entropy

MDPI

Article

Advancing Federated Learning through Verifiable Computations and Homomorphic Encryption

Bingxue Zhang, Guangguang Lu, Pengpeng Qiu, Xumin Gui and Yang Shi *

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China; zhangbingxue@usst.edu.cn (B.Z.); 213330769@st.usst.edu.cn (G.L.); 213330798@st.usst.edu.cn (P.Q.); 212260503@st.usst.edu.cn (X.G.)
* Correspondence: 221240073@st.usst.edu.cn

Abstract: Federated learning, as one of the three main technical routes for privacy computing, has been widely studied and applied in both academia and industry. However, malicious nodes may tamper with the algorithm execution process or submit false learning results, which directly affects the performance of federated learning. In addition, learning nodes can easily obtain the global model. In practical applications, we would like to obtain the federated learning results only by the demand side. Unfortunately, no discussion on protecting the privacy of the global model is found in the existing research. As emerging cryptographic tools, the zero-knowledge virtual machine (ZKVM) and homomorphic encryption provide new ideas for the design of federated learning frameworks. We have introduced ZKVM for the first time, creating learning nodes as local computing provers. This provides execution integrity proofs for multi-class machine learning algorithms. Meanwhile, we discuss how to generate verifiable proofs for large-scale machine learning tasks under resource constraints. In addition, we implement the fully homomorphic encryption (FHE) scheme in ZKVM. We encrypt the model weights so that the federated learning nodes always collaborate in the ciphertext space. The real results can be obtained only after the demand side decrypts them using the private key. The innovativeness of this paper is demonstrated in the following aspects: 1. We introduce the ZKVM for the first time, which achieves zero-knowledge proofs (ZKP) for machine learning tasks with multiple classes and arbitrary scales. 2. We encrypt the global model, which protects the model privacy during local computation and transmission. 3. We propose and implement a new federated learning framework. We measure the verification costs under different federated learning rounds on the IRIS dataset. Despite the impact of homomorphic encryption on computational accuracy, the framework proposed in this paper achieves a satisfactory 90% model accuracy. Our framework is highly secure and is expected to further improve the overall efficiency as cryptographic tools continue to evolve.

Keywords: federated learning; zero-knowledge virtual machine; homomorphic encryption; verifiability; model privacy

Citation: Zhang, B.; Lu, G.; Qiu, P.; Gui, X.; Shi, Y. Advancing Federated Learning through Verifiable Computations and Homomorphic Encryption. *Entropy* **2023**, *25*, 1550. https://doi.org/10.3390/e25111550

Academic Editor: Shu-Chuan Chu

Received: 11 September 2023
Revised: 1 November 2023
Accepted: 4 November 2023
Published: 16 November 2023

1. Introduction

Nowadays, machine learning has become an indispensable tool in many fields, but data privacy and security issues are the main reasons preventing its widespread deployment [1]. Federated learning, as one of the three main technological routes for privacy computing, focuses on protecting data privacy and realizing the availability but invisibility of data [2].

Unlike traditional centralized machine learning processes, federated learning protects the data privacy of learning nodes. It utilizes the ideas of local computing and gradient aggregation to enable multiple participants to jointly construct a model with global performance.

The basic federated learning framework includes a central aggregator and multiple distributed learning nodes. The aggregator controls the entire training process, including

managing learning nodes, aggregating gradients, and maintaining the global model. Before each round of training begins, the aggregator broadcasts the current global model to the learning nodes. The learning node utilizes its private data to execute training algorithms locally, generate model update gradients, and then submit them to the central aggregator. After collecting gradient information from multiple learning nodes, the aggregator generates a new global model based on a certain aggregation strategy [3]. This process will undergo multiple iterations until the predetermined number of training rounds or convergence conditions are reached.

However, due to its distributed nature, the federated learning process may be affected by malicious nodes. For example, distributed nodes may forge false data or provide inaccurate model training results to the aggregator [4], which can seriously degrade the accuracy of machine learning models.

Several recent studies have proposed introducing zero-knowledge proof techniques for federated learning frameworks [5]. They encode the distributed machine learning process as a series of arithmetic and Boolean circuits. Distributed nodes generate verifiable zero-knowledge proofs along with local models. This approach allows the aggregator to verify the integrity of the machine learning process before aggregating a new global model.

A common problem with these solutions is that developers need to learn about cryptography and develop zero-knowledge circuits to accommodate different classes of machine learning protocols [6]. This process is extremely challenging for complex machine learning protocols. In the last two years, the emergence of the zero-knowledge virtual machine (ZKVM) has greatly simplified the design of circuits for developers. The ZKVM is a zero-knowledge virtual environment deployed on a computer that has the capability to generate proofs for arbitrary computations and subsequently verify them on the fly [7].

In this paper, we introduce the ZKVM into federated learning. We construct the ZKVM on all the nodes, where the learning nodes are created as the provers in the ZKP process, and the aggregators are the verifiers. First, in order to demonstrate the universality of the proposed framework, we implement several mainstream machine learning algorithms in ZKVM and compare the verification costs. Second, considering that learning nodes usually do not have high-performance hardware resources, we discuss the memory required for verifying large-scale machine learning tasks. We find that the framework proposed in this paper is able to realize the verification of arbitrary large-scale computational tasks on smaller devices. Finally, we analyze the security of the framework under a variety of malicious behaviors and demonstrate that this framework is highly secure.

In addition, during the cycle of federated learning, all distributed nodes can easily access the latest aggregated global model, which cannot effectively protect the privacy of the global model [8]. In a practical business environment, we would like to have the final machine learning results available only to the demand side.

We note that homomorphic encryption can solve the data privacy problem under multi-party co-computation [9]. It can preserve the structure of the original data so that it can still perform algebraic operations in the ciphertext state. We implement the BGV fully homomorphic encryption scheme in ZKVM. First, we encrypt the initial model weights so that the distributed computational nodes perform the machine learning process in the ciphertext space and the resulting local model remains encrypted. The actual results can be obtained by only aggregating the global model and decrypting it using the private key.

The contributions of this paper are summarized as follows:

1. We introduce the ZKVM for the first time in the federated learning, which can generate verifiable proofs for machine learning tasks of multiple classes and arbitrary sizes. Meanwhile, we discuss the security of the framework for different malicious behaviors.

2. We implement the BGV fully homomorphic encryption scheme in ZKVM. We encrypt the model information to ensure that federated learning nodes always collaborate in the ciphertext space. This protects model privacy during local training and transmission processes.

3. We propose and implement a new federated learning framework. We conducted extensive experiments on the IRIS dataset to summarize the computational cost and learning performance.

