Gradient Disaggregation: Breaking Privacy in Federated Learning by Reconstructing the User Participant Matrix

Maximilian Lam [1] **Gu-Yeon Wei** [1] **David Brooks** [1] **Vijay Janapa Reddi** [1] **Michael Mitzenmacher** [1]

Abstract

We show that aggregated model updates in federated learning may be insecure. An untrusted central server may disaggregate user updates from sums of updates across participants given repeated observations, enabling the server to recover privileged information about individual users' private training data via traditional gradient inference attacks. Our method revolves around reconstructing participant information (e.g: which rounds of training users participated in) from aggregated model updates by leveraging summary information from device analytics commonly used to monitor, debug, and manage federated learning systems. Our attack is parallelizable and we successfully disaggregate user updates on settings with up to thousands of participants. We quantitatively and qualitatively demonstrate significant improvements in the capability of various inference attacks on the disaggregated updates. Our attack enables the attribution of learned properties to individual users, violating anonymity, and shows that a determined central server may undermine the secure aggregation protocol to break individual users' data privacy in federated learning.

1. Introduction

Federated learning is a method for collaboratively learning a shared model across multiple participants and enhances privacy by limiting data sharing (McMahan & Ramage, 2017; Hard et al., 2018; Bonawitz et al., 2019; Konečný et al., 2017; 2015). Participants' data privacy is preserved by sending model updates rather than raw data, which limits the amount of information that is exposed to the central server. In the context of applications, federated learning participants are edge devices such as users' smart phones or wearables, and maintaining the integrity of their data is a critical issue. Already, federated learning has been deployed by many major companies in various privacy sensitive applications including sentiment learning, next word prediction, health monitoring, content suggestion, and item ranking (Hard et al., 2018; Li et al., 2020; Bonawitz et al., 2019). Guaranteeing data privacy in these scenarios is becoming increasingly important as the topic of privacy becomes more heavily scrutinized by the greater public and by government regulations (McMahan & Ramage, 2017; McCabe, 2013; FTC, 2019).

Recent research has shown that model updates may unintentionally leak information about their respective training examples (Geiping et al., 2020; Melis et al., 2019; Zhu et al., 2019). A central server that obtains participants' model updates may perform inference attacks to learn significant information about participants' training data, violating the core privacy principles of the federated learning paradigm. To address this critical privacy flaw, researchers have introduced methods leveraging secure multiparty computation to limit the central server's visibility into individual participants' model updates. Notably, secure aggregation (Segal et al., 2017; So et al., 2020) has emerged as a standard security protocol which ensures that the central server may see only the final sum of model updates, rather than any individual update by itself. Thus, information learned from the aggregated model update may not be attributed to a specific user, which offers a layer of privacy against the central server. Additionally, by aggregating updates over tens to hundreds or thousands of users, updates are obfuscated to a point where most inference attacks are rendered ineffectual (Melis et al., 2019; Geiping et al., 2020; Zhu et al., 2019).

The secure aggregation protocol is secure only to the degree that it hides individual participants' model updates. A procedure that disaggregates individual participants' updates or gradients from their sum would undermine the secure aggregation protocol and unveil the aforementioned privacy vulnerability. In this work, we develop a method for gradient disaggregation, showing that secure aggregation offers little privacy protection against an adversarial server seeking to undermine individual users' data privacy. Our key insight is that participant information (e.g: which rounds of training

[1]Harvard University, Cambridge, MA. Correspondence to: Maximilian Lam <maxlam@g.harvard.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

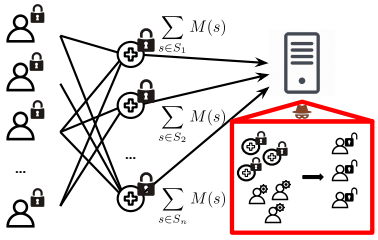


Figure 1. Our gradient disaggregation attack observes multiple rounds of aggregated model updates and leverages side channel information in the form of summary analytics collected by federated learning systems (how often users participated across certain training rounds) to uncover individual users' private model updates, undermining the secure aggregation protocol. Code: https://github.com/gdisag/gradient_disaggregation.

users participated in) is derivable from aggregated model updates, when observing multiple rounds of training and leveraging summary analytics. We can reconstruct this information and use it to recover participants' individual model updates (see Figure 1). Our contributions are as follows:

- We introduce and formulate the gradient disaggregation problem as a constrained binary matrix factorization problem. Leveraging summary analytics collected by federated learning systems, we demonstrate that our disaggregation attack can exactly recover the user participant matrix on up to thousands of participants, revealing the model update of each user. Additionally, we show that gradient disaggregation works even in the presence of significant noise and allows us to disaggregate aggregated model updates that were generated by federated averaging.
- We leverage gradient disaggregation to significantly improve the quality of traditional inference attacks on model updates. We show that without gradient disaggregation, inference attacks often fail to recover meaningful information on updates aggregated across tens to hundreds of users; with gradient disaggregation, we show successful recovery of users' privileged data from their disaggregated model updates.

2. Related Work

2.1. Secure Aggregation

Secure aggregation is a method based on secure multiparty computation and is a key privacy measure deployed in federated learning systems. Secure aggregation ensures that the central server sees only the final aggregate of model updates across users while guaranteeing that no participants' updates are revealed in the clear (Segal et al., 2017). The secure aggregation protocol enhances privacy by obfuscating a user's model update with many other users' updates, limiting inference attacks such as those in (Melis et al., 2019; Geiping et al., 2020; Zhu et al., 2019). This obfuscation also ensures that information learned from the aggregated model update may not be attributed to an individual user. Concretely, on the issue of attribution, the secure aggregation paper states: "Using a Secure Aggregation protocol to compute these weighted averages would ensure that the server may learn only that one or more users in this randomly selected subset wrote a given word, but not which users" (Segal et al., 2017).

Our work on gradient disaggregation undermines the secure aggregation protocol by showing that, through observing multiple rounds of collected data and leveraging side channel information (specifically, user participation frequency as collected by federated learning systems), individual updates may be reconstructed from their overall sums. While secure aggregation has been proven to be cryptographically secure, leaking no information which is not leaked by the aggregated model update itself (Segal et al., 2017), the key insight of our attack is that participant information (e.g: which rounds each user participated in) is derivable from the aggregated model updates and reconstructing it allows us to in turn recover individual model updates.

