Basil: A Fast and Byzantine-Resilient Approach for Decentralized Training

Ahmed Roushdy Elkordy **Saurav Prakash** **A. Salman Avestimehr** [*]
ECE Department
University of Southern California (USC)

1 Introduction and Overview of `Basil`

Decentralized machine learning methods are becoming core aspects of many important applications thanks to the large amounts of data generated on and held by the edge devices. We consider a decentralized training setup that does not rely on a central coordinator (e.g. parameter server). Instead, it only requires on-device computations on the edge nodes and peer-to-peer communications. The general optimization problem in this setting over r distributed devices is given as follows: $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left[f(\mathbf{x}) := \frac{1}{r}\sum_{i=1}^{r} f_i(\mathbf{x})\right]$, where $\mathbf{x}$ is the optimization variable, and $f_i(\mathbf{x})$ is the expected loss function of node i such that $f_i(\mathbf{x}) = \mathbb{E}_{\zeta_i\sim\mathcal{P}_i}[l_i(\mathbf{x},\zeta_i)]$. Here, $l_i(\mathbf{x},\zeta_i) \in \mathbb{R}$ denotes the loss function for model parameter $\mathbf{x} \in \mathbb{R}^d$ for a given realization (x_i, y_i) of ζ_i which is generated from a distribution $\mathcal{P}_i$.

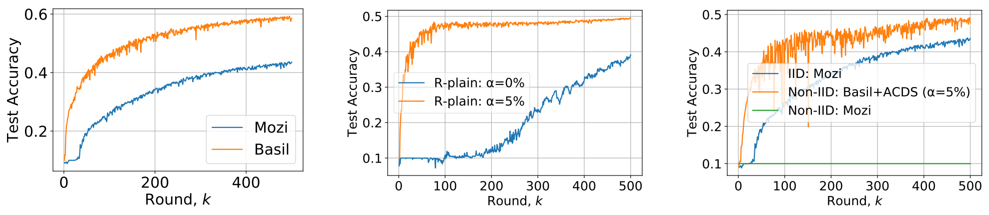
Although decentralized training provides many benefits, its decentralized nature makes it vulnerable to performance degradation due to system failures, malicious nodes and data heterogeneity [1]. Specifically, one of the key challenges for solving the decentralized training problem given above is the different threats that can alter the learning process, such as the software/hardware errors and adversarial attacks. Particularly, some of the clients can become faulty due to software bugs, hardware components which may behave arbitrarily, or even get hacked during training, sending arbitrary or malicious values to other clients, thus severely degrading the overall convergence performance. Such faults, where client nodes arbitrarily deviate from the agreed-upon protocol, are called Byzantine faults [2]. To mitigate Byzantine nodes in a graph-based decentralized setup where nodes are randomly connected by each other, some Byzantine robust optimization algorithms have been introduced recently, e.g., [3, 4]. In these algorithms, each node combines the set of models received from its neighbours by using robust aggregation rules, to ensure that the training is not impacted by the Byzantine nodes. However, to the best of our knowledge, none of these algorithms have considered the scenario when the data distribution at the nodes is heterogeneous. Data heterogeneity makes the detection of Byzantine nodes a daunting task, since it becomes unclear whether the model drift can be attributed to a Byzantine node, or to the very heterogeneous nature of the data. Even in the absence of Byzantine nodes, data heterogeneity can degrade the convergence rate [1].

1.1 Contributions

We summarize our main contributions below:

- We propose `Basil`[2], a fast and computationally efficient Byzantine robust algorithm for decentralized training. Unlike the prior Byzantine robust decentralized algorithms that are based on parallel training over a random graph, `Basil` algorithm achieves Byzantine robustness in decentralized training over a logical ring. Therefore, the training process in `Basil` is sequential, in which each node becomes active and performs model training only when it receives the model from its counterclockwise neighbour. Since nodes need not be active during the whole training time, `Basil` is

[*]Email: `{aelkordy,sauravpr,avestime}@usc.edu`.
[2]Basil I the Macedonian was a prominent ruler of the Byzantine Empire who reigned from 867 to 886.



(a) Gaussian Attack (IID setting) (b) No Attack (non-IID setting) (c) Gaussian Attack

Figure 1: A highlight of the performance benefits of `Basil`, compared with state-of-the-art (Mozi) [3], for CIFAR10 under different settings: In Figure (a), we can see the superior performance of `Basil` over Mozi with $\sim 16\%$ improvement of the test accuracy under Gaussian attack in the the IID setting. Figure (b) demonstrates that the test accuracy in the non-IID setting by using sequential training over the ring topology can be increased by up to $\sim 10\%$ in the absence of Byzantine nodes, when each node shares only 5% of its local data with other nodes. Figure (c) shows that ACDS on the top of Basil not only provides Byzantine robustness to Gaussian attack in the non-IID setting, but also gives higher performance than Mozi in the IID setting. Furthermore, Mozi for the non-IID setting completely fails in the presence of this attack. Further details are given in Appendix H.

quite suitable for scenarios where nodes, being phones or IoT devices, have limited computational resources, and thus remain dormant unless they are triggered to carry out their updates. In `Basil`, the defense technique to filter out Byzantine nodes is a performance-based strategy, wherein each node evaluates a received set of models from its counterclockwise neighbours by using its own local dataset to select the best candidate.

- We provide the theoretical guarantees of `Basil`. In particular, we show that Basil for convex loss functions in the IID data setting has a linear convergence rate with respect to the product of the number of benign nodes and the total training rounds over the ring. In other words, our theoretical result demonstrates a scalable performance for Basil with respect to the number of nodes.
- Using neural network with real-world CIFAR10 dataset, we demonstrate that `Basil` provides up to $\sim 16\%$ higher test accuracy when compared with Mozi, the state-of-the-art Byzantine-resilient decentralized learning scheme over graph, under different Byzantine attacks in the IID setting. In Figure 1(a), a sample result has been illustrated.
- For extending the superior benefits of `Basil` to the scenario when data distribution is non-IID across devices, we propose Anonymous Cyclic Data Sharing (ACDS) to be applied on top of `Basil`. To the best of our knowledge, no prior decentralized Byzantine robust algorithm has considered the scenario when the data distribution at the nodes is non-IID. ACDS allows each node to share a random fraction of its local non-sensitive dataset (e.g., landmarks images captured during tours) with all other nodes, while guaranteeing anonymity of the node identity. There are multiple real-world use cases where anonymous data sharing is sufficient to meet the privacy concerns of the users [3]. The proposed ACDS [4] scheme guarantees anonymous data sharing under the assumptions that there is no collusion between nodes, while providing robustness to Byzantine failures resulting from software/hardware errors or other non-malicious errors when it is integrated with `Basil`.
- We experimentally demonstrate that even when each node shares only 5% of its local data with all other nodes, the test accuracy for the naive sequential training over ring in the non-IID setting can be increased by up to $\sim 10\%$ in the absence of Byzantine nodes (Figure. 1(b)). Furthermore, we demonstrate that using ACDS on top of `Basil` provides resiliency to Byzantine behaviors unlike Mozi in the non-IID setting (Figure. 1(c)).

[3]For example, mobile users maybe fine with sharing some of their own text data, which does not contain any personal and sensitive information with others, as long as their personal identities remain anonymous. Another example is sharing of some non-private data (such as landmark images) collected by a person with others. In this scenario, although data itself is not generated at the users, revealing the identity of the users can potentially leak private information such as personal interests, location, or travel history. Our proposed ACDS strategy is suitable for such scenarios as it guarantees that the owner identity of each shared data point is kept hidden.

[4]Some other anonymous data sharing schemes (e.g., [5]) may fail in the presence of Byzantine faults as discussed in Appendix A, unlike our proposed ACDS which is robust to these faults.

2 Problem Statement

2.1 Decentralized system model

In this section, we formally define the decentralized learning system in the presence of Byzantine faults. In particular, we consider a decentralized learning setting in which a set $\mathcal{N} = \{1, \ldots, N\}$ of $|\mathcal{N}| = N$ nodes collaboratively train a machine learning (ML) model $\mathbf{x} \in \mathbb{R}^d$, where d is the model size, based on all the training data samples $\cup_{n \in \mathcal{N}} Z_n$ that are generated and stored at these distributed nodes, where the size of each local dataset is $|Z_i| = D_i$ data points. In this decentralized setup, we assume that there is no central parameter server, and consider the setup where training process is carried out in a sequential fashion over a clockwise directed ring. Each node in this ring topology takes part in the training process when it receives the model from the previous counterclockwise node. In Appendix, we propose a method in which nodes can consensually agree on a random ordering on a logical ring at the beginning of the training process, so that each node knows the logical ring positions of the other nodes. Therefore, without loss of generality, we assume for notation simplification that the indices of nodes in the ring are arranged in ascending order starting from node 1. In this setup, each node can send its model update to any set of users in the network.

In this decentralized setting, an unknown β-proportion of nodes can be Byzantine, where $\beta \in (0, 1)$, meaning they can send arbitrary and possibly malicious results to the other nodes. We denote $\mathcal{R}$ (with cardinality $|\mathcal{R}| = r$) and $\mathcal{B}$ (with cardinality $|\mathcal{B}| = b$) as the sets of benign nodes and Byzantine nodes, respectively. Furthermore, Byzantine nodes are uniformly distributed over the ring due to consensus-based random order agreement. Finally, we assume nodes can authenticate the source of a message, so no Byzantine node can forge its identity or create multiple fake ones [6].