The rest of the paper is organized as follows: Section 2 summarizes the preparatory knowledge on federated learning, zero-knowledge proofs, and homomorphic encryption. Section 3 describes the implementation of the system. Section 4 sets up experiments to evaluate the framework proposed in this paper. Section 5 explains the limitations. Section 6 summarizes the whole paper.

2. Background

In this section, we provide a brief introduction to the concepts and cryptographic primitives related to federated learning.

2.1. Federated Learning

In 2016, Google Research pioneered the concept of federated learning [10]. Federated learning is essentially a distributed machine learning method, which aims to realize that all parties work together to train machine learning models without exchanging raw data, and to improve the model effect through a series of aggregation algorithms [11]. Federated learning can connect data silos together and effectively build a data ecosystem. It has become one of the important technologies for mining data value in various fields such as healthcare [12], finance [13], and the Internet of Things [14].

However, as data privacy becomes more and more of a concern, federated learning frameworks alone can no longer meet user needs. For example, malicious learning nodes may tamper with the algorithm execution process or submit false learning results, which can directly affect the performance of federated learning. In addition, learning nodes can easily obtain the global model. In practical applications, we would like to obtain the results of federated learning by only one party. Unfortunately, existing research lacks a discussion of the above issues.

In this paper, we combine the state-of-the-art cryptography tools, zero-knowledge proofs, and homomorphic encryption to address the above issues and construct a more complete federated learning framework.

2.2. Zero-Knowledge Proofs and Zero-Knowledge Virtual Machine

Zero-knowledge proofs (ZKP) are used to solve trust problems between two parties in scenarios where no third-party trusted institution is involved. The state-of-the-art idea is non-interactive zero-knowledge proofs [15]. The proof process of this scheme requires only one data transfer, which greatly reduces the communication time.

In federated learning scenarios, ZKP can validate the legitimacy of data circulation and manipulation. For example, Ghodsi Z. et al. proposed to generate proofs for local machine learning processes [16]. They encode machine learning protocols into the circuit program. When a learning node returns the local model to the aggregator, it needs to provide a verifiable proof at the same time. The proof is used to verify the integrity of the local machine learning task execution before aggregating the global model. However, designing machine learning protocols directly based on circuit languages is difficult due to complex cryptographic principles. For example, the federated learning framework proposed by Abla Smahi et al. based on zk-SNARKs circuit language is only applicable to the federated support vector machine [17].

In this paper, we introduce the ZKVM for the first time to address the above problem. The ZKVM is a virtual machine that runs trusted code and generates verifications of the output [18]. It is generalized in that it lowers the development threshold for zero-knowledge circuits and is able to generate proofs for arbitrary applications or computations on the fly.

The existing ZKVMs are mainly classified into three types: mainstream, EVM-equivalent [19], and ZK-optimized, and their differences are shown in the Table 1.

Table 1. Classification and differences of ZKVMs.

	Existing Expertise/Tooling	Blockchain Focused	Performant
Mainstream (WASM, RISC-V)	Lots	No	Maybe
EVM-equivalent (EVM bytecode)	Some	Yes	No
ZK-optimized (new instruction set)	No	Yes	Yes

This article is based on the popular RISC Zero project. RISC ZKVM is essentially a verifiable virtual machine that operates similarly to a real embedded RISC-V microprocessor [20]. The RISC ZKVM takes care of the underlying cryptography and supports the provision of proofs to arbitrary applications that can run on the RISC-V architecture. Developers need only focus on building the federated machine learning process.

As shown in Figure 1, the RISC ZKVM application consists of a host program and a guest program. The host program can provide input to the guest as needed. The guest program generates a zk-proof after execution. Anyone in possession of a copy of the proof can verify the execution of the guest program and read its publicly shared output.

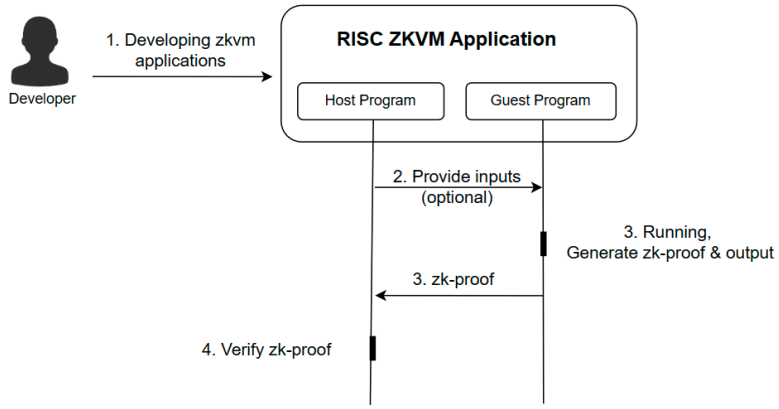


Figure 1. Components of RISC ZKVM application.

We provide a brief description of how ZKVM guarantees computational integrity. The proof system of RISC Zero is built in terms of an execution trace and several constraints that enforce checks of computational integrity. When performing a machine learning task, the execution trace is a record of the full state of the machine at each clock cycle of the computation. They represent the running state of the processor and effectively check the integrity of RISC-V memory operations. Constraints are low-degree polynomial relations over the values of the constraints. The execution trace is valid if and only if each constraint evaluates to 0. For example, $(k)(k-1) = 0$ enforces k to be either 0 or 1. These constraints enforce that the execution of ZKVM is consistent with the RISC-V Instruction Set Architecture (ISA) [21].

In summary, thousands of constraints are first used to enforce the integrity of the RISC-V ISA. Then, at a higher level, constraints are used to enforce that each phase of the machine learning program in ZKVM performs as required.

The ZKVMs are the future, enabling developers to focus on the design of the application itself without paying too much attention to circuits [22]. Constructing ZKVMs on distributed federated learning nodes bridges the gap between zero-knowledge proofs and machine learning programs, making verification of complex machine learning tasks possible.

2.3. Private Set Intersection

Private Set Intersection (PSI) is a privacy-preserving protocol designed to compute the intersection of two parties without revealing their private sets. It also can output the result of a function f computed on the intersection [23].

Secret sharing (SS) and oblivious transfer (OT) are the two key techniques used to construct the PSI protocol. SS requires each participant to secretly divide their input data into data shards and distribute them to others. All participants use their received data shards for calculation and interaction. OT is a secure protocol that protects the privacy of communication between both parties. The sender encrypts n messages and sends them to the receiver, but the receiver can only decrypt k of them. Meanwhile, the sender cannot determine which messages the receiver has decrypted.

PSI-based protocol allows for broader data collaboration and analysis. Some applications that have been proposed are secure computation of medical data [24], security event information sharing [25], etc.