2.2. Analytics in Federated Learning Systems

Infrastructure to support, debug, and manage federated learning systems is critical to their functioning. (Bonawitz et al., 2019) outlines the design of Google's federated learning systems and describes its core components and protocols. A key aspect of their infrastructure is the collection of device analytics. Notably, (Bonawitz et al., 2019) collect several important device metrics such as how often devices performed training, how much memory devices used during training, etc (Bonawitz et al., 2019). These metrics ensure that users' devices are not oversubscribed (draining battery) and may be used to debug device performance issues. Device analytics play a critical role in maintaining user experience quality: "Device utility to the user is mission critical, and degradations are difficult to pinpoint and easy to wrongly diagnose. Using accurate analytics to prevent federated training from negatively impacting the device's utility to the user accounts for a substantial part of our engineering and risk mitigation costs." (Bonawitz et al., 2019)

In our work, we leverage summary information from device analytics – specifically how often a user performed training – to assist disaggregating gradients, breaking privacy. Note while (Bonawitz et al., 2019) points out that device analytics contain no personally identifiable information, these reports nevertheless provide crucial information that links gradient information collected across rounds, facilitating our attack on disaggregating gradients.

2.3. Product Identification by Solving Linear Inverse Problem

Recently, independent of our work, research has shown that individual item product prices may be recovered given customer's transaction history by optimizing a linear inverse problem (Fleder & Shah, 2020). Under certain conditions (e.g: assuming that in the transaction history each item was purchased by itself at least once) their approach recovers these item prices with high precision and allows them to reveal customers' spending habits. Specifically, given a corpus of sums of item prices from customers' transaction histories, (Fleder & Shah, 2020) utilizes a subset sum algorithm to uncover the individual prices of the transaction and to identify the products themselves.

Our work on gradient disaggregation and the work in (Fleder & Shah, 2020) solve the same core problem: uncovering individual values given observations of their sums. While their work recovers prices of items, our work analogically reconstructs participants' model gradients. However, a key distinction in (Fleder & Shah, 2020) is the assumption that each item must be purchased individually at least once. This makes their approach unsuitable for disaggregating aggregated model updates as, under the secure aggregation protocol, each aggregated update is composed of more than one participant's model updates.

2.4. Data Leakage from Model Updates

Recent research has shown that model updates and gradients leak significant amounts of information. Information leaked by model updates ranges from specific properties to entire data samples (Melis et al., 2019; Zhu et al., 2019; Geiping et al., 2020; Shokri et al., 2017; Qian & Hansen, 2020; Wei et al., 2020; Lyu et al., 2020; Athalye et al., 2018; Mengkai et al., 2020; Hitaj et al., 2017). Methods to recover this information from gradients are broadly categorized as inference attacks, and prior works have demonstrated the effectiveness of inference attacks on small batches of gradients, across various modalities ranging from image to text (Shokri et al., 2017), on both shallow and deep networks (Geiping et al., 2020).

In the context of federated learning, these methods suffer decreased efficacy with larger aggregates (> 100) (Melis et al., 2019; Zhu et al., 2019; Geiping et al., 2020). Our work on gradient disaggregation facilitates these attacks by de-obfuscating these updates and by enabling attribution of learned properties to specific users.

2.5. Privacy Attacks in Federated Learning

Recent works have introduced various privacy attacks on federated learning. Broadly, these attacks are performed by a malicious central server or by participants with influence over model training (Lyu et al., 2020). Threats from an adversarial central server typically involve extracting private information via inference attacks as described in the previous section. Attacks by adversarial participants, on the other hand, involve influencing the model training process to alter the behavior of the trained model (e.g: model poisoning, backdoors)(Wang et al., 2020; Bagdasaryan et al., 2019; Fung et al., 2020; Blanchard et al., 2017).

Our work on gradient disaggregation falls under the category of an attack performed by a malicious central server. Specifically, gradient disaggregation breaks the secure aggregation protocol and enables a central server to perform inference attacks on individual participants' model updates.

3. Gradient Disaggregation

3.1. Problem Statement, Threat Model and Assumptions

Gradient disaggregation involves uncovering individual participants' model updates given observations of their sums. Concretely, on round r the central server receives

$$G_{aggregated_{[r,:]}} = \sum_{s \in S_r} M(s)$$

where S_r is the selected participants on round r, and $M(s)$ are the model updates. The goal of gradient disaggregation is, acting as an adversarial central server, to recover $M(s)$ given $G_{aggregated}$ (aggregated gradients across n rounds).

Our threat model and assumptions are as follows:

- The central server is adversarial but is limited in its ability to modify the training protocol. Specifically, we assume the central server may fix its model across rounds. Such a scenario is realistic in a case where an attacker has read access to corporation servers (e.g: to collect round model update data) and limited influence over when the global model is updated (e.g: to fix the model across rounds). An adversarial central server is a major threat model in federated learning (Lyu et al., 2020; Li et al., 2020; Kairouz et al., 2019)
- Client selection / device participation (S_r) is somewhat random and is a subset of the total number of users. This matches the federated learning protocol which selects a random fraction of devices to participate in each round of training (Bonawitz et al., 2019; McMahan & Ramage, 2017; Li et al., 2020).
- The central server has access to side channel information in the form of summary analytics (specifically, how often users participated across certain federated learning rounds). Device and summary analytics are a

core part of federated learning systems and infrastructure (Bonawitz et al., 2019).

3.2. Gradient Disaggregation by Reconstructing the User Participant Matrix

A central server that observes aggregates of users' updates that are constant across rounds obtains

$$G_{aggregated} = PG_{individual} \tag{1}$$

where $G_{aggregated} \in \mathbb{R}^{n \times d}$ are the final aggregated dimension d gradients the server collected across n rounds; $P \in \{0,1\}^{n \times u}$ is the user participant matrix across the n rounds with u total participants specifying which users participated in which rounds; and $G_{individual} \in \mathbb{R}^{u \times d}$ contains per user individual gradients. Hence, recovering $G_{individual}$ may be viewed as a matrix factorization problem where the left term is binary.