2.2 Model training

The set $\mathcal{R}$ of benign nodes use their own datasets to train a shared model by solving the optimization problem given in the introduction in the presence of Byzantine nodes. The general update rule in this setting is given as follows. At the k-th round, the current active node i updates the global model according to:

$$\mathbf{x}_k^{(i)} = \bar{\mathbf{x}}_k^{(i)} - \eta_k^{(i)} g_i(\bar{\mathbf{x}}_k^{(i)}), \tag{1}$$

where $\bar{\mathbf{x}}_k^{(i)} = \mathcal{A}(\mathbf{x}_v^{(j)}, j \in \mathcal{N}, v = 1, \ldots, k)$ is the selected model by node i according to the aggregation rule $\mathcal{A}$, $g(\bar{\mathbf{x}}_k^{(i)})$ is the stochastic gradient computed by node i by using a random sample from its local dataset Z_i, and $\eta_k^{(i)}$ is the learning rate in round k used by node i.

Threat model: Byzantine node $i \in \mathcal{B}$ could send faulty or malicious update $\mathbf{x}_k^{(i)} = ^*$, where * denotes that $\mathbf{x}_k^{(i)}$ can be an arbitrary vector in $\mathbb{R}^d$.

Our goal is to design an algorithm for the decentralized training setup discussed earlier, while mitigating the impact of the Byzantine nodes. Towards achieving this goal, we propose `Basil` that is described next.

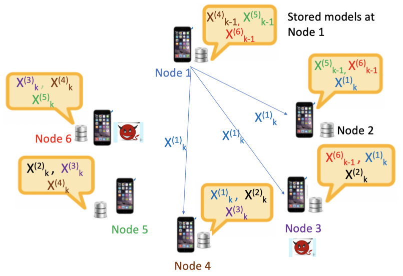


Figure 2: `Basil` scheme with $N = 6$ nodes where node 3 and node 6 are Byzantines. Node 1, the current active benign node in the k-th round, selects one model out of its stored 3 models which gives the lowest loss when it is evaluated on a min-batch from its local dataset Z_1. After that, node 1 updates the selected model by using the same mini-batch according to (5) before broadcasting it to the next 3 clockwise neighbours.

3 The Proposed Basil Algorithm

Now, we describe `Basil`, our proposed approach for mitigating both malicious updates and faulty updates in the IID setting, where the local dataset Z_i at node i consists IID data samples from a distribution $\mathcal{P}_i$, where $\mathcal{P}_i = \mathcal{P}$ for $i \in \mathcal{N}$, and characterize the complexity of `Basil`. After that, we extend `Basil` to the non-IID setting by integrating it to our proposed Anonymous Cyclic Data Sharing scheme.

3.1 Basil for IID setting

As illustrated in Figure 2, `Basil` leverages sequential training over the logical ring to mitigate the effect of Byzantine nodes. At a high level, in the k-th round, the current active node carries out the model update step in (5), and then broadcasts its updated model to the next $S = b + 1$ clockwise nodes, where b is the worst case number of Byzantine nodes. We note that broadcasting each model to the next $b+1$ neighbours is crucial to make sure that the benign subgraph, which is generated by excluding the Byzantine nodes, is connected. Connectivity of the benign subgraph is important as it ensures that each benign node can still receive information from a few other non-faulty nodes, i.e., the good updates can successfully propagate between the benign nodes. Even in the scenario where all Byzantine nodes come in a row, broadcasting each updated model to the next S clockwise neighbours allows the connectivity of benign nodes.

We now describe how the aggregation rule $\mathcal{A}_{\texttt{Basil}}$ in `Basil`, that a node i implements for obtaining the model $\bar{\mathbf{x}}_k^{(i)}$ for carrying out the update in (5), works. Node i stores the S latest models from its previous S counterclockwise neighbours. As highlighted in the previous paragraph, the reason for storing S models is to make sure that each stored set of models at node i contains at least one good model. When node i is active, it implements our proposed performance-based criteria to pick the best model out of its S stored models. In the following, we formally define our model selection criteria:

Definition 1 *(`Basil` Aggregation Rule) In the k-th round over the ring, let $\mathcal{N}_k^i = \{\mathbf{y}_1, \ldots, \mathbf{y}_S\}$ to be the set of S latest models from the S counterclockwise neighbours of node i. We define ζ_i to be a random sample from the local dataset Z_i, and let $l_i(\mathbf{y}_j) = l_i(\mathbf{y}_j, \zeta_i) \in \mathbb{R}$ to be the loss function of node i evaluated on the model $\mathbf{y}_j \in \mathcal{N}_k^i$, by using a random sample ζ_i. The proposed `Basil` aggregation rule is defined as $\bar{\mathbf{x}}_k^{(i)} = \mathcal{A}_{\texttt{Basil}}\left(\mathcal{N}_k^i\right) = \arg\min_{\mathbf{y} \in \mathcal{N}_k^i} \mathbb{E}\left[l_i(\mathbf{y}, \zeta_i)\right]$.*

In practice, node i can sample a mini-batch from its dataset and leverage it as validation data to test the performance (i.e., loss function value) of each the neighboring S models, and set $\bar{\mathbf{x}}_k^{(i)}$ to be the model with the lowest loss among the S stored models. As demonstrated in our experiments, this practical mini-batch implementation of the `Basil` criteria in Definition 1 is sufficient to mitigate Byzantine nodes in the network, while achieving superior performance over state-of-the-art.

We note that the connectivity parameter S can be relaxed $S < b+1$ while guaranteeing the success of Basil with high probability (Proposition 2 in Appendix C).

3.2 Theoretical guarantees

We derive the convergence guarantees of `Basil` under these standard assumptions: convexity, smoothness of the loss functions when the data distribution is IID (formal assumptions along with the proof of Theorem 1 are presented in Appendix D).

Theorem 1 *`Basil` with a fixed learning rate $\eta = \frac{1}{L}$ at all users achieves linear convergence with a constant error as follows:*

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{s=1}^{T}\mathbf{x}^s\right)\right] - f(\mathbf{x}^*) \leq \frac{||\mathbf{x}^0 - \mathbf{x}^*||^2 L}{2T} + \frac{1}{L}\sigma^2, \tag{2}$$

where $T = Kr$, K is the total number of rounds over the ring and r is the number of benign nodes. Here $\mathbf{x}^s$ represents the model after s update steps starting from the initial model $\mathbf{x}^0$, where $s = rk + i$ with $i = 1, \ldots, r$ and $k = 0, \ldots, K-1$. Furthermore, $\mathbf{x}^*$ is the optimal solution for the optimization problem given in the introduction and σ^2 is the stochastic gradient variance bound.

Remark 1 *The error bound for `Basil` decreases with increasing the total number of benign nodes $r = \beta N$, where $\beta \in (0,1)$.*

To extend `Basil` to be robust against software/hardware faults in the non-IID setting, i.e. the local dataset Z_i at node i consists of data samples from a distribution $\mathcal{P}_i$ with $\mathcal{P}_i \neq \mathcal{P}_j$ for $i, j \in \mathcal{N}$ and $i \neq j$, we gives the high level discussion of our Anonymous Cyclic Data Sharing scheme (ACDS) in the following subsection.

3.3 Generalizing Basil to non-IID setting via Anonymous Cyclic Data Sharing

We propose the ACDS scheme that allows each node to anonymously share α fraction of its non-sensitive local dataset with other nodes. At a high level, this is done by first arranging nodes in different rings. Then, within each ring, each node sends a batch of its selected sharable dataset to the next clockwise node. The next node then selects a batch from its own sharable dataset, and uniformly shuffles the combination of its batch and the received dataset from its counterclockwise node. The accumulated dataset at the last node in each ring is shared with all other nodes in all other groups. The process is repeated till the α fraction of the local dataset is shared between all users. The detailed discussion of ACDS along with its anonymity guarantees are presented in Appendix E.

References

[1] P. Kairouz, H. B. McMahan, and e. a. Brendan, “Advances and open problems in federated learning,” *preprint arXiv:1912.04977*, 2019.

[2] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, p. 382–401, Jul. 1982.

[3] S. Guo, T. Zhang, X. Xie, L. Ma, T. Xiang, and Y. Liu, “Towards byzantine-resilient learning in decentralized systems,” *preprint arXiv:2002.08569*, 2020.

[4] Z. Yang and W. U. Bajwa, “Bridge: Byzantine-resilient decentralized gradient descent,” *preprint arXiv:1908.08098*, 2019.

[5] L. A. Dunning and R. Kresman, “Privacy preserving data sharing with anonymous id assignment,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 2, pp. 402–413, 2013.

[6] M. Castro, B. Liskov, and et al., “Practical byzantine fault tolerance,” *OSDI*, vol. 173–186, 1999.

[7] J. Regatti, H. Chen, and A. Gupta, “Bygars: Byzantine sgd with arbitrary number of attackers,” *preprint arXiv:2006.13421*, 2020.

[8] C. Xie, S. Koyejo, and I. Gupta, “Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6893–6901.