2.4. Trusted Execution Environment

Trusted execution environment (TEE) uses the method of isolating some of the hardware and software resources to build a secure area on a computing device to ensure the protection of sensitive data and operations [26].

TEE implementations are usually based on hardware technology. Therefore, on different system architectures (x86, ARM, RISC-V) (x86, Arm, RISC-V), different software interfaces, and security boundaries need to be designed. In addition, the expensive hardware cost has become one of the barriers to the widespread deployment of TEE.

As shown in Table 2, we have compared several cryptographic protocols in terms of computational complexity and communication cost.

Table 2. Computational complexity and communication costs of cryptographic protocols.

Cryptography Tools	Hardware Dependent	Computational Complexity	Communication Rounds	Communication Cost
TEE	Yes	Lower	Lower	Lower
SS	No	Lower	Higher	Moderate
HE	No	Higher	Constant	Lower
OT	No	Higher	Moderate	Lower

We discard the TEE scheme because the learning nodes are usually built on different system architectures, which requires more effort to consider the combination of hardware and software technologies.

In order to minimize the number of communications between learning nodes and aggregators, we finally chose the HE. The aggregator encrypts the global model so that only a constant number of communication rounds are required to delegate the machine learning task to each learning node. Throughout the entire federated learning process, the global model and gradient information are always calculated in the ciphertext space, and ultimately only the demand side decrypts to obtain the machine learning results.

2.5. Homomorphic Encryption

Homomorphic encryption is a special form of encryption that allows algebraic operations to be performed directly on the ciphertext and the result of the computation remains the ciphertext. It is truly a fundamental solution to the problem of confidentiality when delegating data and its operations to a third party [27].

Depending on the ciphertext operations supported, they can be categorized into semi-homomorphic schemes and fully homomorphic schemes. Semi-homomorphic schemes refer to those that support only additive, multiplicative, or a limited number of full homomorphic operations [28]. This is far from sufficient for complex machine learning tasks. It was not until 2009 that Gentry et al. proposed a strictly full homomorphic encryption scheme (FHE) for the first time [29], which provides a broader application prospect for data privacy protection and secure computing.

Currently, the mainstream fully homomorphic encryption schemes include BGV, BFV, GSW, CCKKS, etc.

BFV is similar to BGV, and both of them need to solve the problem of ciphertext dimensionality expansion brought by homomorphic multiplication through key switching. Meanwhile, the BGV scheme needs to control the noise growth by using modulus switching [30]. Unlike the above schemes, the ciphertext form of GSW is a matrix [31]. It does not have the problem of ciphertext multiplication dimension growth. GSW is theoretically simpler, but the performance is not as good as the BGV and BFV. The subsequent TFHE, FHEW, etc., are based on GSW optimization. In order to perform operations on floating-point numbers, Cheon et al. proposed the CKKS scheme to generate approximate results [32].

Suitable FHE algorithms should be selected according to different scenarios. In machine learning, performance, and accuracy are the main factors to be considered. BGV is considered to be the most efficient scheme among the current algorithms. In addition, BGV maps integers to polynomials, which can satisfy the requirement of computational accuracy.

Therefore, we introduce the BGV scheme for global model privacy protection in the federated learning process. First, the developer encrypts the initial model weights and sends them down to the learning nodes. Then, the learning node performs the machine learning task and submits model updates in the ciphertext state. Even if a malicious node or external attacker steals the machine learning results, it cannot obtain meaningful model information. This approach ensures the privacy of federated learning results during computation and transmission.

We implemented the widely used asymmetric cryptography scheme BGV in ZKVM, and the security of this scheme is based on the ring learning with errors (RLWE) problem [33]. Its plaintext space is defined as $R_p = Z_q \frac{[X]}{<\varnothing_m(X)>}$, where $\varnothing_m(X)$ is an m-dimensional cyclotomic polynomial, p is a prime number, and q is a large integer. This article sets the basic parameters m, p, and q of BGV as 16, 33554432, and 1073741824, respectively.

The steps of the BGV scheme are briefly described below:

- $BGV.Setup(1^\lambda)$: Input security parameter λ, and output program parameter $params$.
- $BGV.KeyGen(params)$: Input the $params$ and output the public key pk as well as the private key sk.
- $BGV.Enc_{pk}(msg)$: Input the message $msg \in R_p$ and output the encrypted message $c \in R_q$. This ciphertext consists of two parts, $c = c[0]+c[1]$.
- $BGV.dec_{sk}(c)$: Input ciphertext $c \in R_q$. Output plaintext message msg.
- $BGV.Add(c_a, c_b)$: Let c_a and c_b be the encrypted message of a and b. Homomorphic addition requires only the addition of the corresponding components:

$$c_a + c_b = (c_a[0], c_a[1]) + (c_b[0], c_b[1]) = (c_a[0] + c_b[0], c_a[1] + c_b[1]) = c_{a+b} \quad (1)$$

- $BGV.Mul(c_a, c_b)$: Homomorphic multiplication is computed via the tensor product of the ciphertext vectors and causes the length of the ciphertext to grow exponentially:

$$c_{a \cdot b} := c_a \bigotimes c_b := (c_a[0]c_b[0], c_a[0]c_b[1] + c_a[1]c_b[0], c_a[1]c_b[1]) \quad (2)$$

In order to solve the efficiency problem caused by the growing dimensionality of the ciphertexts, Zvika Brakerski et al. proposed a method called the "relinearization". The decryption operation of a long ciphertext C_1 with a secret key S_1 is converted into a short ciphertext C_2 that is decrypted by a different secret key S_2 [34]. In a recent study, Hiroki Okada et al. proposed that arbitrary binary functions can be realized between ciphertexts by polynomial interpolation [35].

3. System Design

This section provides a detailed description of the proposed framework and workflow.

3.1. System Overview

In this paper, we construct ZKVM on all federated learning nodes based on the RISC Zero project.

Considering the cost of zk-proof generation and verification, we only implement the machine learning algorithm as a guest program and generate zk-proofs for it. Eventually, when the aggregator receives multiple model updates and proofs, it will use the verified model updates to aggregate a new global model.

As shown in Figure 2, we describe the framework in terms of two components of federated learning. The local view includes the components running on the distributed learning nodes. The global view considers the central aggregator that communicates with all learning nodes.

