mathematics

Article

Blockchain-Based Practical and Privacy-Preserving Federated Learning with Verifiable Fairness

Yitian Zhang [1,†], Yuming Tang [2,†], Zijian Zhang [2,3], Meng Li [4], Zhen Li [1,3,*], Salabat Khan [5], Huaping Chen [6] and Guoqiang Cheng [6]

[1] School of Computer Science & Technology, Beijing Institute of Technology, Beijing 100081, China
[2] School of Cyberspace Science & Technology, Beijing Institute of Technology, Beijing 100081, China
[3] Southeast Institute of Information Technology, Beijing Institute of Technology, Putian 351100, China
[4] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China
[5] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[6] Qianxin Technology Group Company, Beijing 100044, China
* Correspondence: zhen.li@bit.edu.cn
† These authors contributed equally to this work.

Abstract: Federated learning (FL) has been widely used in both academia and industry all around the world. FL has advantages from the perspective of data security, data diversity, real-time continual learning, hardware efficiency, etc. However, it brings new privacy challenges, such as membership inference attacks and data poisoning attacks, when parts of participants are not assumed to be fully honest. Moreover, selfish participants can obtain others' collaborative data but do not contribute their real local data or even provide fake data. This violates the fairness of FL schemes. Therefore, advanced privacy and fairness techniques have been integrated into FL schemes including blockchain, differential privacy, zero-knowledge proof, etc. However, most of the existing works still have room to enhance the practicality due to our exploration. In this paper, we propose a Blockchain-based Pseudorandom Number Generation (BPNG) protocol based on Verifiable Random Functions (VRFs) to guarantee the fairness for FL schemes. Next, we further propose a Gradient Random Noise Addition (GRNA) protocol based on differential privacy and zero-knowledge proofs to protect data privacy for FL schemes. Finally, we implement both two protocols on Hyperledger Fabric and analyze their performance. Simulation experiments show that the average time that proof generation takes is 18.993 s and the average time of on-chain verification is 2.27 s under our experimental environment settings, which means the scheme is practical in reality.

Keywords: blockchain; verifiable random functions; differential privacy; zero-knowledge proof; federated learning

MSC: 94A60

Citation: 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. https://doi.org/10.3390/math11051091

Academic Editor: Antanas Cenys

Received: 8 January 2023
Revised: 9 February 2023
Accepted: 13 February 2023
Published: 22 February 2023

1. Introduction

The application of machine learning has achieved much success in various fields such as finance, education, healthcare, etc. However, traditional machine learning algorithms need to collect training data in a centralized manner, which brings data privacy problems. Efforts towards decentralized data collection and protecting data privacy have led to federated learning. This has enabled blockchain to become a hotspot in the area of FL recently, because blockchain systems naturally support decentralization. That is, the decentralized feature of blockchain systems has meant that it can be widely used in healthcare, education, finance, and many other fields, including many convergent applications with federated learning.

In the previous scenarios, since the participants cannot fully trust each other, and there is no trusted third party, the participants have to only build and maintain the chain of trust

based on the blockchain consensus protocols and accomplish their own missions using machine learning. When it comes to practical applications, as the focus of blockchain is to ensure the integrity and immutability of information, confidentiality and privacy needs to be guaranteed by designing custom methods, such as cryptographic methods and privacy-preserving methods, according to the needs of a specific scenario. For example, [1] implemented a digit management system based on blockchain and zero-knowledge proof. Benhanmouda et al. [2] introduced secure multi-party computation (SMC) into the blockchain platform to support private data storage. Jia et al. [3] presents a data protection aggregation scheme based on blockchain, differential privacy (DP), and homomorphic encryption.

However, there still exist technical challenges in terms of efficiency and security in the aforementioned works. The ZKP-based FL schemes are hard to apply in complex AI algorithms due to the large size of the proof. The SMC-based FL schemes require the assumption that all the participants are honest, whereas in practice there may be malicious participants, leading to incorrect computation results. The DP-based FL schemes, which add perturbation to the information, lack formal proof of information privacy. Moreover, in practical applications, participants are not always honest. For instance, when not being detected, selfish participants will not provide real data and would like to obtain others' data, because the participants are curious and want to analyze others' privacy according to their data. For example, the models may leak information about the individual data on which they were trained [4]. This challenge prompts the emergence of quality-aware federated learning schemes such as [5].

In order to explore a practical and privacy-preserving solution to the aforementioned technical challenge, we propose two protocols and implement an FL scheme. Our contributions are summarized below:

1. We propose a Blockchain-based Pseudorandom Number Generation (BPNG) protocol, which is a zk-SNARK based blockchain verifiable random function, to enable the participants to generate a verifiable random number. Using BPNG, the participants can generate computationally unpredictable verifiable random numbers.
2. We propose a Gradient Random Noise Addition (GRNA) protocol based on differential privacy and zk-SNARK. Using GRNA, we can prove that the generated random number is indeed computed according to BPNG and that the computation result is a random number satisfying a specific distribution obtained with the seed constructed as we prescribe.
3. We quantitatively evaluate the performance of the proposed protocols and give the performance and privacy analysis of them.

The rest of this paper is organized as follows. Section 2 reviews related works. Section 3 presents our system model, adversarial model, and design goal. Section 4 briefly recalls the definition of verifiable random functions, the zero-knowledge succinct non-interactive argument of knowledge, and differential privacy. We present the two proposed protocols in Section 5. In Section 6 we provide the privacy analysis of our scheme. In Section 7 we show the privacy analysis and performance analysis. Section 8 draws the conclusion.

2. Related Works

In this section, we will review recent works about privacy-preserving federated learning (FL), as well as the application of verifiable fairness in FL.

Various techniques have been studied and integrated into privacy-preserving federated learning, including blockchain, differential privacy (DP), secure multiparty computation (SMC), etc.

Blockchain-based privacy-preserving FL schemes take advantage of blockchain's immutability and integrity. For example, [6] proposed a blockchain-based privacy-preserving FL framework. In this framework, the blockchain was used to interconnect multiple FL components. It used a distributed ledger of transactions to record information flows, where the immutability of the blockchain helped to provide data provenance. Furthermore, under this work, a malicious client assumption can be adapted instead of a semi-honest client

assumption, and it also made the contribution-based incentive mechanisms possible. In [7], LearningChain, a machine learning model which is decentralized and free of a central server, is proposed as a privacy-preserving and secure system. It designs the Stochastic Gradient Descent (SGD) algorithm to be decentralized and uses it to learn a general predictive model on a blockchain platform. An I-nearest aggregation algorithm is presented to defend against potential Byzantine attacks. Pokhrel et al. [8] proposed an autonomous federated learning (FL) scheme based on blockchain, which is applied to efficient and privacy-aware vehicular communication networking. In this scheme, the local model updates of the on-vehicle machine learning model are exchanged and verified distributedly. It then presents a solution to the data provider to perform machine learning processes while not exposing the privacy of data.