[9] ——, “Zeno++: Robust fully asynchronous SGD,” in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 10 495–10 503.

[10] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, “Machine learning with adversaries: Byzantine tolerant gradient descent,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Curran Associates Inc., 2017, p. 118–128.

[11] 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. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 5650–5659.

[12] A. R. Elkordy and A. S. Avestimehr, “Secure aggregation with heterogeneous quantization in federated learning,” *arXiv preprint arXiv:2009.14388*, 2020.

[13] K. Pillutla, S. M. Kakade, and Z. Harchaoui, “Robust aggregation for federated learning,” *preprint arXiv:1912.13445*, 2019.

[14] L. Zhao, S. Hu, Q. Wang, J. Jiang, S. Chao, X. Luo, and P. Hu, “Shielding collaborative learning: Mitigating poisoning attacks through client-side detection,” *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.

[15] S. Prakash and A. S. Avestimehr, “Mitigating byzantine attacks in federated learning,” *arXiv preprint arXiv:2010.07541*, 2020.

[16] 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. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 3521–3530.

[17] J. woo Lee, J. Oh, S. Lim, S.-Y. Yun, and J.-G. Lee, “Tornadoaggregate: Accurate and scalable federated learning via the ring-based architecture,” *preprint arXiv:1806.00582*, 2020.

[18] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” *preprint arXiv:1806.00582*, 2018.

[19] A. Krizhevsky, “Learning multiple layers of features from tiny images,” 2009.

[20] Y. LeCun, C. Cortes, , and C. Burges, “The mnist database of handwritten digits,” *http://yann.lecun.com/exdb/mnist/*, 1998.

Appendices

Overview

In the following, we summarize the content of this supplementary document:

- In section A, we present some of the realted works
- In Section B, we describe how nodes consensually agree on their order on the ring in `Basil`.
- In Section C, we provide the communication, computations and storage cost of `Basil` Basil. Additionally, we show in Proposition 2 how the connectivity parameter S of `Basil` can be relaxed to $S < b + 1$, while guaranteeing the success of `Basil`.
- In Section D, we prove the convergence guarantees of `Basil`.
- In Section E, we formally present our proposed ACDS scheme.
- In Section F, we describe how `Basil` can be robust to nodes dropout.
- In Section G, we explain Mozi [3], the recent Byzantine robust decentralized algorithm.
- In Section H, we provide the numerical experiments by using MNIST dataset and CIFAR10 to show the performance of `Basil`.

A Related works

Many Byzantine robust strategies have been proposed recently for the distributed training setup (federated learning) where there is a central server to orchestrate the training process [7–15]. These Byzantine robust optimization algorithms combine the gradients received by all workers using robust aggregation rules, to ensure that training is not impacted by malicious nodes. Some of these strategies [10–13] are based on distance-based approaches, while some others are based on performance-based criteria [7–9, 15]. The key idea in distance-based defense solutions is to filter the updates that are far from the average of the updates from the benign nodes. It has been shown that distance-based solutions are vulnerable to the sophisticated attack proposed in [16]. In this attack, Byzantine nodes could create gradients that are malicious but indistinguishable from benign gradients in distance. On the other hand, performance-based filtering strategies rely on having some auxiliary dataset at the server to evaluate the model received from each node.

Compared to the large number of Byzantine robust training algorithms for distributed training in the presence of a central server, there have been only a few recent works on Byzantine resiliency in the decentralized training setup with no central coordinator. In particular, to address Byzantine failures in a decentralized training setup over a random graph under the scenario when the data distribution at the nodes is IID, the authors in [4] propose using a trimmed mean distance-based approach called BRIDGE to mitigate Byzantine nodes. However, the authors in [3] demonstrate that BRIDGE is defeated by the hidden attack proposed in [16]. To solve the limitations of the distance-based approaches in the decentralized setup, [3] proposes an algorithm called Mozi in which a combination of performance-based and distance-based stages are used to mitigate the Byzantine nodes, where the performance-based stage at a particular user leverages only its local dataset. As demonstrated numerically in [3], the combination of these two strategies allows Mozi to defeat the Byzantine attack proposed in [16]. However, Mozi is not suitable for the training over resource-constrained edge devices, as the training is carried out in parallel and nodes remain active all the time. In contrast, `Basil` is a fast and computationally efficient Byzantine robust algorithm, which leverages a novel sequential, memory assisted and performance based criteria for training over a logical ring while filtering the Byzantine users.

Data heterogeneity in the decentralized setting has been studied in some recent works, e.g., [17] in the absence of Byzantine nodes. In particular, the authors of TornadoAggregate [17] propose to cluster users into groups based on an algorithm called Group-BY-IID and CLUSTER where both use EMD (earth mover distance) that can approximately model the learning divergences between the models to complete the grouping. However, EMD function relies on having a publicly shared data at each node which can be collected similarly as in [18]. In particular, to improve training on non-IID data in federated learning, [18] proposed sharing of small portions of users' data with the server. The parameter server pools the received subsets of data thus creating a small subset of the data distributed at the clients, which is then globally shared between all the nodes to make the data distribution close to IID. However, the aforementioned data sharing approach is considered insecure in scenarios where users are fine with sharing some of their datasets with each other but want to keep their identities anonymous, i.e., data shares should not reveal who the data owners are.

As a final remark, we point out that for anonymous data sharing, [5] proposed an approach which is based on utilizing a secure sum operation along with anonymous ID assignment (AIDA). This involves computational operations at the nodes such as polynomial evaluations and some arithmetic operations such as modular operations. Thus, this algorithm may fail in the presence of Byzantine faults during these computations. Particularly, computation errors or software bugs can be present during the AIDA algorithm thus leading to the failure of anonymous ID assignment, or during the secure sum algorithm which can lead to distortion of the shared data.

B Order Agreement over the Ring

In practice, nodes can consensually agree on their order on the ring by using the following simple steps. 1) All nodes first share their IDs with each other, and we assume WLOG that nodes' IDs can be arranged in ascending order, and Byzantine nodes cannot forge their identities or create multiple fake ones [6]. 2) Each node locally generates the order permutation for the nodes' IDs by using pseudo random number generator (PRNG) initialized via a common seed (e.g., N). This ensures that all nodes will generate the same IDs order for the ring.

C Communication, Computation and Storage complexities of `Basil`

Proposition 1. *The communication, computation and storage complexities of `Basil` scheme are all $\mathcal{O}(Sd)$ for each node in each iteration, where d is the model size.*

Proof. Each node receives and stores the latest S models, calculates the loss by using each model out of the S stored models, and broadcasts its updated model to the next S clockwise neighbours. Thus, this results in $\mathcal{O}(Sd)$ communication, computation and storage costs. □

The costs in Proposition 1 can be reduced by relaxing the connectivity parameter S to $S < b + 1$ while guaranteeing the success of `Basil` (benign subgraph connectivity) with high probability, as formally presented in Proposition 2:

Proposition 2. *The number of models that each benign node needs to broadcast, store and evaluate for ensuring the connectivity and success of `Basil` can be relaxed to $S < b + 1$ while guaranteeing the success of `Basil` (benign subgraph connectivity) with high probability.*

Proof. This can be proven by showing that the benign subgraph, which is generated by removing the Byzantine nodes, is connected with high probability when each node broadcasts its updated model to the next $S < b + 1$ clockwise neighbours instead of $b + 1$ neighbours. Connectivity of the benign subgraph is important as it ensures that each benign node can still receive information from a few other non-faulty nodes. Hence by letting each node store and evaluate the latest S model updates, this ensures that each benign node has the chance to select one of the benign updates.

More formally, when each node broadcasts its model to the next S clockwise neighbors, we define A_j to be the failure event in which S Byzantine nodes come in a row where j is the starting node of these S nodes. When A_j occurs, there is at least one pair of benign nodes that have no link between them. The probability of A_j is given as follows:

$$\mathbb{P}(A_j) = \prod_{i=0}^{S-1} \frac{(b-i)}{(N-i)} = \frac{b!(N-S)!}{(b-S)!N!}, \tag{3}$$

where the second equality follows from the definition of factorial, while b, and N are the number of Byzantine nodes and the total number of nodes in the system, respectively. Thus, the probability of having a disconnected benign subgraph in `Basil`, i.e., S Byzantine nodes coming in a row, is given as follows:

$$\mathbb{P}(\text{Failure}) = \mathbb{P}(\bigcup_{j=1}^{N}(A_j)) \overset{(a)}{\leq} \sum_{j=1}^{N}\mathbb{P}(A_j) \overset{(b)}{=} \frac{b!(N-S)!}{(b-S)!(N-1)!}, \tag{4}$$

where (a) follows from union bound and (b) follows from (3). □

In order to further illustrate the impact of choosing S on the probability of failure given in (4), we consider the following numerical examples. Let the total number of nodes in the system be $N = 100$, where $b = 33$ of them are Byzantine, and the storage parameter $S = 15$. The failure event probability in (4) turns out to be $\sim 4 \times 10^{-7}$, which is negligible. For the case when $S = 10$, the probability of failure becomes $\sim 5.34 \times 10^{-4}$, which remains reasonably small.

