Mitigation of Sybil-based Poisoning Attacks in Permissionless Decentralized Learning

Brandon A Mosqueda González, Omar Hasan, Lionel Brunie

Mitigation of Sybil-based Poisoning Attacks in Permissionless Decentralized Learning

1st Brandon A. Mosqueda González
University of Lyon, CNRS
INSA Lyon, LIRIS, UMR5205, F-69621
Lyon, France
brandon.mosqueda-gonzalez@insa-lyon.fr

2nd Omar Hasan
University of Lyon, CNRS
INSA Lyon, LIRIS, UMR5205, F-69621
Lyon, France
omar.hasan@insa-lyon.fr

3rd Lionel Brunie
University of Lyon, CNRS
INSA Lyon, LIRIS, UMR5205, F-69621
Lyon, France
lionel.brunie@insa-lyon.fr

***Abstract*—Decentralized learning enables collaborative machine learning with enhanced privacy by allowing participants to train models locally and share updates for aggregation instead of sharing raw data. However, such systems are vulnerable to poisoning attacks that may compromise the learning process. This threat becomes even more severe when combined with sybil attacks, where adversaries contribute numerous malicious updates with minimal effort, amplifying their impact. To overcome these challenges, particularly in the permissionless setup, we propose SyDeLP, a blockchain-enabled protocol for decentralized learning. SyDeLP integrates byzantine tolerant aggregation for poisoning mitigation with a novel Verifiable Delay Puzzle to counter sybil attacks requiring Proofs of Work to participate. Honest behavior is incentivized by dynamically reducing puzzle difficulty, decreasing the computational burden for honest nodes over time. Empirical evaluations conducted on two benchmark datasets across four types of poisoning attack demonstrate that SyDeLP consistently outperforms existing solutions in terms of poisoning resilience.**

***Index Terms*—Decentralized Learning, Permissionless, Blockchain, Adaptive Difficulty, Verifiable Delay Function, Verifiable Delay Puzzle**

I. Introduction

Federated learning enables multiple devices to collaboratively train a machine learning model without having to share raw data. Instead, the model is iteratively refined by exchanging and aggregating model updates [1]. However, this process relies on a central server to orchestrate communication among participants, introducing limitations such as single point of failure and various security risks [2], [3]. To remove the dependency on a central server, decentralized learning has emerged as a more autonomous alternative. In this paradigm, the participants operate in a peer-to-peer network, sharing updates with a subset of neighbors. The models are locally combined and disseminated through a gossip protocol guaranteeing the convergence of the model [4].

Despite its advantages, the decentralized nature of these systems increases vulnerability to malicious behavior. A significant threat is the poisoning attack, where adversaries produce and share erroneous models, negatively impacting the aggregated models. Although there has been substantial progress in decentralized learning to mitigate known threats [2], [5], existing solutions often rely on trusted parties. However, this reliance re-introduces a central point of failure and creates new vulnerabilities, as trusted entities can be compromised or act maliciously [6].

In a permissionless setting, where no centralized entity governs access control, additional challenges arise. For example, in the absence of such centralized authority, the alternative access mechanism has to prevent disproportionate influence of participants by creating multiple fake entities under their control, a phenomenon known as sybil attack [7]. Poisoning attacks, when combined with sybil attacks, can become severe threats, allowing attackers to gain full control of the aggregation phase and its outcomes [8].

In this paper, we address the problem of mitigating sybil-based poisoning attacks in decentralized learning under a permissionless setup. Our proposal eliminates the centralized entities while providing robust security guarantees against such attacks.

A. Contributions

The main contributions of this paper are summarized as follows:

1) We propose the first protocol, to the best of our knowledge, that addresses Sybil-based Poisoning Attacks (SPAs) in the permissionless setting of decentralized learning. Unlike existing solutions, our protocol provides security guarantees against a wider range of poisoning attacks without relying on trusted parties. To prevent the disproportionate creation of sybils, we enforce individual Proofs of Work for each participant, integrated with blockchain technology and a Byzantine Tolerant Aggregation (BTA) function.
2) We introduce a novel Adaptive Proof of Work (APoW) mechanism that dynamically adjusts the difficulty of required Proofs of Work, rewarding honest clients by reducing their computational overhead while penalizing possible poisoning attempts with increased difficulty. This is achieved through combining the BTA function, a Verifiable Delay Function (VDF), and our custom difficulty adjustment function.
3) We conduct extensive experiments on two distinct learning tasks, evaluating our protocol against four types of poisoning attacks, three of which are sybil-based. Our solution outperforms state-of-the-art protocols in terms

of resilience to these attacks. The code will be made available as an open-source project for reproducibility.

B. Outline

The rest of the paper is structured as follows: Section II introduces the background, Section III discusses the related work, Section IV presents SyDeLP, the proposed protocol, Section V analyzes the security of the proposal, Section VI details the conducted experiments, Section VII discusses the results, and, Section VIII presents the conclusions.

II. Background

A. Decentralized Learning

Decentralized learning [4] is a distributed framework in which a set of nodes collaboratively train a machine learning model by iteratively exchanging model parameters. Each node optimizes its local model with respect to its individual data set and, through aggregated updates, aims to approximate a globally optimal model. Formally, a set of clients $\mathcal{C}$ solve the following stochastic optimization problem:

$$\arg\min_{w} \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mathcal{L}_c(X_c, w) \tag{1}$$

where w represents the model parameters, $\mathcal{L}_c$ is the local loss function of client c (e.g., cross-entropy loss or negative log likelihood loss), and X_c is its local dataset. In each round, every client trains its local model on its private dataset for a number of epochs and then shares it with neighboring nodes. Upon receiving the models from its neighbors, each client aggregates both its model and the received models to form an updated model, which is used for the next round of training. This process repeats for a predetermined number of iterations allowing the model to converge.

B. Poisoning attacks

The poisoning attack involves the active manipulation of data or model parameters to produce incorrect models that achieve a secondary goal. We distinguish four types of poisoning attacks:

1) **Targeted**: The goal is to induce the global model to misclassify specific inputs into incorrect categories by introducing a *backdoor* [9]. A common approach, as described by Lin et al. [10], involves collecting normal training data and then modifying the elements of a target class to align with the desired class. The resulting model can accurately classify normal data but misclassifies instances of the target class.
2) **Untargeted**: The objective is to decrease the accuracy of the global model, hindering its convergence [11]. This is accomplished by deliberately modifying the local model contributions so that, when aggregated with honest models, they deviate from the optimal solution. For instance, Lin et al. [12] propose the Zero-grad attack, where a malicious client submits models that cancel out the honest updates during aggregation, resulting in a zero vector.
3) **Random**: This is a specific case of untargeted attacks in which the model updates are not deliberately crafted to deviate from the optimal solution. Instead, the adversary skips the training part and just sends random vectors drawn from a Gaussian distribution [13]. The larger the variance of the distribution, the stronger the attack.
4) **Sybil-based**: This refers to any poisoning attack that is combined with a Sybil attack [7]. The primary goal of the attacker is to impersonate multiple users with different identities to gain disproportionate influence in the system. This enables the attacker to conduct more potent poisoning attacks [8]. We distinguish two subtypes of sybil-based attacks: (1) Uniform: where the sybil attacker uses identical model parameters for all the controlled sybils, and (2) Diverse: where sybils generate completely independent models.