Though FL is capable of preventing participants from leaking local data, it is still possible for attackers to learn their personal information by analyzing the uploaded parameters. Differential privacy (DP) provided an approach to prevent information leakage. Wei et al. [9] proposed a novel framework based on DP, in which clients add artificial noise to the local model updates before aggregation. Truex et al. [10] adopts local differential privacy (LDP) into a federated learning system and achieves a formal privacy guarantee. It made the existing LDP protocols applicable in federated learning. However, the aforementioned works mainly focused on how to make existing DP mechanisms applicable in FL frameworks, and at which stage the artificial noise can be added to the private information.

SMC was also used for privacy-preserving FL. Xu et al. [11] proposed an approach named HybridAlpha. It uses an SMC protocol based on functional encryption to implement a privacy-preserving federated learning system. This is the first privacy-preserving federated system that prevents certain inference attacks using functional encryption. Based on a chained SMC technique, a privacy-preserving FL framework termed chain-PPFL was proposed in [12]. A chain-based frame is constructed to enable masked information transferred among the participants under the protection of a single-masking mechanism. Benhamouda et al. [2] explored supporting private data on Hyperledger Fabric by using SMC. In this scheme, peers encrypted their local data before storing it on the ledger and used SMC when the local data were required in a transaction.

There are still several other approaches except for the above techniques. In [13], a privacy-preserving federated learning framework for mobile systems was proposed and implemented. It utilized Trusted Execution Environments (TEES) on both the client's side and the server's side to hide the model updates in the learning algorithms from adversaries. Chen et al. [14] also proposed a scheme based on TEES, in which causative agents can be detected. Jiang et al. [15] proposed a privacy-preserving federated learning scheme with membership proof. The membership proofs are generated by leveraging cryptographic accumulators to accumulate user IDs, then the users can verify the proofs on a public blockchain where they are issued. Sun et al. [16] proposed a privacy-preserving personalized incentive scheme for federated learning. The scheme termed Pain-FL can provide workers with customized payments for the leakage cost of privacy while ensuring the model is well performing. In this contract-based scheme, participants agree on a predefined contract with the server in each training round of FL which includes the level of privacy-preservation and the payment method, and then the worker can receive the rewards after contributing to the model.

3. Problem Statement

3.1. System Model

The system comprises (1) a blockchain platform, (2) a chaincode that implements two protocols that we designed for the participants to invoke during the training process, and (3) the participants of federated learning and their personal data.

All the notations of the parameters mentioned in the system are summarized in Table 1. For the blockchain platform, B represents the blockchain platform where the chaincode is deployed. It can generate $TxBinding, pk, sk$, which denote the random number generated

by the blockchain to identify transactions and a pair of public and private keys, to serve as the inputs for VRF. For the chaincode part, it implements the BPNG and GRNA protocols. The VRF represents the verifiable random functions that we use in the protocols, and it can generate R_p and P_v, which denote the public random number and its proof generated by VRF. For the participants, C_i represents the i_th participant in the system, x_i denotes the gradient computed by C_i. C_i uses the hash function H to generate its private random number r_i and the proof P_i, which is the seed to generate DP noise d_i added to the gradient. r_i is in the range f. x_i' denotes the gradient after adding the DP noise. Finally, we obtain ZK_{pk} and ZK_{vk} representing the proving key and the verifying key for the zk-SNARK proof, respectively.

Table 1. Summary of notations.

Notations	Descriptions
C_i	The i^{th} participant in the system
B	The chaincode deployed on the blockchain
VRF	Verifiable random functions
pk	Public key generated by chaincode
sk	Private key generated by chaincode
TxBinding	A unique representation of a specific transaction, generated as a HEX-encoded string of SHA256 hash using the concatenation of the transaction's nonce, creator, and epoch. Provided by Hyperledger Fabric.
P_v	Proof generated by VRF
R_p	Random number generated by VRF
x_i	Gradient computed by i^{th} participant during the process of machine learning
r_i	Random number generated by i^{th} participant
H	Hash function used by participants to generate r_i
P_i	Proof generated by i^{th} participant for verifying r_i
d_i	Laplace-distributed random number mapped from r_i
f	The range of r_i
x_i'	The gradient after adding Laplace noise
ZK_{pk}	The proving key for zk-SNARK proof generation
ZK_{vk}	The verifying key for zk-SNARK proof verification

3.2. Adversarial Model

Our privacy-preserving federated learning scheme aims at preserving system participants' private input from being exposed or inferred with by others during the training process. So, we set the adversaries to be curious-but-honest, which means the participants run the protocols as we designed but they will probably try to obtain sensitive information from others' public data. The adversaries will not break the protocol execution sequence or have the ability to compromise the blockchain on which the system runs.

3.3. Design Goals

Our goal is to design a practical and privacy-preserving federated learning scheme with verifiable fairness, thus we have the design goals described below.

Privacy: Participants in the federated learning system want to contribute their data and profit from the model without exposing their data, thus they can protect the privacy of their

data sources and meanwhile keep their data competitive. Our scheme should preserve the participants' real private data from being exposed to others.

Security: Participants' identities should be authenticated, and the system should provide data security so that the data cannot be modified to ensure that the participants can verify proofs of others' at any time and receive idempotent results.

Verifiable Fairness: The process of adding noises to the private data should be conducted in a fair manner and be verifiable to all the participants to avoid fake data being uploaded to the model. This is important for system assurance.

Efficiency: The system provides proof of generating efficiency and on-chain verification efficiency. Meanwhile, it should make the communication cost as low as possible.

4. Preliminaries

4.1. Verifiable Random Functions

Verifiable random functions were first presented in [17]. They combine unpredictability and verifiability by extending the construction of pseudorandom functions found in [18]. In brief, it is the public key version of a keyed cryptographic hash. For a specific VRF, the verifiable random number R_p can only be generated by the holder of the private key sk, but it can be verified by anyone who knows the corresponding public key pk. Generally, VRF can be used to provide privacy against offline enumeration attacks on data stored in hash-based data structures. Here, we follow the definition of VRF in [17] as follows:

Definition 1. *Let G, F, and V be polynomial-time algorithms, where*

- *G (the function generator) is probabilistic and its input is the security parameter k. It outputs two binary strings (the public key PK and private key SK);*
- *$F = (F_1, F_2)$ (the function evaluator) is deterministic and its input is two binary strings (SK and the input x to the VRF). It outputs two binary strings (the value $F_1(SK, x)$ of the VRF on x and the corresponding $proof = F_2(SK, x)$);*
- *V (the function verifier) is probabilistic. It receives four binary strings (PK, x, v, and $proof$) as the input, and outputs a bool value, YES or NO.*

VRFs are designed to satisfy the following security properties [19]:

1. **Uniqueness**. For any given public key PK and input α, there is a unique VRF output β that is valid.
2. **Collision Resistance**. Finding two inputs α_1 and α_2 that have the same output β should be computationally impossible.
3. **Pseudorandomness**. Pseudorandomness ensures that if an adversarial verifier receives a VRF output β without its corresponding VRF proof π, then β is indistinguishable from a random value.

