electronics

MDPI

Article

Secure and Flexible Privacy-Preserving Federated Learning Based on Multi-Key Fully Homomorphic Encryption

Jiachen Shen, Yekang Zhao, Shitao Huang and Yongjun Ren *

School of Computer Science, School of Cyber Science and Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China
* Correspondence: 002315@nuist.edu.cn

Abstract: Federated learning avoids centralizing data in a central server by distributing the model training process across devices, thus protecting privacy to some extent. However, existing research shows that model updates (e.g., gradients or weights) exchanged during federated learning may still indirectly leak sensitive information about the original data. Currently, single-key homomorphic encryption methods applied in federated learning cannot solve the problem of privacy leakage that may be caused by the collusion between the participant and the federated learning server, whereas existing privacy-preserving federated learning schemes based on multi-key homomorphic encryption in semi-honest environments have deficiencies and limitations in terms of security and application conditions. To this end, this paper proposes a privacy-preserving federated learning scheme based on multi-key fully homomorphic encryption to cope with the potential risk of privacy leakage in traditional federated learning. We designed a multi-key fully homomorphic encryption scheme, mMFHE, that encrypts by aggregating public keys and requires all participants to jointly participate in decryption sharing, thus ensuring data security and privacy. The proposed privacy-preserving federated learning scheme encrypts the model updates through multi-key fully homomorphic encryption, ensuring confidentiality under the CRS model and in a semi-honest environment. As a fully homomorphic encryption scheme, mMFHE supports homomorphic addition and homomorphic multiplication for more flexible applications. Our security analysis proves that the scheme can withstand collusive attacks by up to $N-1$ users and servers, where N is the total number of users. Performance analysis and experimental results show that our scheme reduces the complexity of the NAND gate, which reduces the computational load and improves the efficiency while ensuring the accuracy of the model.

Keywords: privacy preservation; federated learning; multi-key fully homomorphic encryption

check for updates

Citation: Shen, J.; Zhao, Y.; Huang, S.; Ren, Y. Secure and Flexible Privacy-Preserving Federated Learning Based on Multi-Key Fully Homomorphic Encryption. *Electronics* **2024**, *13*, 4478. https://doi.org/10.3390/electronics13224478

Academic Editor: Subir Halder

Received: 9 October 2024
Revised: 29 October 2024
Accepted: 13 November 2024
Published: 14 November 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The training of machine learning models usually requires the centralization of dispersed data to a single server or data center, a practice that not only presents technical challenges, such as communication costs and latency for large data transfers, but more importantly also raises significant concerns about privacy and data protection [1,2]. With the popularity of smartphones and various smart devices, a significant volume of individual data have been accumulated on the devices, which will be of great value if we can make use of them to enhance the quality of service and user experience under the premise of protecting privacy [3]. Google's research team developed the federated learning (FL) concept in response to these challenges [4,5]. The fundamental premise of FL is that every device (or node) trains a model locally with the data it holds, and only sends updates to the model (e.g., gradient or parameter updates) to the server. The server is tasked with aggregating the aforementioned updates, updating the global model, and subsequently disseminating the enhanced model to the participating devices. In this way, the data are not transmitted from the local device, which serves to safeguard the user's privacy. Since

its introduction in 2016, FL has evolved from a preliminary concept to an active research field and has found practical applications in healthcare [6,7], meta-universe [8,9], IoT [10], and many other areas.

However, it has been shown that in FL, model updates (e.g., gradients or weights) exchanged during model training may indirectly disclose confidential information about the original data, even though the data are retained locally [11,12]. For this reason, privacy-preserving federated learning (PPFL), as an extension of FL, further reduces potential privacy risks by introducing techniques such as differential privacy (DP), secure multi-party computation (SMPC), and homomorphic encryption (HE). DP protects the privacy of personal data by adding noise to perturb the training data or model parameters to ensure that modifications to individual data samples do not significantly affect the output results [13]. Nevertheless, the incorporation of noise inevitably impedes the rate of convergence of the model and diminishes the accuracy of the model within the context of aggregation [14]. SMPC, as a cryptographic method, permits the participation of multiple parties in the computation of a function while maintaining the confidentiality of their respective input data; however, SMPC usually requires complex cryptographic operations and frequent network communication, which may lead to higher computational costs and delays, especially in large-scale systems.

The core advantage of HE is its ability to perform computation while the data remain encrypted, which means that the data can be securely processed without decryption. And compared with DP and SMPC, HE can have a small communication overhead while maintaining model accuracy, but the traditional single-key HE in which all participants share the same encryption and decryption keys presents a significant risk to the privacy and security of the other participants in the event that a malicious user conspires with the server. On the other hand, extant privacy-preserving FL (PPFL) schemes based on multi-key homomorphic encryption (MKHE) in semi-honest environments are constrained in terms of security and application conditions. Specifically, in extreme cases, at least two participants (i.e., $k < N - 1$) are required not to collude with the attacker, where N is the total number of users and k is the number of users who collude with the server. The aforementioned limitations of existing PPFL schemes based on MKHE in semi-honest environments can be attributed to two primary factors. Firstly, the attacker's ability to prevent the leakage of private data is insufficient in terms of security. Secondly, they only support homomorphic addition but not homomorphic multiplication (or they require trustworthy hardware support to perform multiplication), which imposes many restrictions on the FL algorithms used in applications and is not flexible enough.

To address this, we propose a multi-key fully homomorphic encryption (MKFHE) scheme derived from GSW13 [15], named mMFHE, to enhance privacy protection in FL. As an FHE scheme, mMFHE is capable of supporting both the homomorphic addition and multiplication of encrypted data and ensures security by requiring encryption with an aggregated public key and decryption sharing. Additionally, mMFHE allows for embedding multiple plaintext messages within a single ciphertext, thereby improving the efficacy of the encryption and decryption procedures. In this paper, we present a PPFL scheme based on mMFHE. In this scheme, model updates are encrypted using homomorphic encryption before being shared for aggregation. Furthermore, the aggregated result can only be decrypted through the collaboration of all those who contributed to its generation. This renders the scheme resilient to assaults from participants and collusion attacks conducted by participants in conjunction with the server. Finally, we performed experiments based on simulations using actual data to assess the efficacy of mMFHE and the proposed PPFL scheme. Specifically, our contributions are the following:

1. We propose a multi-key FHE scheme, mMFHE, that ensures security by requiring encryption with an aggregated public key and decryption sharing. It supports the encryption, decryption, and homomorphic evaluation of multi-bit messages in a single operation, reducing the complexity of NAND gates and enhancing efficiency.

Additionally, it allows for homomorphic addition and multiplication on ciphertexts, increasing application flexibility.

2. We introduce a PPFL scheme based on mMFHE. This scheme employs multi-key FHE to encrypt model updates, thereby ensuring their confidentiality within a semi-honest environment. The proposed scheme exhibits robustness against attacks by participants and collusion attacks, even when involving up to $k = N - 1$ participants in conjunction with the server.
3. We analyze the security of the PPFL scheme based on mMFHE and perform a comparative analysis to evaluate its efficiency. The results of simulation-based experiments conducted on actual datasets demonstrate that our proposed PPFL scheme can reduce computational load while maintaining accuracy, thereby ensuring efficiency.

We introduce the research status of PPFL and the existing PPFL schemes based on HE in Section 2. Section 3 introduces some prerequisite knowledge related to the mMFHE scheme we proposed. Section 4 describes in detail the mMFHE and PPFL schemes based on mMFHE proposed by us. Section 5 presents a detailed security analysis of the PPFL scheme, and the performance analysis and simulation experiments are discussed in Section 6. We discuss some problems currently faced by PPFL based on mMFHE in Section 7 and give a summary of our work and future prospects in Section 8.

2. Related Work

2.1. PPFL

FL is a decentralized machine learning paradigm, first pioneered by McMahan et al. [4] in 2016, that allows several participants to collaboratively train a joint machine learning model without having to exchange or centrally store their individual datasets. The challenge of data silos, which is common in centralized machine learning, is effectively addressed by this approach. Since then, a number of efforts have emerged to improve FL in a variety of ways, from the design of the underlying algorithms to performance optimization and security enhancements. In the same year, the FedAvg algorithm was proposed [5], which allows each participant to train a model independently using their local data and then send their respective model parameters to a central server for averaging. This process is iterative until the model is in a state of convergence. FedAvg is the original and most basic FL algorithm and is widely used in a variety of application scenarios. In 2018, Li et al. [16] suggested the FedProx algorithm, in which a regular term is added to FedAvg in order to deal with the heterogeneity of the devices in the system. In 2020, Asad et al. [17] presented the FedOpt algorithm, which introduces more advanced optimization strategies such as adaptive gradient methods to optimize the updating and aggregation process of the global model. In 2023, Zhang et al. [18] proposed an Adaptive Locally Aggregated FL (FedALA) approach for personalizing client models in FL by capturing the information required in the global model. In 2024, A Raft consensus protocol based on Cauchy Reed–Solomon (CRS) codes was proposed for adaptive data maintenance in the metaverse of Yu et al. [19]. The protocol reduces the data storage requirements of nodes utilizing code erasure technology.

In FL, model updates (e.g., gradients or weights) exchanged during model training may indirectly disclose confidential information about the original data, despite the data being retained locally [11,12]. For this reason, privacy-preserving federated learning, as an extension of federated learning, further reduces potential privacy risks by introducing encryption and anonymization techniques. Most of the existing work related to privacy-preserving federated learning employs techniques such as DP [20–22], SMPC [23], and HE [24,25].

The DP process ensures the confidentiality of personal data by introducing noise to disrupt either the training data or model parameters, thereby ensuring that alterations to specific data samples do not significantly influence the output. In 2020, Wei et al. [26] put forth a new framework leveraging the principles of DP, which effectively mitigates the threat of information leakage by embedding artificial noise in the client-side parameters prior to aggregation. Concurrently, Truex et al. [27] integrated local differential privacy

(LDP) into federated learning, providing formal privacy guarantees and developing an innovative federated learning system, LDP-Fed. In 2023, He et al. advanced this field by introducing a local differential privacy scheme tailored to address the challenges of fine-grained range differences in weights across different layers of FL models, as well as the issue of privacy budget accumulation leading to budget explosion. Their method, ACS-FL, used adaptive cropping, weight compression, and the reorganization of parameters to train clustered FL models on disparate IoT data. However, the noise introduced by DP inherently impairs the speed of aggregation and diminishes the accuracy of the model during aggregation [28].

Compared to DP, SMPC allows several participants to collaboratively calculate a function without exposing their respective inputs through cryptographic methods and has a lower impact on model accuracy. Bonawitz et al. [25] were the first to introduce SMPC into federated learning for secure aggregation, constructing a confidentiality calculation process in combination with dual masking. In 2020, Li et al. [29] combined a single masking mechanism and a chained communication mechanism and proposed a new PPFL framework on the basis of chained SMPC, which improved the aggregation efficiency to some extent. And for the combined problem of privacy inference attacks and poison attacks that federated learning systems may suffer from, Gehlhar et al. [30] proposed an SMPC-based framework, SafeFL, aiming to evaluate the effectiveness of the FL technique in solving privacy inference and poison attacks. However, SMPC usually requires complex cryptographic operations and frequent network communication, which may lead to high computational cost and latency, especially in large-scale systems.