C. Byzantine Tolerant Aggregation (BTA)

Preventing poisoning attacks is challenging because model updates are generated on remote devices, and access to the data is intentionally restricted. Although, defense mechanisms in the form of Byzantine Tolerant Aggregation functions are proposed in the literature [3], [13]–[16], their purpose is to ensure that the aggregated model reflects the true state of the model despite the presence of malicious (a.k.a byzantine) nodes. They act as poisoning detection mechanisms, aiming to identify and exclude abnormal models from the aggregation phase. The predominant approach involves measuring similarities among all models to discard those that are most dissimilar. For example, Multi-KRUM [13] selects for aggregation the $N - \beta$ model updates with the lowest distance scores, where N is the total number of nodes and β is the expected number of byzantine nodes. The distance score of client i is given by $score(i) = \sum_{i \to j} \|w_i - w_j\|^2$, where $i \to j$ denotes all $j \neq i$ such that w_j belongs to the $N - \beta - 2$ closest vectors to w_i. Multi-KRUM can tolerate up to $\beta \leq \frac{N-4}{2}$ byzantine clients and has theoretical guarantees for convergence.

BTA functions are secure when the number of malicious (and possibly colluding) parties does not exceed the β threshold. However, a sybil-based approach allows an attacker with limited computing power to easily surpass this threshold if no defensive measures are considered, especially in permissionless settings where the system is open and anyone is allowed to participate and access information.

D. Blockchain

Introduced by Nakamoto in 2008 for Bitcoin [17], blockchain is a distributed database where data is stored in the form of digitally signed transactions. It facilitates interactions without trust requirements (trustless) by allowing participants to verify them without a central authority. These transactions are grouped into blocks, which are identified by unique IDs generated from the hash of their content. The blocks' content include a hash reference to the previous block (except for the first one), creating a cryptographic chain of blocks. This results in an immutable data structure where any change in previously

written information would produce a cascade effect on the hashes of subsequent blocks, thereby ensuring the integrity of the system.

The blockchain is maintained by a set of distributed nodes that keep a local copy. They are responsible for validating incoming transactions and creating new blocks from them. Node maintainers use consensus mechanisms to decide which block is included next. Bitcoin's Proof of Work consensus [17] and Ethereum's Proof of Stake [18] are among the most well-known consensus mechanisms, but there exist numerous alternatives in the literature [19]. Blockchains can be categorized as public (permissionless) or private (permissioned), each with distinct implications for security and accessibility.

E. Proof of Work (PoW)

Proof of Work (PoW) is a well-established technique where a prover demonstrates to a verifier that a certain amount of computational effort has been done within a specified interval of time [20]. PoW has served as the foundation for numerous security protocols [21], [22] and it has gained widespread recognition with the advent of Bitcoin [17], which employed PoW as a mechanism to mitigate sybil attacks in a trustless decentralized system. Beyond Bitcoin, PoW has been effectively utilized in other decentralized protocols for this same purpose like in [23], a peer-to-peer system for preserving access to journals, and [24], a scalable distributed name service resilient to massive byzantine attacks.

The core of PoW is built on the concept of computational puzzles, which are designed to be computationally difficult to solve but straightforward to verify once the solution is found. These puzzles also provide cryptographic security guarantees, preventing cheating in the proof-generation process or precomputing solutions. Various computational puzzle constructions exist in the literature [21], [22], [25], each with unique characteristics. Among these, we emphasize the Verifiable Delay Function (VDF) due to its applicability to this work. VDFs possess these properties [26]: they require a precise number of sequential, non-parallelizable steps for evaluation, have a unique solution, and allow for efficient verification in poly-logarithmic time.

III. Related work

In decentralized learning, the Sybil-based Poisoning Attack (SPA) has not received sufficient attention. The Sybil attack is typically discussed only from the perspective of node maintainers [27]–[29], focusing on maintaining sybil-tolerant consensus. Other works do not address the SPA at all, as they assume a centralized trusted third party to manage access control [30]–[33]. Even in systems that claim to be permissionless, such as [27], [29], [34], where the SPA is more critical and easier to implement, this problem is not considered. BEAS [35] is one of the few examples where, despite assuming a permissioned protocol, the SPA problem is addressed. However, their solution relies on FoolsGold [8], ignoring its limitations in open environments where all clients have access to individual contributions, as is the case with BEAS. Other works, like [30], [36], require participants to provide collateral in the form of stake to take part in the learning process, but the SPA is not discussed, nor are the limitations of their solutions that require a fixed number of participants and a trusted access control authority.

In Table I, we compare our solution, SyDeLP (see Section IV), with other SPA protection solutions from the literature. Notably, only our proposal is designed for permissionless decentralized learning, where nodes can join or leave at any time. In contrast, the other four protocols assume a permissioned system and three of them require a trusted federated learning server. We also compare them in terms of the major assumptions they rely on. For instance, FoolsGold, SybilWall and MAB-RFL, assume that the attacker generates similar models for the controlled sybils, which make them ineffective against diverse SPA attacks, where this assumption does not hold. Other approaches, including ours, assume that honest nodes will produce similar models instead, which has been experimentally demonstrated [13], [14], [37]. This assumption avoids restricting sybil detection to specific types of attacks, providing broader applicability. On the other hand, our solution is the only one that imposes a limited time to submit model contributions each iteration.

TABLE I
Comparison of SPA defense mechanisms in the literature with properties, assumptions and types of attacks resilience.

		Major assumptions						**Poisoning type**				**SPA type**	
System	Max sybils tolerated	Permissionless	Trusted server	Restricted models access	Sybil models similarity	Honest models similarity	Restricted contribution time	SPA targeted	SPA untargeted	SPA random	Solitary (no sybil-based)	Uniform	Diverse
FoolsGold [8]	∞	○	●	●	●	○	○	●	○	○	○	●	○
SybilWall [38]	∞	○	○	○	●	○	○	●	○	○	○	●	○
Jiang et al. [39]	$\frac{N-2}{2}$	○	●	●	○	●	○	○	●	○	○	●	●
MAB-RFL [37]	$\frac{N-2}{2}$	○	●	○	●	○	○	●	●	○	●	●	○
SyDeLP (ours)	$\frac{N-4}{2}$	●	○	○	○	●	●	●	●	●	●	●	●

Finally, we compare the protocols based on the protection they offer against different poisoning attacks, solitary attacks, and types of SPAs. Our protocol is the only one covering all categories. MAB-RFL also provides good coverage, but it lacks protection against random poisoning and the diverse sybil-based approach. The other three protocols offer less coverage.

Under the assumptions outlined in Table I, both FoolsGold and SybilWall claim to resist an unbounded number of sybils.

However, even an attacker with minimal computational power could submit an arbitrarily large number of updates at no cost, while verifiers would need to expend substantial computing resources to validate them. In contrast, our approach discards updates before they are even considered for evaluation unless the attacker has invested significantly more computational resources.

Our protocol is neither a Byzantine detection nor a sybil detection mechanism; rather, it is designed to be implemented on top of existing BTA functions to make SPAs more computationally expensive for attackers, thus providing enhanced security guarantees.

IV. Sybil-resistant Decentralized Learning Protocol (SyDeLP)

In this section, we introduce the Sybil-resistant Decentralized Learning Protocol (SyDeLP) that integrates Adaptive Proof of Work (APoW) to mitigate SPAs.

A. Overview

We identify two sets of nodes: training nodes $\mathcal{C}$ and verifier nodes $\mathcal{V}$. Training nodes own private datasets, train models locally, and submit model contributions by solving Proofs of Work. Verifier nodes maintain the network by receiving model contributions, verifying them, and aggregating them. In a coupled setup, nodes handle both training and verification, while in a decoupled setup, these roles are separated. Our protocol supports both configurations.

We use blockchain technology to provide a secure and tamper-proof record of the training process. This enables an open and trustless environment where all activities within the protocol can be independently verified. Verifier nodes serve as blockchain maintainers, responsible for recording the training process. Each block represents a single iteration and includes the individual model contributions as well as the aggregated global model. Alternatively, models can be stored off-chain on decentralized file systems and linked to transactions on-chain [40], [41] to reduce storage overhead, but for simplicity, we will assume they are written directly on the blockchain.

