Mitigation of Sybil-based Poisoning Attacks in Permissionless Decentralized Learning

Brandon A. Mosqueda González, Omar Hasan, Lionel Brunie
University of Lyon, CNRS
INSA Lyon, LIRIS, UMR5205, F-69621
Lyon, France
{brandon.mosqueda-gonzalez, omar.hasan, 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. To overcome this challenge, 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 Verifiable Delay Function to counter sybil attacks requiring Proofs of Work (PoW) to participate. Honest behavior is incentivized by dynamically reducing PoW difficulty, decreasing the computational burden for honest nodes over time. Empirical evaluations conducted on a benchmark dataset across three types of poisoning attacks demonstrate that SyDeLP consistently outperforms existing solutions in terms of resilience.**

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

I. Introduction

Decentralized learning is a recent paradigm to collaboratively train machine learning models without having to share raw data. Participants operate in a peer-to-peer network where the model is iteratively refined by exchanging and aggregating model updates through a gossip protocol, which guaranteess the convergence of the model [1].

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 results. Although there has been substantial progress in decentralized learning to mitigate this threat [2], [3], existing solutions often rely on trusted parties.

In a permissionless setting, where no centralized entity governs access control, additional challenges arise. For instance, the sybil attack [4], where malicious participants create multiple fake entities to increase their influence in the system. Amplifying a poisoning attack with a sybil-based strategy leads to more severe consequences, allowing attackers to gain full control of aggregation [5].

In this paper, we address the problem of mitigating the Sybil-based Poisoning Attack (SPA) in a permisionless decentralized learning setup. Our proposal eliminates the centralized entities while providing robust security guarantees.

A. Contributions

The main contributions of this paper are summarized as follows:

1) We propose the first protocol addressing SPAs in permissionless decentralized learning, ensuring security without trusted parties.
2) We introduce a novel Adaptive Proof of Work (APoW) with dynamic difficulty adjustment to mitigate SPAs.
3) We evaluate our protocol on a real-world task against three poisoning attacks, demonstrating superior resilience over state-of-the-art solutions.

II. Background

A. Poisoning attacks

Poisoning attacks manipulate training data or model parameters to achieve a secondary goal. We identify four types: (1) **Targeted**: Degrades the global model's performance on specific classes [6]. (2) **Untargeted**: Reduces accuracy across all classes, hindering convergence [7]. (3) **Random**: Sends random Gaussian-distributed vectors without training [8]. (4) **Sybil-based**: Combines poisoning with a sybil attack [4], either *Uniform*, where all sybils use identical parameters, or *Diverse*, where sybils generate independent models.

B. Byzantine Tolerant Aggregation (BTA)

Byzantine Tolerant Aggregation ensures the aggregated model remains accurate despite malicious (a.k.a byzantine) nodes. They act as poisoning detection mechanisms, aiming to identify and exclude abnormal models from aggregation. For example, Multi-KRUM [8] aggregates 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. Multi-KRUM can tolerate up to $\beta \leq \frac{N-4}{2}$ byzantine clients and has theoretical guarantees for convergence.

III. Related work

In decentralized learning, the Sybil-based Poisoning Attack (SPA) has not received sufficient attention. Most works focus on sybil-tolerant consensus [9], [10] or assume a centralized

trusted party for access control [11], [12], which makes sybil attacks easier to prevent. Even permissionless systems like [9], [10], [13] overlook SPAs, despite their increased vulnerability. Other approaches [11], [14], require participants to stake collateral but do not address SPA risks or the need for a trusted access authority.

Existing SPA defenses, such as FoolsGold [5], SybilWall [15], and MAB-RFL [16], assume permissioned environments and attackers generating similar models, making them ineffective against diverse SPAs. In contrast, BTA functions, which we use in our solution, assume honest nodes produce similar models, a well-demonstrated property [8], [16], [17], allowing broader poisoning detection.

Current methods also suffer from high validation costs, while attackers can submit numerous updates cheaply. Our approach mitigates this by discarding updates early unless attackers invest significantly more computational resources.

