DOI: 10.1561/9781638284772.ch8

Chapter 8

Private Federated Learning

By Kallista Bonawitz, Peter Kairouz, Brendan McMahan and Daniel Ramage

8.1 Introduction

Machine learning and data science are key tools in science, public policy, and the design of products and services thanks to the increasing affordability of collecting, storing, and processing large quantities of data. But centralized collection can expose individuals to privacy risks and organizations to legal risks if data is not properly managed. Starting with early work in 2016 [McM+17; MR17], an expanding community of researchers has explored how data ownership and provenance can be made first-class concepts in systems for learning and analytics in areas now known as FL (federated learning) and FA (federated analytics). With this expanding community, interest has broadened from the initial work on federations of mobile devices to include FL across organizational silos, IoT (Internet of Things) devices, and more. In light of this, Kairouz et al. [Kai+19] proposed a broader definition that emphasized data locality. Since this definition was proposed, the field has continued to mature, revealing new challenges related to scalability, verifiability, and operational complexity. These challenges, particularly in the age of very large foundation models, have motivated a renewed focus on the core privacy properties of a system rather than the specific location of computation. To better capture these evolving principles and aspirations, a new definition has been proposed in [Dal+24]:

> ***Federated learning (FL)*** *is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a service provider. A complete FL system should enable clients to maintain full control over their data, the set of workloads allowed to access their data, and the anonymization properties of those workloads. FL systems should provide appropriate transparency and control to the users whose data is managed by FL clients.*

This updated perspective emphasizes that claims about a system's privacy require a nuanced description of how it addresses a multifaceted set of principles, a topic we will explore throughout this chapter.

An approach very similar in both philosophy and implementation, recently termed federated analytics [RM20], can be taken to allow data scientists to generate analytical insight from the combined information in decentralized datasets. While the focus here is on FL, much of the discussion on technology and privacy applies equally well to FA use cases.

Overview of the Chapter

This chapter provides a brief introduction to key concepts in federated learning and analytics with an emphasis on how privacy technologies may be combined in real-world systems and how their use charts a path toward societal benefit from aggregate statistics in new domains and with minimized risk to the individuals and the organizations who are custodians of the data. After defining FL and contrasting it with traditional centralized learning, we will discuss privacy in federated technologies, examining data minimization techniques (Section 8.2) and data anonymization methods using differential privacy (Section 8.3). We will also track the practical evolution of these technologies, highlighting key production deployments. Finally, the chapter discusses Federated Analytics, which broadens FL for performing data science tasks on decentralized data (Section 8.4), and concludes by examining the open challenges and future directions for the field.

8.1.1 Privacy Principals for Federated Learning and Analytics

To ground a more detailed discussion of FL, let us begin by clarifying the relevant notions of privacy. Privacy is an inherently multifaceted concept, even when restricted to the realm of the products and services offered by a technology company, which is the focus here. Four key components of privacy are highlighted in this context: (1) transparency and consent, (2) data minimization, (3) anonymization of released aggregates, and (4) verifiability.

Transparency and consent are foundational to privacy: they are how users of the product/service both understand and approve of the ways in which their data will

be used. Privacy technology cannot replace transparency and consent, but data-stewardship approaches based on strong privacy technologies make it easier for all parties involved to reason about which types of data usage might be possible (and which are ruled out by design), thereby enabling clearer privacy statements that are simpler to understand, verify, and enforce.

The role of privacy technology becomes more clear when considering specific goals that can be advanced by computation on privacy-sensitive user data; for example, improving a mobile keyboard suggestions based on user input to the virtual keyboard. How can the keyboard be improved in as minimally invasive a manner as possible? The computation goals are primarily the training of ML (machine-learning) models (federated learning) and the calculation of metrics or other aggregate statistics on user data (federated analytics). As we will see, both analytics and (perhaps less obviously) machine learning can be accomplished via appropriately chosen aggregations over (possibly preprocessed) user data. In this context, specializations of three of the above-mentioned broad privacy principles apply:

- *The principle of data minimization*, as applied to aggregations, includes the objective to collect only the data needed for the specific computation (focused collection), to limit access to data at all stages, to process individuals' data as early as possible (early aggregation), and to discard both collected and processed data as soon as possible (minimal retention). That is, data minimization implies restricting access to all data to the smallest set of people possible, often accomplished via security mechanisms, such as encryption at rest and on the wire, access-control lists, and also more nascent technologies such as secure multiparty computation and trusted execution environments, to be discussed later.
- *The principle of data anonymization* captures the objective that the final released output of the computation does not reveal anything unique to an individual. When this principle is specialized to anonymous aggregation, the goal is that data contributed by any individual user to the computation has only a small (limited, measured, and/or mitigated) influence on the final aggregate output. For example, aggregate statistics including model parameters, when released to an engineer—or beyond—should not vary significantly based on whether any particular user's data was included in the aggregation. The XKCD comic shown in Figure 8.1 illustrates a humorous example where this principle is not respected, but this memorization phenomenon has been shown to be a real issue for modern deep networks [Car+19; Car+20].
- *The principle of verifiability* asserts that privacy claims should be verifiable, ideally by users, external auditors, and the service provider itself. Mechanisms supporting verification can include open sourcing relevant code and

Figure 8.1. Randall Munroe humorously captured the risks of allowing one user's data too much influence on the final model in xkcd.com/2169/.

systems designs, public ledgers, trusted execution environment hardware, secure multi-party computation protocols, and alike.

The practical application of these privacy principles in production systems has evolved significantly since the inception of FL. Table 8.1 summarizes this progression, contrasting how core privacy guarantees were implemented in early FL systems (circa 2017–2020) with more recent practices (2021–2024), and it outlines an emerging state for the field, which we will discuss more in Section 8.3.2. The table illustrates a clear trajectory towards stronger, more comprehensive, and externally verifiable privacy protections.

By design, FL structurally embodies data minimization. Figure 8.2 compares the federated approach to more standard centralized techniques. Critically, the federated approach is architecturally designed to prevent the service provider from accessing raw, unaggregated client data. In its classic implementation, this is achieved by making data collection and aggregation inseparable: purpose-specific transformations of client data are computed on-device and sent only for the purpose of immediate aggregation. In newer paradigms that use confidential computing, this same principle is upheld through cryptography, where encrypted client

Table 8.1. A summary of how different privacy principles are addressed under the FL 2017-2020 practice, the FL 2021-2024 practice, and an updated 2025+ goal-state based on the new FL definition we propose. Adapted from [Dal+24].

Privacy Principles	FL 2017–2020	FL 2021–2024	FL 2025–?
Data minimization	Data remain on devices; focused updates and immediate aggregation for model training.	Trusted and cryptographic aggregation methods can additionally guarantee unaggregated updates invisible to the service provider.	Secured data on device or cloud with access verifiably limited to specific workloads and immediately revocable (or within a short TTL).
Data anonymization	No formal anonymization, but messages are collected for the purpose of immediate aggregation.	Distributed DP can provide acceptable utility for some tasks, and protection from an honest-but-curious service provider; central DP can provide better utility, and strong DP protection for the model released to end users but assumes a trusted aggregator.	Achieve the utility of current Central DP approaches, while also offering strong protection against even a malicious service provider; users can verify that only anonymized results are released, and can enforce their privacy preferences.
Transparency and control	Users can choose whether to participate in training, and potentially inspect the on-device binaries and network usage.	Users can additionally inspect the source code of *some* FL instances such as Private Compute Core [Mar+22], while others remain closed source and proprietary.	Users can view a human-readable summary of the purpose and (privacy) properties of any computation their data participated in, and those properties can be verified. Users can make fine-grained choices about which FL workloads to run, or delegate that power to an organization of their choice.
Verifiability and auditability	Where code is open-sourced, it can be inspected; verifying the identity of the code running on devices is possible but difficult.	Same as FL 2017–2020	Client and server-side code verify each others' integrity via remote attestation. Clients can verify the data minimization and anonymization properties of server-side computation. Clients and servers verify each others' authenticity via (ideally independent) Public Key Infrastructure (PKI).

contributions are processed only within a verifiable, trusted execution environment. In either architecture, analysts have no access to per-client messages. Federated learning and federated analytics are instances of a general federated computation schema that embodies these data-minimization practices. This contrasts sharply with the typical approach of centralized processing, which replaces on-device pre-processing and aggregation with bulk data collection, with the primary minimization happening on the server only after the raw data has been logged.