APoW is used to limit the influence of sybil attackers. For each submitted model contribution, training nodes are required to present a PoW of a target difficulty. This difficulty is initially the same for any new node but can be individually reduced or increased over time, this is the adaptive factor. Difficulty information is also recorded in the blockchain, allowing for global verification. The PoW requirement mitigates the impact of sybil attackers because, for each new sybil, the attacker is required to solve a different PoW to be considered. We will later discuss how this is enforced in the protocol. However, PoW alone does not prevent poisoning by nodes presenting proofs, and honest nodes face high computational demands generating these proofs. To address both issues, we use a BTA function to filter out suspicious contributions to assign PoW difficulty reductions for models selected for aggregation.

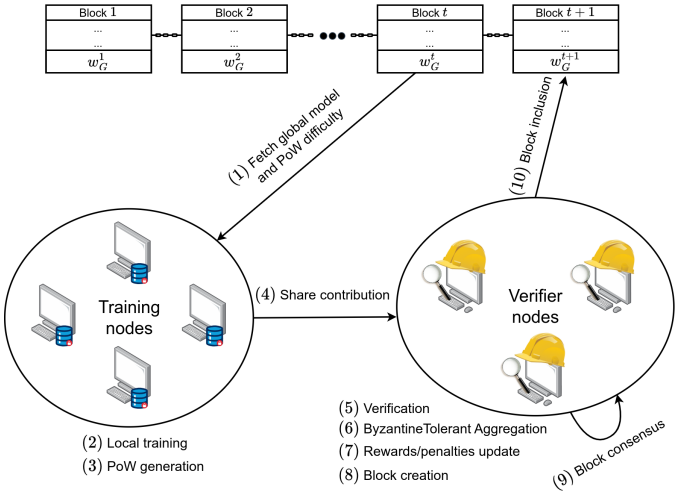

Fig. 1. SyDeLP overview in a single iteration t.

Figure 1 shows a step-by-step illustration of a single iteration of SyDeLP. It starts with training nodes fetching the global model and their current Proof of Work (PoW) difficulties from the blockchain (step 1). Then, they locally update the global model using their local datasets (step 2) and produce the PoW (step 3). The updated model and the PoW are packed into a transaction and shared with verifier nodes (step 4). Verifier nodes validate each incoming transaction (step 5), immediately discarding transactions with invalid Proofs of Work. A BTA function is used to aggregate the models, filtering out potential malicious models (step 6). Training nodes whose models are selected by the BTA function for aggregation receive rewards in the form of difficulty reductions for the next iteration, while owners of unselected models are penalized with an increase in difficulty (step 7). The received transactions, along with the updated difficulties and the global model update, are packaged into a block (step 8). Verifier nodes then use a consensus algorithm to agree on the correct next block (step 9). Finally, the agreed-upon block is added to the blockchain to be used in the next iteration (step 10). This process is repeated a predefined number of times, T, to allow the model to converge.

In Algorithm 1, we present the overall functioning of the protocol. In the following subsections, we described the details of each component.

B. Objectives

The protocol is designed to meet the following objectives:

1) **Sybil-based poisoning attacks mitigation**: By strengthening BTA functions, sybil attacks aiming to introduce poisoning into the global model can be effectively countered.
2) **Permissionless**: The protocol is completely decentralized and open, meaning that anyone can join or leave at any time.
3) **Trustlessness**: No trusted party is required for any purpose in the protocol. Operations can be independently audited and verified with cryptographic guarantees.

Algorithm 1 SyDeLP

for $t \in [1, 2, \ldots, T]$ **do**
  // Transactions creation
  for each $c \in \mathcal{C}$ **in parallel do**
    Get w_G^t and Φ_c from the blockchain.
    // Use $\Phi_c = 0$ for new nodes
    $w_c^{t+1} = w_G^t - \eta \nabla_{w_G^t} \mathcal{L}(X_c, w_G^t)$ *// Local update*
    $w_{sig} = sign(w_c^{t+1}, pk_c, sk_c)$
    $D_c = f(\Phi_c)D$
    $(y, \pi) = VDP(pk_c, w_{sig}, H, D_c)$
    $tx_c^{t+1} = (w_c^{t+1}, \Phi_c, w_{sig}, pk_c, (y, \pi), H)$
    Send tx_c^{t+1} to all $v \in \mathcal{V}$
  end for

  // Block creation
  for each $v \in \mathcal{V}$ **in parallel do**
    // Byzantine/honest labeling
    $\xi = \mathcal{A}(\{w_c^{t+1} \mid c \in \mathcal{C}, \text{is\_valid}(tx_c^{t+1})\})$
    $\mathcal{HC} = \emptyset$ *// Set of honest contributions*
    for each $c \in \mathcal{C}$ **do**
      if $\xi_c == 0$ **then** *// If honest*
        $tx_c^{t+1}(\Phi_c) = tx_c^{t+1}(\Phi_c) + 1$ *// Reward*
        $\mathcal{HC} = \mathcal{HC} \cup \{w_c^{t+1}\}$
      else
        // Penalize
        $tx_c^{t+1}(\Phi_c) = \max(tx_c^{t+1}(\Phi_c) - 1, 0)$
      end if
    end for

    $w_G^{t+1} = FedAVG(\mathcal{HC})$
    $B_v^{t+1} = (\{tx_c^{t+1} \mid c \in \mathcal{C}\}, w_G^{t+1}, H)$
  end for

  Consensus on block state
end for

C. Threat model

Adversary goals. The adversary aims to corrupt the global model by submitting poisoned model updates through sybil identities, either introducing a backdoor or preventing the convergence. Specifically, the attacker's goal is to introduce more than β sybils into the protocol to break the security guarantees of the BTA function $\mathcal{A}$ on aggregation. β refers to the number of byzantine nodes tolerated by $\mathcal{A}$. Privacy considerations such as data reconstruction attacks [42], [43] on individual model updates are not addressed in this work due to the BTA function's requirement to access individual model updates for computing similarities. However, privacy can still be explored by integrating BTA functions with techniques like Differential Privacy, as demonstrated in [44]. We leave this as future work.

Adversary capabilities. The adversary can create an unlimited number of sybils (in the form of pseudo identities), but has constant and limited computing power P, defined as the number of Proofs of Work with initial difficulty D that he can solve in one training iteration, where $P \leq \beta$. The adversary can also generate honest model updates (indistinguishable from other honest updates during validation) and can shift from honest to malicious behavior at any time. For the controlled sybils, we consider the case where the attacker may submit different poisoning contributions for each sybil (even with different poisoning strategies), which is more realistic than the state of the art assumptions regarding sybil models similarities [8], [37], [38].

D. Assumptions

1) In each iteration, training nodes have a limited time τ to locally update the global model and generate the required PoW. This constraint prevents the adversary from computing more than P initial difficulty Proofs of Work per iteration, but requires synchronization among verifier nodes. An approach such as the one proposed by Regnath et al. [45] for trustless date-time synchronization on blockchain may be used to achieve it.
2) Honest nodes have sufficient computing power to meet Assumption 1, enabling them to solve a PoW with difficulty D in time τ. This is complementary to the standard assumption in decentralized learning, where it is assumed that training nodes have enough computing resources to locally train the model.
3) Verifier nodes receive the model updates from all honest training nodes each iteration. This requires that honest training nodes are connected to all verifiers, or at least to one honest verifier, which can then broadcast the updates to the rest of the verifiers via a gossip protocol.
4) Verifier nodes can reach a consensus on the next block by employing a consensus mechanism such as Proof of Work [17], Proof of Stake [18], etc. This implicitly involves the assumptions of the respective consensus mechanisms, such as a majority of honest computing power in the case of PoW.
5) Honest training nodes produce similar model updates to other honest nodes. This assumption is necessary for BTA functions and has been experimentally observed [8], [13]. Our conducted experiments have also confirmed this behavior (see Section VI).