Our protocol is neither a byzantine nor sybil detection mechanism but enhances existing BTA functions, increasing the computational cost of SPAs and improving security.

IV. Sybil-resistant Decentralized Learning Protocol (SyDeLP)

This section introduces the proposed protocol, SyDeLP, which integrates Adaptive Proof of Work (APoW).

A. Overview

SyDeLP defines two node sets: training nodes $\mathcal{C}$ and verifier nodes $\mathcal{V}$. Training nodes own private datasets, train models locally, and submit contributions by solving PoW. Verifier nodes maintain the network, verify contributions, and aggregate models.

We use blockchain technology to provide an open, secure and tamper-proof record of the training process. Verifier nodes serve as blockchain maintainers, recording the training process.

APoW limits sybil influence by requiring PoW for model submissions. Initially uniform, difficulty adjusts individually over time and is recorded on-chain for global verification. Since PoW alone does not prevent poisoning or reduce honest nodes' computational burden, a BTA function is used to filter suspicious models and to assign PoW difficulty reductions to selected contributions.

In Algorithm 1 we present the protocol, where for each of the T iterations nine steps are required: (1) Training nodes fetch the global model and their PoW difficulties from the blockchain. (2) They update the global model using local datasets and generate a PoW. (3) The updated model and PoW are packed into a transaction and shared with verifier nodes. (4) Verifier nodes validate transactions, discarding those with invalid PoWs. (5) A BTA function filters out malicious models before aggregation. (6) Selected models receive PoW difficulty reductions, while unselected ones face increased difficulty. (7) Transactions, updated difficulties, and the global model update are bundled into a block. (8) Verifier nodes reach consensus on the next block. (9) The block is added to the blockchain for the next iteration.

Algorithm 1 SyDeLP

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

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

  Consensus on block state
end for

B. Threat model

Adversary goals. The adversary aims to corrupt the global model by submitting poisoned model updates via sybil identities. The objective is to introduce more than β sybils, which is the security threshold of the BTA function.

Adversary capabilities. The adversary can generate unlimited sybil identities but has a fixed computing power $P \leq \beta$, defined as the number of Proofs of Work with initial difficulty D that he can solve in one training iteration. The adversary can also generate honest model updates and can shift from honest to malicious behavior at any time.

C. Assumptions

1) In each iteration, training nodes have a limited time τ to locally updates the global model and generate the required PoW. This constraint prevents the adversary from computing more than P Proofs of Work (of initial difficulty) per iteration.
2) Honest nodes have sufficient computing power to meet Assumption 1, enabling them to train the model and solve a PoW with difficulty D in time τ.
3) Honest verifier nodes receive the model updates from all honest training nodes each iteration.

D. Initialization

The protocol starts with nodes determining the learning and protocol parameters, such as neural network architecture, through consensus algorithms [9].

After parameter selection, the genesis block is created, storing model hyperparameters, initial global model parameters w_G^1, initial difficulty D, total iterations T, security parameter $\alpha \geq 1$ (see Section V), the BTA function, and its β parameter. Each training node c generates a key pair (pk_c, sk_c) for signing messages. Public keys link nodes to digital identities, tracking their contributions and difficulties. Nodes may change keys but previously earned difficulty reductions cannot be transferred.

E. Transaction creation

At every iteration t, each training node c fetches the current global model w_G^t from the last block and locally update it as $w_c^{t+1} = w_G^t - \eta \nabla_{w_G^t} \mathcal{L}(X_c, w_G^t)$, where η denotes the learning rate, $\mathcal{L}$ the loss function, and $\nabla_{w_G^t}$ the gradient with respect the global model. A digital signature w_{sig} of the resulting model is also generated. To have the model contribution considered, node c must present a PoW that meets its current target difficulty. The difficulty is dynamically adjusted based on the node's previous contributions using their contribution score Φ_c retrieved from its most recent transaction recorded on the blockchain. If a node is contributing for the first time, it sets $\Phi_c = 0$. The target difficulty is then computed as $D_c = \lceil Df(\Phi_c) \rceil$, where f is defined as:

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

Here, f scales the initial difficulty D based on Φ. The security parameter $\alpha \geq 1$ controls the trade-off between security and difficulty reduction. We present a formal discussion on the worst-case attack and the security of f in Section V.

