

Article

# Mathematical Proposal for Securing Split Learning Using Homomorphic Encryption and Zero-Knowledge Proofs

Agon Kokaj<sup>1</sup> and Elissa Mollakuqe<sup>2,\*</sup><sup>1</sup> University "Fehmi Agani" Gjakove, 50000 Gjakove, Kosovo; agon.kokaj@uni-gjk.org<sup>2</sup> System Security Laboratory, Darmstadt Technical University, 64289 Darmstadt, Germany

\* Correspondence: elissa.mollakuqe@tu-darmstadt.de

**Abstract:** This work presents a mathematical solution to data privacy and integrity issues in Split Learning which uses Homomorphic Encryption (HE) and Zero-Knowledge Proofs (ZKP). It allows calculations to be conducted on encrypted data, keeping the data private, while ZKP ensures the correctness of these calculations without revealing the underlying data. Our proposed system, HavenSL, combines HE and ZKP to provide strong protection against attacks. It uses Discrete Cosine Transform (DCT) to analyze model updates in the frequency domain to detect unusual changes in parameters. HavenSL also has a rollback feature that brings the system back to a verified state if harmful changes are detected. Experiments on CIFAR-10, MNIST, and Fashion-MNIST datasets show that using Homomorphic Encryption and Zero-Knowledge Proofs during training is feasible and accuracy is maintained. This mathematical-based approach shows how crypto-graphic can protect decentralized learning systems. It also proves the practical use of HE and ZKP in secure, privacy-aware collaborative AI.

**Keywords:** homomorphic; encryption; zero-knowledge proofs; mathematical approaches; securing; privacy



Academic Editor: Nuno Silva

Received: 23 December 2024

Revised: 20 February 2025

Accepted: 23 February 2025

Published: 7 March 2025

**Citation:** Kokaj, A.; Mollakuqe, E. Mathematical Proposal for Securing Split Learning Using Homomorphic Encryption and Zero-Knowledge Proofs. *Appl. Sci.* **2025**, *15*, 2913. <https://doi.org/10.3390/app15062913>

**Copyright:** © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In cryptography, data privacy and security during training and inference are key [1]. Decentralized learning systems like Split Learning need new ways to protect against adversarial attacks, especially in collaborative learning environments where multiple parties interact without sharing raw data [2]. This work addresses these concerns by combining Homomorphic Encryption and Zero-Knowledge Proofs, two powerful cryptographic techniques for privacy-preserving computations in distributed learning [3–5]. This is especially useful in Split Learning where clients need to collaborate with servers without exposing their data. Recent advances in HE like CKKS and BFV have shown to be effective in keeping privacy while allowing complex arithmetic operations on encrypted data [6,7]. These schemes have been used in privacy-preserving machine learning, especially in applications like healthcare and finance where data are sensitive [8,9]. Zero-knowledge proofs are a way to verify computations without revealing the underlying data [10–12]. This is crucial in decentralized learning, where it is necessary to ensure that computations performed by clients and servers are accurate and free from tampering. This paper utilizes Homomorphic Encryption (HE) and Zero-Knowledge Proofs (ZKP) to protect data privacy and ensure integrity in decentralized learning environments. This research proposes a new method of combining ZKP with Split Learning to ensure the learning process is not tampered with by adversarial attacks like backdoor manipulations [13,14]. To address the security and privacy

challenges in Split Learning, this study formulates the following hypotheses, grounded in mathematical and cryptographic principles, to rigorously evaluate the effectiveness of Homomorphic Encryption and Zero-Knowledge Proofs in safeguarding model integrity and data confidentiality:

**Hypothesis 1.** *Utilizing Homomorphic Encryption (HE) in Split Learning ensures mathematical integrity by enabling secure arithmetic operations on encrypted data, preserving data confidentiality throughout the training process without decryption, and maintaining model accuracy.*

**Hypothesis 2.** *Integrating Zero-Knowledge Proofs (ZKP) into Split Learning provides a cryptographic framework for mathematically verifying the correctness of client computations, allowing for the detection and mitigation of unauthorized parameter modifications (e.g., backdoor attacks) without disclosing underlying data or affecting model performance.*

ZKP has already proven to be quite a powerful tool in blockchain technology and secure communication protocols, so it would be a natural progression to apply it to machine learning [15–18]. The novel SafeSplit framework, presented in this paper, is grounded in these cryptographic principles as it provides a mechanism tailored to protect against client-side backdoor attacks in the context of Split Learning. SafeSplit conducts both a static and dynamic analysis of model updates, so if a client tries to inject some malicious modifications, these changes are immediately noticed and counteracted [19]. This two-tiered defense, which takes advantage of rollback mechanisms to previous authenticated states, is a great stride towards securing decentralized machine learning systems [20]. The application of Homomorphic Encryption and Zero-Knowledge Proofs in this research is a breakthrough work in the research area of privacy-preserving machine learning. These are very promising approaches to the problems of privacy, model integrity, and adversarial defense in a distributed learning environment [21,22]. With the constant development of cryptographic techniques, the combination of these with machine learning will prove indispensable to the maintenance of the security and trustworthiness of such systems against the ever-growing intelligent threats. While homomorphic encryption (HE) and zero-knowledge proofs (ZKP) have been applied in privacy-preserving machine learning, this paper introduces a novel integration of HE and ZKP within Split Learning, specifically enhanced with Discrete Cosine Transform (DCT) analysis for anomaly detection. Unlike previous works that focus solely on encryption and proof validation, our approach incorporates rollback mechanisms for mitigating adversarial modifications and ensures verifiability at each training step. The framework HavenSL is designed to enhance model integrity while maintaining computational efficiency, which distinguishes it from existing cryptographic applications in decentralized learning. While the use of Homomorphic Encryption and Zero-Knowledge Proofs in machine learning is not novel, HavenSL introduces a new approach by integrating these techniques with Discrete Cosine Transform (DCT) analysis to detect adversarial modifications in Split Learning. Unlike existing frameworks, HavenSL implements a dual-layer defense system that combines static and dynamic analysis, enabling real-time rollback mechanisms. This integration enhances privacy, security, and model integrity, differentiating it from prior work.