To approach this matrix factorization problem, we start with the method introduced in (Slawski et al., 2013), which, to the best of our knowledge, is one of the only works to address matrix factorization where the left term is binary. (Slawski et al., 2013) first reconstructs the binary user participant matrix P, then recovers $G_{individual}$ by inverting P from $G_{aggregated}$. As observed by (Slawski et al., 2013), columns of P lie in the image of $G_{aggregated}$. Hence, with $Nul(M)$ as the kernel of a matrix, an approach to solving this factorization problem would be to recover each column p_k of P:

$$\begin{aligned} &\text{Find } p_k \text{ s.t } Nul(G_{aggregated}^T)p_k = 0 \\ &p_k \in \{0,1\}^n \end{aligned} \tag{2}$$

In the context of federated learning, this attempts to recover individually for each user which rounds they participated in. Note that such an optimization procedure can be solved using standard mixed-integer programming frameworks such as (Gurobi Optimization, 2020) and can additionally be parallelized across each user.

However, this approach is not sufficient for gradient disaggregation due to three issues: 1) failure to distinguish between the numerous binary vectors in the image of $G_{aggregated}$, 2) inability to distinguish between user solutions and 3) computational difficulties due to the exponential nature of the optimization problem (recovering p_k is NP hard and (Slawski et al., 2013) reports only being able to solve up to $n = 30$ vectors). To address these issues, we incorporate summary analytics to assist factorization.

3.3. Leveraging Summary Analytics to Reconstruct P

We leverage summary information from device analytics as collected in (Bonawitz et al., 2019) to assist reconstructing P. Specifically, summary analytics that are collected periodically by the central server log how often a specific user participated in training and can be used to narrow down p_k by limiting the total number of participations across certain training rounds (see our Related Works section for details). We capture partial information on participations across rounds by introducing linear constraints: the i'th constraint $C_k^i \in \{0,1\}^n$ specifies for the k'th participant the training rounds for which total number of participations c_k^i is known. For example, knowing that a user participated in training 3 times between rounds 1-5 and 2 times between rounds 6-10 yields $C_1^1 = [1,1,1,1,1,0,0,0,0,0]$, $c_1^1 = 3$, $C_1^2 = [0,0,0,0,0,1,1,1,1,1]$, $c_1^2 = 2$.

We therefore add the individual constraints.

$$C_k^i p_k - c_k^i = 0 \tag{3}$$

After collecting all j constraints and counts across all users, we combine them into

$$C_k = \begin{bmatrix} — & C_k^1 & — \\ & \vdots & \\ — & C_k^j & — \end{bmatrix}, c_k = \begin{bmatrix} c_k^1 \\ \vdots \\ c_k^j \end{bmatrix} \tag{4}$$

Incorporating them into the optimization, we obtain

$$\begin{aligned} &\text{Find } p_k \text{ s.t. } Nul(G_{aggregated}^T)p_k = 0 \\ &p_k \in \{0,1\}^n \\ &C_k p_k - c_k = 0 \end{aligned} \tag{5}$$

We note that it is possible that devices timestamp the exact moment they perform a round of training; in this case, P may be revealed directly through the specificity of the constraints (making the disaggregation problem solvable through a simple linear regression). However, even if devices log only the total number of times they performed training (with no timestamped data) and send these analytics back to the server once every few rounds of participation, the central server may piece together these constraints and incorporate them into the formulation above. In other words, just knowing the number of times particular users performed training and collecting this information periodically (both of which are reasonable based on (Bonawitz et al., 2019)), the central server may obtain enough information to carry out the gradient disaggregation attack. Incorporating summary analytics into the gradient disaggregation attack is significant as it greatly reduces the problem space, allowing a solution to a previously intractable problem.

3.4. Disaggregating Noisy Model Updates

Previously, we assumed users submitted the same model update across every round. However, participants may perform updates composed of multiple steps (e.g: FedAvg) or

their data may change, leading to differences in the updates they submit across rounds. We treat these differences as a form of injected noise.

Accounting for noise, our formulation becomes

$$G_{aggregated} = PG_{individual\_avg} + noise \tag{6}$$

and our goal is to recover for each user the average model update they submitted across rounds $G_{individual\_avg}$. We introduce two changes to reconstruct P in the presence of noise: 1) we use hard-threshold SVD with u singular values to approximate the low rank product $PG_{individual\_avg}$ and 2) we relax our constraint satisfaction problem to minimize the distance of the user participant column to the image of $G_{aggregated}$:

$$\begin{aligned} \min \; & ||Nul(G^T_{aggregated})p_k||^2 \\ & p_k \in \{0,1\}^n \\ & C_k p_k - c_k = 0 \end{aligned} \tag{7}$$

These two changes allow reconstructing P even when the updates user submit across rounds are noisy.

Note that we may have incomplete information for each user; for example, we may have constraints for a user over certain rounds but not others if the infrastructure only provides that information sporadically (or hides it). Additionally, if round participations are inexact (e.g: off by some small error), we may relax the hard constraint $C_k p_k - c_k = 0$ to be a soft constraint: $min||C_k p_k - c_k||^2$ and reweight the objective accordingly. Additionally, we can check whether our solution exactly recovers P by probing the number of optimal solutions returned by the mixed integer programming solver; if the solver returned only one optimal solution (and proved that it is the only one), then this indicates that our reconstruction of P is exact. Our full gradient disaggregation attack which works both for noisy and non-noisy updates is presented in Algorithm 1.