Currently, VRF is widely and mainly used in cryptocurrencies, such as Ethereum. It is used to produce a decentralized random beacon, of which the output is unpredictable to anyone until they become available to everyone. One example VRF specification is [19]. It is in a draft hosted by the Internet Research Task Force (IRTF) Crypto Forum Research Group (CFRG). This work-in-progress draft may become a finalized IETF RFC, and the VRF we used in our protocol is an implementation of this draft.

4.2. Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK)

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) refers to a proving system consisting of three algorithms $(Setup, Prove, Verify)$, which allows a prover to convince a verifier that a statement is true without interactions between them [20,21]. The algorithms $(Setup, Prove, Verify)$ are defined as follows [22]:

- $(pk, vk) \leftarrow Setup(C, 1^\lambda)$. With a security parameter λ and a circuit C, the algorithm generates a pair of keys (pk, vk). pk is the proving key and vk is the verification key.

- $\pi \leftarrow Prove(pk, \overrightarrow{s}, \overrightarrow{w})$ generates a proof π, which reflects that the pair $(\overrightarrow{s}, \overrightarrow{w})$ consisting of statement $\overrightarrow{s}$ and witness $\overrightarrow{w}$ is a satisfying assignment for C.
- $True/False \leftarrow Verify(vk, \overrightarrow{s}, \pi)$. Return true if pi is a valid proof for the statement $\overrightarrow{s}$ with the verification key vk and circuit C. Return false otherwise.

Algorithms $(Setup, Prove, Verify)$ should satisfy the following security properties [22]:

1. **Completeness**. For each valid pair of statement $\overrightarrow{s}$ and witness $\overrightarrow{w}$, $Verify(vk, \overrightarrow{s}, \pi)$ always returns true.
2. **Knowledge Soundness**. If $(\overrightarrow{s}, \overrightarrow{w})$ is not a valid pair of assignment for C, then the probability of $Verify(vk, \overrightarrow{s}, \pi) == True$ is negligible.
3. **Succinctness**. The honestly generated proof size should be polynomial in λ, the running time of $Verify(vk, \overrightarrow{s}, \pi)$ should be polynomial in $\lambda + |\overrightarrow{s}|$.

Note that the arithmetic circuits used in zk-SNARK are computed in a finite field, which means that only positive integers less than some large integer are supported in zk-SNARK's circuits, not negative numbers or fractions.

4.3. Differential Privacy

Differential privacy (DP) is a technique that can share information about a dataset in a way that describes data patterns in the dataset. DP can protect the privacy of individuals in it at the same time. The idea behind DP is that the query function can be designed to make the impact of any single substitution in the query request small enough that any individual information cannot be inferred from the query result. There are several different mechanisms of the implementation of DP [23], namely the Laplace mechanism [24], the Gaussian mechanism [25], the geometric mechanism [26], and the exponential mechanism [27]. The key idea to implementing DP is to design a query function that can add random noise following a chosen distribution on the true sensitive dataset [28]. Here, we follow the definition of DP in [29] as follows:

Definition 2. *A randomized function κ gives (ϵ, δ) differential privacy, if for all datasets D_1 and D_2 differ on at most one element, and all $S \subseteq Range(\kappa)$,*

$$Pr[\kappa(D_1) \in S] \leq \exp(\epsilon) \times Pr[\kappa(D_2) \in S] + \delta \tag{1}$$

where $Range(\kappa)$ denoted the range of function κ.

Generally, the two parameters, ϵ and δ, can quantitatively represent the privacy loss. The closer ϵ and δ approach to zero, the less the privacy leaks.

Using the implementation of the Laplace mechanism, DP can be achieved by adding stochastic noise to the result of the query function, which is drawn from a Laplace distribution [24]. The probabilistic density function of Laplace distribution is as Equation (2).

$$pdf(x) = \frac{1}{2\sigma} e^{-|x-\mu|/\sigma}, x \in (-\infty, +\infty) \tag{2}$$

Here, the parameter μ is often set to 0, and the parameter σ has to be determined by the sensitivity, which stands for the greatest influence of any element in the dataset on the result of the query function [29]. Sensitivity is formally defined as Definition 3.

Definition 3. *For $f : D \rightarrow R^d$, the sensitivity of f is*

$$\Delta f = \max_{D_1, D_2} \|f(D_1) - f(D_2)\| \tag{3}$$

for all D_1 and D_2 differing in at most one element.

To achieve differential privacy, σ should be no more than $\Delta f / \epsilon$ [29].

4.4. TxBinding

$TxBinding$ is a parameter we will use in the BPNG protocol, which is provided by the Hyperledger Fabric API. It is a unique representation of a specific transaction, generated as a HEX-encoded string of SHA256 hash using the concatenation of the transaction's nonce, creator, and epoch. According to the source code of Hyperledger Fabric, this API is implemented as below:

$$H(nonce||creator||epoch) = TxBinding \tag{4}$$

The H in the above equation denotes the hash function, which is SHA256. It first concatenates the nonce, creator, and epoch (selecting the highest four bits and lowest four bits of the epoch because of the length limit of unsigned integer in JavaScript) of a specific transaction. Then, it uses SHA256 to hash the concatenated string and return it HEX-encoded.

The chaincode receives the above information from the transaction proposal, which is a data structure that Hyperledger Fabric designed to store the key information about the transaction, such as the nonce, creator, submitter's signature, and so on. According to the design of the transaction in the blockchain, the information it uses to generate this string is different between any two different transactions, which means $TxBinding$ can identify a unique transaction and it is unpredictable until the associated transaction is created.

In a chaincode proposal, the identity of the submitter needs to be authenticated by the peer so that it can be trusted. However, in some scenarios, the chaincode can only check the identity without the proposal submitter. This value can independently authenticate the identity of the transaction's submitter, which means it can be used to defend against replay attacks.

In our scheme, this value is used as a random number generation seed at the beginning of every computation round during the training process.