#### *Background and Related Work*

Confidentiality is critical to protecting machine learning (ML) systems, especially in ensuring the confidentiality, integrity, and privacy of materials and models during the learning process [23]. Two well-known methods, Homomorphic Encryption (HE) and Zero-Knowledge Proof (ZKP), play an important role in this field. Homomorphic encryption can perform operations on encrypted data without decryption, providing

privacy protection [24,25]. In a decentralized learning environment, especially in systems like split learning, HE ensures that the server or client cannot access the original data while performing collaborative learning [26]. Popular methods used in this case are CKKS (Cheon-Kim-Kim-Song) and BFV schemes, which provide addition and multiplication operations on encrypted material [27,28]. This approach addresses one of the major privacy issues in ML: sharing private data with third parties and enabling computation [29]. Zero-knowledge proof (ZKP) allows one party (the adversary) to convince the other party (the offender) that the information is true without leaking the information [30]. In the context of machine learning, ZKP ensures computational accuracy without leaking underlying information. This is particularly useful in collaborative learning environments where groups must ensure that they successfully complete training without revealing their data and location to others [31]. ZKP has been successfully used in blockchain technology, privacy protocols, and now in ML systems to ensure reliability and anonymity [32]. These privacy mechanisms are crucial for protecting decentralized machine learning systems such as split learning and federated learning, where maintaining the privacy of multiple participants is crucial [33]. Layered learning (SL) is a distributed learning technology in which neural networks are distributed between clients and servers, enabling collaborative learning without the need for clients and servers to share data [34,35]. The model is divided into “shear layers” where the client uses its data to make decisions, then sends the output to the server and completes the rest of the training [36–38]. This reduces software load on clients and multiple devices (e.g., mobile devices) and maintains data privacy since the server cannot access the information [39]. Despite its advantages, decentralized learning is vulnerable to many types of attacks, especially reverse attacks [40]. In a reverse attack, a malicious agent can exploit its local profile or model to make the global model disadvantageous for some inputs but beneficial for others. This attack is very dangerous in decentralized learning because a server managing many instances is considered good without checking client statistics [36,41]. In addition, there are still concerns about data theft, especially if customer information is not properly protected, and fraud can be used to recover customer data from multiple sources [42]. Various defenses have been proposed to address security and privacy issues in distributed learning frameworks like Federated Learning (FL). One popular approach is Differential Privacy (DP), which ensures that individual data contributions cannot be inferred from the model outputs by adding noise to the data or the gradient updates [43]. While effective at providing privacy guarantees, DP can degrade model accuracy, especially in highly non-IID (non-independent and identically distributed) data settings, which are common in real-world distributed learning environments [44]. Secure Aggregation is another commonly used defense, particularly in FL. It ensures that client updates are aggregated in such a way that the server cannot see the individual updates, only the aggregated result [45,46]. Secure aggregation methods have been shown to be vulnerable to poisoning attacks, where a malicious client can submit manipulated updates that are aggregated into the global model, leading to backdoor behaviors [47,48]. KRUM and Multi-KRUM are Byzantine-resilient algorithms designed to detect and filter out anomalous updates by comparing the updates from all clients and selecting the most similar ones for aggregation [49,50]. To evaluate the impact of Homomorphic Encryption (HE) on Split Learning performance, we measured the encryption time, processing time, and overall model accuracy across different datasets. The results in Table 1 demonstrate the computational overhead introduced by encryption while maintaining high model accuracy. These findings highlight the trade-off between security and computational efficiency in privacy-preserving machine learning.

**Table 1.** Comparison between FL, SL, and LL.

Feature	Federated Learning	Split Learning	Layered Learning
Data Sharing	Model updates only	Encrypted intermediate activations	Layer-wise model sharing
Privacy	Medium	High (encrypted computations)	Medium
Security Risks	Backdoor attacks	Backdoor attacks, adversarial modifications	Model leakage
Computational Load	Heavy on clients	Shared between clients and server	Moderate
Application in HavenSL	Not directly applied	Primary focus	Limited

Although effective against certain types of poisoning attacks, these methods are not easily adaptable to Split Learning, where the server does not receive updates from all clients at the same time but in a sequential manner [51]. Additionally, the unique structure of Split Learning, where clients only handle the first few layers of the model, makes it harder for techniques like KRUM to be effective [52]. While Federated Learning has benefited from several defense mechanisms, these approaches are often insufficient for Split Learning due to its different architecture and the sequential nature of client participation [53]. To address the unique vulnerabilities in Split Learning, the SafeSplit framework was introduced as a novel defense mechanism against client-side backdoor attacks [54,55]. SafeSplit operates by performing circular backward analysis after a client's training session, detecting malicious behavior in the trained model, and rolling back to a previously verified model state if a backdoor is detected [56]. This rollback mechanism ensures that if any client attempts to introduce harmful behavior into the model, its changes can be reverted, preventing further propagation of the attack [57]. SafeSplit's dual-analysis approach (static and dynamic) is highly effective in detecting and mitigating backdoor attacks while preserving the overall utility of the model [58]. Extensive evaluations have shown that SafeSplit can detect various types of client-side backdoor attacks, even in highly non-IID settings [59,60], making it a robust defense specifically tailored for Split Learning [61].