Algorithm 1 Gradient Disaggregation

Input: Aggregated gradients $G_{aggregated}$; constraint windows C; constraint sums c, number of users u
Output: Disaggregated gradients $G_{individual\_avg}$

$U, \Sigma, V \longleftarrow SVD(G_{aggregated})$
$G_{denoised} \longleftarrow U\Sigma[0:u]V$
for $i = 1$ **to** u **do**
  $p_i \longleftarrow \min ||Nul(G^T_{denoised})p_i||^2$ s.t. $p_i \in \{0,1\}^n$ and $C_i p_i - c_i = 0$
end for

$P \longleftarrow [p_1, ..., p_u]$
return $LeastSquares(P, G_{aggregated})$

4. Results

4.1. Capabilities and Limitations of Disaggregation

We experimentally validate the capabilities and limitations of our gradient disaggregation procedure across various parameter settings. Note that unlike prior works performing server side attacks on privacy in federated learning, our method leverages participant information and rounds of aggregated gradients. Hence in our experiments we generate this information (across various settings) to understand how our attack behaves under different conditions. We evaluate the following parameters:

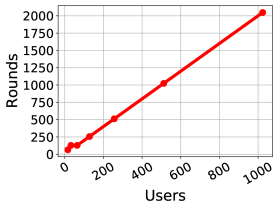
- **Number of Rounds:** Number of rounds of training n
- **Number of Users:** Number of users in system u.
- **Participation Rate:** Fraction of participants chosen to participate in each round.
- **Constraint Granularity:** Granularity of windows across rounds with known participation sums, per user. (E.g: granularity of 10 means we know how many times each user participated across every 10 rounds).
- **Gradient Noise:** Noise of user model updates across rounds.

We run all experiments on a 64-core cpu and use the Gurobi optimizer (Gurobi Optimization, 2020).

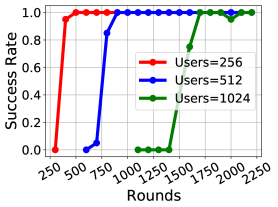
Number of Users

We validate the maximum number of users and rounds we can disaggregate on synthetically generated matrices. $G_{individual}$ is sampled from $\mathcal{N}(0,1)$, P is sampled with sparsity = participation rate = .1, and constraint granularity=10, with no noise between submitted gradients. For users $\in$ {16, 32, 64, 128, 256, 512, 1024} we scan over rounds $\in$ {16, 32, 64, 128, 256, 512, 1024, 2048} and report the minimum number of rounds to successfully disaggregate P with 100% accuracy over 30 trials. Figure 2(a) shows the number of rounds required to exactly recover P across number of users; data shows that we can disaggregate matrices with thousands of user participants with enough observed gradients. Additionally, we plot success rate of reconstructing an individual column for users $\in$ {256, 512, 1024} which is shown in Figure 2(b) which furthermore reinforces that more rounds of observed gradients can increase reconstruction success rate. We also evaluate the relation that rounds vs users has on the runtime of the solver, shown in Figure 3, where we measure the runtime to exactly recover columns of P (with a maximum time limit of 180 seconds per column). Results show that larger P require more time to solve. Additionally fewer rounds leads to slower reconstruction as there are fewer constraints, while too many rounds leads to slower optimization due to large matrix sizes. Note we report time per column, as each

column is solved in parallel.



(a) Rounds required to reconstruct P with 100% accuracy vs number of users.



(b) Success rate of recovering columns of P versus rounds.

Figure 2. Relationship between rounds vs number of users in gradient disaggregation. We successfully disaggregate settings with thousands of users with enough observed aggregated updates.

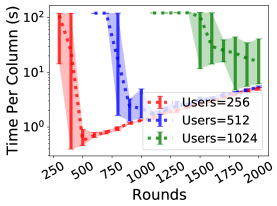


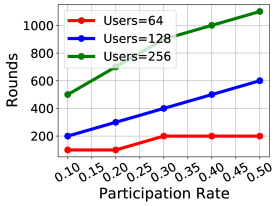
Figure 3. Mean, min, and max times to reconstruct columns of P vs rounds. More users require more time to recover P; too few rounds slows optimization due to lack of constraints; too many rounds slows optimization due to large vector sizes. We disaggregate thousands of users' gradients in minutes on a 64-core cpu.

Participation Rate

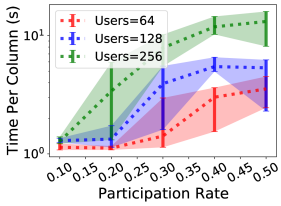
We evaluate the effect of participation rate – the probability that a user is selected to take part in a round of training – on gradient disaggregation. We use the same parameter settings as in the previous section and scan participation rate $\in \{.10, .20, .30, .40, .50\}$ across various numbers of user participants, measuring number of rounds of observed aggregated gradients required to successfully reconstruct P with 100% accuracy across 30 trials. As shown in Figure 4(a), higher participation rate requires more rounds to reconstruct P. Intuitively, more participants per round leads to higher obfuscation of user updates, requiring more rounds to decode. However, as indicated, by observing more rounds of collected gradients, P is eventually reconstructed exactly. We additionally evaluate participation rate's effect on runtime which is shown in 4(b). Higher participation rate makes the reconstruction problem more difficult and hence requires longer to solve. Note that federated learning settings have between tens to hundreds of round participants (Li et al., 2020; Bonawitz et al., 2019) and we have chosen these points to reflect this as accurately as possible.

Constraint Granularity

We evaluate the effect of constraint granularity on gradient disaggregation. We consider granularities $\in \{10, 20, 30, 40, 50\}$. Figure 5(a) shows that coarser constraints make reconstruction more difficult, requiring more rounds of observed aggregated gradients. Additionally, for reference Figure 5(b) shows the histogram of the number of times a user participates within a granularity window at different constraint granularities. Eventually, with enough observed rounds of aggregated gradients, the participant matrix P is exactly recoverable. Our results indicate that less detailed analytics may be compensated for by observing more rounds of aggregated model updates.

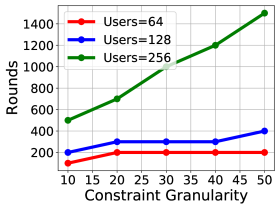


(a) Rounds required to reconstruct P with 100% accuracy vs participation rate.

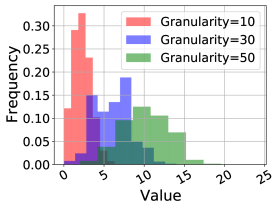


(b) Mean, min, and max runtime to reconstruct columns of P vs participation rate.

Figure 4. Effect of participation rate in gradient disaggregation. Higher participation rate can be compensated for by observing more rounds of aggregated gradients. We recover P even in the presence of many round participants.



(a) Rounds required to reconstruct P with 100% accuracy vs constraint granularity.



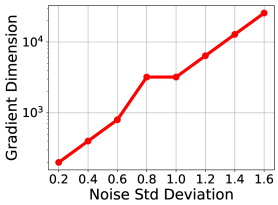
(b) Histogram of number of user participations across constraint windows.