5. Protocols

5.1. Overview

In this section, we introduce two protocols we designed to protect the privacy of the participants. In Section 5.2, we first designed a Blockchain-based Pseudorandom Number Generation (BPNG) protocol to guarantee that the random numbers generated by each participant in the system are indeed randomly generated and not constructed values. Using the random numbers generated by BPNG, in Section 5.3, we designed a Gradient Random Noise Addition (GRNA) protocol to protect the gradient data of each participant by adding noise to it.

5.2. Blockchain-Based Pseudorandom Number Generation Protocol (BPNG)

The function of this protocol is to guarantee that the generation process of the random numbers generated by the participants is verifiable so that participants can verify that the so-called random numbers are indeed randomly generated but not constructed.

To achieve this goal, we first need to introduce verifiable random functions (VRF) that serve to provide a random number seed to the participants. Before using VRF to generate a verifiable random number, a pair of public and private keys are generated by chaincode, with which we obtain pk and sk. Then, we receive a unique blockchain transaction identifier code $TxBinding$ by invoking the Hyperledger Fabric chaincode API and use it as a seed for VRF. Before every computation process starts, we set up a new transaction on the blockchain and use the information of this transaction to generate the $TxBinding$, so we can ensure that $TxBinding$ is unpredictable before the process starts, which means the seed of VRF is unpredictable. With the parameters above, the VRF is used in the following way:

$$VRF(pk, sk, TxBinding) = (P_v, pk, R_p) \tag{5}$$

P_v, pk, and R_p will be uploaded to the blockchain so that participants can verify the public random number R_p with P_v and pk at any time.

After verifying that R_p is credible, the participants can generate their own random number using the given hash function H with the parameters of R_p and their private gradient x_i. The hash function we used in this scheme is the MIMC hash function provided by Zokrates standard library, which is efficient in circuits. We define the usage of the hash function as follows:

$$H(R_p||x_i) = r_i \tag{6}$$

However, we can find that r_i can still be constructed because other participants have no idea whether the generator of r_i executes the protocol honestly, so we introduce zk-SNARK here. With zk-SNARK, the participants can generate their own proofs P_i to prove that they actually executed the protocol honestly and the randomness of r_i becomes provable. The part of zero-knowledge will be described in the next protocol.

Assume that there are M participants in the system. The flow of the algorithm is summarized as shown in Algorithm 1.

Algorithm 1 BPNG Protocol

Require:
$x_i, TxBinding$
Ensure:
pk, P_v, R_p, r_i
We use $VRF()$ and $VRF\_Verify()$ to generate and verify R_p
Generate a pair of pk and sk, get $TxBinding$ from B
$VRF(pk, sk, TxBinding) = (P_v, pk, R_p)$
Publish R_p, pk, P_v to all C_i
for all $C_i, i \in [1, M]$ **do**
  if $VRF\_Verify(R_p, pk, P_v) == TRUE$ **then**
    $r_i = H(R_p||x_i)$
    return r_i
  else
    return FALSE
  end if
end for

5.3. Gradient Random Noise Addition Protocol (GRNA)

The function of this protocol is to input a uniformly distributed random number, and map that random number to a Laplace-distributed random number given by μ, b, and finally output the final result by summing the resulting random number with the gradient of the input.

The first problem we want to solve is how to generate Laplace-distributed random numbers from uniformly distributed random numbers in zk-SNARK. The generation function is shown below:

$$X = \mu - b\,\mathrm{sgn}(U)\ln 1 - 2|U| \tag{7}$$

where U is a uniformly distributed random number of $(-1/2, 1/2]$, $\mathrm{sgn}(x)$ returns the sign of x, and μ, b are the parameters of Laplace distribution. We need to implement the above function in zk-SNARK.

However, we have two problems to solve. The first is that zk-SNARK does not support nonlinear functions such as log or ln, the second is that zk-SNARK does not support fractional calculation. To solve the first problem, we must fit the object function using a polynomial. Because (7) is an odd function, we just need to fit half of the function. We use the Maclaurin series which is shown below to fit the object function where $(U > 0)$. In Equation (7), U is a uniformly distributed random number between $-1/2$ and $1/2$, so

$2|U|$ will be between 0 and 1, but the random number generated in protocol 1 is a discrete random integer. So, we must replace $2|U|$ with x/f, where x is the discrete random integer in the range of $[0, f)$ and f is a constant that marks the upper limit of x. In our experiments, we set the value of f to 1000. In order to make math expectation of the generated random number equal to 0, we let $x\mu = 0$. For the sake of simplicity, we used the function below to obtain the Maclaurin series.

$$y = -b \ln(1 - x/f), (0 \leq x < f) \tag{8}$$

So, the Maclaurin series of (8) is

$$\sum_{i=1}^{n} \frac{b}{if^i} x^i \tag{9}$$

We can see the coefficient of each item of (9) is b/if^i. However, zk-SNARK does not support fractional calculation, so the way to represent each item in zk-SNARK is $[b/if^i] \times x^i$(if $b > if^i$), or $[x^i/[(if^i)/b]]$(if $b < if^i$). Considering that the computation in zk-SNARK is performed in a finite field and that the verification of $a/b \stackrel{?}{=} c$ is actually the process of verifying that $a \stackrel{?}{=} bc$, if b is divisible by a, then the result of b/a is the same as in the integers, otherwise it is not the same. Meanwhile, we cannot guarantee that x^i can be divided exactly by $[if^i/b]$, so we must make sure that $b > if^i$. Then, the final problem is what we want, x', which is the sum of a Laplace-distributed random number r of given parameter b and raw gradient x.

In our solution, the b_{chosen} is set to $5 * 10^{15}$, which is a huge integer, meanwhile, our target is $x' = x + b_{\text{target}}r$, where r is a Laplace-distributed random number with $\mu = 0, b = 1$, so $b_{\text{target}}r$ is a Laplace-distributed random number with $\mu = 0, b = b_{\text{target}}$, but what we obtain is $b_{\text{chosen}}r$, a Laplace-distributed random number with $\mu = 0, b = b_{\text{chosen}}$. Our solution is to enlarge x' with a multiplier $m = b_{\text{chosen}}/b_{\text{target}}$, so the output value will be $x'm = xm + b_{\text{chosen}}r$. The algorithm is shown as Algorithm 2.

Algorithm 2 GRNA Protocol

Require:
x_i, r_i, d_{target}, d_{encoded}, n, f, ZK_{pk}
Ensure:
m, $x_i'm$, P_i
1: $m = d_{\text{encoded}}/d_{\text{target}}$
2: $p_1, \ldots, p_n = b/if^i$(for $i = 1, \ldots, n$)
3: $r'm = \sum_{i=1}^{n} p_i x^i$
4: $x_i'm = x_i m + r'm$
5: Circuit = GenerateCircuit$(p_1, \ldots, p_n)$
6: P_i = GenerateProof(Circuit, r_i, mx_i, mx_i')
7: **return** m, $x_i'm$, P_i

6. Privacy, Security, and Fairness Analysis

In this section, we prove that our scheme achieves all the privacy, security, and fairness goals we mentioned in Section 3.3.