2.2. HE-Based PPFL

The core strength of HE is its ability to perform computation while the data remain encrypted, which means that the data can be securely handled without being decrypted. Zhang et al. [31] proposed a Privacy-Enhanced FL (PEFL) scheme that protects gradients on untrusted servers by encrypting the local gradients of the participant using the Paillier homomorphic cryptosystem. Li et al. [32] constructed an FL security framework on the basis of the threshold Paillier cryptosystem for use in IoT environments [33] and thereby mitigated the negative impact of untrusted users. He et al. [34] presented a privacy-protecting and low-latency FL scheme tailored for edge computing [35], which ensures the privacy of terminal devices by transferring parameters encrypted with an enhanced version of the Paillier homomorphic encryption algorithm, thereby avoiding the transmission of raw data to the edge node. Despite these advancements, this approach relies on single-key homomorphic encryption algorithms, where all devices utilize the same encryption and decryption keys. This architecture poses a significant risk to privacy and security, particularly in scenarios where a malicious participant conspires with the server, potentially compromising the confidentiality of other participants. To better implement the privacy preservation requirements in FL systems in multi-user environments, several works have introduced MKHE. Cai et al. [36] designed a TEE-based MKHE cryptosystem (EMK-BFV) to facilitate PPFL and optimize the operational performance. Ma et al. [37] designed an innovative PPFL scheme based on the multi-key CKKS algorithm learning scheme xMK-CKKS, while Walskaar et al. [38] proposed a more efficient federated learning scheme based on xMK-CKKS by incorporating the Flower framework. Zhang et al. [39] proposed a PPFL scheme VPFL based on the BCP cryptosystem, which is capable of verifying the user's identity and data integrity in a multi-key environment. However, all of the above works are deficient in terms of security; the schemes in reference [36–38] must have at least two participants who do not conspire with the server ($k < N - 1$) to preserve that the honest party's privacy is not compromised in the face of a user–server conspiracy attack in semi-honest environments, whereas [39] presents a two-server scheme that requires the two servers to not be complicit in order to ensure security. In addition, [37–39] only support homomorphic addition, while [36] supports multiplication, but it is essentially a direct-to-plaintext multiplication in a trusted execution environment, rather than homomorphic

multiplication, which requires trusted hardware support; therefore, the above schemes will impose many limitations on the federated learning algorithms used in terms of their applications, and they are not flexible enough.

Table 1 summarizes the above HE-based PPFL scheme and compares it with our proposed scheme, mainly including whether it supports homomorphic multiplication and the security in the face of collusion attacks between users and servers, where k is the number of users who collude with the server, and N is the total number of users participating in FL.

Table 1. Comparison of HE-based PPFL.

Scheme	Base	Homomorphic Addition	Homomorphic Multiplication	Security Against Collusion Attacks
[31,32,34]	Paillier	Yes	No	No
[36]	BFV	Yes	Support for plaintext multiplication in TEE.	$k < N-1$
[37]	MK-CKKS	Yes	No	$k < N-1$
[38]	MK-CKKS	Yes	No	$k < N-1$
[39]	BCP	Yes	No	Requires that the two servers cannot collude
Ours	mMFHE	Yes	Yes	$k = N-1$

3. Preliminaries

3.1. Definitions

For $n \in N$, we denote the set $\{1, \cdots, n\}$ by $[n]$. For any real number $x \in \mathbb{R}$, we define $\lfloor x \rfloor$ as the greatest integer less than or equal to x, and $\lfloor x \rceil := \lfloor x + \frac{1}{2} \rfloor$ as the integer closest to x. Matrices are represented by bold, uppercase letters: $\mathbf{A}$. We use ":=" for deterministic assignments.

Definition 1. *For a distribution $\{\chi_n\}_{n\in N}$ based on integers, if it satisfies $\mathsf{Pr}[|x| \geq B] = negl(\lambda)$, where $negl(\lambda)$ is a negligible function, then we call the distribution B-bounded.*

Theorem 1. *For a range of random variables $x_i (i \in N)$, if it obeys a $B-$bounded distribution, then the random variable $x = \frac{1}{N}\sum_{i=1}^{N} x_i$ also obeys the $B-$bounded distribution.*

Definition 2. *The statistical distance between two distributions A and B over a finite field Ω is $\Delta(X,Y) \overset{\text{stat}}{=} \frac{1}{2}\sum_{t\in\Omega} \mid A(t) - B(t) \mid$. A negligible $\Delta(A,B)$ implies $A \overset{\text{state}}{\approx} B$.*

Definition 3 (Learning with Errors, LWE)**.** *We consider the case of a secret vector $\boldsymbol{s}$ belonging to the discrete cube Z_q^n. The LWE distribution $Z_q^n \times Z_q$ over A_s, χ is established through the following definition: uniformly sample $\boldsymbol{a} \in Z_q^{n\times m}$ and $\boldsymbol{e} \leftarrow \chi$, and then output the pair $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{a}\cdot\boldsymbol{s} + \boldsymbol{e}) mod\ q$.*

Definition 4 (*search.LWE$_{n,q,\chi,m}$*)**.** *For the secret vector $\boldsymbol{s} \in Z_q^n$ and given m independent samples $(\boldsymbol{a_i}, \boldsymbol{b_i}) \in Z_q^n \times Z_q$ to recover $\boldsymbol{s}$, these are selected from the distributions A_s, χ.*

Definition 5 (*Decision.LWE$_{n,q,\chi,m}$*)**.** *Given m independent samples $(\boldsymbol{a_i}, \boldsymbol{b_i}) \in Z_q^n \times Z_q$, these samples are selected from the following two distributions: (1) from A_s, χ; (2) drawn uniformly from $Z_q^n \times Z_q$. The advantage of being able to distinguish between these two types of selection is negligible.*

Definition 6 (Some-are-errorless LWE)**.** *Let $q \geq 1$, $n > 0$, and χ' be a distribution of errors over R. Define $T_q = \{0, \frac{1}{q}, \cdots, \frac{q-1}{q}\}$, where $q \in Z$. The distribution $A'_{s,\chi}$ over $T_q^n \times T_q$ is defined by*

uniformly selecting $\boldsymbol{a} \in T_q^n$ *and* $\boldsymbol{e} \leftarrow \chi'$*, and outputting* $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e})$*. The some-are-errorless LWE problem concerns the distinction between two scenarios:*

(1) All samples are uniformly selected from $T_q^n \times T_q$*.*

(2) A random secret vector $\boldsymbol{s} \in T_q^n$ *is uniformly chosen, with the first* l *samples drawn from* $A'_{s,0}$ *and the remaining samples drawn from* $A'_{s,\chi}$*. In other words, the first* l *samples are of the form* $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{s})$*, which are errorless, while the remaining samples* $(\boldsymbol{a_i}, \boldsymbol{b_i} = \boldsymbol{a_i} \cdot \boldsymbol{s} + \boldsymbol{e_i})$ *for each* $i > l$ *introduce a small error* $\boldsymbol{e}_i$*.*

Theorem 2. *For any* $n, l, q \geq 1 (l \ll n)$*, and an error distribution* χ'*, there exists a problem ranging from* $LWE_{n-1,q,\chi}$ *to* $LWE_{n,q,\chi}$*, a variant some-are-errorless problem. LWE polynomials reduce the success advantage of the problem by up to* $p - n$*, where* p *traverses all* q *prime factors. LWE polynomials, which reduces the success advantage of the problem by at most* $\sum_p p^{-n}$*, where* p *iterates over all the prime factors of* q*. The proof can be found in reference [40].*

Definition 7 (Key Homomorphism)**.** *Key homomorphism in cryptography refers to a property of cryptographic algorithms where transformations on keys can be correlated directly with transformations on ciphertexts [41]. This means that operations performed on keys can have equivalent operations on ciphertexts that preserve the structure of the data being encrypted.*

For example, if there is a cryptographic system that supports key homomorphism, and two keys k_1 and k_2, performing an operation on these keys (like addition or multiplication) to produce a new key k_3 will correlate with a similar operation on ciphertexts encrypted with k_1 and k_2 to produce a new ciphertext that would decrypt correctly under k_3.

In a system that supports multi-key homomorphism, there are several keys $k_1, k_2, \cdots, k_n$. Operations performed across these keys, such as combinations or aggregations (e.g., summation, multiplication), yield a new key k_{new}. The crucial aspect is that an equivalent operation on ciphertexts encrypted with $k_1, k_2, \cdots, k_n$ results in a new ciphertext that, when decrypted with k_{new}, reveals a data transformation that precisely reflects the key operations. This process can support complex data interactions securely and efficiently, providing significant flexibility in distributed cryptographic systems.

3.2. GSW13

To elucidate the distinctions between our proposed mMFHE scheme and prior works, we provide a detailed exposition of the GSW13. The GSW13, a scheme predicated on the LWE problem, is distinguished by its minimal ciphertext expansion during homomorphic operations. We employ the following formal representation for clarity:

Remark 1. *Let us consider the positive integers* m, m', n*, and* q *(with* $m > n\lceil logq \rceil$*) and a matrix* $\mathbf{T} \in \mathbb{Z}_q^{n \times m}$*. It can be demonstrated that there exists a matrix* $\mathbf{G}$*, belonging to the set of* $n \times m$ *matrices over the field of integers modulo* q*, and an inverse function* $\mathbf{G}^{-1}$*, such that* $\mathbf{G^{-1}}(\mathbf{T})$ *is a binary matrix. Furthermore, it can be demonstrated that the matrix* $\mathbf{GG^{-1}}(\mathbf{T})$ *is equal to* $\mathbf{T}$*. The multiplication of a matrix by* $\mathbf{G}$ *results in a bitwise combination of its elements. In contrast, the inverse of* $\mathbf{G}$*, denoted* $\mathbf{G^{-1}}$*, facilitates the bitwise decomposition of these elements. The operational specifics of the GSW13 are outlined as follows:*

- *GSW.Setup*$(1^\lambda, 1^d)$*: Initiate the setup by defining the lattice dimension* $n = n(\lambda, d)$*, where* λ *is the security parameter, and* d *is an integer specifying the maximum circuit depth permissible. Select a noise distribution* $\chi = \chi(\lambda, d)$*, bounded by* B_χ*, and determine a modulus* q *as* $q = B_\chi 2^{\omega(d\lambda log\lambda)}$*. This configuration is chosen to satisfy the Learning With Errors problem* $LWE_{n-1,q,\chi}$*. Set* $m = nlogq + \omega(log\lambda)$ *as the parameter defining the matrix dimensions.*
- *GSW.KeyGen: Generate a uniform random matrix* $\mathbf{B} \in \mathbb{Z}_q^{(n-1)\times m}$ *and a vector* $\mathbf{s}$ *in* $\mathbb{Z}_q^{n-1}$*. Compute the vector* $\mathbf{b}$ *as* $\mathbf{b} = \mathbf{sB} + \mathbf{e}$*, where* $\mathbf{e}$ *is an error vector sampled from a discrete*

distribution over $\mathbb{Z}_q$. *The public key is then given by* $\mathbf{A} = \begin{pmatrix} \mathbf{B} \\ \mathbf{b} \end{pmatrix} \in \mathbb{Z}_q^{n \times m}$, *and the private key is* $sk = \mathbf{t} = (-\mathbf{s}, 1) \in \mathbb{Z}_q^n$.