The ML and analytics goals considered here are compatible with the objective of anonymous aggregation. With ML, the goal is to train a model that generalizes well to all users, without overfitting or memorizing the specifics of any individual's data. Similarly, with statistical queries, the goal is to estimate population-level statistics that are not significantly influenced by any single contribution. However, achieving this is a non-trivial challenge, as modern deep learning models have been shown to be prone to memorizing rare or unique training examples, necessitating the use of explicit privacy-enhancing technologies.

The federated paradigm strengthens its guarantees by combining its architectural design with other techniques—particularly differential privacy (DP) and empirical privacy auditing, which are treated in more depth later—to ensure released aggregates are formally anonymous. This creates a system with built-in, technologically enforced protections. This situation contrasts sharply with the privacy relationship you might have with a bank or health-care provider, where the data anonymization principle may not apply. In those interactions, trust in the provider to use sensitive data only for its intended purpose remains the fundamental tenet.

A third foundational principle, which has grown in importance as FL has matured, is verifiability. While data minimization and anonymization provide strong theoretical protections, they historically required users to trust that the service provider was correctly implementing the protocols. Verifiability addresses this trust gap by enabling users and external auditors to cryptographically confirm that the system is performing computations as promised. This principle became central as technologies like trusted execution environments made it practical to remotely attest to the code running on a server, ensuring that only approved, privacy-preserving operations are ever performed on client data. This shifts the model from simply trusting the provider's policies to relying on verifiable, mathematical guarantees.

8.1.2 Federated Learning Settings and Applications

As indicated earlier, the defining characteristics of FL include keeping raw data decentralized and learning via aggregation. This assumption of locally generated data—often heterogeneous in distribution and quantity—distinguishes FL from more typical data center-based distributed learning settings, where data can be arbitrarily distributed and shuffled, and any worker node in the computation can access any of the data.

The role of a central orchestrator is practically useful and often necessary, as in the case of mobile devices that lack fixed IP addresses and require a central server to mediate device-to-device communication. It further constrains the space of relevant algorithms, and helps to distinguish FL from more general forms of decentralized