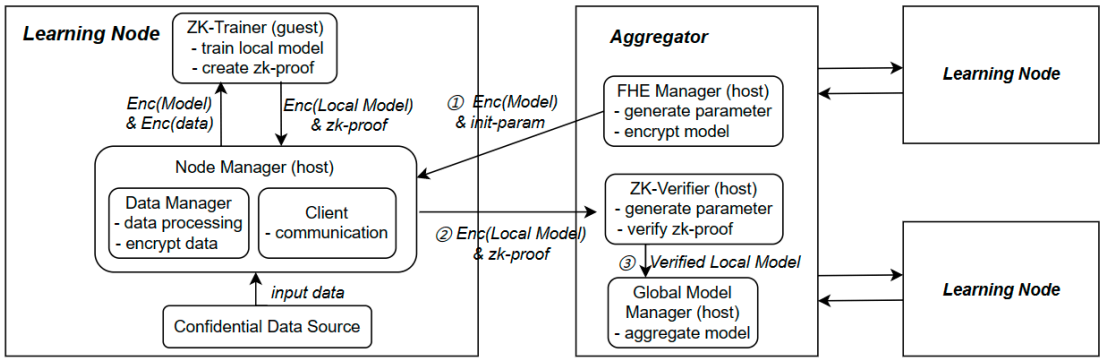


Figure 2. The proposed federated learning framework.

The federated learning process can be summarized in three steps. 1. Learning nodes read encrypted model weights and initialization parameters. 2. Learning nodes perform training tasks to output the ciphertext local model and zk-proof. 3. The aggregator uses the verified local model to aggregate a new global model. The detailed workflow is shown in Figure 3.

(1) Aggregator: The aggregator consists of three components, ZK-Verifier, FHE Manager, and Global Model Manager. It is assumed that the aggregator is a trusted node managed by the task initiator; the aggregation process does not need zero-knowledge proof. In order to reduce the cost of generating proof, all three components are implemented as host programs.

FHE Manager: It is responsible for initializing the BGV scheme. Homomorphic encryption of the global model makes the model information invisible during transmission and local learning.

ZK-Verifier: It is responsible for generating ZKP public parameters such as asymmetric keys, which are used to generate proofs for learning nodes during the machine learning process. It receives and verifies the proof submitted by the learning node.

Global Model Manager: Receiving model updates from all learning nodes, the verified local models are aggregated to produce a new global model.

(2) Learning Nodes: The learning node consists of three components, Data Source, Node Manager, and ZK-Trainer. A learning node becomes a malicious node in a federated learning task when it is attacked. In order to validate the local models submitted by the learning nodes, we implement the zk-Trainer component as a guest program.

Data Source: It is responsible for the storage and management of local privacy data.

Node Manager: It is responsible for the interaction between components. It implements a client that receives ZKP and FHE public parameters from the aggregator and initializes the local execution environment. The data management module reads training

data from the data source, preprocesses, and encrypts it. Finally, the local model updates are forwarded to the central aggregator via Client.

ZK-Trainer: The machine learning task is implemented as a zero-knowledge computation, using encrypted global model and data as input. In addition to outputting the local model, zk-trainer generates a proof for the integrity of this local training process.

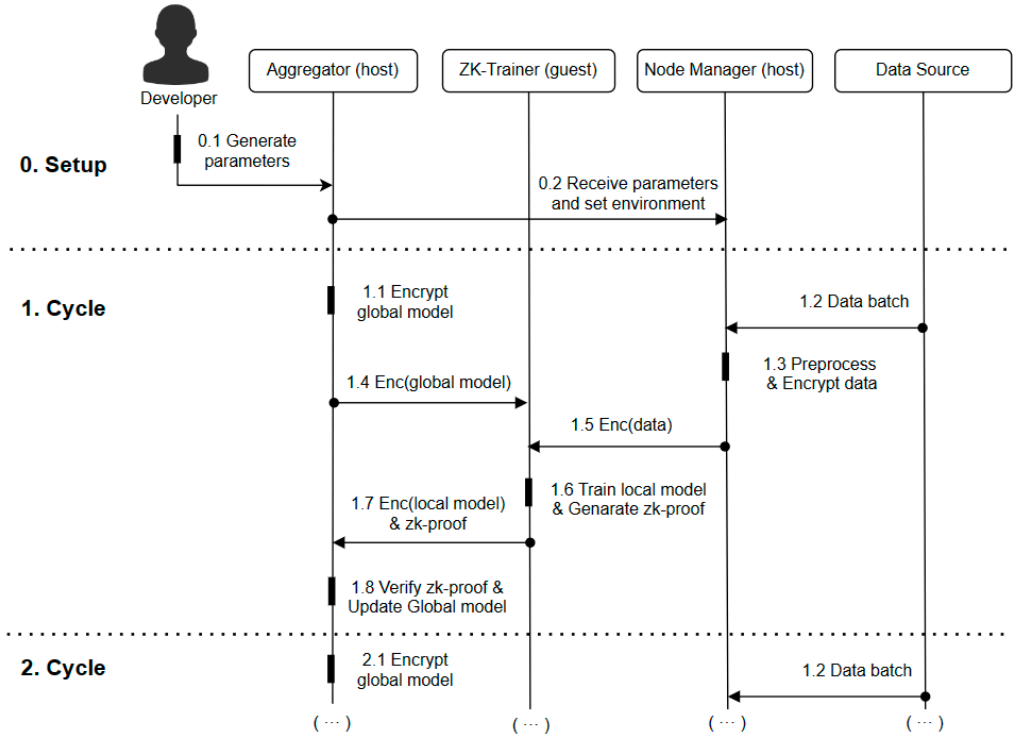


Figure 3. Workflow.

3.2. Workflow

As shown in Figure 3, the workflow of the system consists of a setup phase and a repeating update cycle. For simplicity, the aggregator is represented as a single component.

Setup: In the setup phase, the developer deploys the ZKVM application on all the nodes and initializes the parameters of the BGV scheme. In the federated learning process, the ZK-Trainer is created as the prover and the aggregator as the verifier.

Cycles: At the beginning of each update cycle, the aggregator first homomorphically encrypts the global model (1.1). The node manager of the learning node reads private data from the local data source (1.2). The data are cleaned and sensitive information is removed before homomorphic encryption (1.3). The zk-trainer reads the latest global model from the aggregator (1.4) and receives the processed training data from the node manager (1.5). The zk-trainer performs the zero-knowledge machine learning task to generate the local model and proof (1.6). The proof is sent to the aggregator (1.7) along with the local model. The aggregator first validates the proof with the validation key and then updates the global model with the validated local model (1.8). In successive update cycles, the learning nodes always update the global model in a ciphertext state, thus enabling the protection of global model privacy. At the end of federated learning, only the developer decrypts the global model using the FHE private key to obtain the real machine learning results.

4. Experiments

This section comprehensively evaluates the proposed framework through experiments.

First, we analyze the framework in terms of universality, flexibility, and security. Secondly, we designed the federated learning task and measured the required computational cost and model performance. The detailed experimental environment is shown in Table 3.

Table 3. Software and hardware of the experimental environment.