E. Initialization

The protocol begins with the initialization phase where a set of nodes launches the learning task by determining the initial learning hyperparameters, such as neural network architecture, the number of layers, the learning rate, and the protocol parameters. This can be achieved through consensus algorithms, as in [27].

Once the parameters are decided, the genesis block is created. It contains the model hyperparameters, initial global model parameters w_G^1, which are typically initialized as a vector with random values, the initial (and maximum possible) difficulty D for the individual Proofs of Work, the total number of iterations T, the security parameter $\alpha \geq 1$ (see Section V),

the BTA function to be used, and its β parameter. The difficulty D is defined as the number of operations required to solve the puzzle.

Additionally, each training node c generates a key pair (pk_c, sk_c) to digitally sign messages. The public key will be used to link nodes to digital identities and to track their contributions and difficulties in the blockchain. Note that nodes are free to change their keys at any time, but this will result in the loss of any difficulty reductions they may have obtained.

F. Transaction creation

At every iteration t, each training node c fetches the current global model w_G^t from the last block to locally update it by computing the gradient with respect to its private dataset X_c such that: $w_c^{t+1} = w_G^t - \eta \nabla_{w_G^t} \mathcal{L}(X_c, w_G^t)$, where $\mathcal{L}$ is the loss function and η is the learning rate. Then, the updated model is signed as $w_{sig} = sign(w_c^{t+1}, pk_c, sk_c)$ with the node's key pair to prevent tampering and repudiation when sharing it.

To have the model contribution considered, node c must present a PoW that meets its current target difficulty. Nodes' difficulties are individually updated each iteration depending on their model updates, specifically whether they are selected for aggregation by $\mathcal{A}$ or not. Honest nodes continuously selected will accumulate reductions over time, while a sybil attacker incurs high computational costs to maintain the sybils in the system.

To compute its current difficulty, node c can obtain its contribution score, Φ_c, from its most recent transaction submitted to the blockchain. Nodes contributing for the first time will not have any transaction reference, in this case they use $\Phi_c = 0$. The contribution score Φ_c is then used to compute the current target difficulty as explained in Section IV-G.

The next step involves solving a Verifiable Delay Puzzle (VDP), expressed as $(y, \pi) = VDP(pk_c, w_{sig}, H, D_c)$, where y is the result of the puzzle and π the proof that allows efficient verification of correctness. This function requires D_c sequential (non parallelizable) operations to solve. The puzzle is generated using as input the public key pk_c, the digital signature of the current model w_{sig}, and the hash reference to the last block H. Including the public key watermarks the puzzle, preventing the adversary from reusing proofs across different sybils, as only one model contribution is considered per public key each iteration. Additionally, including the hash reference of the last block prevents the precomputation of proofs since this value cannot be known in advance.

With all the generated data, node c creates a new transaction $tx_c^{t+1} = (w_c^{t+1}, \Phi_c, w_{sig}, pk_c, (y, \pi), H)$ that is sent to the connected verifiers, who are assumed to broadcast the received transactions to all their verifier peers.

G. Adaptive Proof of Work (APoW)

To enforce the second assumption outlined in Section IV-D, which requires honest nodes to update the model and generate a PoW of difficulty D in time τ, we adopt a Verifiable Delay Function puzzle construction. VDFs do not suffer from high variance in the expected number of operations needed to solve the puzzle, unlike other computational puzzles like SHA-256 based Bitcoin's PoW. This allows honest nodes to solve the puzzle in approximately the same time. We use the VDF proposed by Wesolowski [46]:

Let h be a cryptographic hash function, D the difficulty (number of required operations), int a function that maps binary strings to its non-negative integer representation, bin a function that maps non-negative integers to its binary representation, N the modulus of an RSA group G, h_{prime} a hash function that maps any input to a random large prime number and x some data. The non-interactive VDF evaluation works as follows:

1) The prover computes $g = \text{int}(h(\text{“residue”}||x)) \mod N$, where $||$ denotes concatenation, and $y = g^{2^D} \mod N$. Using the Fiat-Shamir heuristic, a non-interactive proof is generated by computing $l = h_{prime}(\text{bin}(g)||\text{bin}(y))$ and $\pi = g^{\lfloor 2^D / l \rfloor} \mod N$. Once finished, the prover sends (x, y, π) to the verifier.
2) The verifier computes the same g and l using x and y. Additionally, he computes $r = 2^D \mod l$. He accepts the proof if $g, y, \pi \in G$ and $\pi^l g^r \mod N = y$, which confirms that the prover correctly computed both π and y.

This VDF construction relies on the sequential-squaring conjecture that computing $g^{2^D} \mod N$ for a uniform g requires at least D sequential steps [47]. The verification of correctness requires essentially two exponentiations and one product $(\pi^l g^r \mod N)$ in the group G which can be efficiently computed guaranteeing that the prover has done D operations.

Note that in this construction, knowing the factorization of N would enable a node to solve the puzzle significantly faster. However, we refer the reader to the original paper [46] where a construction based on the class group of the imaginary quadratic field of unknown order is proposed for scenarios where the private key (the factorization of N) should remain undisclosed.

In our protocol, the data x used to generate the puzzle includes the node's public key, the digital signature of the current model update, and the hash reference of the last block. We refer to this puzzle construction as a Verifiable Delay Puzzle (VDP). The difficulty of the puzzle is individually computed as $D_c = \lceil Df(\Phi_c) \rceil$, for a node c, where f is the difficulty adjustment function given by:

$$f(\Phi) = \left(\frac{T - \alpha}{T - 1} \right)^{\Phi} \tag{2}$$

f takes as input the contribution score and outputs a value in the range $[0, 1]$, which serves as adjustment coefficient of the initial difficulty D. Consequently, $\Phi = 0$ produces $f(0) = 1$, as new nodes always face the maximum difficulty. Note that $f(0) \geq f(1) \geq f(2) \geq \cdots \geq f(n)$.

Here, $\alpha \geq 1$ denotes the security parameter that adjusts the trade-off between security and difficulty reduction. Specif-

ically, α scales the tolerance threshold of $\mathcal{A}$ as $\frac{1}{\alpha}\beta$, against a sybil attacker. This implies that a sybil attacker would need at least $P = \frac{1}{\alpha}\beta + 1$ computational power to compromise the system in a worst-case attack, but the system is still tolerant to β total malicious nodes. We present a formal discussion on the worst-case attack and the security of f in Section V.

Setting $\alpha = 1$ yields $f(\Phi) = \left(\frac{T-1}{T-1}\right)^{\Phi} = 1^{\Phi} = 1$, resulting in a constant maximum difficulty. This case provides optimal security because β is not scaled but requires all nodes to perform proofs of maximum difficulty each time, with no possible reductions. Figure 2 illustrates the evolution of difficulty for a consistently honest-labeled node over 100 iterations with different values of α. Lower values of α (closer to 1) enhance security but limit difficulty reduction. With $\alpha = 2$, there is a balanced trade-off: the protocol can tolerate a sybil attacker with $P = \frac{1}{2}\beta$ computing power, while allowing difficulty reductions for honest nodes. By iteration 70, the difficulty would halve and continue to decrease to less than 0.4.