- *GSW.Encrypt: For a plaintext bit message* μ, *construct a uniform random matrix* $\mathbf{R} \in \{0,1\}^{m \times m}$, *and output the ciphertext* $\mathbf{C} = \mathbf{AR} + \mu\mathbf{G}$.
- *GSW.Decrypt: Define the vector* $\mathbf{w}$ *as consisting of zeros, except for the end position set to* $\lceil q/2 \rceil$. *For a given ciphertext* $\mathbf{C}$, *compute* $v = \mathbf{tCG^{-1}}(\mathbf{w^T}) \approx \mu\lceil q/2 \rceil$. *The decryption yields 0 if* v *is closer to 0 than to* $\lceil q/2 \rceil$, *and 1 otherwise.*
- *GSW.Evaluation: Define homomorphic operations as follows:*

$$\mathbf{ADD}(\mathbf{C}_1, \mathbf{C}_2)\text{: } \textit{Output } \mathbf{C}_1 + \mathbf{C}_2 \in \mathbb{Z}_q^{n \times m}.$$
$$\mathbf{MULT}(\mathbf{C}_1, \mathbf{C}_2)\text{: } \textit{Output } \mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2) \in \mathbb{Z}_q^{n \times m}.$$
$$\mathbf{NAND}(\mathbf{C}_1, \mathbf{C}_2)\text{: } \textit{Output } \mathbf{G} - \mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2).$$

For a more comprehensive analysis and construction details of $\mathbf{G}$ *and* $\mathbf{G}^{-1}$, *readers are referred to additional literature [15].*

4. FL Scheme Based on mMFHE

This section presents our MKFHE scheme, which is based on the key homomorphism of GSW13 under the CRS model, as well as the multi-bit FHE scheme, which is inspired by GSW13 presented by Li et al. [42], and proves that our scheme also satisfies the key linear homomorphism.

4.1. mMFHE

Assuming a CRS model and given the security parameters λ, we set t to the quantity of secret keys and the total number of bits in the message. The i-th participant is designated P_i, $i \in [N]$, and N is the number of participants. It is stipulated that each participant has t messages $\mu_j \in \{0,1\}$, $j \in [t]$. We now give the formal details.

- Setting parameters: $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda, 1^L)$:
  $\mathsf{Setup}(\cdot)$ takes as input the safety parameter λ and the maximum depth L of the circuit. The mode is $q = q(\lambda)$, the dimension of the lattice $n = n(\lambda)$, $m = m(\lambda, L) = O(n \log q)$, and the distribution of the errors $\chi = \chi(\lambda, d)$ such that $(m, n, q, \chi) - LWE$ With the assumption that the security of at least 2^λ is achieved against a known attack, a uniformly randomized matrix $\mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m}$ (as the common string) is chosen. Furthermore, let $l = \lfloor \log q \rfloor + 1$, $M = (n + t) \cdot l$ and output $\mathbf{params} = (n, q, \chi, m)$, $\mathbf{B}$.
- Key generation: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{params})$: For the j-th message μ_j of the i-th participant P_i, select the sample $\mathbf{a}_j^T = (a_{j,1}, \cdots, a_{j,n}) \in \mathbb{Z}_q^{1 \times n}$ and output $\mathsf{sk}_j := \mathbf{s}_j = (\mathbf{I}_j \mid -\mathbf{a}_j^T)^T \in \mathbb{Z}_q^{(n+t) \times 1}$. The important thing to note here is that $\mathbf{v}_j = \mathsf{PowerOf2}(\mathbf{s}_j)$. Most importantly, the private key matrix $\mathbf{sk}_i := \boldsymbol{S}_i = [\mathsf{sk}_1, \cdots, \mathsf{sk}_t] = [\mathbf{s}_1, \cdots, \mathbf{s}_t] \in \mathbb{Z}_q^{(n+t) \times t}$; choose $\mathbf{e}_j \leftarrow \chi^{m \times 1}$, $j \in [t]$, then calculate $\mathbf{b}_j = \mathbf{B} \cdot \mathbf{a}_j + \mathbf{e}_{(p \bmod q)}$, and output $\mathbf{pk}_i := \boldsymbol{A}_i = [\mathbf{b}_1 \mid \cdots \mid \mathbf{b}_t \mid \mathbf{B}] \in \mathbb{Z}_q^{m \times (n+t)}$, where $\mathbf{pk}$ has size $\mathcal{O}(nm \cdot \log^2 q)$. Finally, we observe that $\boldsymbol{A} \cdot \mathbf{s}_i = \mathbf{e}_i$ and $\boldsymbol{A} \cdot \boldsymbol{S} = [\mathbf{e}_1, \cdots, \mathbf{e}_t]$.
- Encryption: $\mathbf{C} \leftarrow \mathsf{Enc}(\mathsf{params}, \mathsf{pk}, \mathbf{M})$: To encrypt a t-bit message, where each bit u_j belongs to the set $0, 1$ and $j \in [t]$, we commence by sampling a uniform matrix $\mathbf{R}$ from the set $\mathbf{R} \leftarrow \{0,1\}^{m \times M}$. Subsequently, the individual bits of the message are embedded into a diagonal matrix $\mathbf{U}$, expressed as $\mathbf{U} = \mathrm{diag}(u_1, \cdots, u_t)$. This matrix $\mathbf{U}$ serves as the basis for constructing the plaintext matrix. The precise method for forming the plaintext matrix will be elaborated as follows:

$$\mathbf{M} = \begin{pmatrix} \mathbf{U}_{t \times t} & \mathbf{0}_{t \times n} \\ \mathbf{0}_{t \times n} & \mathbf{E}_{n \times n} \end{pmatrix} \in \{0,1\}^{(n+t) \times (n+t)}, \tag{1}$$

where the matrices $\mathbf{U} \in \mathbb{Z}_q^{t\times t}$ and $\mathbf{E} \in \{0,1\}^{n\times n}$ are diagonal matrices, i.e., $\mathbf{U} = \mathrm{diag}(u_1, \cdots, u_t)$ and $\mathbf{E} = \mathrm{diag}(1, \cdots, 1)$, which are also the two division matrices of the plaintext matrix $\mathbf{M}$. Upon receipt of the public keys from all participants, each participant is tasked with computing the aggregate public key: $PK = \mathbf{A} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{A}_i$. Calculate and write $\mathbf{C} = \mathbf{M}\cdot\mathbf{G} + \mathbf{A}^T\cdot\mathbf{R} \pmod q \in \mathbb{Z}_q^{(n+t)\times M}$, $\mathbf{G} = \mathrm{BitDecom}\mathbf{p}^{-1}(\mathbf{I}_{n+t})$ $= (g^T \otimes \mathbf{I}_{n+t}) \in \mathbb{Z}_q^{(n+t)\times(n+t)\cdot l}$, where $\mathbf{I}_{n+t}$ denotes the $(n+t)$ dimensional unit matrix; hence, $g^T = [2^0, 2^1, \cdots, 2^{l-1}] \in \mathbb{Z}_q^l, l = \lceil \log q \rceil = \lfloor \log q \rfloor + 1$, for $m \geqslant n\lceil \log q \rceil$, i.e., $m = \mathcal{O}(n(\log q))$.

- Evaluation: Upon receipt of the ciphertexts from all participants, each participant is evaluated for deterministic homomorphism based on the circuit CIR, and a final ciphertext $\hat{\mathbf{C}}$ contains Add and Mult in more detail:
  $\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2)$: Output:

$$\hat{\mathbf{C}} = \mathbf{C}_1 + \mathbf{C}_2 = (\mathbf{M}_1 + \mathbf{M}_2)\mathbf{G} + \mathbf{A}^T(\mathbf{R}_1 + \mathbf{R}_2) \in \mathbb{Z}_q^{(n+t)\times M} \tag{2}$$

  $\mathsf{Mult}(\mathbf{C}_1, \mathbf{C}_2)$: Outputting the matrix product, as $\mathbf{C}_2 = \mathbf{M}_2\cdot\mathbf{G} + \mathbf{A}^T\cdot\mathbf{R}_2$, one obtains

$$\begin{aligned}\mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2) &= \left(\mathbf{M}_1\cdot\mathbf{G} + \mathbf{A}^T\cdot\mathbf{R}_1\right)\cdot\mathbf{G}^{-1}(\mathbf{C}_2) = \mathbf{M}_1\cdot\mathbf{C}_2 + \mathbf{A}^T\cdot\mathbf{R}_1\cdot\mathbf{G}^{-1}(\mathbf{C}_2)\\ &= \mathbf{M}_1\mathbf{M}_2\cdot\mathbf{G} + \mathbf{A}^T\mathbf{R}_1\cdot\mathbf{G}^{-1}(\mathbf{C}_2) + \mathbf{M}_1\mathbf{A}^T\mathbf{R}_2 \in \mathbb{Z}_q^{(n+t)\times M}\end{aligned} \tag{3}$$

  Additionally, this configuration facilitates the computation of homomorphic NAND gates. The operation is executed by generating the output from the expression $\mathbf{G} - \mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2)$.
- Decryption $\mathbf{U} \leftarrow \mathsf{Dec}(\mathsf{params}, \mathsf{sk}, \hat{\mathbf{C}})$: We denote a matrix

$$\mathbf{W}^T = \left(\begin{array}{ccc|c} \lceil q/2\rceil, & \cdots, & 0 & \mathbf{0}^{1\times n}\\ \vdots & \ddots & \vdots & \vdots\\ 0, & \cdots, & \lceil q/2\rceil & \mathbf{0}^{1\times n}\end{array}\right) \in \mathbb{Z}_q^{t\times(n+t)} \tag{4}$$

  Participant P_i constructs its own key matrix $S_i = (\mathbf{s}_1, \cdots, \mathbf{s}_t) = \left(\frac{\mathbf{I}}{-\mathbf{t}_1, \cdots, -\mathbf{t}_t}\right) \in \{0,1\}^{(n+t)\times t}$. According to χ, select random vector $\sigma_j'' \in \mathbb{Z}_q^{n+t-tl}, \sigma_j' = (0, ..., 0, \sigma_j'') \in \mathbb{Z}_q^{n+t}$, and let the random vector matrix $\sigma_i = (\sigma_1', ..., \sigma_j', ..., \sigma_t') \in \mathbb{Z}_q^{(n+t)\times t}$ Generate and publish decryption shares $d_i = S_i^T\cdot\hat{\mathbf{C}} + \sigma_i \in \mathbb{Z}_q^{(n+t)\times t}$. After obtaining all the decryption shares d_i from the other participants, P_i computes

$$D = \frac{1}{N}\sum_{i=1}^{N} d_i = \frac{1}{N}\sum_{i=1}^{N} S_i^T\cdot\hat{\mathbf{C}} + \frac{1}{N}\sum_{i=1}^{N}\sigma_i \triangleq \mathbf{S}^T\hat{\mathbf{C}} + \sigma \tag{5}$$

  where $\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N} S_i$, $\mathbf{S}^T\hat{\mathbf{C}} = \mathbf{S}^T\mathbf{A}^T\mathbf{R} + \mathbf{S}^T\mathbf{M}\mathbf{G} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{e}_{i,1}, ..., \boldsymbol{e}_{i,t})^T\mathbf{R} + \mathbf{S}^T\mathbf{M}\mathbf{G}$ and $\mathbf{V} = \mathbf{S}^T\hat{\mathbf{C}}\cdot\mathbf{G}^{-1}(\mathbf{W}^T)$.
  Output decrypted message $\mathbf{U} = \left|\lfloor\frac{\mathbf{V}}{q/2}\rceil\right|$, where

$$\begin{aligned}\mathbf{V} &= \mathbf{S}^T\hat{\mathbf{C}}\cdot\mathbf{G}^{-1}(\mathbf{W}^T)\\ &= \lceil\frac{q}{2}\rceil\cdot\begin{pmatrix} u_{1,1} & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & u_{t,t}\end{pmatrix} + \frac{1}{N}\begin{pmatrix}\mathbf{e}_1^T\mathbf{R}\\ \vdots\\ \mathbf{e}_t^T\mathbf{R}\end{pmatrix}\cdot\mathbf{G}^{-1}(\mathbf{W}^T)\\ &= \lceil\frac{q}{2}\rceil\cdot\mathbf{U}_{t\times t}\end{aligned} \tag{6}$$

4.2. Key Homomorphism of mMFHE

In order to facilitate comprehension, this section will introduce and prove the key homomorphism of mMFHE in the CRS model.