Software and Hardware	Detailed Information
Rust	1.71.1 (ubuntu20.04)
CPU	12 vCPU Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50 GHz
Random Access Memory	16G
Hard Disk	25G
PyTorch	1.11.0
Cuda	11.3

4.1. Framework Analysis

This section analyzes the proposed framework in terms of universality, flexibility, and security.

4.1.1. Universality

As shown in Table 4, we summarize the recent research.

Table 4. Zero-knowledge proofs combined with machine learning algorithms. Linear regression (LR); support vector machine (SVM); differential privacy (DP); neural network (NN); stochastic gradient descent (SGD).

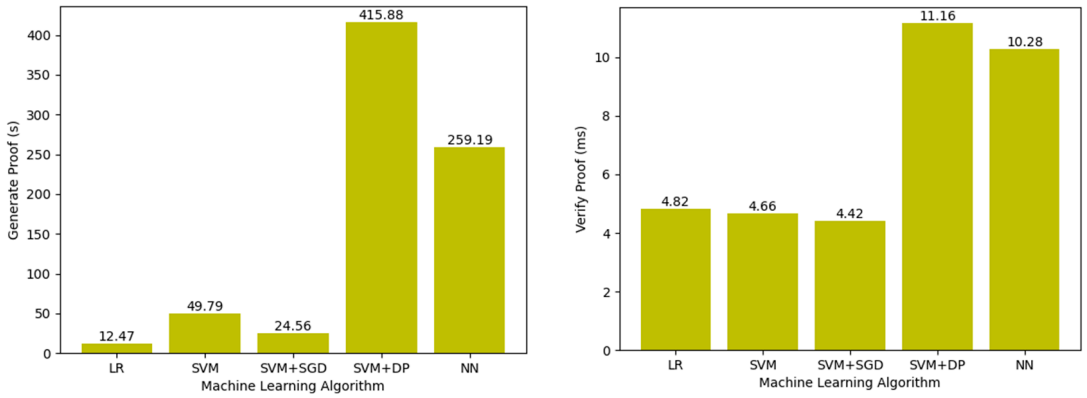
Paper	LR	SVM	DP	NN	SGD
[17]		√			
[36]				√	√
[8]	√				
[37]				√	
[38]			√		
[39]	√	√			
Our	√	√	√	√	√

We use √ to indicate which machine learning algorithms have been implemented in the literature.

Most of the articles utilize circuit languages such as zk-snarks to generate verifiable proofs for the training process of learning nodes. To further implement the incentives, Heiss J. et al. implemented the verification process as smart contracts on the blockchain using zokrates [37]. However, in these studies, developers need to learn specific zk languages and redesign the machine learning algorithms. This development approach severely hinders the integration of applications with zero-knowledge proof techniques. For the first time, we introduce ZKVM in the federated learning framework to achieve zero-knowledge proof for generalized machine learning algorithms.

In order to demonstrate the universality of the framework proposed in this article, we generated proofs for all the machine learning algorithms mentioned above and verified it.

As shown in Figure 4, we summarize the time cost spent by different machine learning algorithms with the same training samples and computational size. It is obvious that the latter two tasks cost more time to generate and verify the proof. This is since differential privacy (DP) introduces additional stochastic algorithms and noise-adding operations. Similarly, the complex structure and backpropagation of neural networks also lead to an increase in time cost.



(**a**) (**b**)

Figure 4. The generation and verification time of proof with different machine learning algorithms: (**a**) generation time of zk-proof; (**b**) verification time of zk-proof.

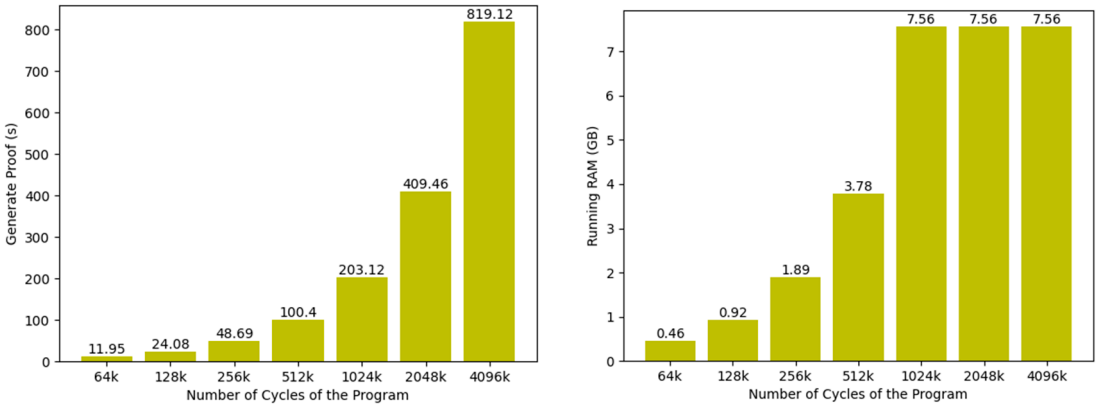
4.1.2. Flexibility

The training and inference of artificial intelligence models require significant hardware resources. However, federated learning nodes with private data are usually ordinary users or organizations that lack high-performance devices [40]. We find that there is still a lack of discussion on how to perform zero-knowledge proofs for operations of arbitrary size.

In the framework proposed in this article, the continuation mechanism automatically divides large programs into smaller parts that are independently calculated and proven. These proofs achieve verification of computational integrity for arbitrary programs on memory-limited devices.

To demonstrate the flexibility of the framework, we gradually increase the cycles of the guest programs. The cycle is the smallest unit computed in the ZKVM circuit, analogous to a clock cycle on a physical central processing unit (CPU). The complexity of a program's execution is measured in cycles as they directly affect the memory, proof size, and time performance of the ZKVM.

As shown in Figure 5, the time required to generate proofs increases linearly as the cycles of the guest program increase. However, the RAM consumed by the program no longer varies when cycles are greater than 1024k. Only when cycles are less than 1024k, the RAM increases linearly. This is because the framework proposed in this paper splits applications with cycles larger than 1024k into smaller computation parts, which limits the memory consumption to 8G. The split part still needs corresponding CPU time to generate proofs, so it does not affect the generation time of zk-proof.



(**a**) (**b**)

Figure 5. The generation time and running RAM with different numbers of cycles: (**a**) generation time of zk-proof; (**b**) running RAM.

Overall, the framework is not limited by the size of computing. No matter how long it takes, ZKVM can always generate zk-proof for arbitrary programs under memory constraints.

4.1.3. Security

This section analyzes the security of the framework under different malicious behaviors. The proposed framework in this paper is built on the premise that the training data is trustworthy. This means that the learning node correctly collects and manages local private data.