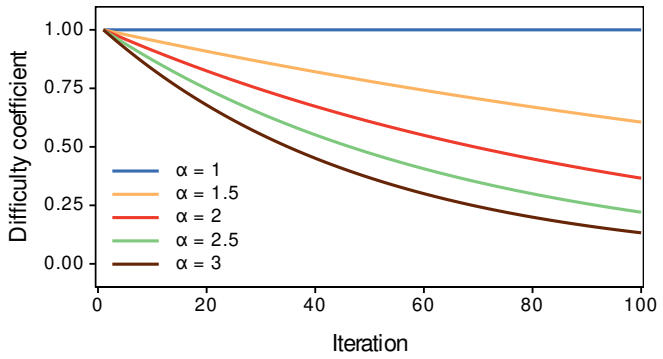

Fig. 2. Difficulty reduction over time for different values of α with $T = 100$

H. Transaction verification

Each time a verifier receives a new transaction, it checks whether the transaction adheres to the protocol rules. In the event of any inconsistency, the transaction is immediately discarded. The verification process consists of: (1) checking that the transaction is within the valid time frame τ for receiving transactions; (2) ensuring that no other valid model contribution has been previously submitted in the current iteration using the provided public key; (3) verifying the digital signature against both the public key and the signed data; and (4) verifying the correctness of the solution y and the proof π of the VDP, which requires verifying that the correct Φ_c was used to calculate the target difficulty for the sending node. Only transactions that satisfy these four conditions are considered and included in the new block. These verification steps were included in line 17 of Algorithm 1 as a high-level function *is_valid*, ensuring that only valid transactions are considered during block creation (from line 15 to 27).

I. Block creation

When the time frame for receiving updates ends, verifier nodes run $\mathcal{A}$ to detect potentially poisoned contributions. We will use Multi-KRUM to implement SyDeLP, but note that any BTA function can be used instead. With Multi-KRUM, the $N - \beta - 2$ models with the lowest distance scores are labeled as honest (with 0), and the rest as byzantine (with 1).

False positives may occur when honest contributions are incorrectly labeled as byzantine due to the nature of BTA functions, which discard the most dissimilar updates even in the absence of attacks. In such cases, these honest nodes will be penalized and they will have to present proofs with higher difficulties in the next iteration. This reflects the security/efficiency trade-off in our protocol. By employing APoW, we strengthen the BTA functions at the cost of increasing computational demands on all participants. The computational burden for securing the protocol does not lie with a single entity but is distributed among all the training clients.

After byzantine/honest labeling, the contribution score of each client is updated accordingly by verifier nodes, increasing by 1 for honest-labeled models and decreasing by 1 for the others. Contribution scores are lower-bounded at 0, which yields the initial difficulty D. This bound prevents nodes from bypassing higher difficulties by rejoining with a new identity to reset their difficulty back to the initial value D.

Then, the set of honest-labeled models $\mathcal{HC}$ is aggregated using element-wise averaging (FedAVG [1]) into a global model $w_G^{t+1} = FedAVG(\mathcal{HC})$. Finally, the set of all transactions $\{tx_c^{t+1} \mid c \in \mathcal{C}\}$ (with the updated contribution scores), the new global model w_G^{t+1}, and the hash reference of the previous block H, are bundled into a new block B_v^{t+1} to be added into the blockchain.

If the verifier nodes follow the protocol rules and correctly compute the protocol's functions, they should arrive at a similar block. However, they will use a consensus mechanism to agree on the next block. Since the choice of consensus mechanism does not directly impact the problem under investigation, any existing consensus mechanism can be employed.

V. Security Analysis

In this section we present a formal security analysis of the difficulty adjustment function f used in SyDeLP to mitigate SPAs. We describe a worst-case attack that will naturally lead to the definition of f, previously introduced in Equation 2, that maintains the resilience to poisoning on aggregation in the presence of a sybil attacker.

A. Worst-case attack

Assume that any poisoning attempt, when the total number of malicious nodes in the system is lower or equal than β can be effectively identified by $\mathcal{A}$. Similarly, assume that honest models generated by the attacker are always labeled as honest by $\mathcal{A}$, producing difficulty reductions for those sybils.

Let $P < \beta$, in the first iteration, an attacker can introduce $S_1 = P$ sybils into the protocol, by definition of P. Then, the attacker generates honest models for all of them, which

will produce difficulty reductions in the next iteration. In the second iteration, the attacker would need to expend work equivalent to $S_1 f(1)$ to keep the initial sybils, where $f(1) < f(0) = 1$ is the reduction coefficient. Since $S_1 f(1) < P$, the attacker could introduce $S_2 = P - S_1 f(1)$ new sybils in this iteration. The attacker can continue doing this until the number of sybils under his control exceeds β, breaking the security guarantee on aggregation of $\mathcal{A}$.

B. Difficulty adjustment function

With a constant maximum difficulty on the PoW, the security of the protocol is guaranteed against a sybil attacker with computing power $P \leq \beta$. When difficulty reduction is considered, we can express this same condition as $\alpha P \leq \beta$, for some $\alpha > 1$, because the attacker would be able to introduce more than P sybils as described in the worst-case attack.

Let S_i be the number of sybils introduced at iteration i, with $S_1 = P$. The goal of the difficulty function f is to bound the total number of sybils introduced after T iterations to at most αP, that is:

$$\sum_{i=1}^{T} S_i \leq \alpha P \tag{3}$$

Suppose that after the first iteration, f can effectively limit the number of new introduced sybils to a constant sP, in each subsequent iteration, where $0 < s < 1$. Then, we can set Equation 3 to the maximum possible value, αP, and express it as:

$$\sum_{i=1}^{T} S_i = P + (T-1)sP = \alpha P \tag{4}$$

Solving for s we have:

$$s = \frac{\alpha - 1}{T - 1} \tag{5}$$

Considering this value of s, we will show that the difficulty adjustment function f, meets the desired security constraint of Equation 3 in the presence of a worst-case attack. First, note that for an always-honest labeled node c, his contribution score at iteration j is $\Phi_c = j - i$, where i is the iteration it started to contribute. Suppose a sybil attacker is following the strategy of the worst-case attack, then the computing power required at iteration j to keep the S_i sybils introduced at iteration i is $S_i f(j-i)$. Given that the attacker's computing power is limited and constant, the following constraint must be met at every iteration j for the work required for all the introduced sybils so far:

$$\sum_{i=1}^{j} S_i f(j-i) \leq P \tag{6}$$

Considering that $S_1 = P$, and $S_i = sP, \forall i > 1$, we can rewrite Equation 6 as:

$$Pf(j-1) + \sum_{i=2}^{j} sPf(j-i) \leq P \tag{7}$$

Solving for $f(j-1)$ we have:

$$\begin{aligned} f(j-1) + s\sum_{i=2}^{j} f(j-i) &\leq 1 \\ f(j-1) &\leq 1 - s\sum_{i=0}^{j-2} f(j-i-2) \end{aligned} \tag{8}$$

Treating this inequality as an equality gives the highest possible value for $f(j-1)$, allowing us to derive a recursive definition for f. To derive a non-recursive definition, we rewrite Equation 8 as a function with parameter Φ as:

$$f(\Phi) = 1 - s\sum_{i=0}^{\Phi-1} f(i) \tag{9}$$

We will show by induction that this expression follows the binomial expansion $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$, with $x = 1$ and $y = -s$.

Theorem 1. *The closed-form expression of Equation 9 is* $f(\Phi) = (1-s)^{\Phi}$.