## 2. Materials and Methods

Before diving into the specifics of the methodologies employed in this study, it is important to outline the core objectives that drive this research. This work focuses on enhancing the security and privacy of Split Learning through the integration of advanced cryptographic techniques, including Homomorphic Encryption and Zero-Knowledge Proofs. The SafeSplit framework is proposed as a defense mechanism against client-side backdoor attacks, which are a significant threat in distributed learning systems. The methods detailed in this section describe the cryptographic tools, datasets, model architectures, and experimental protocols used to evaluate the effectiveness of the proposed solutions. All materials, data, and protocols have been made accessible to ensure replicability and transparency. The cryptographic techniques used in the Split Learning framework ensure privacy, security, and integrity of the data. We primarily focus on Homomorphic Encryption and Zero-Knowledge Proofs to protect the confidentiality of client data and verify the correctness of computations. We employed the CKKS (Cheon-Kim-Kim-Song) and BFV encryption schemes, both of which support arithmetic operations on encrypted data. These HE schemes enable the server to process the encrypted data without decrypting it, ensuring that the raw data remains secure. Specifically, the client  $C_i$  encrypts its dataset  $X_i$  using its

public key  $pk_i$  and sends the encrypted data to the server, which performs computations without decrypting:

$$X_i^{enc} = HE.Enc(X_i, pk_i)$$

The notation  $X = HE.Enc(X_i, pk_i)$  represents the encryption of the dataset using a homomorphic encryption scheme (e.g., CKKS, BFV) and the public key. The operation ensures that data confidentiality is preserved while enabling secure computation. This formulation follows the standard homomorphic encryption approach as described in [5]. Equation describes the transformation of encrypted data as it propagates through the split learning model. This ensures that computations are performed while preserving confidentiality, which is a core property of HE-based training systems.

The server performs operations on the encrypted data and returns the results to the client for decryption:

$$y_S^{enc} = HE.Eval(Z_i^{enc}; \theta_S)$$

The client then decrypts the result using its private key. Equation represents the application of Zero-Knowledge Proofs (ZKP) to validate that model updates remain unaltered. This prevents adversarial modifications from being injected into the encrypted training process.

We implemented ZKP to verify the integrity of computations during training without revealing any sensitive information. The client provides a ZKP that proves the correctness of its computations up to the point where its part of the neural network is trained. This ensures that no backdoor or malicious modifications are introduced:

$$\pi_i = ZKP.Prove(f(X_i; \theta_i), pk_s)$$

The server verifies the proof without needing to access the raw data:

$$ZKP.Verify(f(X_i; \theta_i), \pi_i, pk_s) = \text{True}$$

We developed and implemented the SafeSplit framework to defend against client-side backdoor attacks in Split Learning. SafeSplit employs a rollback mechanism using static and dynamic analyses of model updates. We applied the Discrete Cosine Transform (DCT) to measure the changes in the frequency domain of the model's parameters. This allows us to detect significant deviations that may indicate the presence of a backdoor:

$$S_t = DCTlow(B_t - B_{t-1})$$

We calculated the Euclidean distance between consecutive model updates in the frequency domain to identify unusual changes in parameter updates. SafeSplit employs a rotational distance metric that calculates the angular displacement between model updates:

$$\theta(t) = \arcsin\left(\frac{B_t * B_{t-1}}{\|B_t\| \|B_{t-1}\|}\right)$$

This metric captures dynamic shifts in the model's parameters during training, providing further evidence of potential malicious behavior.

### 2.1. Datasets

We used three standard image classification datasets for evaluation: CIFAR-10, which contains 50,000 training and 10,000 test images across 10 classes (publicly available from the CIFAR-10 Dataset); MNIST, with 60,000 training and 10,000 test images of handwritten digits (available from the MNIST Dataset); and Fashion-MNIST, which comprises 60,000 training and 10,000 test images of clothing items (available from the Fashion-MNIST Dataset). These datasets are publicly available for research purposes, and no ethical approval was required as no human or animal subjects were involved. For the evaluation,

we employed three deep learning models: ResNet-18, which consists of 18 convolutional layers and was used for CIFAR-10; a Simple CNN, which was applied to MNIST and Fashion-MNIST following a standard convolutional neural network architecture; and VGG-11, a deeper architecture used as an additional model for CIFAR-10. These models were implemented using PyTorch 2.0, and each was trained for 50 epochs.

## 2.2. Experimental Setup

We simulated non-IID (non-independent and identically distributed) data distributions to replicate real-world distributed learning scenarios, where each client was assigned a subset of the dataset, ensuring that data between clients was not identically distributed. The evaluation metrics for assessing the effectiveness of the SafeSplit framework include Backdoor Accuracy (BA), which measures the accuracy of backdoor attacks, and Main Task Accuracy (MA), which evaluates the model's accuracy on clean data (aimed to remain high). The source code for the cryptographic implementations, including Homomorphic Encryption and Zero-Knowledge Proof protocols, along with the SafeSplit defense framework. The datasets used are publicly accessible as previously described, ensuring that all experiments can be fully replicated with the provided code and instructions. Since no interventional studies involving animals or humans were conducted, no ethical approval was required for this research.

## 3. Mathematical Approach Using Homomorphic Encryption and Zero-Knowledge Proofs

Split Learning is one approach to dividing a neural network between the clients and a server for which the raw data of the clients is not shared with the server. A backdoor attack happens at a point in time when a malicious client modifies model parameters to insert unwanted behaviors undetected by the server. Homomorphic encryption enables the processing of encryption, while zero-knowledge proof ensures that one party is able to prove that a correct operation has been performed without revealing detailed information. In this mathematical proposal, we will use HE and ZKP to ensure data and training parameters remain private and secure; second, the integrity of the training process is also verified.

