and evaluation of key performance indicator-based multivariate statistics process monitoring approaches," *Journal of Process Control*, vol. 33, pp. 112–126, 2015.

[35] N. P. Smart, "The discrete logarithm problem on elliptic curves of trace one," *Journal of Cryptology*, vol. 12, no. 3, pp. 193–196, 1999.

[36] G. Panti, "A general Lagrange theorem," *American Mathematical Monthly*, vol. 116, no. 1, pp. 70–74, 2009.