Theorem 3. *Consider an environment in the GSW13 encryption scheme where all participants use the same random matrix* $\mathbf{B}$ *to generate their public keys. In this case, the GSW13 scheme is able to realize the additive homomorphism property of the key. Specifically, if all participants* P_i*, for* $i \in [N]$*, construct their own public key as* $\mathbf{A}_i$ *and accumulate these public keys as* $pk = \mathbf{A} = \sum_{i=1}^{N} \mathbf{A}_i$ *for encrypting a single-bit message* μ*, the resulting cipher text is* $\mathbf{C} = \sum_{i=1}^{N} \mathbf{A}_i\mathbf{R} + \mu\mathbf{G}$*. In this configuration, the corresponding valid private key* $sk = \mathbf{t} = \sum_{i=1}^{N} \mathbf{t}_i$ *is able to correctly decrypt the ciphertext* $\mathbf{C}$ *into the message* μ*. The proof of this theorem was given in reference [40], and below, we will show that it still holds in mMFHE.*

Proof. Assume that the participant P_i generates private key matrix $\mathbf{sk}_i := \boldsymbol{S}_i = [\mathsf{sk}_1, \cdots, \mathsf{sk}_t] = [\mathbf{s}_1, \cdots, \mathbf{s}_t]$ according to the $\mathsf{mFHE.KeyGen}()$ algorithm, where $\mathbf{s}_j = (\mathbf{I}_j \mid -\mathbf{a}_j^T)^T$, $\mathbf{a}_j^T = (a_{j,1}, \cdots, a_{j,n}) \in \mathbb{Z}_q^{1\times n}$ are randomly selected samples; the public key matrix is $\mathbf{pk}_i := \boldsymbol{A}_i = [\mathbf{b}_1 \mid \cdots \mid \mathbf{b}_t \mid \mathbf{B}]$, where $\mathbf{b}_j = \mathbf{B} \cdot \mathbf{a}_j + \mathbf{e}_{p \bmod q}$, $\mathbf{e}_j \leftarrow \chi^{m\times 1}, j \in [t]$. Then,

$$\begin{aligned}
\mathbf{A} \cdot \mathbf{S} &= \sum_{i=1}^{N} \boldsymbol{A}_i \cdot \sum_{i=1}^{N} \boldsymbol{S}_i \\
&= \left(\sum_{i=1}^{N} \boldsymbol{b}_{1,i}, ..., \sum_{i=1}^{N} \boldsymbol{b}_{t,i}, N \cdot \boldsymbol{B}\right) \left(\sum_{i=1}^{N} \boldsymbol{s}_{1,i}, ..., \sum_{i=1}^{N} \boldsymbol{s}_{t,i}\right) \\
&= \left(\sum_{i=1}^{N} \boldsymbol{b}_{1,i}, ..., \sum_{i=1}^{N} \boldsymbol{b}_{t,i}, N \cdot \boldsymbol{B}\right) \left(\sum_{i=1}^{N} (\boldsymbol{I}_1| - \boldsymbol{a}_1^T)^T, ..., \sum_{i=1}^{N} (\boldsymbol{I}_t| - \boldsymbol{a}_t^T)^T\right) \\
&= N \sum_{i=1}^{N} (\boldsymbol{e}_{1,i}, ..., \boldsymbol{e}_{t,i}) \approx \mathbf{0}
\end{aligned} \tag{7}$$

In the case that all error vectors $\mathbf{e}_{i,j}$ are small enough, the above equation holds, i.e., we can obtain the plaintext message matrix $\mathbf{U}$ according to the decryption algorithm of mMFHE. It can be seen that under the assumption that $\mathbf{B}$ is CRS, mMFHE still satisfies the additive key homomorphism.

However, the validity of the above theorem relies on two key assumptions: one is that the number of participants N must be kept small, and the other is that all the error vectors $\mathbf{e}_{i,j}$ also need to be sufficiently small. These conditions pose rather stringent constraints in practical applications. In view of this, we propose to consider another form of key homomorphism for a wider range of application scenarios. □

Theorem 4. *The GSW13 encryption scheme possesses the linear homomorphism property of the key. More specifically, if the same setup as before is used and the public key is defined as* $pk = \mathbf{A} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{A}_i$*, and this public key is used to encrypt a single bit message* μ *to obtain the ciphertext* $\mathbf{C}$*, then the corresponding valid private key is* $sk = \mathbf{t} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{t}_i$*. This private key can accurately decrypt the ciphertext* $\mathbf{C}$*, thus recovering the original message* μ*.*

Proof. Similarly, we have

$$\begin{aligned}
\mathbf{A} \cdot \mathbf{S} &= \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{A}_i \cdot \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{S}_i \\
&= \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{e}_{1,i}, ..., \boldsymbol{e}_{t,i}) \approx \mathbf{0}
\end{aligned} \tag{8}$$

It can thus be shown that mMFHE satisfies the linear homomorphism property of the key. □

4.3. Threat Model

In contemporary encryption protocols, it is commonly postulated that participants align with the "honest but curious" model. In accordance with this assumption, participants faithfully execute the prescribed protocol while simultaneously seeking any feasible means to derive confidential data contained within the output generated throughout the protocol's execution.

In our research, we employ MKFHE to safeguard data privacy within a federated learning framework. Specifically, we posit that both the server and all remote participants operate under the honest-but-curious assumption. This implies that while they conscientiously adhere to the protocol's specifications, they concurrently endeavor to derive personal information about other participants from the shared data during the course of the protocol's execution. Additionally, we entertain the possibility of collusion among the participants and the server.

To elucidate, we consider a specific adversarial scenario with $k = N - 1$ participants, where N denotes the total number of participants and k the number of conspirators, collaborating with the server to compromise the confidentiality of a targeted participant. This scenario highlights the potential vulnerabilities and the requisite safeguards necessary in the design of secure federated learning systems.

4.4. Our PPFL Scheme

Building upon the mMFHE scheme proposed in the preceding section of this document, we introduce a privacy-centric FL scheme. This scheme is predicated on the assumption that all participants actively engage and contribute to the model training process in each iteration.

With reference to the FedAvg algorithm, the general process of our PPFL scheme is illustrated in Figure 1. In our proposed PPFL scheme, based on the mMFHE scheme and in referencing the FedAvg algorithm, the process is outlined as shown in Figure 1. During each model aggregation round, clients train the received global model using their local data. Clients independently decide the number of training rounds and parameters based on their resources and data volume. After training, clients encrypt their model parameters using aggregated public key PK and send the encrypted results back to the central server.

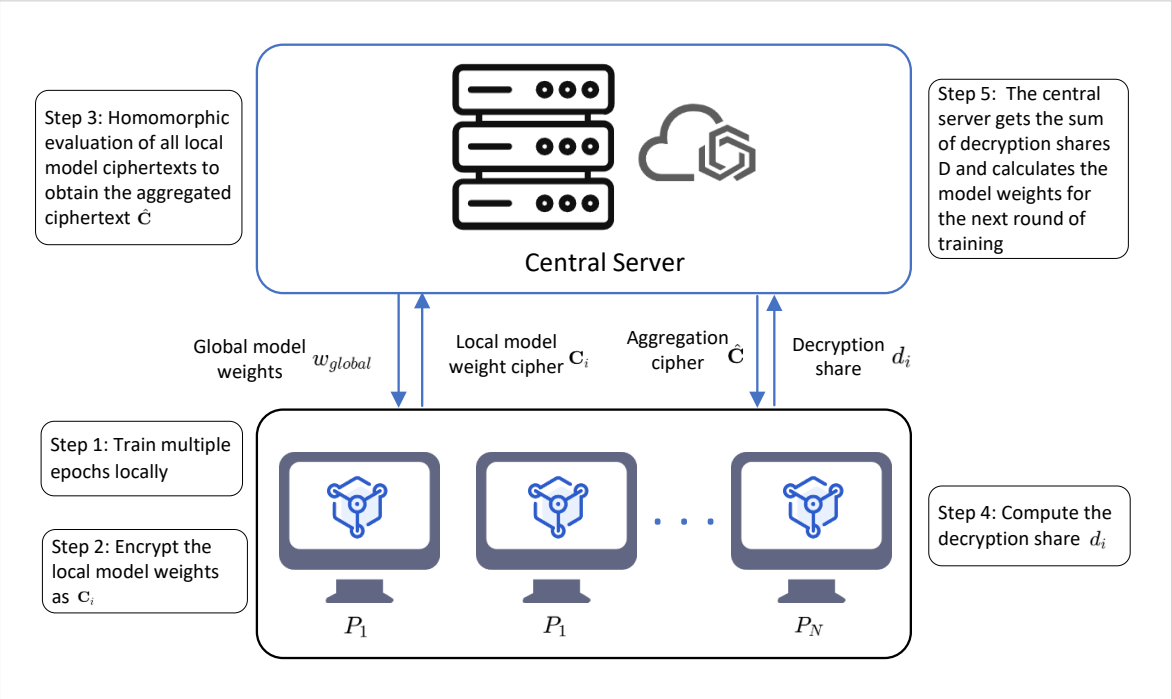

Figure 1. Model of mMFHE-based PPFL scheme.

The central server collects all encrypted model parameters from the clients and performs a ciphertext aggregation to obtain the aggregated result ciphertext $\hat{\mathbf{C}}$. Then, clients use their private key matrices $\mathbf{sk}_i$ and the $\hat{\mathbf{C}}$ from the server to compute decryption shares d_i and send these shares back to the central server. The server aggregates all decryption shares to obtain the plaintext form of the model aggregation result and updates the global model using the average aggregation method. This process is repeated until the model meets the predetermined performance standards or completes the specified number of training rounds. The aforementioned flow of encryption and decryption is illustrated in Figures 2 and 3, and a detailed description of each step is provided below.

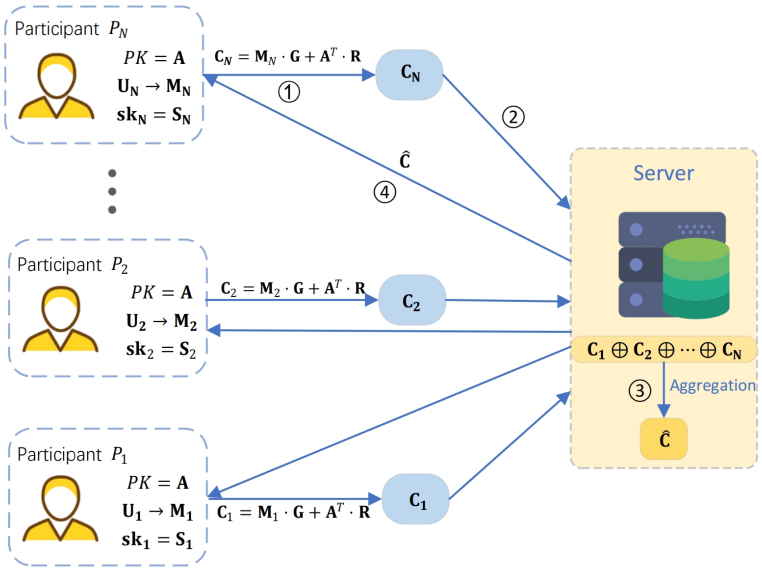

Figure 2. Encryption process.