D Convergence Analysis

In this section, we prove the main Theorem presented in Section 3.2 in the main paper. We start the proofs by stating the main assumptions and the update rule of `Basil`.

Assumption 1 (IID data distribution). *Local dataset Z_i at node i consists of IID data samples from a distribution $\mathcal{P}_i$, where $\mathcal{P}_i = \mathcal{P}$ for $i \in \mathcal{R}$. In other words, $f_i(\mathbf{x}) = \mathbb{E}_{\zeta_i \sim \mathcal{P}_i}[l(\mathbf{x}, \zeta_i)] = \mathbb{E}_{\zeta_j \sim \mathcal{P}_j}[l(\mathbf{x}, \zeta_i)] = f_j(\mathbf{x}) \, \forall i, j \in \mathcal{R}$. Hence, the global loss function $f(\mathbf{x}) = \mathbb{E}_{\zeta_i \sim \mathcal{P}_i}[l(\mathbf{x}, \zeta_i)]$.*

Assumption 2 (Bounded variance). *Stochastic gradient $g_i(\mathbf{x})$ is unbiased and variance bounded, i.e., $\mathbb{E}_{\mathcal{P}_i}[g_i(\mathbf{x})] = \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x})$, and $\mathbb{E}_{\mathcal{P}_i}||g_i(\mathbf{x}) - \nabla f_i(\mathbf{x})||^2 \leq \sigma^2$, where $g_i(\mathbf{x})$ is the stochastic gradient computed by node i by using a random sample ζ_i from its local dataset Z_i.*

Assumption 3 (Smoothness). *The loss functions $f_i^{\prime}s$ are L-smooth and twice differentiable, i.e., for any $\mathbf{x} \in \mathbb{R}^d$, we have $||\nabla^2 f_i(\mathbf{x})||_2 \leq L$.*

Let b^i to be the number of Byzantine nodes out of the S counterclockwise neighbours of node i. We divide the set of stored models $\mathcal{N}_k^i$ at node i in the k-th round into two sets. The first set $\mathcal{G}_k^i = \{\mathbf{y}_1, \dots, \mathbf{y}_{r^i}\}$ contains the benign models, where $r^i = S - b^i$. We consider scenarios with $S = b + 1$, where b is the total number of Byzantine nodes in the network. Without loss of generality, we assume the models in this set are arranged such that the first model is from the closest benign node in the neighbourhood of node i, while the last model is from the farthest node. Similarly, we define the second set $\mathcal{B}_k^i$ to be the set of models from the counterclockwise Byzantine neighbors of node i such that $\mathcal{B}_k^i \cup \mathcal{G}_k^i = \mathcal{N}_k^i$.

The general update rule in `Basil` is given as follows. At the k-th round, the current active node i updates the global model according to the following rule:

$$\mathbf{x}_k^{(i)} = \bar{\mathbf{x}}_k^{(i)} - \eta_k^{(i)} g_i(\bar{\mathbf{x}}_k^{(i)}), \tag{5}$$

where $\bar{\mathbf{x}}_k^{(i)}$ is given as follows

$$\bar{\mathbf{x}}_k^{(i)} = \arg\min_{\mathbf{y} \in \mathcal{N}_k^i} \mathbb{E}\left[l_i(\mathbf{y}, \zeta_i)\right]. \tag{6}$$

In order to prove the main theorem given in Section 3.2 in the main paper, we first prove the following:

Theorem 2 *When the learning rate $\eta_k^{(i)}$ for node $i \in \mathcal{R}$ in round k satisfies $\eta_k^{(i)} \geq \frac{1}{L}$, the expected loss function $\mathbb{E}\left[l_i(\cdot)\right]$ of node i evaluated on the set of models in $\mathcal{N}_k^i$ can be arranged as follows:*

$$\mathbb{E}\left[l_i(\mathbf{y}_1)\right] \leq \mathbb{E}\left[l_i(\mathbf{y}_2)\right] \leq \cdots \leq \mathbb{E}\left[l_i(\mathbf{y}_{r^i})\right] \leq \mathbb{E}\left[l_i(\mathbf{x})\right] \;\forall \mathbf{x} \in \mathcal{B}_k^i, \tag{7}$$

where $\mathcal{G}_k^i = \{\mathbf{y}_1, \dots, \mathbf{y}_{r^i}\}$ is the set of benign models stored at node i. Hence, the `Basil` aggregation rule in (6) is reduced to $\bar{\mathbf{x}}_k^{(i)} = \mathcal{A}_{\texttt{Basil}}(\mathcal{N}_k^i) = \mathbf{y}_1$. Hence, the model update step in (5) can be simplified as follows:

$$\mathbf{x}_k^{(i)} = \mathbf{y}_1 - \eta_k^{(i)} g_i(\mathbf{y}_1), \tag{8}$$

D.1 Proof of Theorem 1

We first show that if node i completed the performance-based criteria in (6) and selected the model $\mathbf{y}_1 \in \mathcal{G}_k^i$, and updated its model as follows:

$$\mathbf{x}_k^{(i)} = \mathbf{y}_1 - \eta_k^{(i)} g_i(\mathbf{y}_1), \tag{9}$$

we will have

$$\mathbb{E}\left[\ell_{i+1}(\mathbf{x}_k^{(i)})\right] \leq \mathbb{E}\left[\ell_{i+1}(\mathbf{y}_1)\right], \tag{10}$$

where $\ell_{i+1}(\mathbf{y}_1) = \ell_{i+1}(\mathbf{y}_1, \zeta_{i+1})$ is the loss function of node $i+1$ evaluated on a random sample ζ_{i+1} by using the model $\mathbf{y}_1$.

The proof of (10) is as follows: By using Taylor's theorem, there exists a γ such that

$$\begin{aligned}\ell_{i+1}(\mathbf{x}_k^{(i)}) =& \ell_{i+1}\left(\mathbf{y}_1 - \eta_k^{(i)} g_i(\mathbf{y}_1)\right) & (11)\\ =& \ell_{i+1}(\mathbf{y}_1) - \eta_k^{(i)} g_i(\mathbf{y}_1)^T g_{i+1}(\mathbf{y}_1) + \frac{1}{2}\eta_k^{(i)} g_i(\mathbf{y}_1)^T \nabla^2 \ell_{i+1}(\gamma)\eta_k^{(i)} g_i(\mathbf{y}_1), & (12)\end{aligned}$$

where $\nabla^2 \ell_{i+1}$ is the stochastic Hessian matrix. By using the following assumption

$$||\nabla^2 \ell_{i+1}(\mathbf{x}_k^{(i)})||_2 \leq L \text{ for all random samples } \zeta_{i+1} \text{ and any model } \mathbf{x} \in \mathbb{R}^d, \tag{13}$$

where L is the Lipschitz constant, we get

$$\ell_{i+1}(\mathbf{x}_k^{(i)}) \leq \ell_{i+1}(\mathbf{y}_1) - \eta_k^{(i)} g_i(\mathbf{y}_1)^T g_{i+1}(\mathbf{y}_1) + \frac{(\eta_k^{(i)})^2 L}{2}||g_i(\mathbf{y}_1)||^2. \tag{14}$$

By taking the expected value of both sides of this expression (where the expectation is taken over the randomness in the sample selection), we get

$$\begin{aligned}\mathbb{E}\left[\ell_{i+1}(\mathbf{x}_k^{(i)})\right] \leq& \mathbb{E}\left[\ell_{i+1}(\mathbf{y}_1)\right] - \eta_k^{(i)}\mathbb{E}\left[g_i(\mathbf{y}_1)^T g_{i+1}(\mathbf{y}_1)\right] + \frac{(\eta_k^{(i)})^2 L}{2}\mathbb{E}||g_i(\mathbf{y}_1)||^2\\ \overset{a}{=}& \mathbb{E}\left[\ell_{i+1}(\mathbf{y}_1)\right] - \eta_k^{(i)}\mathbb{E}\left[g_i(\mathbf{y}_1)^T\right]\mathbb{E}\left[g_{i+1}(\mathbf{y}_1)\right] + \frac{(\eta_k^{(i)})^2 L}{2}\mathbb{E}||g_i(\mathbf{y}_1)||^2\\ \leq& \mathbb{E}\left[\ell_{i+1}(\mathbf{y}_1)\right] - \eta_k^{(i)}\mathbb{E}\left[g_i(\mathbf{y}_1)^T\right]\mathbb{E}\left[g_{i+1}(\mathbf{y}_1)\right] + (\eta_k^{(i)})^2 L\mathbb{E}||g_i(\mathbf{y}_1)||^2\\ \overset{b}{\leq}& \mathbb{E}\left[\ell_{i+1}(\mathbf{y}_1)\right] - \eta_k^{(i)}||\nabla f(\mathbf{y}_1)||^2 + (\eta_k^{(i)})^2 L||\nabla f(\mathbf{y}_1)||^2 + (\eta_k^{(i)})^2 L\sigma^2\\ =& \mathbb{E}\left[\ell_{i+1}(\mathbf{y}_1)\right] - ||\nabla f(\mathbf{y}_1)||^2\left(\eta_k^{(i)} - (\eta_k^{(i)})^2 L\right) + (\eta_k^{(i)})^2 L\sigma^2 \qquad (15)\end{aligned}$$

where (a) follows from that the samples are drawn from independent data distribution, while (b) form Assumption 1 along with

$$\begin{aligned}\mathbb{E}||g_i(\mathbf{y}_1)||^2 =& \mathbb{E}||g_i(\mathbf{y}_1) - \mathbb{E}\left[g_i(\mathbf{y}_1)\right]||^2 + ||\mathbb{E}[g_i(\mathbf{y}_1)]||^2\\ \leq& \sigma^2 + ||\nabla f(\mathbf{y}_1)||^2. \qquad (16)\end{aligned}$$