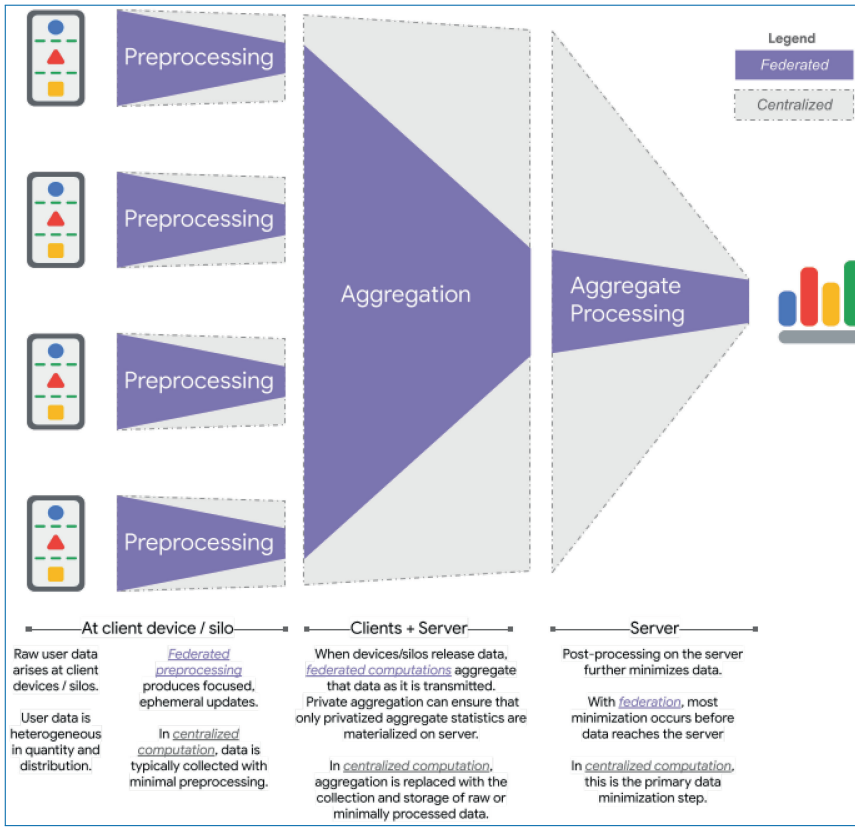

Figure 8.2. Federated learning and federated analytics are instances of a general federated computation schema that embodies data minimization practices. The more typical approach of centralized processing replaces on-device preprocessing and aggregation with data collection, with the primary minimization happening on the server during the processing of the logged data.

learning, including peer-to-peer approaches. From the basic definition, two FL settings have received particular attention:

- Cross-device FL, where the clients are large numbers of mobile or IoT devices.
- Cross-silo FL, where the clients are a typically smaller number of organizations, institutions, or other data silos.

Table 8.2, adapted from Kairouz et al. [Kai+19], summarizes the key characteristics of the FL settings, highlights some of the key differences between the cross-device and cross-silo settings, as well as contrasting with data center distributed learning.

Cross-device FL is now used by both Google [Bon+19] and Apple [Pau+21] for Android and iOS phones, respectively, for many applications such as mobile keyboard prediction; cross-device FA is being explored for problems such as health

Table 8.2. Typical characteristics of federated learning settings in contrast to traditional single-data center distributed learning. Adapted from [Kai+19].

	Data-center Distributed Learning	Cross-Silo Federated Learning	Cross-Device Federated Learning
Setting	Training a model on a large but "flat" dataset. Clients are compute nodes in a single cluster or data center.	Training a model on siloed data. Clients are different organizations (e.g., medical or financial) or data centers in different geographical regions.	The clients are a very large number of mobile or IoT devices.
Data Distribution	Data is centrally stored, so it can be shuffled and balanced across clients. Any client can read any part of the dataset.	Data is generated locally and remains decentralized. Each client stores its own data and cannot read the data of other clients. Data is not independently or identically distributed.	
Orchestration	Centrally orchestrated.	A central orchestration server/service organizes the training, but never sees raw data.	
Distribution Scale	Typically 1 – 1000 clients.	Typically 2 – 100 clients.	Up to 10^{10} clients.
Client Properties	Clients are reliable and almost always available to participate in computations. Clients may be directly addressed, and can maintain state across computation rounds.		Clients are often unavailable and can only be accessed by random sampling from available devices. For large populations a single client will typically only participate once in a given computation.

research (e.g., Google Health Studies). Since these early systems were described, the deployment of FL in production has accelerated significantly. At Google, FL powers numerous features in the Gboard mobile keyboard, including next-word prediction, smart compose, and emoji suggestions [Har+18; Xu+23]. Further applications include keyword spotting for virtual assistants [Har+22], smart text selection in Android [HK23], and smart reply in Android Messages [Goo20b]. Apple uses federated learning for features like scene identification in Photos [App23] and understanding aggregate trends for Apple Intelligence [App25], while Meta has developed systems for applications such as Ad prediction [Hub+22; Sto+22].

Cross-silo FL has received considerable attention as well. Health and medical applications are a primary motivation, with significant investments from Nvidia, IBM, and Intel, as well as numerous startups. Another application that is on the rise is finance, with investments from WeBank, Credit Suisse, Intel, and others.

8.1.3 Algorithms for Cross-device Federated Learning

Modern ML approaches, particularly deep learning, are generally data hungry and compute intensive, and so the feasibility of the federated training of production-quality models was far from a foregone conclusion. Much of our early work, particularly [McM+17] focused on establishing a proof of concept. This work introduced the federated averaging algorithm, which continues to see widespread use, though many variations and improvements have been subsequently proposed.

The core idea builds on the classic SGD (stochastic gradient descent) algorithm, which is widely used for the training of ML models in more traditional settings. The model is given as a function from training examples to predictions, parameterized by a vector of model weights, and a loss function that measures the error between the prediction and the true output (label). SGD proceeds by sampling a batch of training examples (typically from 10s to 1000s), computing the average gradient of the loss function with respect to the model weights, and then adjusting the model weights in the opposite direction of the gradient. By appropriately tuning the size of the steps taken on each iteration, SGD can be shown to have desirable convergence properties, even for nonconvex functions.