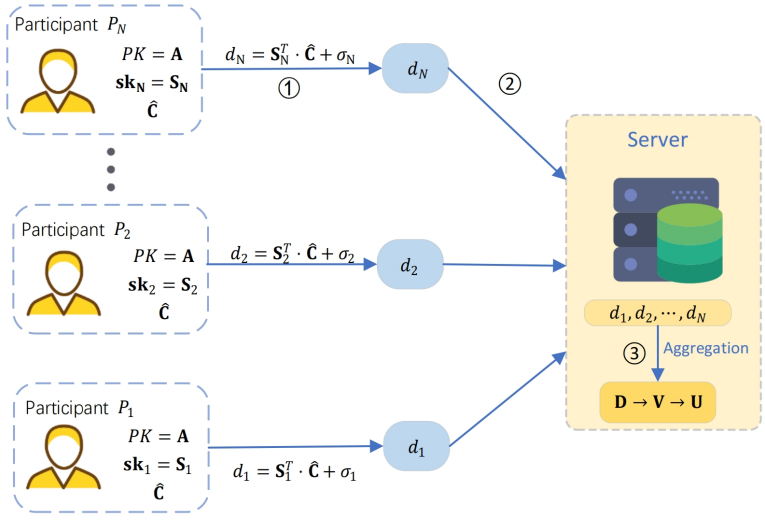

Figure 3. Decryption process.

Initialization: The central server performs the initialization of the global model; executes the Setup$(\cdot)$ function; sets the ciphertext mode q, the lattice dimensions n, m, the error distribution χ, and the length of the message t according to λ and L; selects the uniform random matrix $\mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m}$; and returns public parameters params $= (n, q, \chi, m)$, public

matrix $\mathbf{B}$, and t. Each remote client P_i ($i \in [N]$) selects the sample $\mathbf{a}_j^T = (a_{j,1}, \cdots, a_{j,n}) \in \mathbb{Z}_q^{1\times n}$ in order to generate the private key matrix:

$$\mathbf{sk}_i := \boldsymbol{S}_i = [\mathsf{sk}_1, \cdots, \mathsf{sk}_t] = [\mathbf{s}_1, \cdots, \mathbf{s}_t] \in \mathbb{Z}_q^{(n+t)\times t} \tag{9}$$

where $\mathsf{sk}_j := \mathbf{s}_j = (\mathbf{I}_j \mid -\mathbf{a}_j^T)^T \in \mathbb{Z}_q^{(n+t)\times 1}$ is the j-th bit of the message corresponding to the key, $j \in [t]$. Subsequently, select $\mathbf{e}_j \leftarrow \chi^{m\times 1}$ and compute $\mathbf{b}_j = \mathbf{B} \cdot \mathbf{a}_j + \mathbf{e}_{p \ mod \ q}$ to generate and return the public key matrix to the centralized server:

$$\mathbf{pk}_i := \boldsymbol{A}_i = [\mathbf{b}_1 \mid \cdots \mid \mathbf{b}_t \mid \mathbf{B}] \in \mathbb{Z}_q^{m\times(n+t)} \tag{10}$$

The central server receives the public key matrix from all participating clients, calculates

$$PK = \mathbf{A} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{A}_i \tag{11}$$

and returns the aggregated public key PK to all participants (clients).

Step 1: Local Training: At the beginning of each aggregation round, each participant P_i receives the global model weights w_{global} from the central server. They then utilize their own locally held data to train the model, and after enough epochs, the participant P_i generates a locally held model with the weights w_i.

Step 2: Model Weight Encryption: Let the t-bit message $u_i \in \{0,1\}$ be the encoded plaintext input of w_i and generate the corresponding plaintext matrix $\mathbf{M}_i$ (see Section 4.1 above); participant P_i samples a homogeneous matrix $\mathbf{R}_i \leftarrow \{0,1\}^{m\times M}$ and encrypts $\mathbf{M}_i$ with the aggregation public key $PK = \mathbf{A}$ to obtain the ciphertext

$$\mathbf{C}_i = \mathbf{M}_i \cdot \mathbf{G} + \mathbf{A}^T \cdot \mathbf{R} \pmod q \in \mathbb{Z}_q^{(n+t)\times M} \tag{12}$$

and sends C_i to the central server.

Step 3: Homomorphic Evaluation: The central server performs homomorphic evaluation on the ciphertexts after receiving the ciphertexts with model weights sent by each participant. For ease of understanding, here in this paper, there is an example of homomorphic addition, which yields

$$\hat{\mathbf{C}} = \sum_{i=1}^{N} \mathbf{C}_i = \sum_{i=1}^{N} \mathbf{M}_i\mathbf{G} + \mathbf{A}^T \sum_{i=1}^{N} \mathbf{R}_i \in \mathbb{Z}_q^{(n+t)\times N}; \tag{13}$$

The server then sends $\hat{\mathbf{C}}$ to all participants. And if we want to perform homomorphic multiplication, in the case of two participants,

$$\mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2) = \mathbf{M}_1\mathbf{M}_2 \cdot \mathbf{G} + \mathbf{A}^T\mathbf{R}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2) + \mathbf{M}_1\mathbf{A}^T\mathbf{R}_2 \in \mathbb{Z}_q^{(n+t)\times M} \tag{14}$$

Step 4: Calculation of decryption share: In MKFHE, the decryption of the ciphertext necessitates the input of all participating members; in order to decrypt $\hat{\mathbf{C}}$, it is necessary for each participant P_i to calculate its decryption share d_i using its private key matrix S_i:

$$d_i = \boldsymbol{S}_i^T \cdot \hat{\mathbf{C}} + \sigma_i \in \mathbb{Z}_q^{(n+t)\times t} \tag{15}$$

where $\sigma_i = \left(\sigma'_1, \ldots, \sigma'_j, \ldots, \sigma'_t\right) \in \mathbb{Z}_q^{(n+t)\times t}$ is the matrix of random vectors, and σ'_j is the error vector chosen from χ.

Step 5: Model Aggregation: The server calculates the aggregation of the decryption shares after obtaining the decryption shares d_i of all participants and obtains

$$D = \frac{1}{N}\sum_{i=1}^{N} d_i = \frac{1}{N}\sum_{i=1}^{N} S_i^T \cdot \hat{\mathbf{C}} + \frac{1}{N}\sum_{i=1}^{N} \sigma_i \triangleq \mathbf{S}^T\hat{\mathbf{C}} + \sigma \tag{16}$$

where $\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N} S_i$. Then, the server computes $\mathbf{V} = \mathbf{S}^T\hat{\mathbf{C}} \cdot \mathbf{G}^{-1}(\mathbf{W}^T)$, obtains the decrypted message $\mathbf{U} = \left| \lfloor \frac{\mathbf{V}}{q/2} \rceil \right|$, decodes $\mathbf{U}$ to obtain the aggregation of the local model weights of each participant, and computes the new weight, which will be used as the global model weight in the next round.

5. Security Analysis

This section presents a discussion of the manner in which the presented scheme ensures the confidentiality of the model weights within the FL system. This, in turn, ensures the privacy of the data hosted on the distributed devices by each FL participant. In order to characterize the security of our scheme for each potential adversary in the CRS model, we will employ the following theorem.

Theorem 5 (Semantic Safety)**.** *mMFHE is IND-CPA-safe if the parameters params* $= (n, q, \chi, m, t)$ *are chosen to align with the difficulty presumption of the* $LWE_{n,m,q,\chi}$ *problem, and* $m = O(n \log q)$.

Proof. (1) The public key matrix $\boldsymbol{A} = [\mathbf{b}_1 \mid \cdots \mid \mathbf{b}_t \mid \mathbf{B}]$ and the $m \times (n+t)$-rank matrices uniformly chosen from $\mathbb{Z}_q$ are statistically indistinguishable. A detailed proof of this is given in reference [42]. (2) The ciphertext matrices $\mathbf{C}$ and the $(n+t) \times M$-rank matrices selected uniformly from $\mathbb{Z}_q$ are computationally indistinguishable. A detailed proof of this is given in reference [43]. □

Theorem 6 (Security Against Honest-but-Curious Servers)**.** *In the framework of our proposed scheme, it is established that a server, operating under the honest-but-curious model, is incapable of deducing any private information from any of the remote participants.*

Proof. In the mMFHE-based PPFL scheme, the remote participant P_i will send two kinds of information to the server, i.e., C_i and d_i. In Step 2, P_i will encrypt its own locally held model gradient to obtain C_i using mMFHE and send it to the server, where $C_i = \mathbf{M}_i \cdot \mathbf{G} + \mathbf{A}^T \cdot \mathbf{R}(\mathrm{mod} q)$. From Theorem 5, mMFHE is IND-CPA-secure, so C_i will not disclose to the server any of the M_i information.

In Step 4, P_i partially decrypts the aggregated ciphertext $\hat{\mathbf{C}}$ using the private key matrix S_i to obtain the decryption share $d_i = S_i^T \cdot \hat{\mathbf{C}} + \sigma_i$ and sends it to the server, which aggregates the decryption shares of all parties to obtain $D = \mathbf{S}^T\hat{\mathbf{C}} + \sigma$. Since the first $(n+t) \times t$ components in σ_i and σ are subject to B_χ boundedness, these two equations form the some-are-errorless LWE problem. Therefore, in Steps 4 and 5, P_i publishes its d_i without revealing its private key or the aggregation key. After the aggregation is decrypted, the server is only able to ascertain the total sum of the model weights, rather than the specific weights of the individual participants.

It thus follows that our scheme ensures that the individual weights remain confidential, thereby guaranteeing the privacy of data belonging to various participants distributed across remote devices. It is assured that the server, upon receiving information, remains unable to deduce any private details regarding the participants. □

Theorem 7 (Security for Honest-but-Curious Users)**.** *Within the framework of the mMFHE-based PPFL scheme, it is established that an honest-but-curious user is incapable of deriving any private information about other users by intercepting shared information.*

Proof. In our proposed model, the model update for each participant P_i is secured using the mMFHE encryption. Each device independently selects an S_i to generate a corresponding A_i. Concurrently, all participants collaborate to obtain an aggregated public key, which is then employed for the encryption of their model weights. To further enhance security, an error is introduced into the decryption share, thereby protecting the keys of each user. With Theorem 5, the mMFHE has been proven to be IND-CPA-secure. Consequently, an honest-but-curious user is unable to extract meaningful information from data uploaded by other participants, thus maintaining confidentiality across the network. □

Theorem 8 (Security Against User–Server Collusion:)**.** *Assume a scenario where $k \leq N-1$ users engage in collusion with the server, yet do not disclose the model updates from other users, where N represents the total number of users and k is the number of users in collusion with the server.*

Proof. In addressing a scenario with a static semi-malicious adversary A involving exactly $N-1$ corrupt users conspiring with the server, we establish a Probabilistic Polynomial-Time (PPT) simulator, denoted as Sim, and designate P_h to represent the sole honest participant. The simulator Sim undertakes the following operations on behalf of the honest party:

In Step 2, the simulator Sim employs a zero value as a placeholder for the genuine input of the honest participant P_h during the encryption process. Subsequently, Sim acquires the inputs and private keys of the $N-1$ compromised parties from the "evidence tape". These inputs are then supplied to an "ideal machine", from which the output y is derived. Additionally, Sim is capable of retrieving the homomorphically computed ciphertext $\hat{\mathbf{C}}$ from the server.