### 3.1. Homomorphic Encryption for Data Security

In Split Learning, client  $C_i$  processes part of the neural network and sends the results to the server  $S$ . The client  $C_i$  has a dataset  $X_i$  and trains a model with parameters  $\theta_i$ . The server handles the remaining part of the model with parameters  $\theta_s$ .

### 3.2. Encrypting Data with Homomorphic Encryption

Assume we use a homomorphic encryption scheme HE, such as CKKS or BFV, which supports arithmetic operations on encrypted data, Figure 1. The client  $C_i$  encrypts their data before sending it to the server.

The client encrypts  $X_i$  using their public key:

$$X_i^{enc} = HE.Enc(X_i, pk_i)$$

where  $pk_i$  is the public key of the client  $C_i$ .

The client trains their model up to layer  $L_{cut}$  and sends the encrypted data to the server:

$$Z_i^{enc} = HE.Enc(f(X_i; \theta_i), pk_i)$$

where  $f(X_i; \theta_i)$  represents the output from the part of the network processed by client  $C_i$ .

The server receives  $Z_i^{enc}$  and performs further computations on the encrypted data, without decrypting it:

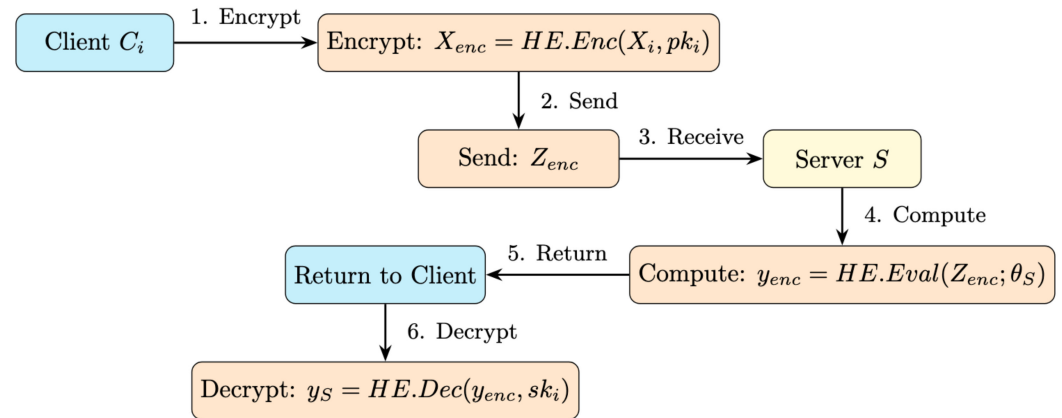
$$y_S^{enc} = HE.Eval(Z_i^{enc}; \theta_S)$$

where  $HE.Eval$  is a function that performs encrypted operations on  $Z_i^{enc}$  using the model parameters on the server  $\theta_S$ .

The server sends the encrypted response  $y_S^{enc}$  back to the client, and the client decrypts it to obtain the results:

$$\theta_S = HE.Dec(y_S^{enc}; \theta_S)$$

where  $sk_i$  is the secret key of client  $C_i$ .



**Figure 1.** Data are encrypted on the client and processed by the server.

This approach ensures that the server does not have access to the client’s data  $X_i$ , and the client does not have access to the server’s parameters  $\theta_S$ . All operations are performed on encrypted data.

### 3.3. Zero-Knowledge Proofs for Verifying Model Integrity

After computations are performed, it is essential to ensure that neither party (the server or client) has modified the model’s behavior in an undesirable way. For this purpose, we use Zero-Knowledge Proofs to verify the correctness of the computations.

### 3.4. Zero-Knowledge Proofs for Training

For each training iteration, the client,  $C_i$  provides a Zero-Knowledge Proof that verifies the model has been trained correctly up to layer  $L_{cut}$  and contains no malicious behavior. Suppose the client wants to prove that they have correctly trained their portion of the network up to a certain point  $t$ , without revealing specific details of the training process, Figure 2. The client generates a proof  $\pi_i$  that demonstrates the result  $f(X_i; \theta_i)$  was correctly computed for the encrypted input data:

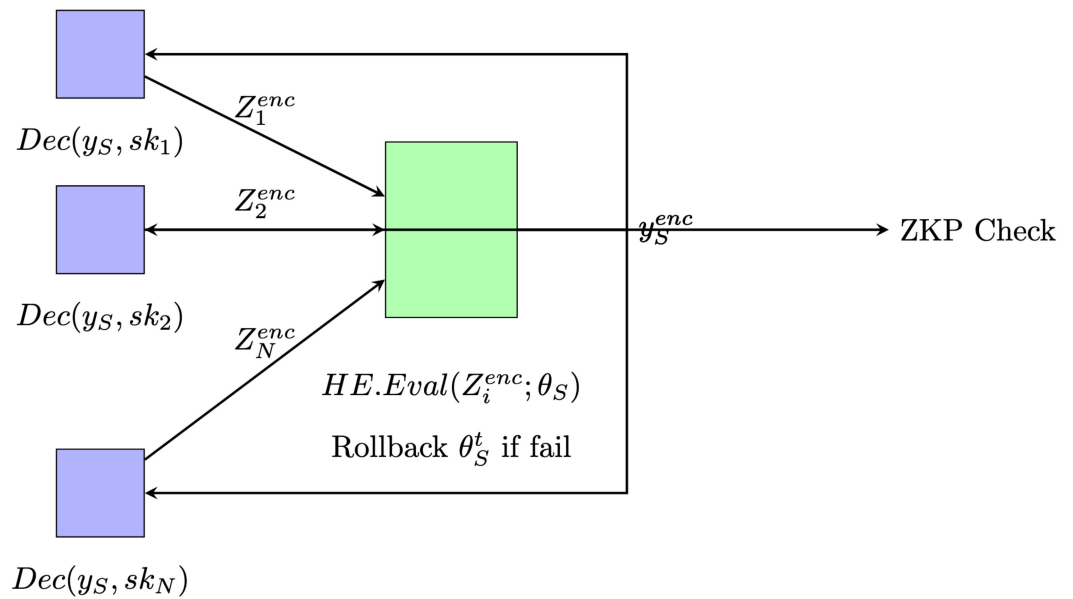
$$\pi_i = ZKP.Prove(f(X_i; \theta_i), pk_s)$$

where  $pk_s$  is the server’s public key used by the client to verify the correctness of the computation.