Let $C_k^i = \frac{||\nabla f(\mathbf{y}_1)||^2}{||\nabla f(\mathbf{y}_1)||^2 + \sigma^2} = \frac{||\mathbb{E}[g_i(\mathbf{y}_1]||^2}{||\mathbb{E}[g_i(\mathbf{y}_1)]||^2 + \sigma^2}$, which implies that $C_k^i \in [0, 1]$. By selecting the learning rate as $\eta_k^{(i)} \geq \frac{1}{L}C_k^i$, we get

$$\mathbb{E}\left[l_{i+1}(\mathbf{y}_1)\right] \leq \mathbb{E}\left[l_{i+1}(\mathbf{y}_1)\right]. \tag{17}$$

Note that, nodes can just use a learning rate $\eta \geq \frac{1}{L}$, since $C_k^i \in [0, 1]$, while still achieving (17). This completes the first part of the proof.

By using (17), it can be easily seen that the update rule in equation (5) can be reduced to the case where each node updates its model based on the model received from the closest benign node (9) in its neighborhood, where this follows from using induction.

Let's consider this example. Consider a ring with N nodes and by using $S = 3$ while ignoring the Byzantine nodes for a while (assume all nodes are benign nodes). We consider the first round $k = 1$. With a little abuse of notations, we can get the following, the updated model by node 1 (the first node in the ring) $\mathbf{x}_1 = h(\mathbf{x}_0)$ is a function of the initial model $\mathbf{x}_0$ (updated by using the model $\mathbf{x}_0$). Now, node 2 has to select one model from the set of two models $\mathcal{N}_k^2 = \{\mathbf{x}_1 = h(\mathbf{x}_0), \mathbf{x}_0\}$. The selection is performed by evaluating the expected loss function of node 2 by using the criteria given in (6) on the models on the set $\mathcal{N}_k^2$. According to (17), node 2 will select the model $\mathbf{x}_1$ which results in lower expected loss. Now, node 2 updates its model based on the model $\mathbf{x}_1$, i.e., $\mathbf{x}_2 = h(\mathbf{x_1})$. After that, node 3 applies the aggregation rule in (6) to selects one model from this set of models $\mathcal{N}_k^3 = \{\mathbf{x}_2 = h(\mathbf{x_1}), \mathbf{x}_1 = h(\mathbf{x_0}), \mathbf{x}_0\}$. By using (17) and Assumption 1, we get

$$\mathbb{E}[l_3(\mathbf{x}_2)] \leq \mathbb{E}[l_3(\mathbf{x}_1)] \leq \mathbb{E}[l_3(\mathbf{x}_0)], \tag{18}$$

and node 3 model will be updated according to the model $\mathbf{x}_2$, i.e., $\mathbf{x}_3 = h(\mathbf{x}_2)$.

More generally, the set of stored benign models at node i is given by $\mathcal{N}_k^i = \{\mathbf{y}_1 = h(\mathbf{y}_2), \mathbf{y}_2 = h(\mathbf{y}_3), \ldots, \mathbf{y}_{r^i} = h(\mathbf{y}_{r^i-1})\}$, where r^i is the number of benign models in the set $\mathcal{N}_k^i$. According to (17), we will have the following

$$\mathbb{E}\left[l_i(\mathbf{y}_1)\right] \leq \mathbb{E}\left[l_i(\mathbf{y}_2)\right] \leq \cdots \leq \mathbb{E}\left[l_i(\mathbf{y}_{r^i})\right] \leq \mathbb{E}\left[l_i(\mathbf{x})\right] \;\; \forall \mathbf{x} \in \mathcal{B}_k^i, \tag{19}$$

where the last inequality in (19) follows from the fact that the Byzantine nodes are sending faulty models and their expected loss is supposed to be higher than the expected loss of the benign nodes.

According to this discussion and by removing the Byzantine nodes thanks to (19), we can only consider the benign subgraph which is generated by removing the Byzantine nodes according to the discussion in Section 3.1 in the main paper. Note that by letting each active node send its updated model to the next $b + 1$ nodes, where b is the total number of Byzantine nodes, the benign subgraph can always be connected. By considering the benign subgarph (the logical rings without Byzantine nodes), we assume without loss of generality that the indices of benign nodes in the ring are arranged in ascending order starting from node 1 to node r. In this benign subgraph, the update rule will be given as follows

$$\mathbf{x}_k^{(i)} = \mathbf{x}_k^{(i-1)} - \eta_k^{(i)} g_i(\mathbf{x}_k^{(i-1)}), \tag{20}$$

D.2 Proof of main theorem (Theorem 1) given in the main paper

By using Taylor's theorem, there exists a γ such that

$$\begin{aligned}
f(\mathbf{x}_k^{(i+1)}) &\overset{a}{=} f\left(\mathbf{x}_k^{(i)} - \eta_k^{(i)} g_{i+1}(\mathbf{x}_k^{(i)})\right) \\
&= f(\mathbf{x}_k^{(i)}) - \eta_k^{(i)} \left(g_{i+1}(\mathbf{x}_k^{(i)})\right)^T \nabla f(\mathbf{x}_k^{(i)}) \\
&\quad + \frac{1}{2}\eta_k^{(i)} \left(g_{i+1}(\mathbf{x}_k^{(i)})\right)^T \nabla^2 f(\gamma) \eta_k^{(i)} g_{i+1}(\mathbf{x}_k^{(i)}) \\
&\overset{b}{\leq} f(\mathbf{x}_k^{(i)}) - \eta_k^{(i)} \left(g_{i+1}(\mathbf{x}_k^{(i)})\right)^T \nabla f(\mathbf{x}_k^{(i)}) + \frac{L}{2}\eta_k^{(i)} ||g_{i+1}(\mathbf{x}_k^{(i)}))||^2
\end{aligned} \tag{21}$$

where (a) follows from the update rule in (20), while f is the global loss function in equation (1) in the main paper, and (b) from Assumption 3 where $||\nabla^2 f(\gamma)|| \leq L$. Given the model $\mathbf{x}_k^i$, we take expectation over the randomness in selection of sample ζ_{i+1} (the random sample used to get the model $\mathbf{x}_k^{(i+1)}$). We recall that ζ_{i+1} is drawn according to the distribution $\mathcal{P}_{i+1}$ and is independent of the model $\mathbf{x}_k^i$). Therefore, we get the following set of equations:

$$\begin{aligned}
\mathbb{E}[f(\mathbf{x}_k^{(i+1)})] &\leq f(\mathbf{x}_k^{(i)}) - \eta_k^{(i)} \mathbb{E}\left[(g_{i+1}(\mathbf{x}_k^{(i)}))\right]^T \nabla f(\mathbf{x}_k^{(i)}) + \frac{(\eta_k^{(i)})^2 L}{2} \mathbb{E}||g_{i+1}(\mathbf{x}_k^{(i)}))||^2 \\
&\overset{a}{\leq} f(\mathbf{x}_k^{(i)}) - \eta_k^{(i)} ||\nabla f(\mathbf{x}_k^{(i)})||^2 + \frac{(\eta_k^{(i)})^2 L}{2} ||\nabla f(\mathbf{x}_k^{(i)})||^2 + \frac{(\eta_k^{(i)})^2 L}{2} \sigma^2 \\
&= f(\mathbf{x}_k^{(i)}) - ||\nabla f(\mathbf{x}_k^{(i)})||^2 \left(\eta_k^{(i)} - \frac{(\eta_k^{(i)})^2 L}{2}\right) + \frac{(\eta_k^{(i)})^2 L}{2} \sigma^2 \\
&\overset{b}{\leq} f(\mathbf{x}_k^{(i)}) - \frac{\eta_k^{(i)}}{2} ||\nabla f(\mathbf{x}_k^{(i)})||^2 + \frac{\eta_k^{(i)}}{2} \sigma^2
\end{aligned} \tag{22}$$

where (a) follows from (16), and (b) by selecting $\eta_k^{(i)} \leq \frac{1}{L}$. Furthermore, in the proof of Theorem 1, we choose the learning to be $\eta_k^{(i)} \geq \frac{1}{L}$. Therefore, the learning rate will be given by $\eta_k^{(i)} = \frac{1}{L}$. By the convexity of the loss function f, we get the next inequality from the inequality in (22)

$$\mathbb{E}[f(\mathbf{x}_k^{(i+1)})] \leq f(\mathbf{x}^*) + \langle \nabla f(\mathbf{x}_k^{(i)}), \mathbf{x}_k^{(i)} - \mathbf{x}^* \rangle - \frac{\eta_k^{(i)}}{2} ||\nabla f(\mathbf{x}_k^{(i)})||^2 + \frac{\eta_k^{(i)}}{2}\sigma^2 \tag{23}$$