Using this gathered information, Sim computes the simulated partial decryption result for the honest party P_h as follows:

$$\boldsymbol{\rho}'_h \leftarrow Sim(y, \hat{\mathbf{C}}, h, \{\mathbf{sk}_i\}_{i\in[N]\setminus\{h\}}) \tag{17}$$

In Step 4, rather than forwarding the authentic decryption result, Sim transmits this simulated partial decryption result to the server. We define a series of hybrid games to prove the indistinguishability between the real and simulated scenarios, i.e., $IDEAL_{F,Sim,Z} \overset{\text{comp}}{\approx} REAL_{\pi,A,Z}$, where Z represents a specific environment. In this context, the game $REAL_{\pi,A,Z}$ represents the execution of protocol π_f in the real environment Z with a semi-honest adversary. The game $HYB_{\pi,A,Z}$ is essentially the same as the game $REAL_{\pi,A,Z}$, but assumes that P_h obtains all private keys $\{\mathbf{sk}_i\}$ for $i \in [N]\setminus\{h\}$ after Step 2, and in Step 4, it sends the simulated partial decryption $\rho'_h \leftarrow Sim(y, \hat{\mathbf{C}}, h, \{\mathbf{sk}_i\}_{i\in[N]\setminus\{h\}})$ to the server instead of the real decryption. In contrast, the game $IDEAL_{F,Sim,Z}$ replaces the real input of P_h with 0 for encryption and sends it to the server in Step 2, while the other Steps remain consistent with the game $HYB_{\pi,A,Z}$. □

Lemma 1. $REAL_{\pi,A,Z} \overset{\text{stat}}{\approx} HYB_{\pi,A,Z}$

Proof. The distinction between the two scenarios is marked by the substitution of the simulated decryption ρ'_h for the actual partial decryption ρ_h conducted by participant P_h. Therefore, let $\mathbf{V} = \mathbf{U}\lceil\frac{q}{2}\rceil + \boldsymbol{e}'$, with the simulated decryption algorithm being

$$\begin{aligned}\boldsymbol{\rho}'_h &= N\cdot\mathbf{U}\cdot\lceil\frac{q}{2}\rceil + N\boldsymbol{e}' - \sum_{i\neq h}\mathbf{S}_i\hat{\mathbf{C}}\mathbf{G}^{-\mathbf{1}}(\mathbf{W}^T) + \boldsymbol{\sigma}'_h \\ &= N\cdot\mathbf{U}\cdot\lceil\frac{q}{2}\rceil + N\boldsymbol{e}' + \boldsymbol{\sigma}'_h - \sum_{i\neq h}\mathbf{V}_i\end{aligned} \tag{18}$$

where $\boldsymbol{e}' \leftarrow \chi$, $\boldsymbol{\sigma}'_h \leftarrow \chi$.

As $\mathbf{V} = \frac{1}{N}\sum_{i\in[N]}\mathbf{V}_i = \mathbf{U}\cdot\lceil\frac{q}{2}\rceil + \boldsymbol{e}' \Rightarrow N\boldsymbol{e}' = \sum_{i\in[N]}\mathbf{V}_i - N\cdot\mathbf{U}\cdot\lceil\frac{q}{2}\rceil$, the real decryption of P_h is

$$\begin{aligned}\boldsymbol{\rho}_h &= \mathbf{V}_h + \boldsymbol{\sigma}_h \\ &= \sum_{i\in[N]}\mathbf{V}_i - \sum_{i\neq h}\mathbf{V}_i + \boldsymbol{\sigma}_h \\ &= N\boldsymbol{e}' + N\cdot\mathbf{U}\cdot\lceil\frac{q}{2}\rceil - \sum_{i\neq h}\mathbf{V}_i + \boldsymbol{\sigma}_h\end{aligned} \tag{19}$$

where $\boldsymbol{\sigma}_h \leftarrow \chi$.

It can be readily observed that the values $\boldsymbol{\sigma}_h$ and $\boldsymbol{\sigma}'_h$ are statistically indistinguishable, thereby establishing that $\boldsymbol{\rho}_h$ and $\boldsymbol{\rho}'_h$ are likewise indistinguishable from one another. Therefore, it is established that $REAL_{\pi,A,Z} \overset{\text{stat}}{\approx} HYB_{\pi,A,Z}$. □

Lemma 2. $HYB_{\pi,A,Z} \overset{\text{comp}}{\approx} IDEAL_{F,Sim,Z}$

Proof. In these two games, only the ciphertext generated by P_h differs. In accordance with Theorem 5, the ciphertexts are deemed computationally indistinguishable, thereby rendering the two games indistinguishable as well. By Lemmas 1 and 2, we can conclude that $IDEAL_{F,Sim,Z} \overset{\text{comp}}{\approx} REAL_{\pi,A,Z}$. This means that even if the semi-honest adversary A corrupts $N-1$ users and colludes with the server, A still cannot infer the private information of the only honest participant P_h. □

6. Performance Analysis and Experimentation

6.1. Performance Analysis

This section presents an analysis of the performance of the proposed mMFHE protocol and makes some comparisons with existing related schemes.

mMFHE is implemented based on the GSW13 scheme. Since homomorphic NAND is realized through a combination of homomorphic addition and homomorphic multiplication, we can evaluate its efficiency by analyzing the complexity of NAND. In comparison to other FHE schemes, the mMFHE scheme exhibits a time complexity of $\tilde{O}(N(nd)^{\omega})$ for evaluating NAND gates, where n represents the lattice dimension, d denotes the depth of the NAND circuit being evaluated, and $\omega < 2.3727$ is a fixed constant [44]. In parallel research, detailed in reference [45], another LWE-based FHE scheme, referred to as Bv11, serves as the foundation for constructing SMC protocols via threshold decryption. The Bv11 scheme achieves a complexity of $\tilde{O}(n^3 d)$ for evaluating NAND gates. This positions it as marginally less efficient than mMFHE, particularly when the lattice dimension n is large. More importantly, the ciphertexts in mMFHE are in matrix form. This implies a lower expansion rate for ciphertext size, reduced time consumption, and the ability to avoid evaluation key operations in homomorphic evaluation, which are often the most time- and space-consuming aspects of FHE and related applications.

In recent years, several FHE schemes based on GSW13 have emerged, such as in [15,40]. These schemes follow a "matrix cascading" approach to construct joint ciphertexts, resulting in large ciphertext sizes and computational assumptions. Moreover, the aforementioned schemes are single-bit; if a participant's input is t bits, the two schemes need to be executed t times. In contrast, the scheme proposed in this paper only requires a single execution, making it more time-efficient than existing schemes. The details are shown in Table 2, where "CTE Ratio" represents the ciphertext expansion ratio, n is the lattice dimension, N is the number of users, EK indicates whether the key needs to be evaluated, and "NAND TCP" is the time complexity consumed by each NAND gate.

Table 2. Comparison of computational performance of related FHE schemes.

Scheme	Base	CTE Ratio	EK	NAND TCP
[15]	GSW13	$O(1)$	No	$\tilde{O}(tN(nd)^w)$
[46]	NTRU	$O(1)$	Yes	Depend on N
[45]	Bv11	$(n+1)\log q$	Yes	$\tilde{O}(n^3d)$
[40]	GSW13	$O(1)$	No	$\tilde{O}(t(nd)^w)$
Ours	GSW13	$O(1)$	No	$\tilde{O}((nd)^w)$

6.2. Experimentation and Evaluation

In this study, we tested and evaluated the performance of mMFHE- and mMFHE-based PPFL, respectively. In addition, a series of comparative tests were conducted to demonstrate the performance.

We implemented our scheme and tested its performance using the following settings:

Simulation experiment settings: The hardware used was 13th Gen Intel®Core™ i9-13900HX 2.20 GHz with 32 GB of memory. We used a Ubuntu Server 18.04 LTS as the server operating system and Ubuntu 18.04 LTS as the user operating system. We used three datasets: FEMINIST [47], EMINIST [48], and FashionMINIST [49], as shown in Table 3, to train the model. We used a fully connected neural network (CNN) as the model. The input layer consisted of 784 nodes. The output layer used the softmax activation function. The hidden layer had a five-layer structure and used the ReLu activation function. Our model was trained using the Adam optimizer [50] with a learning rate of 0.01 and 20 local epochs in one round of aggregation. The value of the security parameter q was $2^{31} - 1$.

Table 3. Datasets.

Dataset	Input Size	Sample Num	Partion
FEMINIST	28×28	805,263	3500 users
EMINIST	28×28	814,255	Custom
FashionMINIST	28×28	70,000	Custom

To verify the effectiveness of our scheme, we implemented MK-CKKS-based FL, TEE-based FL, and traditional FL to compare them with the mMFHE-based FL to show the performance of our proposed scheme in terms of computational cost, memory overhead, communication cost, and accuracy. Traditional federated learning has no additional privacy protection for model updates. MK-CKKS-based federated learning encrypts model updates through MK-CKKS, but there is a privacy risk due to the use of noise flooding technology during decryption.

Computational cost: We compared mMFHE with MK-CKKS in terms of homomorphic addition, homomorphic multiplication, encryption, and decryption. MK-CKKS is a multi-key version of the MKFHE scheme CKKS. Federated learning based on MK-CKKS is updated through the MK-CKKS encryption model, but there is a privacy risk due to the use of noise flooding technology during decryption. As can be seen from Figure 4, the cost of mMFHE in the encryption and decryption stages is roughly the same as that of MK-CKKS, but it has certain advantages overall. We discuss the computational cost when the number of addition and multiplication operations changes in Figures 5 and 6. We note that the time required for mMFHE to perform homomorphic addition is linearly related to the number of executions, and is better than the MK-CKKS scheme. The multiplication efficiency of mMFHE is slightly lower than that of MK-CKKS at the beginning, but as the number of multiplication operations increases, MK-CKKS needs to perform complex operations such as the rescaling of the noise, which leads to a decrease in multiplication efficiency and is lower than the multiplication execution efficiency of mMFHE.

Memory consumption: Compared with mMFHE and MK-CKKS, TEE-based FL migrates the entire privacy process to secure memory. MK-CKKS and mMFHE use multi-key HE running in ordinary memory to protect privacy. As shown in Figure 7, we tested the

memory overhead of the three. Due to the lower ciphertext expansion rate of mMFHE, the memory overhead of mMFHE is greater than the secure memory of TEE-based FL, but less than the ordinary memory of MK-CKKS. Although the memory overhead of TEE-based FL is smaller than that of mMFHE, it cannot guarantee security in the face of collusion attacks between users and servers. In addition, TEE needs to use an expensive page swap mechanism in secure memory. Therefore, we believe that the memory overhead of mMFHE is acceptable as a price to pay for higher security.

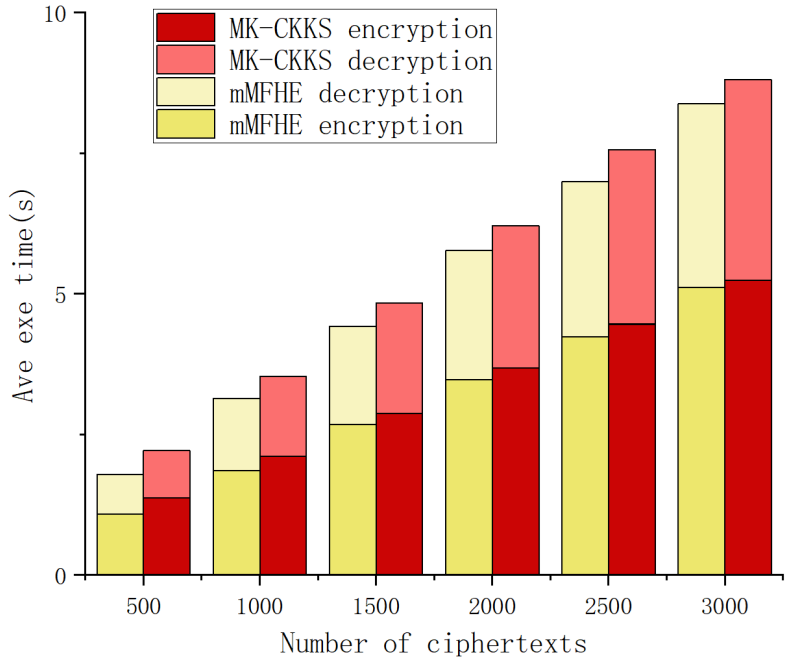

Figure 4. Encryption and decryption.