Figure 5. Effect of constraint granularity in gradient disaggregation. Coarser constraints can be compensated by observing more rounds of aggregated gradients.

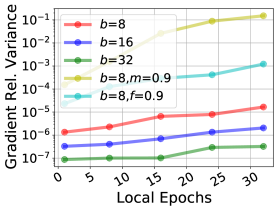
Noisy Model Updates / FedAvg

We address the scenario where model updates submitted by users are noisy across rounds, which may be due to the stochasticity of the optimization (e.g: the FedAvg algorithm). Initial experiments synthetically generate user ground truth gradients and inject noise into them at aggregation time. We initialize user vectors sampled from $\mathcal{N}(0, 1)$ then inject noise sampled from $\mathcal{N}(0, \sigma)$ to each user's vector at aggregation time, measuring the gradient dimension required to exactly reconstruct P. We perform the experiment with 100 users, a participation rate of .1 and constraint granularity of 10, with a 600 second time limit on reconstructing each column of P.

Figure 6(a) shows the minimum gradient dimension (d) that is required to exactly reconstruct P with 100% success rate. Note that unlike prior experiments, increased noise may be compensated for by incorporating a higher number of the parameters of the model update (rather than observing more rounds of gradients). As even the smallest neural network models contain thousands or millions of parameters (Han et al., 2016; 2015; Howard et al., 2017), this indicates that the attack may handle significant levels of noise. Furthermore, note that the dimension of the model update does not significantly affect solver time as the nullspace of $G_{aggregated}$ is computed only once and reused across users.



(a) Number of parameters of the gradient dimension required to recover P exactly.



(b) Relative noise of FedAvg updates on Cifar10 LeNet; batch size b, momentum m, dataset fraction f.

Figure 6. Effect of noise on gradient disaggregation.

Dataset Size D	Batch Size b	Local Epochs e						
		1	2	4	8	16	32	64
64	8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	16	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
128	8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	16	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
64 (momentum=.9)	8	.99	1.0	1.0	1.0	1.0	1.0	1.0
	16	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
128 (momentum=.9)	8	1.0	1.0	1.0	1.0	1.0	1.0	.96
	16	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
64 (fraction=.9)	8	.06	.66	.97	.99	.98	.99	.85
	16	1.0	1.0	1.0	1.0	1.0	1.0	.99
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
128 (fraction=.9)	8	1.0	1.0	1.0	1.0	1.0	1.0	.90
	16	.59	.96	1.0	1.0	1.0	1.0	.99
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 1. Fraction of P reconstructed with FedAvg model updates (users=100, rounds=200, Cifar10 LeNet, participant rate=.1, granularity=10, time limit per column=10 min). We exactly reconstruct P in the majority of FedAvg settings.

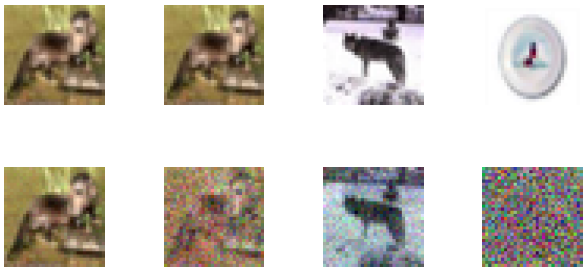
Additionally, we perform experiments on gradient disaggregation using model updates generated by the FedAvg algorithm (McMahan et al., 2017), on Cifar10 (Krizhevsky, 2009) with a LeNet neural network (SGD lr=.01). FedAvg performs multiple epochs of training over the participant's dataset before sending the final model difference back to the central server. We evaluate gradient disaggregation on updates generated by FedAvg over various parameter settings: local batchsize b, epochs e, user dataset size D (see (McMahan et al., 2017) for more details on these parameters); additionally, we simulate a shift in data distribution by randomly sampling a fraction f of participants' total data set during computation of model updates; finally we test disaggregation on updates generated with and without SGD momentum m. Figure 6(b) shows that relative variance of model updates ($D = 128$) increases with epochs of training, with momentum and with a shifting data distribution. However, as Table 1 shows we can reconstruct P exactly in nearly all cases. The failure cases happen at lower (≤ 1) or higher epochs (≥ 64) of training. At lower epochs, we believe parameters of the update are smaller and less distinguished from each other, making reconstruction more difficult; at higher epochs, reconstruction is more difficult as updates are more noisy. With $2 - 32$ epochs, we are generally able to exactly recover P across the settings.

4.2. Gradient Inversion Attacks with Disaggregation

We evaluate the benefits of gradient disaggregation on two methods to invert images from their gradients. Generally, gradient inversion methods optimize image data x',y' to match the target gradient ∇W : $\arg\min_{x',y'} ||\frac{\partial l(F(x',W),y')}{\nabla W} - \nabla W||^2$ (Zhu et al., 2019). This optimization grows exponentially more difficult with larger aggregates (Geiping et al., 2020; Zhu et al., 2019); we use gradient disaggregation to reduce the aggregate and improve the quality of the inverted images. To quantitatively measure quality, we use PSNR as in (Geiping et al., 2020). In our results we only show the reconstructed image with the smallest corresponding PSNR to a ground truth image for space.

We perform the attack in (Zhu et al., 2019) on an MLP network on Cifar100 and show the effect of inversion with and without gradient disaggregation across multiple users with each user having 1 image in their dataset (submitting full gradients of that image). Figure 7 shows the closest reconstructed image to a user's data example and Table 2 shows the corresponding PSNR achieved. With gradient disaggregation, we recover the target user's exact gradient and hence the reconstructed image is high quality. Without disaggregation, reconstruction quality degrades significantly.

We furthermore perform the attack in (Geiping et al., 2020) to invert noisy FedAvg updates. Figure 8 and Table 3 show the results of inverting fedavg updates with local epochs = 4, batch size = 16, user data set size = 64, with and without gradient disaggregation (100 users, 2 layer MLP). With gradient disaggregation we achieve similar quality as inverting a single model update, whereas inverting an update aggregated over multiple users (users=10) significantly degrades reconstruction quality.