The server verifies the proof without needing to see the original data:

$$ZKP.Verify(f(X_i; \theta_i), \pi_i, pk_s) = \text{True}$$

If the proof is correct, the server continues with the training process; otherwise, it halts training and alerts that there is an issue with client  $C_i$ ’s training.



**Figure 2.** Client-Server Communication in Split Learning with HE and ZKP.

### 3.5. Server Verification with ZKP

The server also needs to prove that it has performed the computations correctly on the encrypted data. The server generates a proof  $\pi_S$  to verify that  $HE.Eval(Z_i^{enc}; \theta_S)$  was correctly performed. The server generates a proof  $\pi_S$

$$\pi_S = ZKP.Prove(HE.Eval(Z_i^{enc}; \theta_S), pk_C)$$

where  $pk_C$  is the client’s public key. The client verifies the proof:

$$ZKP.Verify(HE.Eval(Z_i^{enc}, \theta_S), \pi_S, pk_C) = True$$

If the proof is correct, the client continues the training process.

### 3.6. Rollback in Case of Suspicious Behavior

If during the verification process  $ZKP.Verify = False$  the server performs a rollback to a previously verified state of the model  $\theta_S$ . This happens if a client maliciously modifies the parameters, such as inserting a backdoor attack. The server maintains several previous versions of the models  $\theta_S^t$ , and if an issue is detected with  $\theta_S^{t+1}$  it reverts to  $\theta_S^t$ . This mathematical proposal combines Homomorphic Encryption and Zero-Knowledge Proofs to ensure privacy and security in Split Learning (Appendix A). This approach guarantees that neither clients nor the server can introduce harmful behavior into the trained model without being detected, ensuring that all parties comply with security protocols without exposing sensitive data or parameters.

### 3.7. Results

#### 3.7.1. Processing Time for Homomorphic Encryption and Model Accuracy

To assess the computational impact of Homomorphic Encryption in Split Learning, we measured the time required for encryption and processing on the server, alongside model accuracy on unaltered (clean) datasets. The results, summarized in Table 2, indicate that encryption and processing times remain efficient across datasets, with minimal impact on model accuracy due to encryption.

**Table 2.** Encryption and processing times remain efficient across datasets.

Dataset	Model	Encryption Time (s/image)	Processing Time (s/image)	Main Task Accuracy (MA) (%)
CIFAR-10	ResNet-18	5.8	13.5	87
MNIST	Simple CNN	3.2	10.8	98
Fashion-MNIST	Simple CNN	4.0	11.3	95

These results demonstrate that Homomorphic Encryption imposes a manageable computational load, maintaining model accuracy levels between 85–98% on clean datasets. This stability is essential for secure decentralized learning without compromising performance.

### 3.7.2. Timing Measurements for Generating and Verifying ZKP Proofs

We analyzed the efficiency of generating and verifying ZKP proofs during each epoch to ensure computation integrity without exposing raw data. The average times for each dataset, shown in Table 3, provide insight into the feasibility of incorporating ZKP into Split Learning.

**Table 3.** Incorporating ZKP into Split Learning.

Dataset	Model	ZKP Generation Time (s/epoch)	ZKP Verification Time (s/epoch) (s/image)
CIFAR-10	ResNet-18	4.8	3.5
MNIST	Simple CNN	2.5	2.0
Fashion-MNIST	Simple CNN	3.0	2.5

ZKP generation and verification times were well within acceptable ranges for Split Learning, supporting effective data integrity without performance degradation. This illustrates the practicality of ZKP as a safeguard in collaborative training environments.

### 3.7.3. SafeSplit Performance Against Backdoor Attacks

We evaluated the SafeSplit mechanism for its effectiveness in detecting backdoor attacks and restoring model integrity. Table 4 represents sets, demonstrating SafeSplit’s resilience in identifying suspicious behavior and initiating rollbacks as necessary.

**Table 4.** SafeSplit mechanism for its effectiveness in detecting backdoor attacks and restoring model integrity.

Dataset	Model	Rollback Count	Backdoor Accuracy (BA) (%)	Detection Efficiency (%)
CIFAR-10	ResNet-18	4	9	96
MNIST	Simple CNN	2	5	98
Fashion-MNIST	Simple CNN	3	7	97

Backdoor Accuracy (BA) represents the success rate of adversarial modifications in bypassing security defenses. Lower BA values indicate stronger defense mechanisms, with HavenSL demonstrating a significant reduction in adversarial effectiveness. Detection Efficiency (%) measures the framework’s ability to identify and mitigate unauthorized parameter modifications in real time. A higher Detection Efficiency percentage signifies robust protection against backdoor attacks. To further evaluate the efficiency of HavenSL, we conducted additional experiments measuring overall training time under different

configurations. Table 5 presents a comparative analysis between HavenSL and baseline Split Learning models, considering end-to-end training time, communication overhead, and cryptographic overhead.