We now back-substitute $g_i(\mathbf{x}_k^{(i)})$ into (23) by using $\mathbb{E}[g_{i+1}(\mathbf{x}_k^{(i)})] = \nabla f(\mathbf{x}_k^{(i)})$ and $||\nabla f(\mathbf{x}_k^{(i)})||^2 \leq \sigma^2 - \mathbb{E}||g_{i+1}(\mathbf{x}_k^{(i)})||^2$:

$$\begin{aligned}\mathbb{E}[f(\mathbf{x}_k^{(i+1)})] \leq & f(\mathbf{x}^*) + \langle \mathbb{E}[g_{i+1}(\mathbf{x}_k^{(i)})], \mathbf{x}_k^{(i)} - \mathbf{x}^* \rangle - \frac{\eta_k^{(i)}}{2} \mathbb{E}||g_{i+1}(\mathbf{x}_k^{(i)})||^2 + \eta_k^{(i)}\sigma^2 \\ = & f(\mathbf{x}^*) + \mathbb{E}[\langle [g_{i+1}(\mathbf{x}_k^{(i)})], \mathbf{x}_k^{(i)} - \mathbf{x}^* \rangle - \frac{\eta_k^{(i)}}{2} ||g_{i+1}(\mathbf{x}_k^{(i)})||^2] + \eta_k^{(i)}\sigma^2. \end{aligned} \tag{24}$$

by completing the square of the middle two terms to get:

$$\begin{aligned}\mathbb{E}[f(\mathbf{x}_k^{(i+1)})] \leq & f(\mathbf{x}^*) + \mathbb{E}\left[\frac{1}{2\eta_k^{(i)}}\left(||\mathbf{x}_k^{(i)} - \mathbf{x}^*||^2 - ||\mathbf{x}_k^{(i)} - \mathbf{x}^* - \eta_k^{(i)} g_{i+1}(\mathbf{x}_k^{(i)})||^2\right)\right] + \eta_k^{(i)}\sigma^2. \\ = & f(\mathbf{x}^*) + \mathbb{E}\left[\frac{1}{2\eta_k^{(i)}}\left(||\mathbf{x}_k^{(i)} - \mathbf{x}^*||^2 - ||\mathbf{x}_k^{(i+1)} - \mathbf{x}^*||^2\right)\right] + \eta_k^{(i)}\sigma^2. \end{aligned} \tag{25}$$

For K rounds and r benign nodes, we note that the total number of SGD steps are $T = Kr$. We let $s = kr + i$ represent the number of updates happen to the initial model $\mathbf{x}^0$, where $i = 1, \dots, r$ and $k = 0, \dots, K-1$. Therefore, $\mathbf{x}_k^{(i)}$ can be written as $\mathbf{x}^s$. With the modified notation, we can now take the expectation in the above expression over the entire sampling process during training and then by summing the above equations for $s = 0 \dots, T-1$, while taking $\eta = \frac{1}{L}$, we have the following:

$$\sum_{s=0}^{T-1}\left(\mathbb{E}[f(\mathbf{x}^{s+1})] - f(\mathbf{x}^*)\right) \leq \frac{L}{2}\left(||\mathbf{x}_0 - \mathbf{x}^*||^2 - \mathbb{E}[||\mathbf{x}_k^{(T)} - \mathbf{x}^*||^2]\right) + \frac{1}{L}T\sigma^2 \tag{26}$$

By using the convexity of $f(.)$, we get

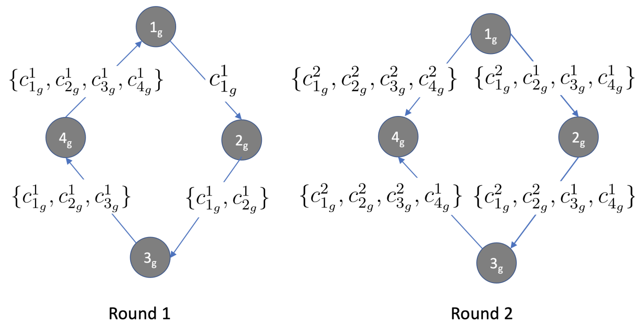
$$\begin{aligned}\mathbb{E}\left[f\left(\frac{1}{T}\sum_{s=1}^{T}\mathbf{x}^s\right)\right] - f(\mathbf{x}^*) \leq & \frac{1}{T}\sum_{s=0}^{T-1}\mathbb{E}\left[f(\mathbf{x}^{s+1})\right] - f(\mathbf{x}^*) \\ \leq & \frac{||\mathbf{x}^0 - \mathbf{x}^*||^2 L}{2T} + \frac{1}{L}\sigma^2. \end{aligned} \tag{27}$$

E Generalizing Basil to non-IID setting via Anonymous Cyclic Data Sharing

We propose Anonymous Cyclic Data Sharing scheme, a scheme that can be integrated on the top of `Basil` to guarantee robustness against software/hardware faults in the non-IID setting. This scheme allows each node to anonymously share α fraction of its local non-sensitive dataset with other nodes. In other words, ACDS guarantees that the owner of the shared data is kept hidden from all other nodes under no collusion between nodes.

ACDS starts by first clustering the N nodes into G groups of rings, where the set of nodes in each group is denoted by $\mathcal{N}_g = \{1_g, \dots, n_g\}$ where node 1_g is the starting node in ring g, and each group has the same number of nodes, $n = \frac{N}{G}$. Each node $i \in \mathcal{N}$ divides its data set Z_i into sensitive (Z_i^s) and non-sensitive (Z_i^{ns}) portions. For $\alpha \in (0, 1)$, every node chooses αD data points from its local non-sensitive data at random, and then divides these data points into H batches denoted by $\{c^{(1)}, \dots, c^{(H)}\}$, where each batch is of size $M = \frac{\alpha D}{H}$ data points.

As shown in Figure 3, within group g and during the first round $\tau = 1$, node 1_g sends the first batch $c_{1_g}^{(1)}$ to its clockwise neighbour, node 2_g, which shuffles the data points in this batch with the data points from its first batch $c_{2_g}^{(1)}$ before sending them together to node 3_g. The cyclic sharing over



1_g
2_g
3_g
4_g
$\{c^1_{1_g}, c^1_{2_g}, c^1_{3_g}, c^1_{4_g}\}$
$c^1_{1_g}$
$\{c^1_{1_g}, c^1_{2_g}, c^1_{3_g}\}$
$\{c^1_{1_g}, c^1_{2_g}\}$
Round 1
$\{c^2_{1_g}, c^2_{2_g}, c^2_{3_g}, c^2_{4_g}\}$
$\{c^2_{1_g}, c^1_{2_g}, c^1_{3_g}, c^1_{4_g}\}$
$\{c^2_{1_g}, c^2_{2_g}, c^2_{3_g}, c^1_{4_g}\}$
$\{c^2_{1_g}, c^2_{2_g}, c^1_{3_g}, c^1_{4_g}\}$
Round 2

Figure 3: Two rounds for the ACDS scheme within group g with $n = 4$ nodes.

the ring proceeds in the following general procedure in which node i_g shuffles the data points in the received set of batches $\{c^{(1)}_{1_g}, \ldots, c^{(1)}_{(i-1)_g}\}$ from its counterclockwise neighbour node $(i-1)_g$ with the data points from its own batch $c^{(1)}_{i_g}$, before sending them together to node $(i+1)_g$. After completing the first round, the master node in each group, node 1_g, shares the received set of n_g batches $\{c^{(1)}_{1_g}, \ldots, c^{(1)}_{n_g}\}$ with the other groups. This can be done by either sending these batches to the master nodes in the other groups who broadcast them within their groups, or by directly sharing them with all other nodes in the other groups.

In round $\tau > 1$ as shown in Figure 3, node 1_g after receiving the set of batches $\{c^{(\tau-1)}_{1_g}, \ldots, c^{(\tau-1)}_{n_g}\}$ from node n_g, it replaces its batch $c^{(\tau-1)}_{1_g}$ with $c^{(\tau)}_{i_g}$, which is the τ-th batch on its list, and shuffles all the set of data points in the new set of batches $\{c^{(\tau)}_{1_g}, c^{(\tau-1)}_{1_g} \ldots, c^{(\tau-1)}_{n_g}\}$ before sending them to node 2_g. More generally, in the τ-th round, node i_g replaces its batch $c^{(\tau-1)}_{i_g}$ with $c^{(\tau)}_{i_g}$, which is the τ-th batch on its list, and shuffles all the set of data points in the new set of batches $\{c^{(\tau)}_{1_g}, \ldots, c^{(\tau)}_{i_g}, c^{(\tau-1)}_{(i+1)_g}, \ldots, c^{(\tau-1)}_{n_g}\}$ before sending them to node $(i+1)_g$. The collected data in each group are shared within the other groups after each round in the same way as mentioned earlier. The total number of rounds until completing the ACDS scheme is H rounds, where H is the total number of batches to be shared by each node. Note that data sharing by using ACDS strategy only needs to be performed once when the training is initialized, i.e., this is one time cost; however, as we demonstrate later in Section H, this dramatically improves the performance.