Proof. We verify the base case for $\Phi = 0$, so $f(0) = (1-s)^0 = 1$. Now, assume the statement holds for some $\Phi = k$, we get: $f(k) = (1-s)^k$. Finally, we show that it is also true for $\Phi = k+1$. From the recursive definition of f (Equation 9) and the induction hypothesis we have:

$$f(k+1) = 1 - s\sum_{i=0}^{k} f(i) = 1 - s\sum_{i=0}^{k} (1-s)^i \tag{10}$$

The summation $\sum_{i=0}^{k} (1-s)^i$ is the well known geometric series $\sum_{i=0}^{n} a_0 r^i = a_0 \frac{1-r^{n+1}}{1-r}$ with initial value $a_0 = 1$ and common ratio $r = 1 - s$, so it is equivalent to:

$$\sum_{i=0}^{k} (1-s)^i = \frac{1-(1-s)^{k+1}}{1-(1-s)} = \frac{1-(1-s)^{k+1}}{s} \tag{11}$$

substituting in 10 we have:

$$\begin{aligned} f(k+1) &= 1 - s\left(\frac{1-(1-s)^{k+1}}{s}\right) \\ &= 1 - \left(1-(1-s)^{k+1}\right) \\ &= (1-s)^{k+1} \end{aligned} \tag{12}$$

which is what we aimed to prove. Therefore, our initial hypothesis $f(\Phi) = (1-s)^{\Phi}$ is correct. □

Substituting the value of s from Equation 5, f becomes:

$$f(\Phi) = \left(1 - \frac{\alpha - 1}{T - 1}\right)^{\Phi} = \left(\frac{T - \alpha}{T - 1}\right)^{\Phi} \tag{13}$$

which is the same expression presented in Equation 2. Note that this definition of f meets the security requirements, allowing an attacker with computing power P to introduce at most $\alpha P \leq \beta$ sybils after T iterations.

VI. Evaluation

In this section, we describe the experiments conducted to compare SyDeLP with two state-of-the-art SPA mitigation protocols. The objective of this comparison is to assess resilience against different types of poisoning attacks and their impact on model utility. We evaluated a fixed setup with one-third of the network nodes being malicious. The implementation of the blockchain component and the evaluation of SyDeLP under worst-case attack scenarios are left for future work. The code to reproduce the experiments can be found on Github at https://github.com/brandon-mosqueda/sydelp.

A. Datasets and models

MNIST [48]: A widely used dataset for evaluating decentralized learning, consisting of 70,000 images of handwritten digits, with 60,000 designated for the training set and 10,000 for the testing set. Each image is a grayscale, 28 x 28-pixel square, resulting in a total of 784 pixels per image. This dataset is used for multi-class classification, where each class corresponds to a decimal digit from 0 to 9. We employ a well-known deep learning multilayer perceptron architecture [1], which includes one hidden layer with 100 units using the ReLU activation function and an output layer of 10 units using the softmax activation function. For all the experiments, we set the learning rate to 0.001, the batch size to 16, and the local epochs to 10.

UCI SMS Spam Collection [49]: Similar to the the experiments conducted in [50], we adopt this dataset for binary classification on text data. The SMS Spam dataset comprises 5,572 english SMS messages categorized as spam (13%) or not spam (87%). We randomly split the dataset into training and testing sets, allocating 80% and 20% of the data, respectively, while preserving the original labels proportions (spam and not spam). The model used is a Long-Short Term Memory (LSTM) with an embedding dimension of 64 and an LSTM layer with 64 units. The output layer consists of a single unit with sigmoid activation function. For all the experiments we set the learning rate to 0.001, the batch size to 8 and the local epochs to 5.

B. Data distribution

We conducted experiments exclusively under the non-independent and identically distributed (non-IID) data partitioning scenario, as it is the most realistic setting. The training sets in both datasets were divided into partitions using the Dirichlet distribution, which has been used for generating non-IID data partitions on decentralized learning simulations [16], [35], [38]. This approach provides a flexible mechanism to control the degree of data heterogeneity among clients through the concentration parameter $\sigma \in \mathbb{R}^+$. Lower values of σ results in higher degree of non-IID partitions, while higher values approach the uniform distribution. The Dirichlet distribution naturally creates imbalanced partitions, affecting both the number of samples and the class distribution within each partition. We set $\sigma = 0.1$ to achieve a high degree of non-IID in the partitions.

C. Attacks

We evaluated three sybil-based and one non sybil-based poisoning attack to assess the effectiveness of the defense mechanisms. All the attacks were carried out under both scenarios, uniform and diverse (see Section II-B).

Label flipping [51]: In this attack, the malicious nodes change all the labels of a source class to a target class in their local datasets. The goal is to induce the model to misclassify samples of the source class as the target class. For the MNIST dataset, we set the source class to 1 and the target class to 9. For the SMS Spam dataset, the source class is 1 (spam) and the target class is 0 (not spam).

Sign flipping [52]: This is an untargeted attack where the malicious clients normally train their models but then they multiply their updates by a negative constant σ, inverting the direction of the actual gradients. We set $\sigma = -3$ for the experiments.

Random [13]: As detailed in Section II-B, in the random attack, malicious nodes generate and share random vectors drawn from the Gaussian distribution. For our experiments, we use Gaussian distribution parameters $\mu = 3$ and $\sigma = 2$.

Solitary random: This is the only non-sybil based attack. It is similar to the random attack but involves only four malicious nodes.

D. Evaluated protocols

No defense: This refers to the baseline setup where no defense mechanism is employed. For consistency and better comparison, we assume that, at each iteration, all the models are aggregated into a global model to compute the metrics, similar to SyDeLP and MAB-RFL.

MAB-RFL [37]: Proposed for federated learning, in MAB-RFL, the model selection process for aggregation is modeled as an extended multi-armed bandit (MAB) problem, where high quality updates are more likely to be selected. A sybil detection mechanism based on a graph constructed from cosine similarities is proposed, while non sybil-based attacks are addressed using an approach based on Principal Component Analysis and agglomerative clustering. For the SMS Spam dataset we set the protocol parameters $\alpha = 0.1$, $c_{max} = 0.93$ and $c_{min} = 0.93$, while for the MNIST dataset, we used $\alpha = -0.1$, $c_{max} = 0.7$ and $c_{min} = 0.3$. For further details about these parameters, we refer the reader to the original paper [37].

SybilWall [38]: This system adapts the core concepts of FoolsGold [8] for decentralized learning, removing the need for a central server. In SybilWall, contributions are locally

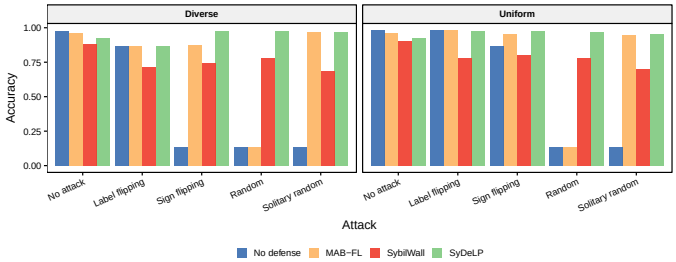

Fig. 3. Model accuracy of SMS Spam dataset.

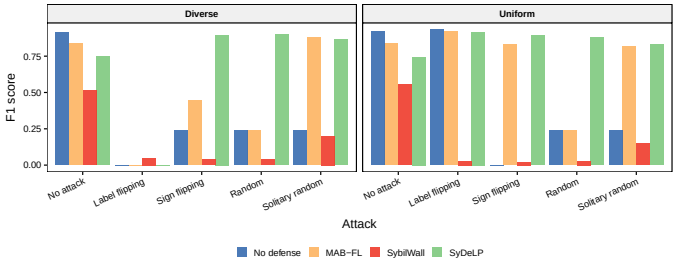

Fig. 4. F1 score on SMS Spam dataset.