The simplest extension of SGD to the federated setting would be to broadcast the current model weights to a random set of clients, have them each compute the gradient of the loss on their local data, average these gradients across clients at the server, and then update the global model weights. SGD, however, often requires 10^5 or more iterations to produce a high-accuracy model. Back-of-the-envelope calculations suggest a single iteration might take minutes in the federated setting, implying federated training might take between a month and a year—outside the realm of practicality.

The key idea of federated averaging is intuitive: Decrease communication and startup costs by taking multiple steps of SGD locally on each device, and then average the resulting models (or model updates) less frequently. If models are averaged after each local step, this reduces to SGD (and is probably too slow); if models are averaged too infrequently, they might diverge and averaging could produce a worse model. Is there a sweet spot in between? Empirically, the 2017 paper [McM+17] showed that the answer is yes, demonstrating that moderate-sized language models (e.g., for next-word prediction) and image-classification models could be trained in fewer than 1,000 communication rounds. This reduces the expected training time to a few days—still much slower than would be possible with a high-performance compute cluster on centralized data, but within the realm of feasibility for real-world production use.

This algorithm also demonstrates the key privacy point mentioned earlier—that model training can be reduced to the (repeated) application of a federated aggregation (the averaging of model gradients or updates), as in Figure 8.2.

In practice, FedAvg and its variants fit into a generalized two-stage optimization framework where clients perform local updates using a client optimizer, and the server applies the aggregated update using a server optimizer. This flexible structure allows FL systems to incorporate advances from centralized training. For example, adaptive optimizers like Adam or Yogi can be used on the server to significantly improve performance, particularly for language tasks. This framework is also compatible with the integration of privacy technologies like differential privacy, as we will see in Section 8.3.

8.1.4 Workflows and Systems for Cross-device Federated Learning

Having a feasible algorithm for FL is a necessary starting point, but making cross-device FL a productive approach for ML-driven product teams requires much more. Based on Google's experience deploying cross-device FL across multiple Google products, the typical workflow often includes the following steps:

1. **Identifying a problem well suited for FL.** Typically this means a moderately sized (1-50 MB) on-device model is desired; training data potentially available on-device is richer or more representative than data available in the data center; there are privacy or other reasons to prefer not to centralize the data; and the feedback signal (labels) necessary to train the model are readily available on-device (for example, a model for next-word prediction can naturally be trained based on what users type if they ignore predicted next words; an image-classification model would be harder to train unless interaction with the app naturally led to labeled images).
2. **Model development and evaluation.** As with any ML task, choosing the right model architecture and hyperparameters (learning rates, batch sizes, regularization) is critical to success in FL. The challenge can be bigger in the federated setting, which introduces a number of new hyperparameters (e.g., number of clients participating in each round, how many local steps to take before averaging). Often the starting point is to do coarse model selection and tuning using a simulation of FL based on proxy data available in the data center. Final tuning and evaluation must be conducted using federated training on real devices, however, as the differences in data distribution, real-world device fleet characteristics, and many other factors are impossible to capture fully in simulation. Evaluation must also be conducted in a federated manner: independent from the training process, the candidate global model

is sent to (held-out) devices so that accuracy metrics can be computed on these devices' local datasets and aggregated by the server (both simple averages and histograms over per-client performance are important). Taken together, these needs give rise to two key infrastructure requirements: (1) providing high-performance FL simulation infrastructure that allows a smooth transition to running on real devices; (2) a cross-device infrastructure that makes it easy to manage multiple simultaneous training and evaluation tasks.

3. **Deployment.** Once a high-quality candidate model is selected in step 2, the deployment of that model (e.g., making user-visible next-word predictions in a mobile keyboard) typically follows the same procedures that are used for a data center-trained model: additional validation and testing (potentially including manual quality assurance), live A/B testing to compare to the previous production model, and a staged rollout to the full device fleet (potentially several-orders-of-magnitude more devices than actually participated in the training of the model).