When the developer designs the federated learning task as a guest program, this program uniquely corresponds to an image ID. The learning node outputs the local model and verifiable proof after executing the guest program. This proof includes information such as image ID, execution trace, and output hash. During this process, malicious nodes may intentionally tamper with the guest program or return incorrect local training results to the aggregator. As shown in Table 5, we analyzed the security of the framework under different malicious behaviors.

Table 5. Security analysis of frameworks under different malicious behaviors.

Malicious Behavior	Security Analysis
Modify Program	If the program is modified before execution, the image ID in the proof will not match what is expected.
Modify Execution	If the execution is modified, then the execution trace will be invalid. For example, run ZKVM in a debugger and start changing random memory.
Modify Output	If the output is modified, then the output hash will not match the hash recorded in the proof.

Overall, the aggregator's verification of proof identifies all behaviors of malicious nodes. Aggregating the new global model using only the verified local model guarantees the correctness of the federated learning results.

4.2. Federated Learning Performance

The federated learning framework proposed in this paper is generic and applicable to any distributed machine learning task. To gain insight into the performance of this framework, we implemented a feedforward neural network with a single hidden layer [41], whose structure is shown in Figure 6, and the sizes of the input and hidden layers are set to 6 and 10, respectively.

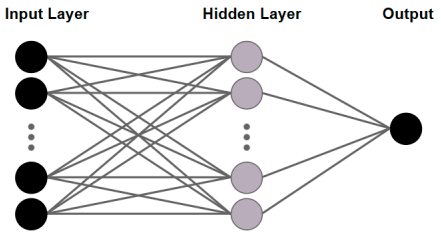


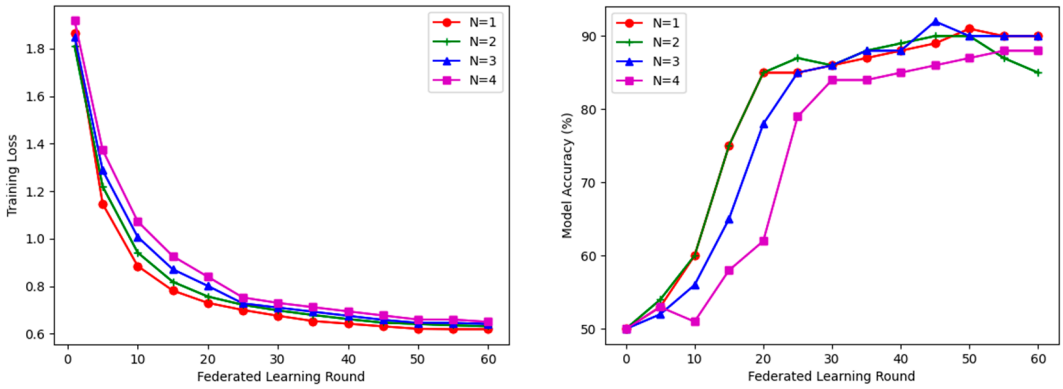
Figure 6. Feedforward neural network with one hidden layer.

4.2.1. Model Accuracy

We construct a binary classification task after screening the classical IRIS dataset. First, we randomly distribute the dataset on N federated learning nodes, and then perform the local learning process in the ciphertext state, and finally use the average aggregation algorithm to obtain the federated learning results. We measured the model accuracy under different numbers of learning nodes.

Since ZKVM is essentially a virtual machine, only encryption and homomorphic operations have an impact on the model performance. As shown in Figure 7, with different parameter configurations, the global model was able to successfully converge and achieve high accuracy as the number of federated learning rounds increased. For example, the

model achieves a 90% accuracy when the number of joint learning nodes is set to either 1 or 3. This is considered a satisfactory result considering the impact of the FHE scheme on computational accuracy and error. The accuracy of the model can be further improved by constructing neural networks with more complex structures or by using more appropriate aggregation algorithms.



(**a**) (**b**)

Figure 7. The loss and accuracy of the global model trained with different numbers of learning nodes: (**a**) model loss; (**b**) model accuracy.

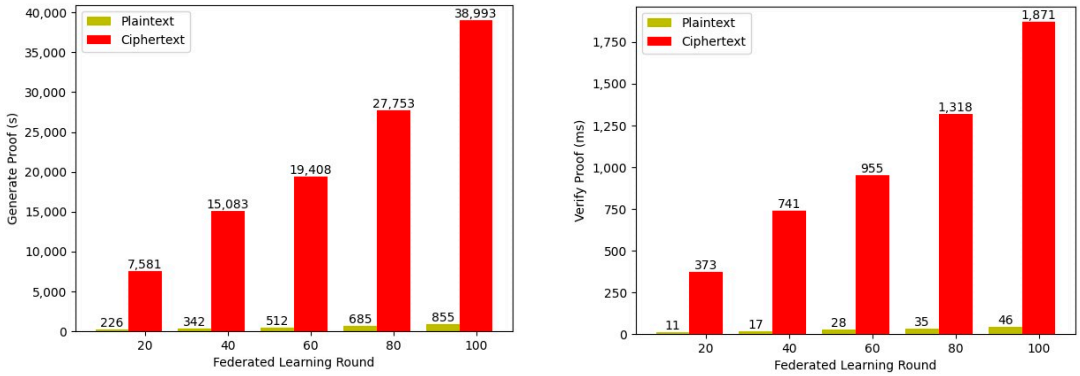
4.2.2. Computational Cost

In addition to model performance, computational cost is crucial for the evaluation of federated learning frameworks.

We encrypt each neuron in the model as two polynomials with the highest power of four. Then, forward propagation and backpropagation are performed in plaintext and ciphertext states, respectively, to update the model information. We implemented these machine learning tasks as guest programs in ZKVM that need to be proved. Then, the required computational cost was measured under different federated learning rounds.

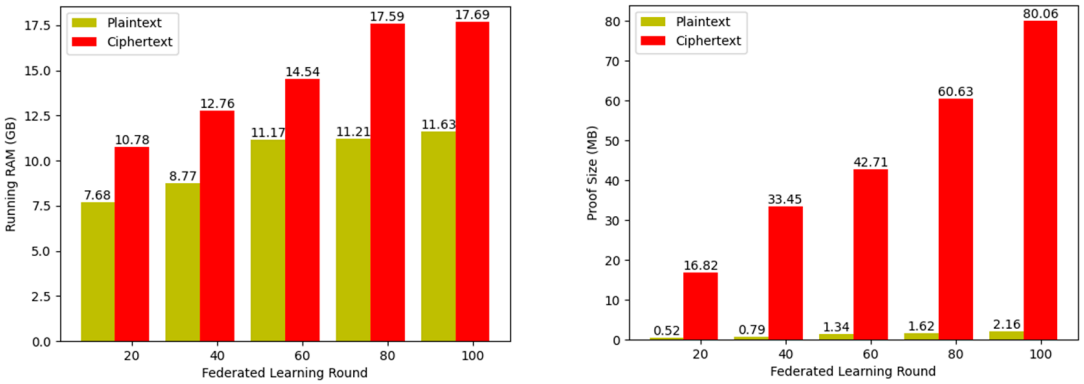
We summarize the relationship between the number of federated learning rounds and the generation and verification time of proof in Figure 8. The learning process in the ciphertext state takes more time to generate and verify proof than the plaintext operations. The reason for this is due to a simple plaintext algebraic operation transformed into many operations between polynomials after homomorphic encryption of the model information. Although generating proof takes a considerable amount of time, the verification time is only in milliseconds.