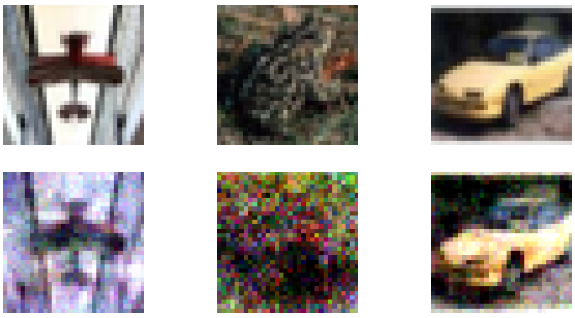


(a) users=1 (or with disaggregation) (b) users=2 (no disaggregation) (c) users=4 (no disaggregation) (d) users=32 (no disaggregation)

Figure 7. Recovered images from gradients across users (top image is the closest ground truth). Gradient disaggregation recovers individual users' exact gradients, hence, performing the gradient inversion attack with gradient disaggregation on multiple users yields the same quality as performing the attack on just one user. Without disaggregation, gradient inversion fails on gradients aggregated across more users.

	users=1	users=2	users=4	users=32
PSNR	36.5	18.8	13.9	6.1

Table 2. Corresponding PSNR scores against ground truth for Figure 7



(a) users=1 (no disaggregation) (b) users=10 (no disaggregation) (c) users=100 (disaggregated)

Figure 8. Recovered images from FedAvg updates across users (top image is closest ground truth). Gradient disaggregation enables high quality inversion on noisy FedAvg updates aggregated across many users; unlike disaggregation on exact gradients, disaggregation on noisy updates recovers the average update submitted across rounds, and we are able to reconstruct high quality images on noisy updates aggregated across many users. Without disaggregation, inversion on updates aggregated over multiple users (users=10) significantly degrades quality.

	users=1	users=10	users=100 (disaggregated)
PSNR	16.0	13.3	18.6

Table 3. Corresponding PSNR scores against ground truth for Figure 8.

4.3. Property Inference Attacks with Disaggregation

We demonstrate gradient disaggregation on property inference attacks as in (Melis et al., 2019). We train a gender model on the LFW dataset (Huang et al., 2007) and a model to predict whether participants' FedAvg updates (local epochs=4, batchsize=8, data size per user=32) on the gender model contain people of a specific race (hence the attacker's goal may be to learn a participants' images' race from the application). As in (Melis et al., 2019) only the target's dataset contains a significant proportion (p=.5) of images with the specific race and the goal is to determine whether the target's update is present in the aggregated updates over various numbers of users.

Figure 9 shows the AUC score of the attack across various numbers of users with and without gradient disaggregation. AUC score quickly degrades with more users; however, with gradient disaggregation high AUC score is maintained across increased numbers of participants as each user's model update is disaggregated exactly, allowing the property inference attack to be performed on each user separately. We note that the requirement in (Melis et al., 2019) that only the target has the particular data distribution is a limiting assumption, as many participants' data may exhibit the property of interest. With gradient disaggregation, learned properties are attributed to individual participants, enabling the central server to build profiles of users, violating anonymity.

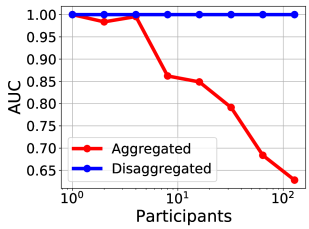


Figure 9. Property inference with and without gradient disaggregation (main task: gender classification, auxiliary task: identifying images of specific race) on FedAvg updates. Gradient disaggregation enables property inference on individual model updates and maintains high AUC score across increased number of users.

5. Discussion

We introduce gradient disaggregation, a method to disaggregate model updates from sums of model updates given repeated observations and access to summary information from device analytics. Our attack is capable of disaggregating model updates over thousands of users and we apply it to augment existing attacks such as gradient inversion and property inference. Our attack undermines the secure aggregation protocol.

Our findings show that summary metrics such as participa-

tion frequency may, when combined with gradient information, be used as an attack vector to undermine individual users' data privacy in federated learning systems. Ways to mitigate this attack include: injecting noise into model updates to reduce efficacy of disaggregation, using differential privacy on the collected device metrics to make reconstruction more difficult, and reducing or eliminating the collection of device analytics. These mitigation strategies may hinder the management of federated learning systems, and employing these techniques to increase privacy must be balanced with the costs to utility. We hope that bringing awareness to the privacy risks of side channel information in federated learning infrastructure will assist in designing secure federated learning systems.

6. Acknowledgements

We are grateful for the helpful discussions with members of Harvard Edge Computing group and VLSI group. This work was supported by the Application Driving Architectures (ADA) Research Center, a JUMP Center cosponsored by SRC and DARPA. Michael Mitzenmacher was supported in part by NSF grants CCF-1563710, CCF-1535795, and DMS-2023528, and by a gift to the Center for Research on Computation and Society at Harvard University. Maximilian Lam was supported by the Ashford Fellowship.

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, July 2018. URL https://arxiv.org/abs/1802.00420.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning, 2019.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 119–129. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf.

Bonawitz, K. A., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C. M., Konečný, J., Mazzocchi, S., McMahan, B., Overveldt, T. V., Petrou, D., Ramage, D., and Roselander, J. Towards federated learning at scale: System design. In *SysML 2019*, 2019. URL https://arxiv.org/abs/1902.01046. To appear.

Fleder, M. and Shah, D. I know what you bought at chipotle for $9.81 by solving a linear inverse problem. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(3), November 2020. doi: 10.1145/3428332. URL https://doi.org/10.1145/3428332.

FTC. Ftc privacy restrictions facebook. https://www.ftc.gov/news-events/press-releases/2019/07/ftc-imposes-5-billion-penalty-sweeping-new-privacy-restrictions, 2019.

Fung, C., Yoon, C. J. M., and Beschastnikh, I. Mitigating sybils in federated learning poisoning, 2020.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients – how easy is it to break privacy in federated learning?, 2020.

Gurobi Optimization, L. Gurobi optimizer reference manual, 2020. URL http://www.gurobi.com.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1135–1143, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

Hard, A., Kiddon, C. M., Ramage, D., Beaufays, F., Eichner, H., Rao, K., Mathews, R., and Augenstein, S. Federated learning for mobile keyboard prediction, 2018. URL https://arxiv.org/abs/1811.03604.