E.1 Anonymity guarantees of ACDS

In the first round of the scheme, node 2_g will know that the source of the received batch $c^1_{1_g}$ is node 1_g. Similarly and more general, node i_g will know that each data points in the received set of batches $\{c^1_{1_g}, \ldots, c^1_{(i-1)_g}\}$ is generated by one of the previous $i-1$ counterclockwise neighbours. However, in the next $H-1$ rounds, each received data point by any node will be equally likely generated from any one of the remaining $n-1$ nodes in this group. Hence, the size of the candidate pool from which each node could take a guess for the owner of each data point is small specially for the first set of nodes in the ring. In order to provide anonymity for the entire data and decrease the risk in the first round of the ACDS scheme, the size of the batch can be reduced to just one data point. Therefore, in the first round node 2_g will only know one data point from node 1_g. This comes on the expense of increasing the number of rounds. Another key consideration is that the number of nodes in each group trades the level of anonymously with the communication cost. In particular, the communication cost per node in the ACDS scheme is $\mathcal{O}(n)$, while the anonymous level, which we measure by the number of possible candidate for a given data point, is $(n-1)$. Therefore, increasing n, i.e., decreasing the number of group G, will decrease the communication cost but increase the anonymous level.

F Joining and Leaving of Nodes

`Basil` can handle the scenario of 1)node dropouts out of the N available nodes 2) nodes rejoining the system.

F.1 Nodes dropout

For handling node dropouts, we allow for extra communication between nodes. In particular, each active node can broadcast its model to the $S{=}b{+}d{+}1$ clockwise neighbours, where b and d are respectively the number of Byzantine nodes and the worst case number of dropped nodes, and each node can store only the latest $b{+}1$ model updates. By doing that, each benign node will have at least 1 benign update even in the worst case where all Byzantine nodes appear in a row and d (out of S) counterclockwise nodes drop out.

F.2 Nodes rejoining

To address a node *rejoining* the system, this rejoined node can re-broadcast its ID to all other nodes. Since benign nodes know the correct order of the nodes (IDs) in the ring according to Section B, each active node out of the $L{=}b{+}d{+}1$ counterclockwise neighbours of the rejoined node sends its model to it, and this rejoined node stores the latest $b{+}1$ models. We note that handling participation of new fresh nodes during training is out of scope of our paper, as we consider mitigating Byzantines in decentralized training with a *fixed* number of N nodes

G Mozi

We consider Mozi [3] a recent Byzantine resilient approach for parallel decentralized training. This decentralized training setup is defined over undirected graph: $\mathcal{G} = (V, E)$, where V denotes a set of N nodes and E denotes a set of edges representing communication links. Filtering Byzantine nodes is done over two stages for each training iteration. At the first stage, each benign node performs a distance-based strategy to select a candidate pool of potential benign nodes from its neighbors. This selection is performed by comparing the Euclidean distance of its own model with the model from its neighbours. In the second stage, each benign node performs a performance-based strategy to pick the final nodes from the candidate pool resulted from stage 1. It reuses the training sample as the validation data to compute loss function value of each model. It selects the models whose loss values are smaller than the value of its own model, and calculates the average of those models as the final updated value. Formally, the update rule in Mozi is given by

$$\mathbf{x}_{k+1}^{(i)} = \alpha \mathbf{x}_k^{(i)} + (1-\alpha)\mathcal{R}_{\text{Mozi}}(\mathbf{x}_k^{(j)}, j \in \mathcal{N}_i) - \eta \nabla f_i(\mathbf{x}_k^{(i)}), \tag{28}$$

where $\mathcal{N}_i$ is the set of neighbours of Node i, $\nabla f_i(\mathbf{x}_k^{(i)})$ is the local gradient of node i evaluated on a random sample from the local dataset of node i while using its own model, k is the training round, and $\mathcal{R}_{\text{Mozi}}$ is given as follows:

$$\mathcal{R}_{\text{Mozi}} = \begin{cases} \frac{1}{\mathcal{N}_{i,k}^r} \sum_{j \in \mathcal{N}_{i,k}^r} \mathbf{x}_k^{(j)} & \text{if } \mathbf{x}_{k+1}^{(i)} \neq \phi \\ \mathbf{x}_k^{j^*}, & \text{Otherwise} \end{cases} \tag{29}$$

where there are two stages of filtering:

$$\text{stage (1)}\ \ \mathcal{N}_{i,k}^s = \arg \min_{\substack{\mathcal{N}^* \subset \mathcal{N}_i, \\ \mathcal{N}^* = \rho_i |\mathcal{N}_i|}} \sum_{j \in \mathcal{N}_i} ||\mathbf{x}_k^{(j)} - \mathbf{x}_k^{(i)}||,$$

$$\text{stage (2)}\ \ \mathcal{N}_{i,k}^r = \bigcup_{\substack{j \in \mathcal{N}_{i,k}^s \\ \ell_i(\mathbf{x}_k^{(j)}) \leq \ell_i(\mathbf{x}_k^{(i)})}} j, \text{ and } j^* = \arg \min_{j \in \mathcal{N}_{i,k}^s} \ell_i(\mathbf{x}_k^{(i)}).$$

H Numerical Experiments

H.1 Setting

Schemes We consider four schemes as described next. *G-plain*: This is for graph based topology. At the start of each round, nodes exchange their models with their neighbors. Each node then finds the average of its model with the received neighboring models and uses it to carry out an SGD step over its local dataset. *R-plain*: This is for ring based topology with $S = 1$. The current active node carry out an SGD step over its local data set by using the model received from its previous counterclockwise neighbour. *Mozi*: This is the prior state-of-the-art for mitigating Byzantines in decentralized training over graph, and is described in Section G. *Basil*: This is our proposal.

Datasets and Hyperparameters There are a total of 100 nodes, in which 67 are benign. For decentralized network setting for simulating Mozi and G-plain schemes, we follow a similar approach as described in the experiments in [3]. Specifically, we first assign connections randomly among benign nodes with a probability of 0.4, and then randomly assign connections from the benign nodes to the Byzantine nodes, with a probability of 0.4. Furthermore, we set the Byzantine ratio for benign nodes as $\rho = 0.33$.

For Basil and R-plain, the nodes are arranged in a logical ring, and 33 of them are randomly set as Byzantines. Furthermore, we set $S = 10$ for Basil which gives us the guarantees discussed in Proposition 1. We use a decreasing learning rate of $0.03/(1 + 0.03\,k)$. We consider CIFAR10 [19] and use a neural network with 2 convolutional layers and 3 fully connected layers. The details are included in the following paragrpah (Model). The training dataset is partitioned equally among all nodes. Furthermore, we report the worst test accuracy among the benign clients in our results. We also conduct similar evaluations on a simpler MNIST dataset [20]. The experimental results lead to the same conclusion.

Models We provide the details of the neural network architectures used in our experiments. For MNIST, we use a model with three fully connected layers, and the details for the same are provided in Table 1. Each of the first two fully connected layers is followed by ReLU, while softmax is used at the output of the third one fully connected layer.

Table 1: Details of the parameters in the architecture of the neural network used in our MNIST experiments.

Parameter	**Shape**
fc1	784×100
fc2	100×100
fc3	100×10

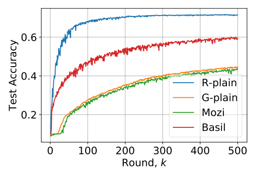
For CIFAR10 experiments in the main paper, we consider a neural network with two convolutional layers, and three fully connected layers, and the specific details of these layers are provided in Table 2. ReLU and maxpool is applied on the convolutional layers. The first maxpool has a kernel size 3×3 and a stride of 3 and the second maxpool has a kernel size of 4×4 and a stride of 4. Each of the first two fully connected layers is followed by ReLU, while softmax is used at the output of the third one fully connected layer.

We initialize all biases to 0. Furthermore, for weights in convolutional layers, we use Glorot uniform initializer, while for weights in fully connected layers, we use the default Pytorch initialization.

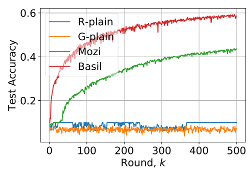
Table 2: Details of the parameters in the architecture of the neural network used in our CIFAR10 experiments.

Parameter	**Shape**
conv1	$3 \times 16 \times 3 \times 3$
conv2	$16 \times 64 \times 4 \times 4$
fc1	64×384
fc2	384×192
fc3	192×10

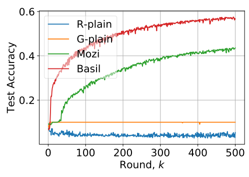
Byzantine Attacks We consider a variety of attacks, that are described as follows. *Gaussian Attack*: Each Byzantine node replaces its model parameters with entries drawn from a Gaussian distribution with mean 0 and standard distribution $\sigma = 1$. *Random Sign Flip*: We observed in our experiments that the naive sign flip attack, in which Byzantine nodes flip the sign of each parameter before exchanging their models with their neighbors, is not strong in the R-plain scheme. To make the sign-flip attack stronger, we propose a layer-wise sign flip, in which Byzantine nodes randomly choose to flip or keep the sign of the entire elements in each neural network layer. *Hidden Attack*: This is a sophisticated attack that degrades the performance of distance-based defense approaches, as proposed in [16] and described in detail in Appendix. Essentially, the Byzantine nodes are assumed to be omniscient, i.e. they can collect the models uploaded by all the benign clients. Byzantine nodes then design their models such that they are undetectable from the benign ones in terms of the distance metric, while still degrading the training process. For hidden attack, as the key idea is to exploit similarity of models from benign nodes, thus to make it is more effective, the Byzantine nodes launch this attack after 20 rounds of training.