As shown in Figure 9, the running memory required by the guest program shows a slower growth as the number of federated learning rounds increases. The reason is that model training is a repetitive arithmetic process, and our guest program based on the rust language implementation has a better memory management mechanism, so the running memory changes less. However, due to the increase in computational complexity, the size of the proof increases linearly with the number of federated learning rounds.



(**a**) (**b**)

Figure 8. The generation and verification time of proof with different numbers of federated learning round: (**a**) generation time of zk-proof; (**b**) verification time of zk-proof.



(**a**) (**b**)

Figure 9. The running RAM and proof size with different numbers of federated learning round: (**a**) running RAM; (**b**) proof size.

5. Limitation

The federated learning framework proposed in this paper verifies the integrity of the machine learning process and protects the privacy of the global model. However, this paper still has some limitations.

First, to protect model privacy, we implement the BGV fully homomorphic encryption scheme in ZKVM. We propose to encrypt the initial model and perform local learning in the ciphertext space. This encryption scheme acting on integers has an impact on the computational accuracy, which reduces the results of federated learning. Second, encrypting model weight as polynomials leads to an increase in computation and requires more time and memory for the validation of machine learning tasks. Nevertheless, as mentioned above, our framework still achieves high model accuracy. With future developments in cryptography, we believe the limitations of this framework can be well addressed.

6. Conclusions

In this paper, we propose and implement a federated learning framework based on ZKVM and homomorphic encryption. We implement the federated learning task as a guest program in ZKVM, which verifies the integrity of local model training. In addition, we propose to encrypt the global model, and the learning node outputs the local model in the ciphertext space, which protects the privacy of the global model during training and transmission.

Our framework has broad applicability and can generate zero-knowledge proofs for machine learning tasks of any class and size. At the same time, our framework is highly secure and can effectively identify various behaviors of malicious nodes.

We evaluate the training cost and model performance of federated learning tasks on feedforward neural networks. We find that after implementing machine learning as a ZKP task, it takes more time to generate proof. In contrast, the time required to verify proof is only measured in milliseconds. Furthermore, although homomorphic encryption has an impact on the calculation accuracy, the framework proposed in this paper nevertheless achieves a satisfactory 90% model accuracy.

In the future work, we will work on improving the computational efficiency and accuracy of homomorphic encryption schemes to increase the usability of this framework. In addition, the continuous development of ZKVM technology will help to reduce the time and resources required for generating proofs, which is conducive to further improving the overall efficiency of the framework.

Author Contributions: Methodology, B.Z. and G.L.; investigation, P.Q. and X.G.; writing—original draft, G.L.; writing—review and editing, B.Z. and Y.S.; supervision, B.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant Number 62007024).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Ullah, Z.; Al-Turjman, F.; Mostarda, L.; Gagliardi, R. Applications of artificial intelligence and machine learning in smart cities. *Comput. Commun.* **2020**, *154*, 313–323. [CrossRef]
2. Boobalan, P.; Ramu, S.P.; Pham, Q.V.; Dev, K.; Pandya, S.; Maddikunta, P.K.R.; Gadekallu, T.R.; Huynh-The, T. Fusion of federated learning and industrial Internet of Things: A survey. *Comput. Netw.* **2022**, *212*, 109048. [CrossRef]
3. Wen, J.; Zhang, Z.; Lan, Y.; Cui, Z.; Cai, J.; Zhang, W. A survey on federated learning: Challenges and applications. *Int. J. Mach. Learn. Cybern.* **2023**, *14*, 513–535. [CrossRef]
4. Zhu, J.; Cao, J.; Saxena, D.; Jiang, S.; Ferradi, H. Blockchain-empowered federated learning: Challenges, solutions, and future directions. *ACM Comput. Surv.* **2023**, *55*, 1–31. [CrossRef]
5. Buyukates, B.; He, C.; Han, S.; Fang, Z.; Zhang, Y.; Long, J.; Farahanchi, A.; Avestimehr, S. Proof-of-Contribution-Based Design for Collaborative Machine Learning on Blockchain. *arXiv* **2023**, arXiv:2302.14031.
6. Bellés-Muñoz, M.; Isabel, M.; Muñoz-Tapia, J.L.; Rubio, A.; Baylina, J. Circom: A Circuit Description Language for Building Zero-knowledge Applications. In *IEEE Transactions on Dependable and Secure Computing*; IEEE: Hong Kong, China, 2022.
7. Arun, A.; Setty, S.; Thaler, J. Jolt: SNARKs for Virtual Machines via Lookups. *Cryptol. Eprint Arch.* **2023**. Available online: https://eprint.iacr.org/2023/1217 (accessed on 9 August 2023).
8. Rückel, T.; Sedlmeir, J.; Hofmann, P. Fairness, integrity, and privacy in a scalable blockchain-based federated learning system. *Comput. Netw.* **2022**, *202*, 108621. [CrossRef]
9. Gorantala, S.; Springer, R.; Gipson, B. Unlocking the Potential of Fully Homomorphic Encryption. *Commun. ACM* **2023**, *66*, 72–81. [CrossRef]
10. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
11. Moshawrab, M.; Adda, M.; Bouzouane, A.; Ibrahim, H.; Raad, A. Reviewing Federated Learning Aggregation Algorithms; Strategies, Contributions, Limitations and Future Perspectives. *Electronics* **2023**, *12*, 2287. [CrossRef]
12. Nguyen, D.C.; Pham, Q.V.; Pathirana, P.N.; Ding, M.; Seneviratne, A.; Lin, Z.; Dobre, O.; Hwang, W.-J. Federated learning for smart healthcare: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–37. [CrossRef]
13. Zheng, Z.; Zhou, Y.; Sun, Y.; Wang, Z.; Liu, B.; Li, K. Applications of federated learning in smart cities: Recent advances, taxonomy, and open challenges. *Connect. Sci.* **2022**, *34*, 1–28. [CrossRef]
14. Ghimire, B.; Rawat, D.B. Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things. *IEEE Internet Things J.* **2022**, *9*, 8229–8249. [CrossRef]
15. Li, W.-H.; Zhang, Z.-Y.; Zhou, Z.-B.; Deng, Y. An Overview on Succinct Non-interactive Zero-knowledge Proofs. *J. Cryptol. Res.* **2022**, *9*, 379–447.