6.1. Privacy

In BPNG protocol, the protocol is secure if we can ensure that

(1) For each participant, the random number R_p generated by VRF is unpredictable.
(2) Without revealing the random number r_i and input x_i of i^{th} participant, that r_i is generated using R_p and x_i as the hash preimage should be verifiable.

For (1), in the BPNG protocol, R_p is generated by VRF using a pair of pk and sk and the $TxBinding$ provided by the blockchain. Among them, the pair of keys are generated

by chaincode before the computation process starts, and the $TxBinding$ is generated at the setup of a transaction which means it is unpredictable before its publication on the blockchain. So, if the probability of key collision in a finite key generation time is negligible, then R_p is unpredictable because the VRF is preimage-resistant. In BPNG, we use EdDSA to generate the key pair, which means that the protocol's security depends on the security properties of EdDSA, which can meet our needs.

For (2), we can use zk-SNARK to solve this problem, which means that the private parameters are secure as long as zk-SNARK achieves knowledge soundness.

In the GRNA protocol, the protocol is secure if we can ensure that

1. Without revealing the random noise d_i of i^{th} participant, d_i is generated using r_i is verifiable.
2. Without revealing random noise d_i and private input x_i of i^{th} participant, the published x_i' equals $d_i + x_i$ is verifiable.

Obviously, in GRNA, we need some kind of SNARK protocol to meet our needs, so we use zk-SNARK to solve the problems, which means that if zk-SNARK achieves the security properties as it was designed previously, the protocol is secure as well.

We note that there is a zk-SNARK part in both protocols, and we can combine these two parts into one, which will not affect the security of any protocol because of the continuity of the two protocols.

Since the security of the protocols is guaranteed, we need to consider the settings of privacy parameters in our protocols. In general, we want to adjust the settings of our parameters to a condition that can protect participants' privacy while ensuring the usability of the machine learning process.

First, we must explain why we use $\alpha = 0.002 \times 0.998^{epoch}$ as learning rate. As shown in Figure 1, it is obvious that with a given privacy budget $\epsilon = 1$, the lower the learning rate the more likely the model is to converge and the slower the model converges. In order to use the highest possible privacy budget, we set the learning rate to a relatively low $\alpha = 0.002 \times 0.998^{epoch}$.

Second, we need to verify the effect of adding noise on the convergence of the machine learning model. Figure 2 shows the graph of the variation of loss with epochs for different privacy budgets, which shows that the loss function cannot converge until the privacy budget ϵ is big enough.

Third, we want to evaluate the effect of adding noise on privacy protection. We calculate the norm of the un-noised gradient submitted by the model and count its distribution, denoted as D, and denote the distribution of the norm of the noised gradient as D'. Furthermore, we compute the distribution of the difference between the last submitted gradient and the current gradient and denote the one from un-noised gradient D_δ, and the one from noised D'_δ. To show the difference between noised gradient and unnoised gradient, we calculate the KL divergence of D, D' and D_δ, D'_δ under different ϵ settings, the result are shown in Figure 3.

6.2. Security

We implement our scheme using Hyperledger Fabric, which is a permissioned blockchain platform. Unlike other public blockchain platforms, Fabric registers participants on the blockchain before the network starts, and the participants are divided into different channels according to the setup settings, which means they can only access the data inside their channels. After the network starts running, the registered users can access the blockchain data and chaincode in their registered channel using their private keys. So, the participants are all pre-authenticated and their operations will be logged on the blockchain.

Meanwhile, as a blockchain platform, Hyperledger Fabric also makes the data hard to tamper with and thus provides data security. In short, as long as the Hyperledger Fabric achieves the features as it was designed, the security of our scheme can be implemented as well.

6.3. Fairness

We achieve verifiable fairness through the BPNG protocol. According to the BPNG protocol, participants can verify the generation process of the random number r_i, which means participants can find out whether someone is cheating in the learning process, thus implementing verifiable fairness. We provide the analysis of the BPNG protocol in Section 6.1.

6.4. Summary

We presented privacy, security, and fairness analyses. As we stated above, the privacy, security, and fairness of our scheme rely mostly on the cryptographic techniques we use in the two protocols and, for now, these techniques are secure under the conditions we use them. Therefore, we can say that all of the design goals we mentioned in Section 3.3 are implemented.