To solve the PoW, node c must solve a Verifiable Delay Function (VDF), expressed as $(y, \pi) = VDF(pk_c, w_{sig}, H, D_c)$, where y is the result of the function, π the proof that allows efficient verification, and H the hash of the last block. The VDF requires D_c sequential (non parallelizable) operations to solve. The VDF includes the public key pk_c, preventing the adversary from reusing proofs across different sybils, as only one model contribution is considered per public key each iteration. Additionally, by including H, precomputation of VDF evaluations are avoided. Finally, node c creates a transaction $tx_c^{t+1} = (w_c^{t+1}, \Phi_c, w_{sig}, pk_c, (y, \pi), H)$ that is sent to verifier nodes.

We adopt the Verifiable Delay Function (VDF) construction proposed by Wesolowski [18]. Further technical details are provided in the extended version of this paper [19].

F. Block creation

Upon receiving a transaction, the verifier checks protocol compliance and discards any non-compliant transactions. Verification includes: (1) verifying the transaction is within the valid time frame τ; (2) ensuring no prior valid contribution exists for the public key in the current iteration; (3) checking the digital signature against the public key and signed data; and (4) validating the solution y and proof π of the VDF, ensuring the correct Φ_c was used for target difficulty. Only transactions meeting these conditions are included in the new block.

When the time frame for receiving updates ends, verifier nodes run BTA to detect potentially poisoned contributions. The contribution score of clients selected for aggregation is increased by 1, while for non-selected clients is decreased by 1. Then, the set of honest labeled models $\mathcal{HC}$ is aggregated into a global model $w_G^{t+1} = FedAVG(\mathcal{HC})$. $FedAVG$ refers to the coordinate-wise average of the models as in [20]. Finally, a block $B_v^{t+1} = (\{tx_c^{t+1} \mid c \in \mathcal{C}\}, w_G^{t+1}, H)$ is added to the blockchain via consensus.

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 (Equation 1), that maintains the resilience to poisoning on aggregation in the presence of a sybil attacker.

A. Worst-case attack

Assume BTA can identify poisoning attempts when malicious nodes are $\leq \beta$ and that honest models from an attacker are always labeled as honest, reducing difficulty for sybils.

Let $P < \beta$. In iteration 1, an attacker introduces $S_1 = P$ sybils (by definition of P), generating honest models to reduce difficulty in iteration 2. To maintain sybils, the attacker expends work $S_1 f(1)$, with $f(1) < 1$. Since $S_1 f(1) < P$, he can introduce $S_2 = P - S_1 f(1)$ new sybils. This continues until sybils exceed β, breaking BTA's security guarantee.

B. Difficulty adjustment function

With constant maximum PoW difficulty, security holds for $P \leq \beta$. With difficulty reduction, the same condition can be expressed as $\alpha P \leq \beta$, for some $\alpha > 1$. Let S_i be the number of sybils introduced at iteration i, with $S_1 = P$, and limiting new sybils per iteration to sP $(0 < s < 1)$, security holds if:

$$\sum_{i=1}^{T} S_i = P + (T-1)sP = \alpha P \tag{2}$$

Solving for s we have:

$$s = \frac{\alpha - 1}{T - 1} \tag{3}$$

For an always-honest node c, its contribution score at iteration j is $\Phi_c = j - i$, where i is its starting iteration. Thus, an attacker maintaining S_i sybils at iteration j requires work $S_i f(j-i)$. Given the attacker's limited computing power, the constraint at iteration j is:

$$Pf(j-1) + \sum_{i=2}^{j} sPf(j-i) \leq P \tag{4}$$

Solving for $f(j-1)$:

$$f(j-1) \leq 1 - s\sum_{i=0}^{j-2} f(j-i-2) \tag{5}$$

Rewriting in terms of Φ:

$$f(\Phi) = 1 - s\sum_{i=0}^{\Phi-1} f(i) \tag{6}$$

Theorem 1. *The closed-form solution of Equation 6 is* $f(\Phi) = (1-s)^{\Phi}$.