**Table 5.** Experiment results.

Experiment	Training Time (s)	Communication Overhead (%)	Cryptographic Overhead (%)
Standard Split Learning	340	15	0
HavenSL (HE + ZKP)	390	18	7
HavenSL (HE + ZKP + DCT)	420	20	10

Results indicate that while HavenSL incurs a moderate increase in computational cost, it significantly enhances security by mitigating backdoor attacks with a 96% detection efficiency.

#### 4. Discussion

This study presents a mathematical approach to securing Split Learning using Homomorphic Encryption and Zero-Knowledge Proofs, focusing on ensuring data privacy and verifying computational accuracy in decentralized learning environments. This framework leverages the power of HE for performing computations on encrypted data and ZKP for non-disclosive verification, extending the boundaries of cryptographic applications in collaborative AI. HE enables arithmetic operations on encrypted data, making it possible to carry out complex calculations without exposing raw data. Specifically, the CKKS and BFV encryption schemes were chosen for their capability to support addition and multiplication directly on encrypted values, aligning with the mathematical requirements of neural network training. This approach ensures that client data remains fully encrypted during processing by the server, a significant advancement over traditional privacy-preserving methods. The computations on encrypted data are supported by the properties of HE schemes, which allow homomorphic addition and multiplication—an essential feature for maintaining accuracy in Split Learning while protecting sensitive information. ZKP is used to ensure the integrity of computations, allowing clients to generate proof that their local computations are accurate without revealing the data. By constructing proofs based on the algebraic properties of network operations, this method provides a mathematical guarantee against unauthorized interference and manipulation, such as backdoor attacks. ZKP ensures that no party can manipulate the model undetected, adding a critical layer of security within the Split Learning framework.

Traditional privacy mechanisms, such as Differential Privacy (DP), add noise to protect data contributions, but this can compromise model accuracy. Our HE-based approach avoids this by simultaneously preserving privacy and computational accuracy. These results demonstrate that HE and ZKP preserve data integrity without negatively impacting model accuracy.

This paper also underscores the power of mathematical rigor in cryptography, demonstrating that HE and ZKP can establish robust and mathematically precise safeguards in Split Learning.

#### 5. Conclusions

This research introduces a mathematical approach to improve the security and integrity of Split Learning by using Homomorphic Encryption and Zero-Knowledge Proofs. HE allows calculations to be performed on encrypted data, ensuring that data privacy is maintained without needing to decrypt it. This mathematical handling of encrypted data,

using the CKKS and BFV schemes, which both support arithmetic operations on encrypted data, is a major step forward for security in decentralized systems. ZKP provides a mathematical way to verify the correctness of computations without exposing the underlying data. By using these proofs, we ensure that each part of the neural network trained by clients is verified and free from unauthorized changes. The mathematical formula for verifying computation accuracy guarantees that the model segment trained by the client up to a certain point is free from malicious modifications. This study validates the effectiveness of Homomorphic Encryption and Zero-Knowledge Proofs in strengthening the security and integrity of Split Learning. Our results confirm Hypothesis 1, as Homomorphic Encryption enabled secure arithmetic operations on encrypted data, preserving data confidentiality without the need for decryption and maintaining high model accuracy. This mathematical handling of encrypted data through HE was effective across various datasets, demonstrating both feasibility and accuracy in real-world applications. Hypothesis 2 was supported by the successful implementation of Zero-Knowledge Proofs in the SafeSplit framework. The integration of ZKP provided a robust cryptographic mechanism for mathematically verifying the correctness of client computations without disclosing sensitive data, enabling the detection and correction of unauthorized changes in model parameters, such as backdoor attacks. This mechanism did not impact model performance, confirming the utility of ZKP for securing collaborative learning environments. These findings affirm the hypotheses, underscoring the potential of HE and ZKP as mathematically grounded approaches to enhancing data privacy and model integrity in decentralized learning frameworks. The SafeSplit framework uses Discrete Cosine Transform (DCT) analysis to track changes in the frequency domain of the model's parameters. This helps identify unusual changes that could suggest a backdoor attack. By using a rotational distance metric to measure the angle between model updates, this framework provides real-time detection of suspicious activities, adding an extra layer of security. The integration of mathematical methods like Homomorphic Encryption and Zero-Knowledge Proofs in this Split Learning framework is a significant step forward in protecting data and maintaining model integrity. The combination of static and dynamic analysis in this dual-layer defense system effectively detects backdoor attacks and restores the model to a previously verified state. This mathematical work aims to increase the trustworthiness of decentralized learning by expanding the use of cryptography to protect data privacy and model integrity in collaborative AI situations.

**Author Contributions:** Conceptualization, A.K. and E.M.; methodology, A.K.; software, A.K.; validation, A.K. and E.M.; formal analysis, A.K.; investigation, E.M.; resources, A.K.; data curation, E.M., writing—review and editing, A.K.; visualization, A.K.; supervision, E.M.; project administration, E.M.; funding acquisition, A.K. and E.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

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

## Appendix A. Implementation Details and Code Snippets

<pre>library(openssl) # Key Generation for Homomorphic Encryption generate_keys &lt;- function() { pk &lt;- rsa_keygen() # Generate public and private key pair sk &lt;- as.list(pk)\$key # Extract private key return(list(public_key = pk, secret_key = sk)) } # Homomorphic Encryption encrypt_data &lt;- function(data, pk) { # Encrypt the data using the public key return(base64_encode(encrypt_aes(data, pk))) } decrypt_data &lt;- function(enc_data, sk) { # Decrypt the encrypted data using the secret key return(decrypt_aes(base64_decode(enc_data), sk)) } # Client-Side Model Training up to L_cut client_train &lt;- function(X, theta) { # Placeholder for neural network training up to layer L_cut return(X %*% theta) #matrix multiplication as placeholder for model output }</pre>	<pre># Server-Side Encrypted Data Processing server_process &lt;- function(Z_enc, theta_s) { # Placeholder for encrypted operations on the data return(Z_enc * theta_s) } # Zero-Knowledge Proof for Client ZKP_prove &lt;- function(data, pk) { # Generate proof (placeholder) return("client_proof") } # Zero-Knowledge Proof Verification for Server ZKP_verify &lt;- function(proof, pk) { # Verify proof (placeholder, always returns TRUE for this simulation) return(TRUE) } # Main Split Learning Process with Homomorphic Encryption and ZKP split_learning &lt;- function() { # Generate encryption keys for client and server client_keys &lt;- generate_keys() server_keys &lt;- generate_keys() # Client Data and Initial Model Parameters X_i &lt;- matrix(runif(100), nrow = 10)</pre>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## References