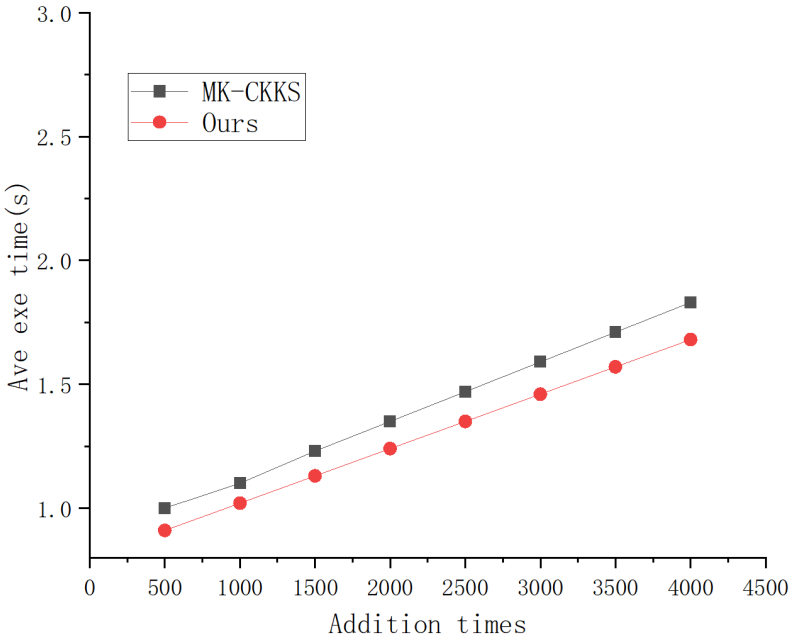

Figure 5. Addition.

Communication overhead: We further evaluated the total communication overhead of a round in mMFHE-based PPFL. The data transmitted from the user to the server were mainly encrypted in the model. We tested the communication cost when the model size is at 10^4 to 10^6. Table 4 lists the ciphertext and communication consumption corresponding

to the number of model parameters, and the communication overhead was about 32.2 MB when the model size reached 7027860, which is within the acceptable range.

Accuracy: Model accuracy usually refers to the proportion of correct predictions made by the global model on the test dataset. We evaluated the model accuracy of mMFHE-based PPFL and compared it with the traditional FL scheme and MK-CKKS-based FL. As shown in Figure 8,when we increase the local epoch to L = 40, the mMFHE-based scheme provides an accuracy of 97.1%, which is very close to the accuracy of the federated learning scheme (97.9%) and higher than that of MK-CKKS. This is because MK-CKKS uses approximate calculations when encrypting and has the problem of error accumulation, which also proves the accurate decryption of the mMFHE scheme.

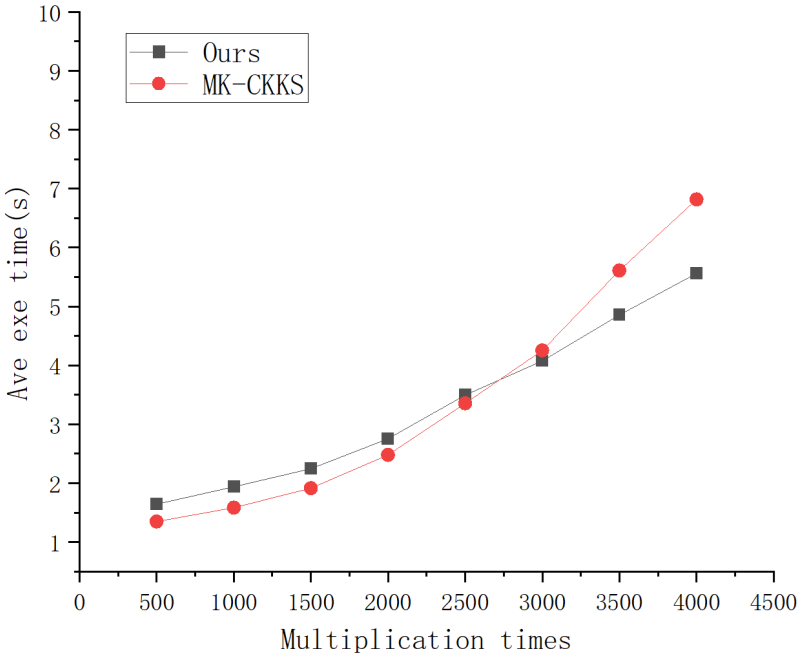

Figure 6. Multiplication.

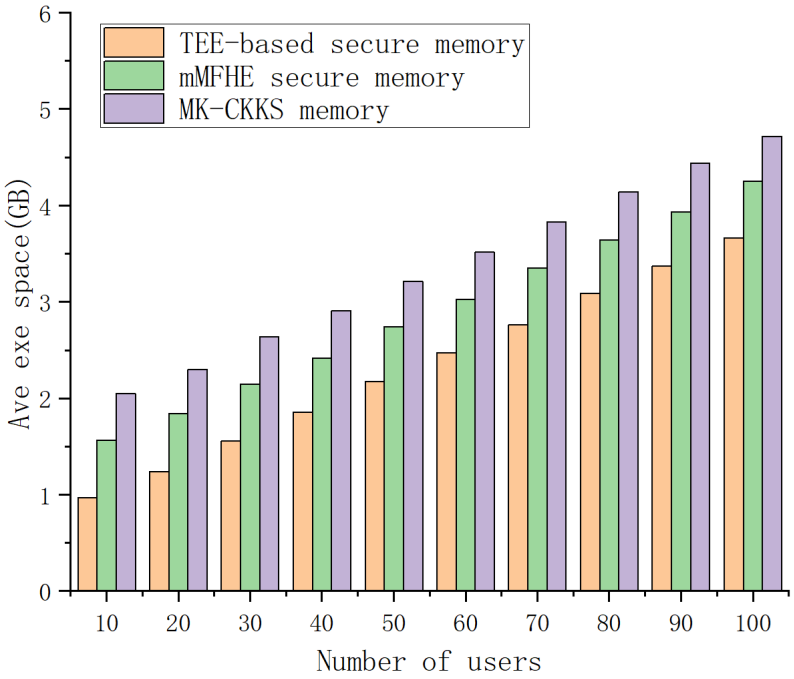

Figure 7. Memory consumption.

Table 4. Communication overhead

Model Size	Ciphertexts Num	Communication Overhead
50,670	63	232 KB
616,420	762	2.8 MB
7,027,860	8660	32.2 MB

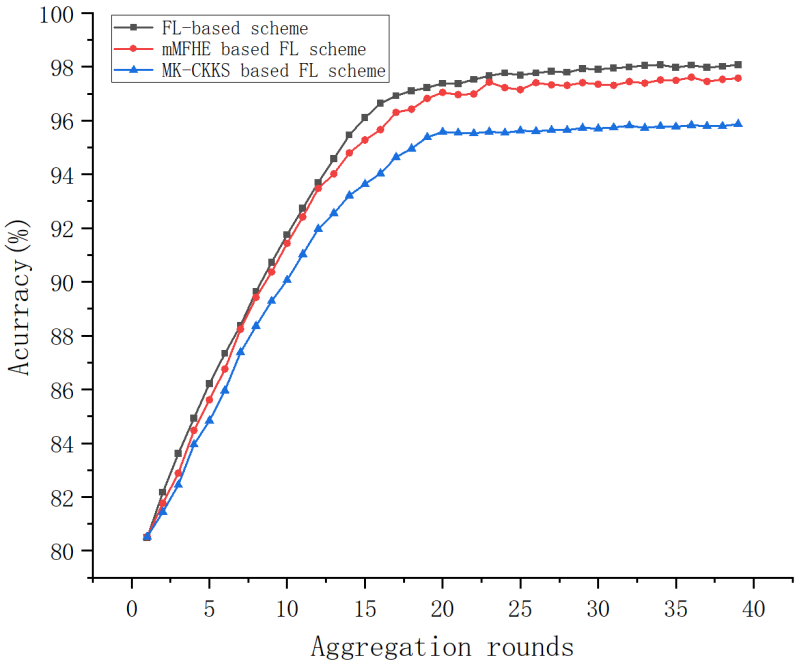

Figure 8. Model accuracy.

7. Discussion

We will discuss some of the current problems faced by mMFHE-based FL with future research directions in this section.

Transmission penalties for encrypted data: In FL based on HE, the encrypted transmission of the data does not directly result in penalties. HE is a form of encryption that allows computations to be performed directly on encrypted data and results in encrypted results, a process that does not require decrypting the data. This provides strong guarantees for protecting data privacy and is particularly suitable for use in scenarios where data privacy is critical, such as in FL. However, the use of HE introduces some other challenges and costs, such as computational efficiency vs. latency. HE operations are typically more complex and time-consuming than non-encryption operations, which can increase the overall computational requirements and latency of the learning task, and we have also shown in our experiments the time spent on the encryption and decryption processes—which is not required in non-encrypted FL. In addition, advanced encryption techniques require more computational resources, and we also show in our experiments the memory consumption required by mMFHE. These factors may impose "penalties" in terms of system performance and cost, especially in resource-limited devices or applications that require real-time response, which also leads to the lack of real-time applicability of mMFHE compared to TEE-based FL.

Reducing complexity: To address the above-mentioned problems of mMFHE in terms of computational efficiency and overhead, we plan to explore the possibility of applying hybrid cryptography in FL in our subsequent research, i.e., combining HE and other types of cryptography (e.g., symmetric encryption or SMPC) to share the computational load. For example, HE could be used for highly computationally demanding parts, while more efficient encryption schemes could be used for other parts. Alternatively, a selective

encryption strategy can be applied to select the appropriate encryption granularity and strength based on the sensitivity of the data and the computational requirements, thus reducing the unnecessary computational burden.

Key management: In MKFHE-based FL, managing multiple keys during multi-party decryption may lead to errors, which remains a challenging issue. We plan to introduce the trusted execution environment as the key management center (KMC) in the subsequent improvement scheme, which is responsible for key generation and management, and select an appropriate key distribution protocol to ensure the security of the key distribution process from the KMC to the user side.

Scalability: In mMFHE-based FL, scalability becomes an important issue as the number of clients increases. The proposed system may face complexity issues as the number of participants increases. To address this issue, we plan to introduce a dynamic participant management mechanism that holds dynamic participant joining and exiting. Through dynamically adjusting the number and load of participants, the computational load and network communication pressure of the system can be effectively balanced. This can be combined with a key management mechanism, where new participants need to go through a secure registration process to obtain the necessary authentication and authorization when joining. In mMFHE, this typically involves distributing or generating a set of keys (public and private) and sharing the public key with the rest of the system. When a participant exits, a logout mechanism is required to ensure that their keys and sensitive data are no longer used by the system and to remove their public key from any shared lists or databases.

Interoperability: Interoperability is a key issue in mMFHE-based FL systems, especially when multiple different organizations or platforms are involved. Addressing interoperability issues mainly involves ensuring effective communication and data sharing between different systems while protecting data privacy and security. In our subsequent work, we plan to design a scalable and flexible system architecture that can accommodate different types of computing nodes and data storage solutions. Such an architecture should support modular and plug-in extensions to facilitate the integration of new technologies or third-party services.