16. Ghodsi, Z.; Javaheripi, M.; Sheybani, N.; Zhang, X.; Huang, K.; Koushanfar, F. zPROBE: Zero Peek Robustness Checks for Federated Learning. *arXiv* **2022**, arXiv:2206.12100.
17. Smahi, A.; Li, H.; Yang, Y.; Yang, X.; Lu, P.; Zhong, Y.; Liu, C. BV-ICVs: A privacy-preserving and verifiable federated learning framework for V2X environments using blockchain and zkSNARKs. *J. King Saud Univ.—Comput. Inf. Sci.* **2023**, *35*, 101542. [CrossRef]
18. Dokchitser, T.; Bulkin, A. Zero Knowledge Virtual Machine step by step. *Cryptol. Eprint Arch.* **2023**. Available online: https://eprint.iacr.org/2023/1032 (accessed on 9 August 2023).
19. Bayan, T.; Banach, R. Exploring the Privacy Concerns in Permissionless Blockchain Networks and Potential Solutions. *arXiv* **2023**, arXiv:2305.01038.
20. Bruestle, J.; Gafni, P. RISC Zero ZKVM: Scalable, Transparent Arguments of RISC-V Integrity. Available online: https://dev.risczero.com/proof-system-in-detail.pdf (accessed on 9 August 2023).
21. Cui, E.; Li, T.; Wei, Q. Risc-v instruction set architecture extensions: A survey. *IEEE Access* **2023**, *11*, 24696–24711. [CrossRef]
22. Botrel, G.; El Housni, Y. Faster Montgomery multiplication and Multi-Scalar-Multiplication for SNARKs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2023**, *2023*, 504–521. [CrossRef]
23. Pinkas, B.; Schneider, T.; Tkachenko, O.; Yanai, A. Efficient circuit-based PSI with linear communication. In *Advances in Cryptology—EUROCRYPT 2019, Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, 19–23 May 2019*; Springer International Publishing: Cham, Switzerland, 2019; pp. 122–153.
24. Wang, Y.W.; Wu, J.L. A Privacy-Preserving Symptoms Retrieval System with the Aid of Homomorphic Encryption and Private Set Intersection Schemes. *Algorithms* **2023**, *16*, 244. [CrossRef]
25. Stefanov, E.; Shi, E.; Song, D. Policy-enhanced private set intersection: Sharing information while enforcing privacy policies. In *Public Key Cryptography—PKC 2012, Proceedings of the 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, 21–23 May 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 413–430.
26. Ménétrey, J.; Göttel, C.; Pasin, M.; Felber, P.; Schiavoni, V. An exploratory study of attestation mechanisms for trusted execution environments. *arXiv* **2022**, arXiv:2204.06790.
27. Joshi, B.; Joshi, B.; Mishra, A.; Arya, V.; Gupta, A.K.; Peraković, D. A comparative study of privacy-preserving homomorphic encryption techniques in cloud computing. *Int. J. Cloud Appl. Comput. (IJCAC)* **2022**, *12*, 1–11. [CrossRef]
28. Lin, H.; Chen, C.; Hu, Y. Privacy-protected aggregation in federated learning based on semi-homomorphic encryption. In Proceedings of the 3rd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC 2023), Wuhan, China, 31 March–2 April 2023; Volume 12717, p. 127171J.
29. Gentry, C. *A Fully Homomorphic Encryption Scheme*; Stanford University: Stanford, CA, USA, 2009.
30. Mahato, G.K.; Chakraborty, S.K. A comparative review on homomorphic encryption for cloud security. *IETE J. Res.* **2023**, *69*, 5124–5133. [CrossRef]
31. Gupta, S.; Cammarota, R.; Rosing, T.Š. Memfhe: End-to-end computing with fully homomorphic encryption in memory. In *ACM Transactions on Embedded Computing Systems*; Association for Computing Machinery: New York, NY, USA, 2022.
32. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology—ASIACRYPT 2017, Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017*; Springer International Publishing: Cham, Switzerland, 2017; pp. 409–437.
33. Geelen, R.; Vercauteren, F. Bootstrapping for BGV and BFV Revisited. *J. Cryptol.* **2023**, *36*, 12. [CrossRef]
34. Masahiro, Y. *Fully Homomorphic Encryption without Bootstrapping*; LAP LAMBERT Academic Publishing: Saarbrücken, Germany, 2015.
35. Morimura, K.; Maeda, D.; Nishide, T. Improved integer-wise homomorphic comparison and division based on polynomial evaluation. In Proceedings of the 17th International Conference on Availability, Reliability and Security, Vienna, Austria, 23–26 August 2022; pp. 1–10.
36. Marzo, S.; Pinto, R.; McKenna, L.; Brennan, R. Privacy-Enhanced ZKP-Inspired Framework for Balanced Federated Learning. In *Artificial Intelligence and Cognitive Science, Proceedings of the 30th Irish Conference, AICS 2022, Munster, Ireland, 8–9 December 2022*; Springer Nature: Cham, Switzerland, 2022; pp. 251–263.
37. Heiss, J.; Grünewald, E.; Tai, S.; Haimerl, N.; Schulte, S. Advancing blockchain-based federated learning through verifiable off-chain computations. In Proceedings of the 2022 IEEE International Conference on Blockchain (Blockchain), Espoo, Finland, 22–25 August 2022; pp. 194–201.
38. Zhang, Y.; Tang, Y.; Zhang, Z.; Li, M.; Li, Z.; Khan, S.; Chen, H.; Cheng, G. Blockchain-Based Practical and Privacy-Preserving Federated Learning with Verifiable Fairness. *Mathematics* **2023**, *11*, 1091. [CrossRef]
39. Xing, Z.; Zhang, Z.; Li, M.; Liu, J.; Zhu, L.; Russello, G.; Asghar, M.R. Zero-Knowledge Proof-based Practical Federated Learning on Blockchain. *arXiv* **2023**, arXiv:2304.05590.
40. Abreha, H.G.; Hayajneh, M.; Serhani, M.A. Federated learning in edge computing: A systematic survey. *Sensors* **2022**, *22*, 450. [CrossRef]
41. Chen, Y.; Zhang, C.; Liu, C.; Wang, Y.; Wan, X. Atrial fibrillation detection using a feedforward neural network. *J. Med. Biol. Eng.* **2022**, *42*, 63–73. [CrossRef]