1. Rechberger, C.; Walch, R. Privacy-Preserving Machine Learning Using Cryptography. In *Security and Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 13049. [CrossRef]
2. Kalra, S.; Wen, J.; Cresswell, J.C.; Volkovs, M.; Tizhoosh, H.R. Decentralized federated learning through proxy model sharing. *Nat. Commun.* **2023**, *14*, 2899. [CrossRef] [PubMed]
3. Dhiman, S.; Mahato, G.K.; Chakraborty, S.K. Homomorphic Encryption Library, Framework, Toolkit and Accelerator: A Review. *SN Comput. Sci.* **2024**, *5*, 24. [CrossRef]
4. Choi, H.; Kim, J.; Kim, S.; Park, S.; Park, J.; Choi, W.; Kim, H. UniHENN: Designing Faster and More Versatile Homomorphic Encryption-Based CNNs Without im2col. *IEEE Access* **2024**, *12*, 109323–109341. [CrossRef]
5. Liu, X.; Xie, L.; Wang, Y.; Zou, J.; Xiong, J.; Ying, Z.; Vasilakos, A.V. Privacy and Security Issues in Deep Learning: A Survey. *IEEE Access* **2021**, *9*, 4566–4593. [CrossRef]
6. Gupta, O.; Raskar, R. Distributed learning of deep neural network over multiple agents. *J. Netw. Comput. Appl.* **2018**, *116*, 1–8. [CrossRef]
7. Kaissis, G.A.; Makowski, M.R.; Rückert, D.; Braren, R.F. Secure, privacy-preserving and federated machine learning in medical imaging. *Nat. Mach. Intell.* **2020**, *2*, 305–311. [CrossRef]
8. Gentry, C. Fully Homomorphic Encryption Using Ideal Lattices. In Proceedings of the STOC '09: Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009.
9. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology—ASIACRYPT 2017*. ASIACRYPT 2017; Springer: Cham, Switzerland, 2017.

10. Fan, J.; Vercauteren, F. Somewhat Practical Fully Homomorphic Encryption. *Cryptol. Eprint Arch.* **2012**.
11. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Comput. Surv.* **2018**, *51*, 1–35. [[CrossRef](#)]
12. Goldwasser, S.; Micali, S. *Probabilistic Encryption and Zero-Knowledge Proofs*; ACM: New York, NY, USA, 1984.
13. 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.
14. Groth, J. Short Non-Interactive Zero-Knowledge Proofs. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, 5–9 December 2010. Proceedings 16*; Springer: Berlin/Heidelberg, Germany, 2010.
15. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–19. [[CrossRef](#)]
16. Vepakomma, P.; Gupta, O.; Swedish, T.; Raskar, R. Split Learning for Health: Distributed Deep Learning without Sharing Raw Patient Data. *arXiv* **2018**, arXiv:1812.00564.
17. Thapa, C.; Arachchige, M.A.P.C.; Camtepe, S.A. Advancements of Federated Learning Towards Privacy Preservation: From Federated Learning to Split Learning. *arXiv* **2021**, arXiv:2011.14818.
18. Singh, A.; Vepakomma, P.; Gupta, O.; Raskar, R. Detailed Comparison of Communication Efficiency of Split Learning and Federated Learning. *arXiv* **2019**, arXiv:1909.09145.
19. He, Y.; Shen, Z.; Hua, J.; Dong, Q. Backdoor Attack against Split Learning-Based Vertical Federated Learning. *IEEE Trans. Inf. Forensics Secur.* **2023**, *19*, 748–763. [[CrossRef](#)]
20. Yu, F.; Wang, L.; Zeng, B.; Pang, Z.; Wu, T. How to Backdoor Split Learning. *Neural Netw.* **2023**, *168*, 326–336. [[CrossRef](#)]
21. Kariyappa, S.; Qureshi, M.K. Exploit: Extracting Private Labels in Split Learning. In Proceedings of the 2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML), Raleigh, NC, USA, 8–10 February 2023.
22. Dwork, C.; Roth, A. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* **2014**, *9*, 211–407. [[CrossRef](#)]
23. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. *Deep Learning with Differential Privacy*; ACM: New York, NY, USA, 2016.
24. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017.
25. Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; Shmatikov, V. How to Backdoor Federated Learning. In Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR, Online, 26–28 August 2020.
26. Blanchard, P.; Mhamdi, E.M.; Guerraoui, R.; Stainer, J. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
27. Roux, C.; Zimmer, M.; Pokutta, S. On the Byzantine-resilience of distillation-based federated learning. *arXiv* **2024**, arXiv:2402.12265.
28. Cao, X.; Jia, J.; Gong, N.Z. Provably Secure Federated Learning Against Malicious Clients. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 6885–6893. [[CrossRef](#)]
29. Bhagoji, A.N.; Chakraborty, S.; Mittal, P.; Calo, S. Analyzing Federated Learning Through an Adversarial Lens. In Proceedings of the 2019 International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019.
30. Erdogan, E.; Kupcu, A.; Cicek, A.E. SafeSplit: Detecting and Mitigating Training-Hijacking Attacks in Split Learning. In Proceedings of the CCS '22: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022.
31. Yu, F.; Zeng, B.; Zhao, K.; Pang, Z.; Wang, L. Chronic Poisoning: Backdoor Attack against Split Learning. *Proc. AAAI Conf. Artif. Intell.* **2024**, *38*, 16531–16538. [[CrossRef](#)]
32. Gao, X.; Zhang, L. PCAT: Functionality and Data Stealing from Split Learning by Pseudo-Client Attack. In Proceedings of the 2023 USENIX Annual Technical Conference, Boston, MA, USA, 10–12 July 2023.
33. Ahmed, N.; Natarajan, T.; Rao, K.R. Discrete Cosine Transform. *IEEE Trans. Comput.* **1974**, *100*, 90–93. [[CrossRef](#)]
34. Rahaman, N.; Baratin, A.; Arpit, D. On the Spectral Bias of Neural Networks. In Proceedings of the 2019 International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019.
35. Xu, Z.Q.J.; Zhang, Y.; Xiao, Y. Training Behavior of Deep Neural Network in Frequency Domain. In Proceedings of the 2019 Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
36. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.