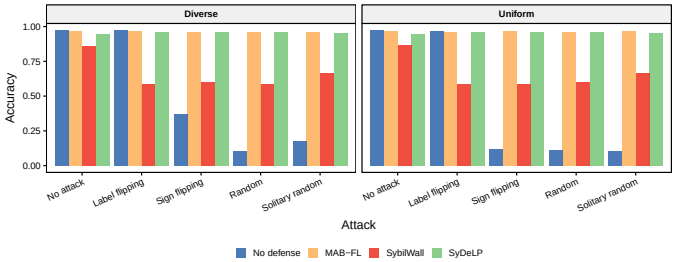

Fig. 5. Model accuracy of MNIST dataset.

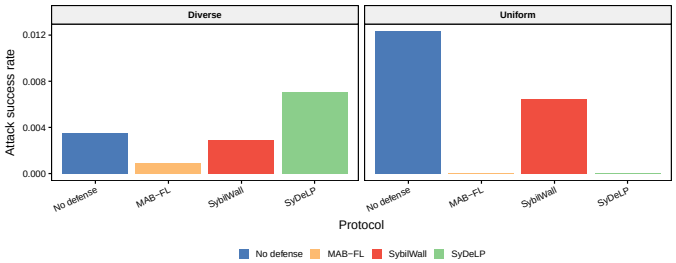

Fig. 6. Attack success rate on MNIST dataset.

weighted based on their cosine similarity with other received contributions. Updates with high similarity are assumed to originate from sybil attackers and are assigned lower weights. A probabilistic gossip mechanism is introduced to incorporate information from farther nodes, increasing the likelihood of detecting sybil updates. We evaluated this system on a random regular graph with degree 8 and a distant propagation relevance parameter $\lambda = 0.8$. As this is the only system where aggregation does not occur globally, we report metrics aggregated solely from honest nodes.

SyDeLP: We evaluated our proposal using Multi-KRUM as BTA function with $\beta = 40$. No simulations on the actual blockchain and PoW implementation were conducted, so setting a value for α is not required here.

E. Metrics

We evaluate the model accuracy for both datasets, as both are classification tasks (Figures 3 and 5). For the MNIST dataset, under the label flipping attack, we also report the attack success rate in Figure 6, defined as the proportion of source class samples in the testing misclassified as the target class [8]. On the other hand, for the SMS Spam dataset, we use the F1 score to evaluate the balance between the precision and recall (Figure 4). Note that the F1 score also captures the attack success rate in this case as it involves binary classification. Metrics are reported for the models in the final iteration on the testing set to assess the overall effectiveness of the learning process.

F. Setup

For all the experiments, we use 100 nodes and 100 iterations. In the three sybil-based attacks, the number of malicious nodes is set to 33, while in the solitary attack, only 4 malicious nodes are used. We implemented the experiments in Python 3.9 using the Keras 3.6 [53] library with Tensorflow 2.17 [54] as backend for the deep learning models.

The experiments were conducted on a computer running Ubuntu 22.04, equipped with an Intel i9-13950HX 13th Gen processor featuring 22 cores and clock speeds of up to 5.3 GHz.

G. Experimental results

UCI SMS Spam Collection: As shown in Figure 3, the three evaluated defense mechanisms produce a similar accuracy in the absence of attacks. However, accuracy significantly decreases under attacks when no defense or SybilWall is employed. Given the unbalanced nature of this dataset, accuracy alone cannot capture the effectiveness of the protocols against poisoning attacks. Therefore, in Figure 4, we present the results in terms of F1 score, which better reflects the impact of attacks on the underrepresented label (spam). Without attacking, MAB-RFL outperforms the other defenses. However, under diverse poisoning attacks, MAB-RFL exhibits a noticeable performance reduction. This is expected as both MAB-RFL and SybilWall were designed for uniform attacks, assuming that sybil attackers produce similar updates. Notably, even in a uniform attack, the random attack is only mitigated by SyDeLP. Finally, none of the defense mechanism could mitigate the label flipping attack on the diverse scenario. This failure is reflected with the almost zero F1 score, as all the spam samples were incorrectly classified as non-spam. We hypothesize that label flipping identification on binary classification tasks (or text data) cannot be captured by similarity measures. Further investigation is needed to address this case.

MNIST: This dataset exhibits more stable results across the evaluated cases. Figure 5 shows the accuracy results. The model's accuracy is not significantly impacted under any attack when using SyDeLP and MAB-RFL. In contrast, SybilWall presents an abrupt reduction when facing the attacks. It is noteworthy that the label-flipping attack is entirely unsuccessful in this dataset. Even without defense mechanism, the model's accuracy remains unaffected and the attack success rate remains low, as shown in Figure 6. Among the evaluated protocols, SyDeLP shows the highest attack success rate for the diverse case, but it is negligible since the attack success rate is less than 1%.

VII. Discussion

For the experimental evaluation on the robustness of SyDeLP against SPAs, we performed extensive experiments and compare it with two solutions in the literature. Our results demonstrate that SyDeLP maintains performance similar to the no-attack scenario across a broader range of attacks. MAB-RFL also performs comparably, except under diverse SPAs on the SMS Spam dataset. However, a critical limitation of MAB-RFL is the lack of practical guidelines for setting its key parameters, c_{min} and c_{max}, which define the cosine similarity thresholds for identifying sybils.

For the MNIST dataset, we used the parameters provided in the original work. However, for the SMS Spam dataset, which was not part of the original evaluation, determining appropriate values required several exploratory experiments that were not trivial. We observed high cosine similarity among honest updates through the learning process, requiring $c_{max} = c_{min} = 0.93$. Lower values disrupted the training process entirely. Finding such parameters is not a practical task in a real-world scenario without incurring significant costs.

In contrast, SyDeLP's parameters, β and α, are easier to define as they directly control the security-cost trade-off. β represents the total number of tolerated malicious nodes, while $\frac{1}{\alpha}\beta$ represents the computational resources required to compromise the system under the worst-case attack.

Addressing privacy concerns is essential, as privacy leakage remains a significant challenge in decentralized learning. Although studies have suggested that BTA functions are compatible with differential privacy [29], [30], [35], other approaches, such as cryptographic-based methods, should also be investigated.

Finally, we would like to explore additional reward mechanisms for incentivizing honest participation beyond difficulty reductions. A natural extension would be incorporating monetary incentives, similar to cryptocurrency systems like Bitcoin. For instance, nodes could be rewarded with digital assets based on the frequency with which their models are selected for aggregation. These rewards could be distributed iteratively or at the conclusion of the learning process, fostering continued and honest participation.

VIII. Conclusion

We proposed SyDeLP, a decentralized learning protocol designed to mitigate Sybil-based poisoning attacks (SPAs) in permissionless settings. By combining Byzantine Tolerant Aggregation functions for poisoning detection with Adaptive Proof of Work to counter sybil attacks, SyDeLP introduces a robust framework for securing decentralized learning systems. The protocol's security is governed by two comprehensive parameters, β and α, which enable a practical balance between resilience and computational efficiency.

Through experimental evaluation on two benchmark datasets, we compared SyDeLP with two baseline protocols across ten attack scenarios. The results demonstrate that SyDeLP consistently outperforms alternatives in terms of robustness and overall model utility.

While we are optimistic that this work represents a significant step toward trustless decentralized learning systems with verifiable operations, further research is needed. Specifically, trustless initialization and efficient verification methods for training correctness remain open challenges. Additionally, relaxing current assumptions, such as the need for synchronization and access to global aggregation, could expand the applicability of SyDeLP to peer-to-peer settings where only local aggregation is feasible.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016. [Online]. Available: https://arxiv.org/abs/1602.05629

[2] E. Hallaji, R. Razavi-Far, M. Saif, B. Wang, and Q. Yang, "Decentralized federated learning: A survey on security and privacy," *IEEE Transactions on Big Data*, vol. 10, no. 2, pp. 194–213, 2024.