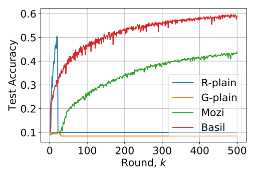
(a) No Attack



(b) Gaussian Attack

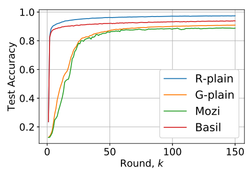


(c) Random Sign Flip

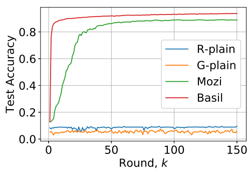


(d) Hidden Attack

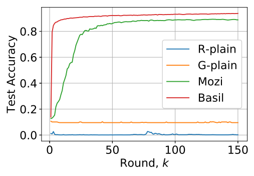
Figure 4: Illustrating the results for CIFAR10 under IID data distribution setting.



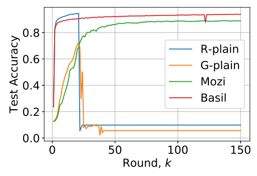
(a) No Attack



(b) Gaussian Attack



(c) Random Sign Flip Attack



(d) Hidden Attack

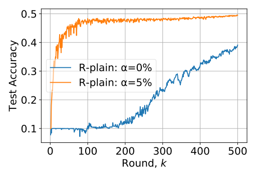
Figure 5: Illustrating the results for MNIST under IID data distribution setting.

H.2 Results

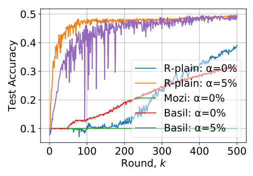
IID Setting We first present the results for the IID data setting. Training dataset is first shuffled randomly and then partitioned among the nodes. As can be seen from Figure 4(a), `Basil` converges much faster than both Mozi and G-plain even in the absence of any Byzantine attacks, illustrating the benefits of ring topology based learning over graph based topology. We note that the total number of gradient updates after k rounds in the two setups are almost the same. We can also see that R-plain gives higher performance than `Basil`. This is because in `Basil`, a small mini-batch is used for performance evaluation, hence in contrast to R-plain, the latest neighborhood model may not be chosen in each round resulting in the loss of some update steps. Nevertheless, Figures 4(b), 4(c) and 4(d) illustrate that `Basil` is not only Byzantine-resilient, it maintains its superior performance over Mozi with $\sim 16\%$ improvement in test accuracy, as opposed to R-plain that suffers significantly. Furthermore, we would like to highlight that as `Basil` uses a performance-based criterion for mitigating Byzantine nodes, it is robust to the Hidden attack as well. Finally, by considering the poor convergence of R-plain under different Byzantine attacks, we conclude that `Basil` is a good solution with the fast convergence, strong Byzantine resiliency and acceptable computation overhead.

Figure As can be seen from Figure 5 that using `Basil` leads to the same conclusion shown in CIFAR10 dataset in terms of its fast convergence, high test accuracy, and Byzantine robustness compared to the different schemes.

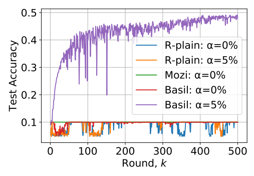
Non-IID setting For simulating the non-IID setting, we sort the training data as per class, partition the sorted data into N subsets, and assign each node 1 partition. By applying ACDS in the absence of Byzantine nodes while trying different values for α, we found that $\alpha = 5\%$ gives a good performance while giving a reasonable small amount of shared data from each node. Figure 6(a) illustrates that test accuracy for R-plain in the non-IID setting can be increased by up to $\sim 10\%$ that when each node shares only $\alpha = 5\%$ of its local data with other nodes. Figure 6(c), and Figure 6(d) illustrate that `Basil` on the top of ACDS with $\alpha = 5\%$ is robust to software/hardware faults represented in Gaussian model and random sign flip. Furthermore, both `Basil` without ACDS and Mozi completely fail in the presence of these faults. This is because the two defenses are using performance criteria



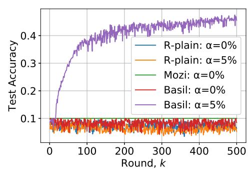
(a) No Attack



(b) No Attack

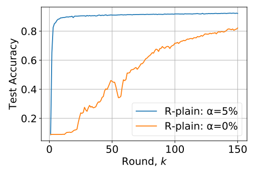


(c) Gaussian Attack

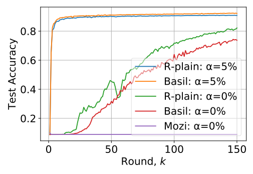


(d) Random Sign Flip

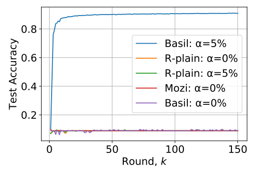
Figure 6: Illustrating the results for CIFAR10 under non-IID data distribution setting.



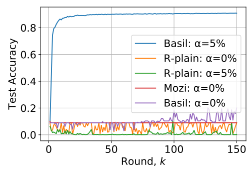
(a) No Attack



(b) No Attack



(c) Gaussian Attack



(d) Random Sign Flip Attack

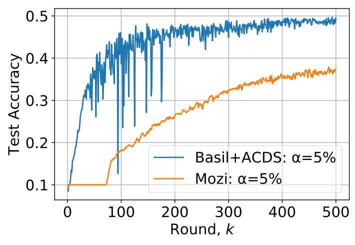
Figure 7: Illustrating the results for MNIST under non-IID data distribution setting.

which is not meaningful in the non-IID setting. In other words, each node has only data from one class, hence it becomes unclear whether a high loss value for a given model can be attributed to Byzantine node, or to the very heterogeneous nature of the data. Additionally, both R-plain with $\alpha = 0$ and $\alpha = 5$ completely fail in the presence of Byzantine nodes.

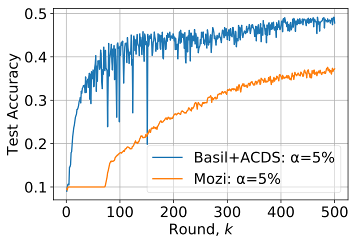
Finally, we can observe in 6(b) that `Basil` with $\alpha = 0$ gives low performance. This confirms that non-IID data distribution degraded the convergence behaviour. For Mozi, the performance is completely degraded, since the way in which Mozi is implemented, each node selects the set of models which gives a lower loss than its own local model, before using them in the update rule. Since performance-based is not meaningful in this setting, each node might end up only with its own model. Hence,the model of each node does not completely propagate over the graph, as also demonstrated in Figure 6(b), where Mozi fails completely. This is different from the ring setting where the model is propagated over the ring.

Similarly, Figure 7 that using `Basil` leads to the same conclusion shown in CIFAR10 dataset in terms of its fast convergence, high test accuracy, and Byzantine robustness compared to the different schemes.

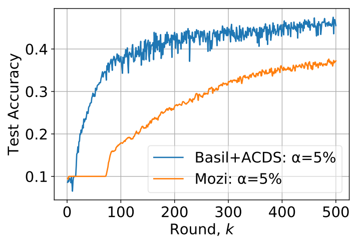
In Figure 6 and Figure 7 in this section, we showed that Mozi performs quite poorly for non-IID data setting, when no data is shared among the clients. We note that achieving anonymity in data sharing in Mozi is an open problem. Nevertheless, in Figure 8 and Figure 9, we further show that even 5% data sharing is done in Mozi, performance remains quite low in comparison to Basil+ACDS.



(a) No Attack

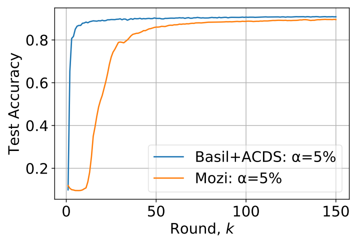


(b) Gaussian Attack

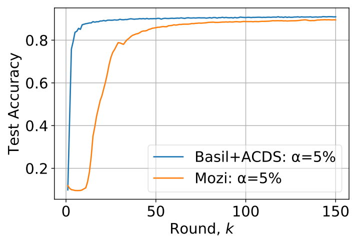


(c) Random Sign Flip Attack

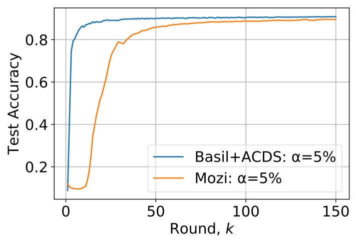
Figure 8: Illustrating the performance of Basil compared with Mozi for CIFAR10 under non-IID data distribution setting with $\alpha = 5\%$ data sharing.



(a) No Attack



(b) Gaussian Attack



(c) Random Sign Flip Attack

Figure 9: Illustrating the performance of Basil compared with Mozi for MNIST under non-IID data distribution setting with $\alpha = 5\%$ data sharing.