8. Conclusions and Future Work

We propose a PPFL on the basis of MKFHE. Specifically, we designed an MKFHE scheme, mMFHE, based on GSW13, which is secure under the CRS model. To deal with the privacy leakage risks associated with GSW13 as a single-key FHE in FL involving multiple users, we improved it through its key homomorphism, transforming it into MKFHE, which supports encryption using an aggregated public key and shared decryption. Additionally, mMFHE supports embedding multiple plaintext messages into a single ciphertext, thereby enhancing encryption and decryption efficiency while maintaining a low ciphertext expansion rate. Moreover, our security analysis demonstrates that the proposed PPFL scheme can resist collusion attacks involving up to $k = N - 1$ users and the server. Performance analysis and experiments further validate its efficiency.

While our scheme effectively defends against collusion attacks between participants and the server in FL, the introduction of MKFHE also brings new challenges, such as the issue of user offline status during the FL process. If a participant goes offline during model training, the absence of their partial decryption results will cause the aggregation decryption of that round of ciphertext to fail. Given the widespread application of FL in fields such as the Internet of Things (IoT), addressing the user offline issue will be a focus of our future work.

Author Contributions: Conceptualization, Y.Z.; methodology, Y.Z.; validation, J.S. and Y.Z.; resources, Y.R.; writing—original draft preparation, J.S. and Y.Z.; writing—review and editing, Y.Z. and S.H.; visualization, Y.R.; supervision, Y.R.; funding acquisition, Y.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China grant number 62072249.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhu, L.; Liu, Z.; Han, S. Deep leakage from gradients. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019) , Vancouver, BC, Canada, 8–14 December 2019; Volume 32.
2. Hitaj, B.; Ateniese, G.; Perez-Cruz, F. Deep models under the GAN: Information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 603–618.
3. Vatter, J.; Mayer, R.; Jacobsen, H.A. The evolution of distributed systems for graph neural networks and their origin in graph processing and deep learning: A survey. *ACM Comput. Surv.* **2023**, *56*, 1–37. [CrossRef]
4. McMahan, H.B.; Yu, F.; Richtarik, P.; Suresh, A.; Bacon, D. Federated learning: Strategies for improving communication efficiency. In Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016; pp. 5–10.
5. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
6. Rieyan, S.A.; News, M.R.K.; Rahman, A.M.; Khan, S.A.; Zaarif, S.T.J.; Alam, M.G.R.; Hassan, M.M.; Ianni, M.; Fortino, G. An advanced data fabric architecture leveraging homomorphic encryption and federated learning. *Inf. Fusion* **2024**, *102*, 102004. [CrossRef]
7. Mantey, E.A.; Zhou, C.; Anajemba, J.H.; Arthur, J.K.; Hamid, Y.; Chowhan, A.; Otuu, O.O. Federated learning approach for secured medical recommendation in internet of medical things using homomorphic encryption. *IEEE J. Biomed. Health Inform.* **2024**, *28*, 3329–3340. [CrossRef] [PubMed]
8. Hou, X.; Wang, J.; Jiang, C.; Meng, Z.; Chen, J.; Ren, Y. Efficient federated learning for metaverse via dynamic user selection, gradient quantization and resource allocation. *IEEE J. Sel. Areas Commun.* **2023**, *42*, 850–866. [CrossRef]
9. Ren, Y.; Lv, Z.; Xiong, N.N.; Wang, J. HCNCT: A cross-chain interaction scheme for the blockchain-based metaverse. *ACM Trans. Multimed. Comput. Commun. Appl.* **2024**, *20*, 1–23. [CrossRef]
10. Issa, W.; Moustafa, N.; Turnbull, B.; Sohrabi, N.; Tari, Z. Blockchain-based federated learning for securing internet of things: A comprehensive survey. *ACM Comput. Surv.* **2023**, *55*, 1–43. [CrossRef]
11. Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 1333–1345.
12. Melis, L.; Song, C.; De Cristofaro, E.; Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 691–706.
13. Sun, L.; Wang, Y.; Ren, Y.; Xia, F. Path signature-based xai-enabled network time series classification. *Sci. China Inf. Sci.* **2024**, *67*, 170305. [CrossRef]
14. Ren, Y.; Zhu, F.; Wang, J.; Sharma, P.K.; Ghosh, U. Novel vote scheme for decision-making feedback based on blockchain in internet of vehicles. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 1639–1648. [CrossRef]
15. Mukherjee, P.; Wichs, D. Two round multiparty computation via multi-key FHE. In *Advances in Cryptology—EUROCRYPT 2016, Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, 8–12 May 2016*; Proceedings 31; Springer: Berlin/Heidelberg, Germany, 2012; pp. 735–763.
16. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. *Proc. Mach. Learn. Syst.* **2020**, *2*, 429–450.
17. Asad, M.; Moustafa, A.; Ito, T. Fedopt: Towards communication efficiency and privacy preservation in federated learning. *Appl. Sci.* **2020**, *10*, 2864. [CrossRef]
18. Zhang, J.; Hua, Y.; Wang, H.; Song, T.; Xue, Z.; Ma, R.; Guan, H. Fedala: Adaptive local aggregation for personalized federated learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 11237–11244.
19. Yu, X.; Liu, R.; Nkenyereye, L.; Wang, Z.; Ren, Y. ACRS-Raft: A Raft Consensus Protocol for Adaptive Data Maintenance in the Metaverse Based On Cauchy Reed-Solomon Codes. *IEEE Trans. Consum. Electron.* **2024**, *70*, 3792–3801. [CrossRef]
20. Zhang, C.; Li, S.; Xia, J.; Wang, W.; Yan, F.; Liu, Y. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 20), Online, 15–17 July 2020; pp. 493–506.
21. Madi, A.; Stan, O.; Mayoue, A.; Grivet-Sébert, A.; Gouy-Pailler, C.; Sirdey, R. A secure federated learning framework using homomorphic encryption and verifiable computing. In Proceedings of the 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS), Hamilton, ON, Canada, 18–19 May 2021; pp. 1–8.

22. Stripelis, D.; Saleem, H.; Ghai, T.; Dhinagar, N.; Gupta, U.; Anastasiou, C.; Ver Steeg, G.; Ravi, S.; Naveed, M.; Thompson, P.M.; et al. Secure neuroimaging analysis using federated learning with homomorphic encryption. In Proceedings of the 17th International Symposium on Medical Information Processing and Analysis, Campinas, Brazil, 17–19 November 2021; Volume 12088, pp. 351–359.
23. Lindell, Y. Secure multiparty computation. *Commun. ACM* **2020**, *64*, 86–96. [CrossRef]
24. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.* **2018**, *51*, 1–35. [CrossRef]
25. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1175–1191.
26. Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.; Poor, H.V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. [CrossRef]
27. Truex, S.; Liu, L.; Chow, K.H.; Gursoy, M.E.; Wei, W. LDP-Fed: Federated learning with local differential privacy. In Proceedings of the third ACM International Workshop on Edge Systems, Analytics and Networking, Heraklion, Greece, 27 April 2020; pp. 61–66.
28. Hu, R.; Guo, Y.; Li, H.; Pei, Q.; Gong, Y. Personalized federated learning with differential privacy. *IEEE Internet Things J.* **2020**, *7*, 9530–9539. [CrossRef]
29. Li, Y.; Zhou, Y.; Jolfaei, A.; Yu, D.; Xu, G.; Zheng, X. Privacy-preserving federated learning framework based on chained secure multiparty computing. *IEEE Internet Things J.* **2020**, *8*, 6178–6186. [CrossRef]
30. Gehlhar, T.; Marx, F.; Schneider, T.; Suresh, A.; Wehrle, T.; Yalame, H. SafeFL: MPC-friendly framework for private and robust federated learning. In Proceedings of the 2023 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 25 May 2023; pp. 69–76.
31. Zhang, J.; Chen, B.; Yu, S.; Deng, H. PEFL: A privacy-enhanced federated learning scheme for big data analytics. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
32. Li, Y.; Li, H.; Xu, G.; Huang, X.; Lu, R. Efficient privacy-preserving federated learning with unreliable users. *IEEE Internet Things J.* **2021**, *9*, 11590–11603. [CrossRef]
33. Ren, Y.; Leng, Y.; Qi, J.; Sharma, P.K.; Wang, J.; Almakhadmeh, Z.; Tolba, A. Multiple cloud storage mechanism based on blockchain in smart homes. *Future Gener. Comput. Syst.* **2021**, *115*, 304–313. [CrossRef]
34. He, C.; Liu, G.; Guo, S.; Yang, Y. Privacy-preserving and low-latency federated learning in edge computing. *IEEE Internet Things J.* **2022**, *9*, 20149–20159. [CrossRef]
35. Ren, Y.; Leng, Y.; Cheng, Y.; Wang, J. Secure data storage based on blockchain and coding in edge computing. *Math. Biosci. Eng.* **2019**, *16*, 1874–1892. [CrossRef] [PubMed]
36. Cai, Y.; Ding, W.; Xiao, Y.; Yan, Z.; Liu, X.; Wan, Z. SecFed: A Secure and Efficient Federated Learning Based on Multi-Key Homomorphic Encryption. *IEEE Trans. Dependable Secur. Comput.* **2023**, *21*, 3817–3833. [CrossRef]
37. Ma, J.; Naas, S.A.; Sigg, S.; Lyu, X. Privacy-preserving federated learning based on multi-key homomorphic encryption. *Int. J. Intell. Syst.* **2022**, *37*, 5880–5901. [CrossRef]
38. Walskaar, I.; Tran, M.C.; Catak, F.O. A practical implementation of medical privacy-preserving federated learning using multi-key homomorphic encryption and flower framework. *Cryptography* **2023**, *7*, 48. [CrossRef]
39. Zhang, Q.; Jing, S.; Zhao, C.; Zhang, B.; Chen, Z. Efficient federated learning framework based on multi-key homomorphic encryption. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing, Proceedings of the 16th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2021), Fukuoka, Japan, 28–30 October 2021*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 88–105.
40. Wang, H.; Feng, Y.; Ding, Y.; Tang, S. A multi-key SMC protocol and multi-key FHE based on some-are-errorless LWE. *Soft Comput.* **2019**, *23*, 1735–1744. [CrossRef]
41. Gentry, C.; Sahai, A.; Waters, B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013, Proceedings of the 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2013*; Proceedings, Part I; Springer: Berlin/Heidelberg, Germany, 2013; pp. 75–92.
42. Li, Z.; Ma, C.; Zhou, H. Multi-key FHE for multi-bit messages. *Sci. China Inf. Sci.* **2018**, *61*, 029101. [CrossRef]
43. Li, Z.; Ma, C.; Morais, E.; Du, G. Multi-bit Leveled Homomorphic Encryption via-Based. In Proceedings of the International Conference on Information Security and Cryptology, Beijing, China, 4–6 November 2016; pp. 221–242.
44. Sun, L.; Li, C.; Ren, Y.; Zhang, Y. A Multitask Dynamic Graph Attention Autoencoder for Imbalanced Multilabel Time Series Classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 11829–11842. [CrossRef]
45. Asharov, G.; Jain, A.; López-Alt, A.; Tromer, E.; Vaikuntanathan, V.; Wichs, D. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Advances in Cryptology–EUROCRYPT 2012, Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, 15–19 April 2012*; Proceedings 31; Springer: Berlin/Heidelberg, Germany, 2012; pp. 483–501.
46. López-Alt, A.; Tromer, E.; Vaikuntanathan, V. 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, New York, NY, USA, 19–22 May 2012; pp. 1219–1234.

47. Caldas, S.; Duddu, S.M.K.; Wu, P.; Li, T.; Konečnỳ, J.; McMahan, H.B.; Smith, V.; Talwalkar, A. Leaf: A benchmark for federated settings. *arXiv* **2018**, arXiv:1812.01097.
48. Cohen, G.; Afshar, S.; Tapson, J.; Van Schaik, A. EMNIST: Extending MNIST to handwritten letters. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 2921–2926.
49. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
50. Kingma, D.P. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.