[3] E. M. El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 3521–3530. [Online]. Available: https://proceedings.mlr.press/v80/mhamdi18a.html

[4] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," 2017. [Online]. Available: https://arxiv.org/abs/1705.09056

[5] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[6] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2023, pp. 175–199.

[7] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[8] A. E. Samy and Š. Girdzijauskas, "Mitigating sybil attacks in federated learning," in *Information Security Practice and Experience*, W. Meng, Z. Yan, and V. Piuri, Eds. Singapore: Springer Nature Singapore, 2023, pp. 36–51.

[9] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *CoRR*, vol. abs/1712.05526, 2017. [Online]. Available: http://arxiv.org/abs/1712.05526

[10] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2938–2948. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

[11] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/ec1c59141046cd1866bbbcdfb6ae31d4-Paper.pdf

[12] F. Lin, Q. Ling, and Z. Xiong, "Byzantine-resilient distributed large-scale matrix completion," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8167–8171.

[13] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[14] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5650–5659. [Online]. Available: https://proceedings.mlr.press/v80/yin18a.html

[15] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1639–1654, 2022.

[16] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," in *Computer Security – ESORICS 2021*, E. Bertino, H. Shulman, and M. Waidner, Eds. Cham: Springer International Publishing, 2021, pp. 455–475.

[17] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2008.

[18] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," https://ethereum.org/en/whitepaper/, 2013, accessed: 2023-07-08.

[19] M. S. Ferdous, M. J. M. Chowdhury, M. A. Hoque, and A. Colman, "Blockchain consensus algorithms: A survey," *arXiv preprint arXiv:2001.07091*, 2020.

[20] M. Jakobsson and A. Juels, *Proofs of Work and Bread Pudding Protocols(Extended Abstract)*. Boston, MA: Springer US, 1999, pp. 258–272. [Online]. Available: https://doi.org/10.1007/978-0-387-35568-9_18

[21] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Annual International Cryptology Conference*. Springer, 1992, pp. 139–147.

[22] D. Stebila, L. Kuppusamy, J. Rangasamy, C. Boyd, and J. Gonzalez Nieto, "Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols," in *Topics in Cryptology–CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*. Springer, 2011, pp. 284–301.

[23] B. Awerbuch and C. Scheideler, "Group spreading: A protocol for provably secure distributed name service," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2004, pp. 183–195.

[24] P. Maniatis, D. S. Rosenthal, M. Roussopoulos, M. Baker, T. J. Giuli, and Y. Muliadi, "Preserving peer replicas by rate-limited sampled voting," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003, pp. 44–59.

[25] B. Cohen and K. Pietrzak, "Simple proofs of sequential work," in *Advances in Cryptology – EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds. Cham: Springer International Publishing, 2018, pp. 451–467.

[26] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Annual international cryptology conference*. Springer, 2018, pp. 757–788.

[27] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 1178–1187.

[28] B. Chen, H. Zeng, T. Xiang, S. Guo, T. Zhang, and Y. Liu, "Esb-fl: Efficient and secure blockchain-based federated learning with fair payment," *IEEE Transactions on Big Data*, 2022.

[29] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, 2021.

[30] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2438–2455, 2021.

[31] M. Xu, Z. Zou, Y. Cheng, Q. Hu, D. Yu, and X. Cheng, "Spdl: A blockchain-enabled secure and privacy-preserving decentralized learning system," *IEEE Transactions on Computers*, vol. 72, no. 2, pp. 548–558, 2023.

[32] C. Fang, Y. Guo, J. Ma, H. Xie, and Y. Wang, "A privacy-preserving and verifiable federated learning method based on blockchain," *Computer Communications*, vol. 186, pp. 1–11, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422000081

[33] V. Mugunthan, R. Rahman, and L. Kagal, "Blockflow: An accountable and privacy-preserving solution for federated learning," *arXiv preprint arXiv:2007.03856*, 2020.

[34] J. Heiss, E. Grünewald, S. Tai, N. Haimerl, and S. Schulte, "Advancing blockchain-based federated learning through verifiable off-chain computations," in *2022 IEEE International Conference on Blockchain (Blockchain)*, 2022, pp. 194–201.

[35] A. Mondal, H. Virk, and D. Gupta, "BEAS: blockchain enabled asynchronous & secure federated machine learning," *CoRR*, vol. abs/2202.02817, 2022. [Online]. Available: https://arxiv.org/abs/2202.02817

[36] W. Boitier, A. D. Pozzo, Álvaro García-Pérez, S. Gazut, P. Jobic, A. Lemaire, E. Mahe, A. Mayoue, M. Perion, T. F. Rezende, D. Singh, and S. Tucci-Piergiovanni, "Fantastyc: Blockchain-based federated learning made secure and practical," 2024. [Online]. Available: https://arxiv.org/abs/2406.03608

[37] W. Wan, S. Hu, J. Lu, L. Y. Zhang, H. Jin, and Y. He, "Shielding federated learning: Robust aggregation with adaptive client selection," *arXiv preprint arXiv:2204.13256*, 2022.

[38] T. Werthenbach and J. Pouwelse, "Towards sybil resilience in decentralized learning," 2023.

[39] Y. Jiang, Y. Li, Y. Zhou, and X. Zheng, "Mitigating sybil attacks on differential privacy based federated learning, 2020," *Cited on*, p. 98, 2010.

[40] C. Li, Q. Shen, C. Xiang, and B. Ramesh, "A trustless federated framework for decentralized and confidential deep learning," in *2022 IEEE 1st Global Emerging Technology Blockchain Forum: Blockchain & Beyond (iGETblockchain)*, 2022, pp. 1–6.

[41] A. Qammar, A. Karim, H. Ning, and J. Ding, "Securing federated learning with blockchain: a systematic literature review," *Artificial Intelligence Review*, vol. 56, no. 5, pp. 3951–3985, 2023.

[42] L. Zhu, Z. Liu, and S. Han, *Deep leakage from gradients*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[43] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 16 937–16 947. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf

[44] R. Guerraoui, N. Gupta, R. Pinot, S. Rouault, and J. Stephan, "Differential privacy and byzantine resilience in sgd: Do they add up?" in *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, 2021, pp. 391–401.

[45] E. Regnath, N. Shivaraman, S. Shreejith, A. Easwaran, and S. Steinhorst, "Blockchain, what time is it? trustless datetime synchronization for iot," in *2020 International Conference on Omni-layer Intelligent Systems (COINS)*. IEEE, 2020, pp. 1–6.

[46] B. Wesolowski, "Efficient verifiable delay functions," in *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*. Springer, 2019, pp. 379–407.

[47] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," 1996.

[48] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," *Pattern Recognition*, vol. 26, no. 4, pp. 741–774, 1993. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[49] T. Almeida and J. Hidalgo, "SMS Spam Collection," UCI Machine Learning Repository, 2011, DOI: https://doi.org/10.24432/C5CC84.

[50] A.-T. Tran, T.-D. Luong, J. Karnjana, and V.-N. Huynh, "An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation," *Neurocomputing*, vol. 422, pp. 245–262, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220315095

[51] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security – ESORICS 2020*, L. Chen, N. Li, K. Liang, and S. Schneider, Eds. Cham: Springer International Publishing, 2020, pp. 480–501.

[52] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 1544–1551.

[53] J. Allaire and F. Chollet, *keras: R Interface to 'Keras'*, 2023, r package version 2.11.1. [Online]. Available: https://tensorflow.rstudio.com/

[54] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2015. [Online]. Available: https://www.tensorflow.org/