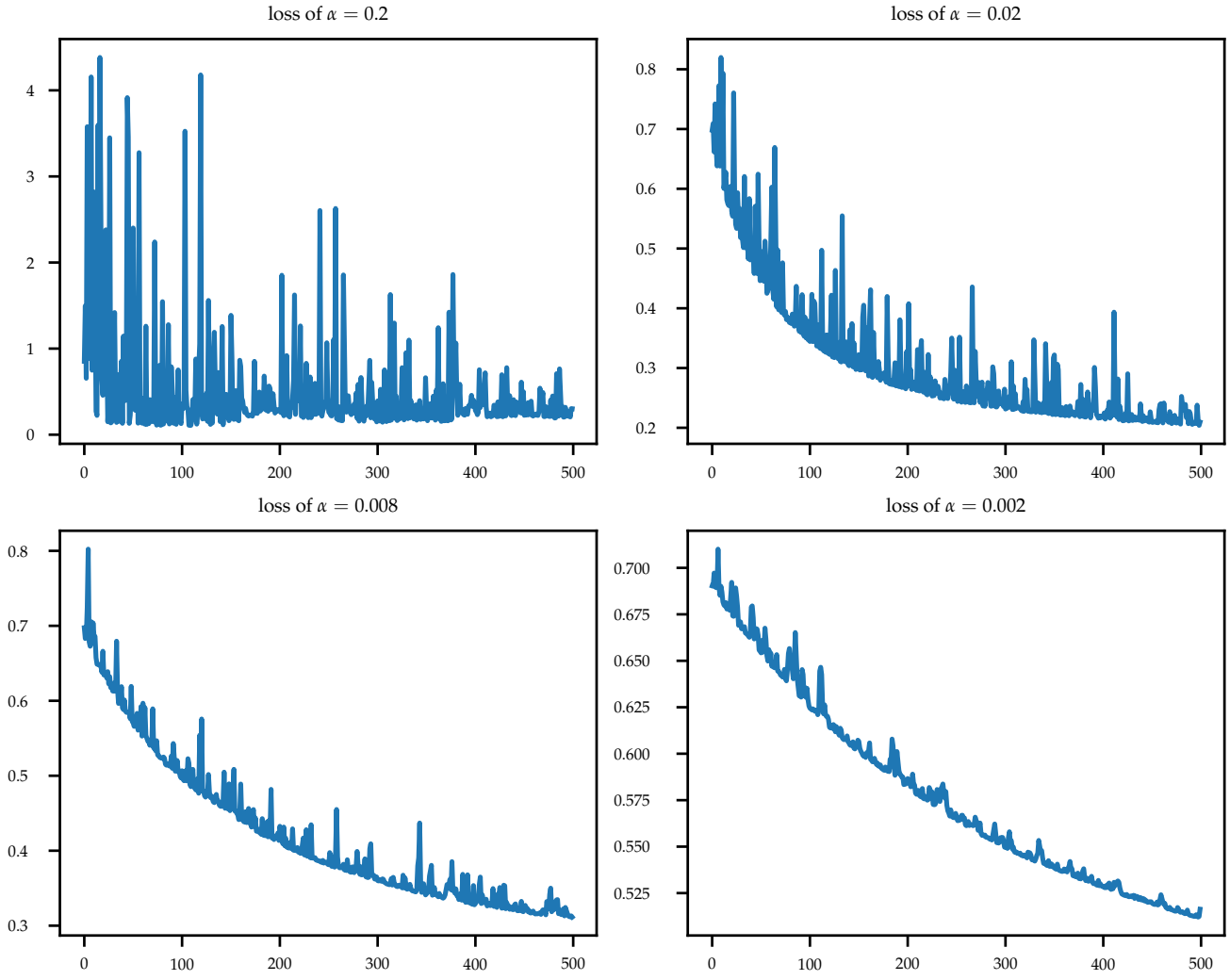

Figure 1. Training convergence under different learning rates.

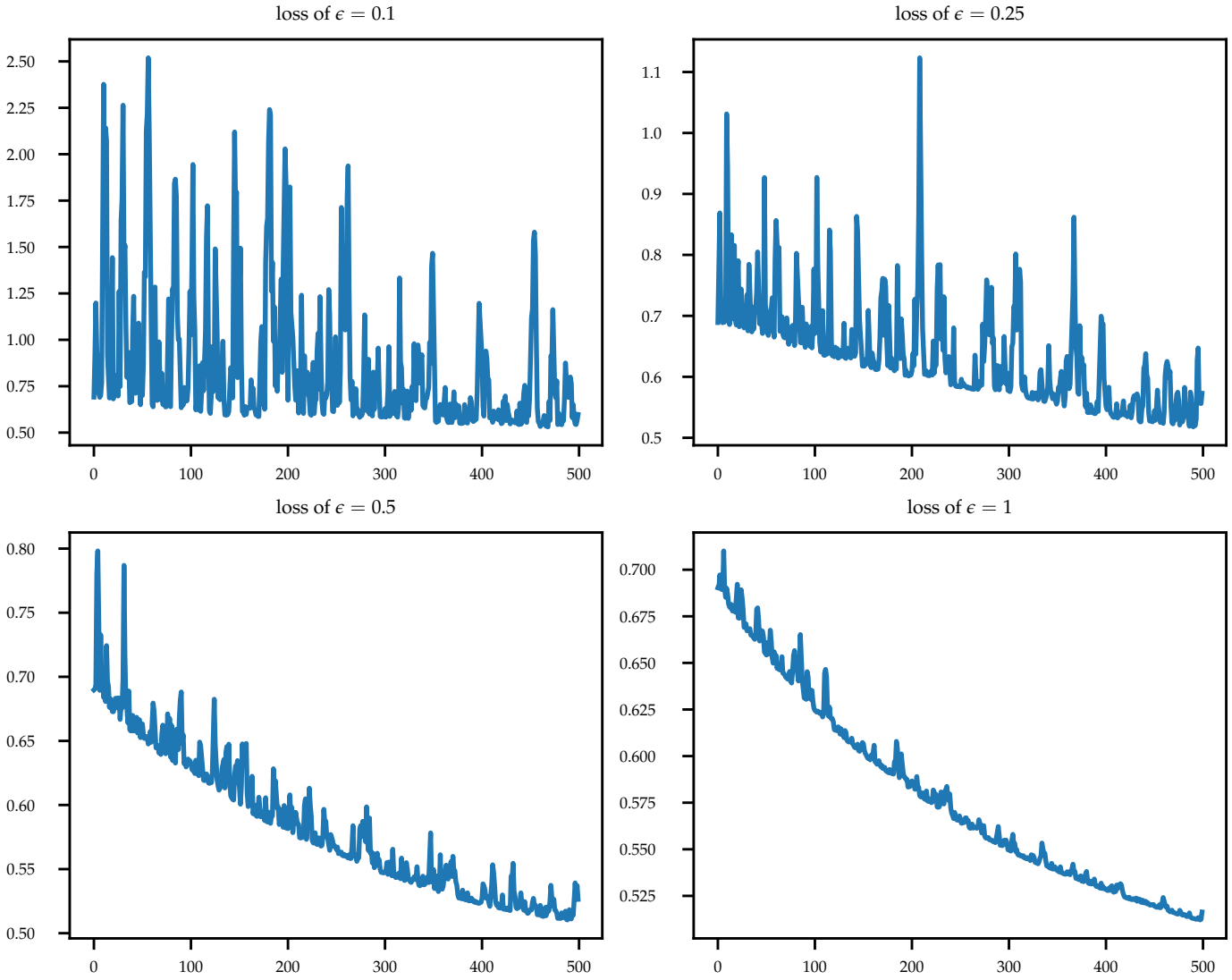

Figure 2. Training convergence under different privacy budgets.

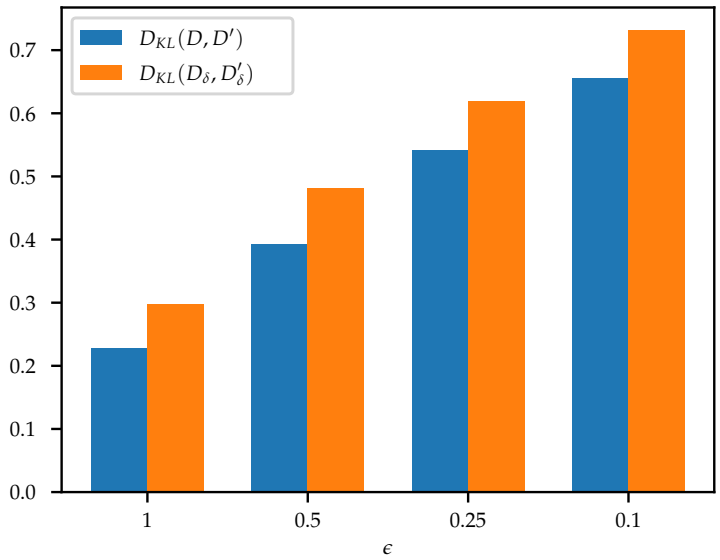

Figure 3. K-L divergence under different privacy budgets.