Hitaj, B., Ateniese, G., and Perez-Cruz, F. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pp. 603–618, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3134012. URL https://doi.org/10.1145/3133956.3134012.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning, 2019.

Konečný, J., McMahan, B., and Ramage, D. Federated optimization:distributed optimization beyond the datacenter, 2015.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, May 2020. ISSN 1558-0792. doi: 10.1109/msp.2020.2975749. URL http://dx.doi.org/10.1109/MSP.2020.2975749.

Lyu, L., Yu, H., and Yang, Q. Threats to federated learning: A survey, 2020.

McCabe, K. E. Just you and me and netflix makes three: Implications for allowing "frictionless sharing" of personally identifiable information under the video privacy protection act. Act, 20 J. Intell. Prop. L. 413, 2013.

McMahan, B. and Ramage, D. Federated learning google. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data, 2017.

Melis, L., Song, C., De Cristofaro, E., and Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706, 2019. doi: 10.1109/SP.2019.00029.

Mengkai, S., Wang, Z., Zhang, Z., Song, Y., Wang, Q., Ren, J., and Qi, H. Analyzing user-level privacy attack against federated learning. *IEEE Journal on Selected Areas in Communications*, PP:1–1, 06 2020. doi: 10.1109/JSAC.2020.3000372.

Qian, J. and Hansen, L. K. What can we learn from gradients?, 2020.

Segal, A., Marcedone, A., Kreuter, B., Ramage, D., McMahan, H. B., Seth, K., Bonawitz, K. A., Patel, S., and Ivanov, V. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, 2017. URL https://eprint.iacr.org/2017/281.pdf.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models, 2017.

Slawski, M., Hein, M., and Lutsik, P. Matrix factorization with binary components. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 26, pp. 3210–3218. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/226d1f15ecd35f784d2a20c3ecf56d7f-Paper.pdf.

So, J., Guler, B., and Avestimehr, A. S. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning, 2020.

Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., yong Sohn, J., Lee, K., and Papailiopoulos, D. Attack of the tails: Yes, you really can backdoor federated learning, 2020.

Wei, W., Liu, L., Loper, M., Chow, K.-H., Gursoy, M. E., Truex, S., and Wu, Y. A framework for evaluating gradient leakage attacks in federated learning, 2020.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients, 2019.

S1. Experiments on "Honest but Curious" Disaggregation Attack (neural network is updated, not fixed)

Prior experiments assumed that the central server fixed the neural network model. In the following experiments, we eliminate this assumption and update the neural network model with model updates computed from participants (via FedAvg). In this scenario, the attack becomes honest but curious as the attacker no longer has to modify the learning protocol (specifically, via fixing the model), but instead only needs to observe the gradient information and summary analytics collected by the server.

Dataset Size D	Batch Size b	Local Epochs e					
		1	2	4	8	16	32
64	8	1.0	1.0	.30	0.0	0.0	0.0
	16	1.0	1.0	1.0	.35	0.0	0.0
	32	1.0	1.0	1.0	1.0	.35	0.0
128	8	1.0	.52	0.0	0.0	0.0	0.0
	16	1.0	1.0	.46	0.0	0.0	0.0
	32	1.0	1.0	1.0	.49	0.0	0.0

Table S1. "Honest but curious" gradient disaggregation on FedAvg updates: neural network model is updated with participants' updates via FedAvg (SGD lr=1e-3) every round. Even with extra added noise from the changing model, our gradient disaggregation attack may reconstruct P exactly in a good portion of settings.

We run our experiments using FedAvg model updates, on Cifar10, with 100 users, 200 rounds, participation rate of .1, constraint granularity of 10, SGD lr of 1e-3, on a LeNet CNN model. Table S1 shows the fraction of P reconstructed across various FedAvg settings. Results show that with lower lr (1e-3), gradient disaggregation can exactly reconstruct P when FedAvg updates are small (1-4 epochs). With more epochs of FedAvg (and larger dataset size or smaller batch size), increased noise prevents reconstruction of P. We additionally tried the experiment with higher lr (1e-2), and disaggregation typically failed due to high noise, even with lower epochs of FedAvg. Our results indicate that under the right set of circumstances, the gradient disaggregation attack can be used in an honest but curious scenario; however, more robust results are achieved if the attacker can fix the neural network model.

S2. Experiments on Partial Constraints

We perform experiments showing our gradient disaggregation attack with partial sets of constraints. Specifically, constraint granularity determines the rounds across which number of participations is known (e.g: if granularity is 10, we know how many times each user participated between rounds 0-9, 10-19, 20-29, etc) and we drop a specific fraction of these constraints (e.g: in prior setting, only knowing participation counts between rounds 0-9, 40-49, etc). Testing this scenario shows the degree to which our method works when only partial summary analytics is given. We enforced a time limit of 10 minutes for solving each column

Table S2 shows the fraction of P reconstructed across various proportions of dropped constraints. Results indicate that even when significant proportions of constraints are dropped, P may be exactly recovered with more rounds of collected aggregated updates. Note that, when rounds $<$ users reconstruction fails due to the rank being less than the number of users.

S3. Experiments on Inexact "Noisy" Constraints

We perform experiments showing our gradient disaggregation attack when the constraints are inexact. For example, if analytics specified that a user participated 5 times between rounds 0-10, when the user actually participated 4 times. To handle noisy constraints, we relax our formulation and convert participation constraints into soft constraints:

$$\begin{aligned} \min \; & ||Nul(G_{aggregated}^{T})p_k||^2 + \lambda(C_k p_k - c_k) \\ & p_k \in \{0,1\}^n \end{aligned} \tag{8}$$

Where λ is a reweighting factor for the participation constraints.

Table S3 shows the results of gradient disaggregation when constraint noise $\mathcal{N}(0, \mu)$ is added to each constraint. As indicated, even in the presence of noise, gradient disaggregation may exactly recover P with enough rounds.