Proof. Base case: $\Phi = 0$, so $f(0) = (1-s)^0 = 1$. Assume for $\Phi = k$, $f(k) = (1-s)^k$. For $\Phi = k+1$:

$$f(k+1) = 1 - s\sum_{i=0}^{k} f(i) = 1 - s\sum_{i=0}^{k}(1-s)^i \tag{7}$$

Using the geometric series formula:

$$\sum_{i=0}^{k}(1-s)^i = \frac{1-(1-s)^{k+1}}{s} \tag{8}$$

Substituting in Equation 7:

$$f(k+1) = 1 - s\left(\frac{1-(1-s)^{k+1}}{s}\right) = (1-s)^{k+1} \tag{9}$$

Hence, $f(\Phi) = (1-s)^{\Phi}$. □

Substituting s from Equation 3:

$$f(\Phi) = (1-s)^{\Phi} = \left(\frac{T-\alpha}{T-1}\right)^{\Phi} \tag{10}$$

which matches Equation 1. This ensures an attacker with computing power P can 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 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. We refer the reader to the extended version [19] where further experiments are included.

Dataset and model: We use the UCI SMS Spam Collection [21], a dataset for binary classification on text data. We randomly split the dataset into training and testing sets, allocating 80% and 20% of the data, respectively. The model used is a Long-Short Term Memory (LSTM) with an embedding dimension of 64 and an LSTM layer with 64 units.

Attacks: We evaluated three poisoning attacks to assess the resilience of the defense mechanisms, namely: Label flipping [6], Sign flipping [22] and Random [8]. They were carried out under both scenarios, uniform and diverse.

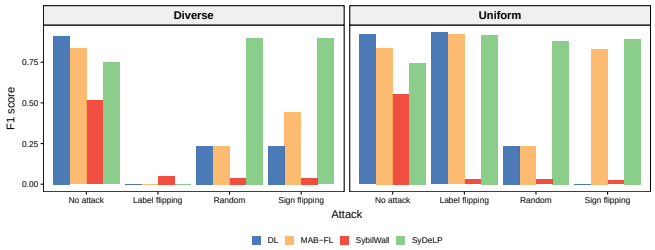

Fig. 1. F1 score on SMS Spam dataset.

Evaluated protocols: No attack (baseline), MAB-RFL [16] and SybilWall [15]. We use Multi-KRUM as BTA function for SyDeLP with $\beta = 40$. As the number of malicious clients is fixed, α is not required.

Metric: We report the F1 score (Figure 1) in the last iteration to evaluate the overall classification performance of the model.

Setup: For all the experiments, we use 100 nodes and 100 iterations. The number of malicious nodes is set to 33. We implemented the experiments in Python 3.9 using the Keras 3.6 library with Tensorflow 2.17 as backend for the deep learning models.

VII. Discussion and Conclusion

Our experimental results demonstrate that SyDeLP maintains performance similar to the baseline scenario across a broader range of attacks. MAB-RFL only performs comparably against uniform SPAs while SybilWall cannot effectively counter poisoning. 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. We hypothesize that label flipping identification on binary classification tasks (or text data) cannot be captured by similarity measures.

For future work, 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.

We are optimistic that this work represents a significant step toward trustless decentralized learning systems with verifiable operations, however, further research is needed. Specifically, trustless initialization and efficient verification methods for training correctness remain open challenges.

VIII. Acknowledgements

This work has been partially supported by the *Agence Nationale de la Recherche* (ANR) of the French government through the France 2030 program with the reference ANR-23-PEIA-005 (REDEEM project).

References

[1] 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

[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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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

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

[16] 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.

[17] 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

[18] 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.

[19] B. A. Mosqueda González, O. Hasan, and L. Brunie, "Mitigation of Sybil-based Poisoning Attacks in Permissionless Decentralized Learning," Jan. 2025, working paper or preprint. [Online]. Available: https://hal.science/hal-04892539

[20] 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

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

[22] 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.