7. Performance Analysis and Comparison

7.1. Datasets, Parameters, Metrics, Setup

For simplicity, we expanded the sample size of the IRIS dataset from 150 to 4050 and trained on the extended IRIS dataset using a logistic regression model to perform a machine learning process. Because logistic regression is a two-classes model and the IRIS dataset includes three classes, for each evaluation we selected two of the three categories, input them into the model for training, and evaluated their performance.

Like many other federal learning schemes, we used stochastic gradient descent algorithm for learning. The total number of samples in the extended IRIS dataset is 4050 (1350 for each class), so we chose two classes each time, and the total number of samples in the dataset for each time was 2700. We chose 80% of them as the training set and 20% of them as the validation set. We trained the model for 500 epochs every time, and we set the learning rate $a = 0.002 \times 0.998^{epoch}$.

We implemented the application based on Hyperledger Fabric, which is a mature permissioned blockchain. We used the Zokrates framework for zk-SNARK proof generating and verifying. Because Zokrates is implemented in rust which can be compiled into WASM and provides an npm package where rust is a recently emerged programming language, WASM is an assembly language that can be executed in a javascript virtual machine and npm is the official package manager for node-js, we implemented Fabric chaincode in Typescript, and we implemented the local application also in Typescript. To parallelize proof generation that is CPU-intensive, we executed multiple subprocesses to generate zk-SNARK proof simultaneously.

We implemented this application using Hyperledger Fabric version 2.3, nodejs version 17.8.0, Zokrates-js version 1.0.39, running on docker versioning 20.10.14.

We conducted our performance analysis on a PC that has an Intel Core i7-8550U CPU and 16G memory running Linux 5.17.1. However, as we ran the blockchain platform in a docker container, the computing resources are limited by docker which means the performance is restricted by docker and we cannot know exactly how many computing resources it holds according to the settings of docker. We can only provide results under the circumstance of docker due to the Hyperledger Fabric setup requirements, the result under real production circumstances needs to be further tested or estimated.

7.2. Result

We evaluated the time taken to generate proofs and on-chain verification by submitting 150 gradients and counting the time spent. The result is shown in Figure 4. The average time proof generating takes is 18.993 s, and the average time of on-chain verification is 2.27 s. It shows that most of the running time is spent on proof generation.

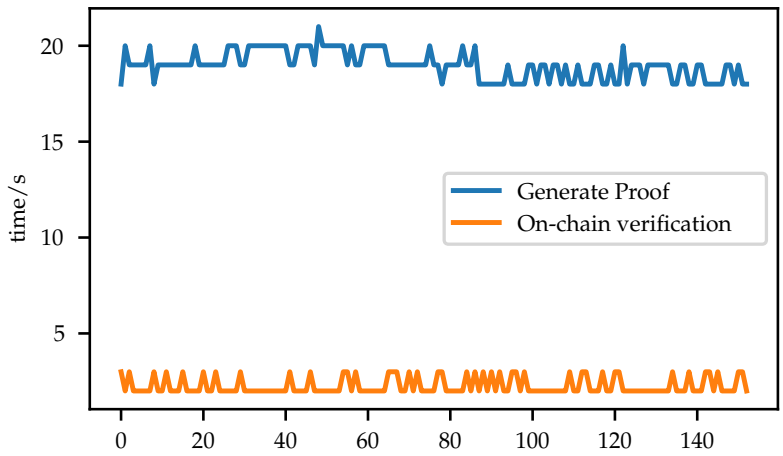

Figure 4. Performance analysis.

The result may seem not practical for a federated learning scheme, but we find that several factors in the test environment settings have a relatively large impact on the performance result. First, we conducted the performance analysis on a laptop that has limited computing resources, which means that our running circumstance is much worse than in the real application. Second, due to the setup requirements of Hyperledger Fabric, we had to run multiple containers in docker to create a Hyperledger Fabric network structure, which greatly limits the speed of computing operation. Third, most of the running time was spent on the proof generation process using Zokrates. We find that Zokrates is more efficient when running outside docker or using its implementation in other programming languages, which means the efficiency of proof generation can be further improved.

According to our estimation, the scheme is practical in real-world applications because the above problems can be solved. We can set up the Hyperledger Fabric network using different devices and aside from the federated learning machines, so there will be no resource limitation by the docker containers and it will be more efficient when assigning different types of jobs to different devices, and meanwhile the proof generation can perform more efficiently.

7.3. Comparison with Existing Approach

We compared our schemes with that of [30], which is another blockchain-based federated learning scheme that uses zero-knowledge proof techniques, but its main goal is to solve the problem of poisoning attacks. Both our scheme and scheme [30] are also blockchain-based federated learning schemes, both use Zokrates as a zero-knowledge proof solution, and both implement complex zero-knowledge proof circuits associated with machine learning algorithms. However, our schemes also have a lot of differences. In our scheme, all that needs to be proven is that the random numbers we use in DP are indeed random; in [30], all that needs to be proven is that the generated gradients are indeed computed by the correct algorithm. Our scheme uses Hyperledger Fabric, and the smart contract for verifying zero-knowledge proofs is implemented using WASM and runs in the native OS. The blockchain platform [30] uses is Ethereum, and the smart contract for verifying zero-knowledge proofs is implemented using EVM assembly [31] and runs in the EVM [31]. In a test with batch size of 10, our scheme takes about 20 s to generate the proof and 3 s to complete the verification, while [30] takes 8 s to generate the proof and 37 s to complete the verification. Considering the differences in the problems solved by the two solutions, the blockchain used and the hardware used, this comparison is for reference only.

7.4. Summary

We used a simple machine learning model to validate our proposed federal learning approach. In order to make the model converge successfully even at higher privacy budgets, we analyzed the effect of varying the learning rate on the convergence of the model at the same privacy budget, and we found that the lower the learning rate, the easier the model converges. We also tested the effect of different privacy budgets on the convergence of the model given a low learning rate.

Moreover, we conducted our performance analysis and found that most of the running time was spent on the proof generation process of Zokrates.

8. Conclusions

In this paper, we have proposed a permissioned blockchain-based federated learning method that protects privacy by adding Laplace-distributed noise to the gradients submitted by federated learning participants. We use zero-knowledge proof to guarantee that the gradients that participants submit are generated from a real value not randomly generated, and the noise added to the gradient is Laplace-distributed not deliberately selected. Experimental analysis shows the machine learning model's convergence under

different learning rates and privacy budgets and the performance of our scheme. Future efforts will focus on applying this scheme to vertical federal learning.