Users	Rounds	Constraint Fraction									
		1	.9	.8	.7	.6	.5	.4	.3	.2	.1
128	256	1	1	1	1	1	.97	.71	.26	.08	.03
	512	1	1	1	1	1	1	1	1	.99	.38
	1024	1	1	1	1	1	1	1	1	1	.99
	2048	1	1	1	1	1	1	1	1	1	1
	4096	1	1	1	1	1	1	1	1	1	1
256	256	0	0	0	0	0	0	0	0	0	0
	512	1	1	1	.98	.94	.63	.09	.02	0	0
	1024	1	1	1	1	1	1	1	.99	.70	.10
	2048	1	1	1	1	1	1	1	1	1	.96
	4096	1	1	1	1	1	1	1	1	1	1
512	256	0	0	0	0	0	0	0	0	0	0
	512	0	0	0	0	0	0	0	0	0	0
	1024	1	.99	.99	.93	.53	.09	0	0	0	0
	2048	1	1	1	1	1	1	1	.99	.55	.01
	4096	1	1	1	1	1	1	1	1	1	.85
1024	256	0	0	0	0	0	0	0	0	0	0
	512	0	0	0	0	0	0	0	0	0	0
	1024	0	0	0	0	0	0	0	0	0	0
	2048	1	1	.99	.95	.41	.01	0	0	0	0
	4096	1	1	1	1	1	1	.99	.81	.1	0

Table S2. Fraction of P recovered with gradient disaggregation on partial constraint information. With more rounds, we can exactly recover P even when a significant fraction of constraints are missing.

Number of Users	Rounds	Constraint Noise		
		.3	.5	1
32	128	1	1	.63
	256	1	1	.97
	512	1	1	1
	1024	1	1	1
	2048	1	1	1
64	128	1	1	.34
	256	1	1	.97
	512	1	1	1
	1024	1	1	1
	2048	1	1	1
128	128	0	0	0
	256	1	1	.7
	512	1	1	.99
	1024	1	1	1
	2048	1	1	1
256	128	0	0	0
	256	0	0	0
	512	1	1	.48
	1024	1	1	1
	2048	1	1	1

Table S3. Fraction of P recovered with gradient disaggregation when constraints are inexact/noisy. Participation rate=.1, λ=.1, constraint granularity = 10. Reconstruction is more successful with more rounds.

S4. Extended Experiments on Participation Rate

We provide extended data on gradient disaggregation against various parameter values of participation rate. Participation rate is the proportion of users that participate in sending model updates per round and impacts how many updates are summed to yield the aggregated update that is observed by the central server. Unless stated, we enforced a time limit of 10 minutes for solving each column; we use a constraint granularity of 10. We show our extended results in Table S4.

Users	Rounds	Participation Rate		
		.1	.2	.4
128	256	1	1	.05
	512	1	1	1
	1024	1	1	1
	2048	1	1	1
	4096	1	1	1
256	256	0	0	0
	512	1	.73	0
	1024	1	1	1
	2048	1	1	1
	4096	1	1	.31
512	256	0	0	0
	512	0	0	0
	1024	1	.34	0
	2048	1	1	1
	4096	1	1	.03
1024	256	0	0	0
	512	0	0	0
	1024	0	0	0
	2048	1	.02	0
	4096	1	.48	0
	5120*	1	1	0

Table S4. Fraction of P recovered via gradient disaggregation for various participation rates. * indicates settings where the time limit for solving each column was increased to 60 minutes (vs 10 minutes). Generally, using more rounds facilitates more successful reconstruction; note that with larger number of users and rounds, success rate decreased due to exceeding the 10 minute time limit.

S5. Extended Experiments on Constraint Granularity

We provide extended data on gradient disaggregation against various parameter values of constraint granularity. Constraint granularity is how precise summary statistics capture user partipation frequency (see main paper for details). Unless stated, we enforced a time limit of 10 minutes for solving each column; we use a constraint granularity of 10. We show our extended results in Table S5.

Users	Rounds	Constraint Granularity			
		10	20	40	80
128	256	1	1	.99	.95
	512	1	1	1	1
	1024	1	1	1	1
	2048	1	1	1	1
	4096	1	1	1	1
256	256	0	0	0	0
	512	1	.94	.27	.02
	1024	1	1	.97	.44
	2048	1	1	1	1
	4096	1	1	1	.86
512	256	0	0	0	0
	512	0	0	0	0
	1024	1	.5	0	0
	2048	1	1	.98	.24
	4096	1	1	.38	.035
1024	256	0	0	0	0
	512	0	0	0	0
	1024	0	0	0	0
	2048	1	.23	0	0
	4096	1	.91	0	0

Table S5. Fraction of P recovered via gradient disaggregation for various constraint granularities.

S6. Extended Experiments on FedAvg Updates (Cifar10)

We provide extended experiments on gradient disaggregation on FedAvg. We show our results in Table S6 and S7.

User Dataset Size	Batch Size	Local Epochs						
		1	2	4	8	16	32	64
384	8	1	1	1	1	1	1	1
	16	1	1	1	1	1	1	1
	32	1	1	1	1	1	1	1
	64	1	1	1	1	1	1	1
384 (mom.=.9)	8	.96	.88	.78	.97	.83	.19	.01
	16	1	1	1	1	1	1	1
	32	1	1	1	1	1	1	1
	64	1	1	1	1	1	1	1
384 (fraction=.9)	8	.98	1	1	1	.99	1	1
	16	1	1	1	1	1	1	1
	32	1	1	1	1	1	1	.99
	64	1	1	1	1	1	1	1
384 (fraction=.8)	8	1	1	1	.99	.83	.92	1
	16	.91	.95	.99	1	.97	.84	.95
	32	.99	.99	1	1	1	1	1
	64	1	1	1	1	1	1	1

Table S6. Fraction of P recovered on FedAvg updates using larger LeNet model (last hidden layer size=512), across various settings (mom.= SGD momentum, fraction=fraction of data sampled from the 384 examples to perform FedAvg over). Users=100, rounds=200, participation rate=.1, constraint granularity=10.

Number of Users	Batch Size	Local Epochs					
		1	2	4	8	16	32
512	16	1	1	1	.996	1	1
1024	16	.998	1	.999	1	1	1

Table S7. Fraction of P recovered on FedAvg updates using larger LeNet model (last hidden layer size=512), With more users. (rounds=200, participation rate=.1, constraint granularity=10). With many users, P is still reconstructable.