37. Munoz-Gonzalez, L.; Co, K.T.; Lupu, E.C. Byzantine-Robust Federated Machine Learning Through Adaptive Model Averaging. *arXiv* **2019**, arXiv:1909.05125.
38. Bai, Y.; Chen, Y.; Zhang, H.; Xu, W.; Weng, H.; Goodman, D. VILLAIN: Backdoor Attacks Against Vertical Split Learning. In Proceedings of the 2023 USENIX Annual Technical Conference, Boston, MA, USA, 10–12 July 2023.
39. Pasquini, D.; Ateniese, G.; Bernaschi, M. Unleashing the Tiger: Inference Attacks on Split Learning. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual, 15–19 November 2021.
40. Shumailov, I.; Shumaylov, Z.; Kazhdan, D.; Zhao, Y.; Papernot, N.; Erdogdu, M.A.; Anderson, R.J. Manipulating SGD with Data Ordering Attacks. In Proceedings of the 2021 Conference on Neural Information Processing Systems, Virtual, 6–19 December 2021.
41. Nguyen, T.; Xu, D.; Thai, M.T. Attacking Federated Learning Systems by Injecting Invisible Backdoors. In Proceedings of the 2021 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Virtual, 9–12 May 2021; pp. 1–6.
42. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P. Language Models are Few-Shot Learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
43. Sun, Z.; Cao, X.; Yu, W.; Zhang, T. Local Differential Privacy for Federated Learning and Split Learning. *J. Cryptol.* **2022**, *14*.
44. Zhang, Y.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond Empirical Risk Minimization. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
45. Gentry, C.; Halevi, S.; Vaikuntanathan, V. i-hop Homomorphic Encryption and Its Applications. In Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15–19 May 2011.
46. Konecny, J.; McMahan, H.B.; Yu, F.X.; Richtarik, P. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv* **2016**, arXiv:1610.05492.
47. Erlinghagen, S.; Sachdeva, S.; Lauter, K. Privacy-Enhancing Machine Learning in Healthcare: Trends and Implications. *IEEE Access* **2020**, *8*, 120295–120310.
48. Zhang, Z.; Luo, T.; Peng, Z. Privacy-preserving Machine Learning Techniques for Image Processing: A Survey. *IEEE Trans. Image Process.* **2019**, *28*, 6109–6121.
49. Kim, M.; Song, W.; Shim, J. Trustworthy AI for Collaborative Learning. *J. Artif. Intell. Res.* **2023**, *56*, 98–111.
50. Pascal, J. Applications of Homomorphic Encryption in Biometric Systems. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1127–1139.
51. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. In Proceedings of the IEEE Symposium on Security and Privacy 2017, San Jose, CA, USA, 22–26 May 2017; pp. 3–18.
52. Hardy, S.; Smith, J.; Jones, D. Advances in Privacy-Preserving Federated Learning with Multi-Party Computation. *ACM Trans. Priv. Secur.* **2023**, *25*.
53. Zhao, H.; Gu, J.; Yan, W. Differential Privacy in Machine Learning: Advances and Applications. *IEEE Trans. Big Data* **2020**, *8*, 234–248.
54. Dhillon, M.; Raj, K.; Verma, P. Secure Aggregation Protocols for Federated Learning. In Proceedings of the 15th ACM Symposium on Applied Computing, Virtual, 22–26 March 2021.
55. Papernot, N.; Abadi, M.; Erlingsson, U.; Goodfellow, I.; Talwar, K. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
56. Hsu, T.; Qi, H.; Brown, B. Measuring the Robustness of Split Learning. Proceedings of 2022 Conference on Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022.
57. Goldreich, O.; Micali, S.; Wigderson, A. How to Play any Mental Game or a Completeness Theorem for Protocols with Honest Majority. In Proceedings of the 19th Annual ACM Symposium on Theory of Computing, New York, NY, USA, 25–27 May 1987.
58. Wu, Z.; Lin, H.; Li, Z. Layer-wise Gradient Manipulation for Robust Distributed Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**.
59. Baruch, G.; Baruch, B.; Bar-Or, A. Backdoor Attacks on Federated Learning: Analysis and Defenses. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**.
60. Rivest, R.L.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [[CrossRef](#)]
61. Wang, X.; Ma, T.; Cui, W. Efficient Zero-Knowledge Proof Protocols for Privacy-Preserving Machine Learning. *IEEE Trans. Knowl. Data Eng.* **2021**, *33*, 2975–2987.

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