Author Contributions: Conceptualization, Z.Z.; Formal analysis, Y.Z.; Funding acquisition, H.C. and G.C.; Methodology, Z.Z., M.L. and S.K.; Software, Y.Z. and Y.T.; Supervision, Z.Z.; Validation, Y.T. and Z.L.; Visualization, Z.L.; Writing—original draft, Y.Z., Y.T. and Z.L.; Writing—review & editing, Z.Z., M.L., S.K., H.C. and G.C. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is supported by High-performance Reliable Multi-party Secure Computing Technology and Product Project for Industrial Internet No.TC220H056.

Data Availability Statement: Not applicable.

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

References

1. Yang, X.; Li, W. A zero-knowledge-proof-based digital identity management scheme in blockchain. *Comput. Secur.* **2020**, *99*, 102050. [CrossRef]
2. Benhamouda, F.; Halevi, S.; Halevi, T. Supporting private data on Hyperledger Fabric with secure multiparty computation. *IBM J. Res. Dev.* **2019**, *63*, 3. [CrossRef]
3. Jia, B.; Zhang, X.; Liu, J.; Zhang, Y.; Huang, K.; Liang, Y. Blockchain-Enabled Federated Learning Data Protection Aggregation Scheme With Differential Privacy and Homomorphic Encryption in IIoT. *IEEE Trans. Ind. Inform.* **2022**, *18*, 4049–4058. [CrossRef]
4. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 3–18. [CrossRef]
5. Deng, Y.; Lyu, F.; Ren, J.; Chen, Y.C.; Yang, P.; Zhou, Y.; Zhang, Y. FAIR: Quality-Aware Federated Learning with Precise User Incentive and Model Aggregation. In Proceedings of the IEEE INFOCOM 2021—IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021; pp. 1–10. [CrossRef]
6. Awan, S.; Li, F.; Luo, B.; Liu, M. Poster: A Reliable and Accountable Privacy-Preserving Federated Learning Framework Using the Blockchain. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 2561–2563. [CrossRef]
7. Chen, X.; Ji, J.; Luo, C.; Liao, W.; Li, P. When Machine Learning Meets Blockchain: A Decentralized, Privacy-preserving and Secure Design. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 1178–1187. [CrossRef]
8. Pokhrel, S.R.; Choi, J. Federated learning with blockchain for autonomous vehicles: Analysis and design challenges. *IEEE Trans. Commun.* **2020**, *68*, 4734–4746. [CrossRef]
9. Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.S.; Poor, H.V. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. [CrossRef]
10. 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; Association for Computing Machinery: New York, NY, USA, 2020; pp. 61–66. [CrossRef]
11. Xu, R.; Baracaldo, N.; Zhou, Y.; Anwar, A.; Ludwig, H. HybridAlpha: An Efficient Approach for Privacy-Preserving Federated Learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, London, UK, 15 November 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 13–23. [CrossRef]
12. 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.* **2021**, *8*, 6178–6186. [CrossRef]
13. Mo, F.; Haddadi, H.; Katevas, K.; Marin, E.; Perino, D.; Kourtellis, N. PPFL: Privacy-Preserving Federated Learning with Trusted Execution Environments. In Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services, Virtual, 24 June–2 July 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 94–108. [CrossRef]
14. Chen, Y.; Luo, F.; Li, T.; Xiang, T.; Liu, Z.; Li, J. A training-integrity privacy-preserving federated learning scheme with trusted execution environment. *Inf. Sci.* **2020**, *522*, 69–79. [CrossRef]
15. Jiang, C.; Xu, C.; Zhang, Y. PFLM: Privacy-preserving federated learning with membership proof. *Inf. Sci.* **2021**, *576*, 288–311. [CrossRef]
16. Sun, P.; Che, H.; Wang, Z.; Wang, Y.; Wang, T.; Wu, L.; Shao, H. Pain-FL: Personalized Privacy-Preserving Incentive for Federated Learning. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3805–3820. [CrossRef]
17. Micali, S.; Rabin, M.; Vadhan, S. Verifiable random functions. In Proceedings of the 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039), New York, NY, USA, 17–18 October 1999; pp. 120–130. [CrossRef]
18. Goldreich, O.; Goldwasser, S.; Micali, S. How to construct random functions. *J. ACM (JACM)* **1986**, *33*, 792–807. [CrossRef]

19. Goldberg, S.; Reyzin, L.; Papadopoulos, D.; Včelák, J. Verifiable Random Functions (VRFs). Internet-Draft draft-irtf-cfrg-vrf-11, Internet Engineering Task Force, 2022. Work in Progress. Available online: https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-vrf-11 (accessed on 6 January 2023).
20. Ben-Sasson, E.; Chiesa, A.; Tromer, E.; Virza, M. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14), San Diego, CA, USA, 20–22 August 2014; pp. 781–796.
21. Groth, J. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In Proceedings of the Advances in Cryptology-ASIACRYPT 2010—16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, 5–9 December 2010; Abe, M., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6477; pp. 321–340.
22. Parno, B.; Howell, J.; Gentry, C.; Raykova, M. Pinocchio: Nearly Practical Verifiable Computation. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, 19–22 May 2013; pp. 238–252. [CrossRef]
23. Ouadrhiri, A.E.; Abdelhadi, A. Differential Privacy for Deep and Federated Learning: A Survey. *IEEE Access* **2022**, *10*, 22359–22380. [CrossRef]
24. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A.D. Calibrating Noise to Sensitivity in Private Data Analysis. In Proceedings of the Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, 4–7 March 2006; Halevi, S., Rabin, T., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 3876, pp. 265–284.
25. Dong, J.; Roth, A.; Su, W.J. Gaussian Differential Privacy. *arXiv* **2019**, arXiv:1905.02383.
26. Ghosh, A.; Roughgarden, T.; Sundararajan, M. Universally Utility-maximizing Privacy Mechanisms. *SIAM J. Comput.* **2012**, *41*, 1673–1693. [CrossRef]
27. McSherry, F.; Talwar, K. Mechanism Design via Differential Privacy. In Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), Providence, RI, USA, 20–23 October 2007; IEEE Computer Society: Washington, DC, USA, 2007; pp. 94–103.
28. Dwork, C.; Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* **2014**, *9*, 211–407. [CrossRef]
29. Dwork, C.; Lei, J. Differential privacy and robust statistics. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, 31 May–2 June 2009; Mitzenmacher, M., Ed.; ACM: New York, NY, USA, 2009; pp. 371–380.
30. 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. [CrossRef]
31. Buterin, V. Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform. 2013. Available online: https://github.com/ethereum/wiki/wiki/White-Paper (accessed on 6 January 2023).