It is worth emphasizing that all the work in step 2 has no impact on the user experience of the devices participating in training and evaluation; models being trained with FL do not make predictions visible to the user unless they go through the deployment step. Ensuring this processing does not otherwise negatively impact the device is a key infrastructure challenge. For example, heavyweight computation might execute only when the devices are idle, plugged in, and on an unmetered Wi-Fi network.

Figure 8.3 illustrates the model development and deployment workflows. Building a scalable infrastructure and compelling developer APIs for these workflows is a significant challenge. A paper by Bonawitz et al. [Bon+19] provides an overview of Google's production system as of 2019. Since then, other large-scale systems have been described by companies like Apple [Pau+21] and Meta [Hub+22], and a new system for "confidential federated computations" was recently introduced by Google [Eic+24; Dal+24]. These systems face common challenges related to scale, client heterogeneity, and availability, but they have adopted different strategies to manage them. For instance, while Google's early system was designed around synchronous rounds (with over-selection of clients to mitigate dropouts), Meta's system uses an asynchronous approach to improve robustness and efficiency.

8.1.5 Privacy for Federated Technologies

FL provides a variety of privacy advantages out of the box. In the spirit of data minimization, the raw data stays on the device, updates sent to the server are focused for a particular purpose, ephemeral, and aggregated as soon as possible. In particular, no non-aggregated data is persisted on the server; end-to-end encryption protects

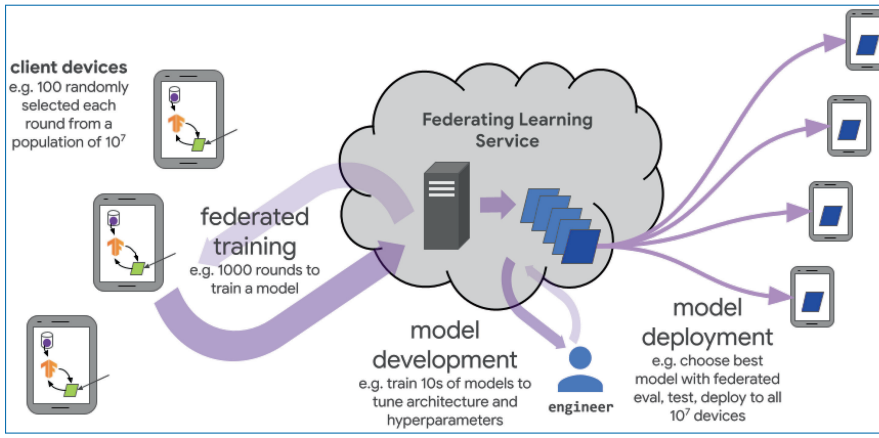

Figure 8.3. Components and phases of a cross-device FL system.

data in transit, and both the decryption keys and decrypted values are held only ephemerally in RAM. ML engineers and analysts interacting with the system can access only aggregated data. The fundamental role of aggregates in the federated approach makes it natural to limit the influence of any individual client on the output, but algorithms need to be carefully designed if the goal is to provide more formal guarantees such as differential privacy.

Researchers at Google and beyond are strengthening the privacy guarantees that an FL system can make. While the basic FL approach has proven feasible and gained substantial adoption, its combination with other techniques described in this section is still far from "on by default for most uses of FL." Even as the state of the art advances, inherent tensions with other objectives (including fairness, accuracy, development velocity, and computational cost) will likely prevent a one-size-fits-all approach to data minimization and anonymization. Thus, practitioners benefit from continued advancement of research ideas and software implementations for composable privacy enhancing techniques. Ultimately, decisions about privacy technology deployment are made by product or service teams in consultation with domain-specific privacy, policy, and legal experts. As privacy technologists, our obligation is two-fold: to enable products to offer more privacy through usable FL systems and, perhaps more importantly, to help policy experts strengthen privacy definitions and requirements over time.

In analyzing the privacy properties of a federated system, it is useful to consider access points and threat models. Building on Figure 8.3, one can ask what private information might an actor learn with access to various parts of the system. With access to the physical device or network? With root or physical access to the servers providing the FL service? To the models and metrics released to the ML engineer? To the final deployed model?

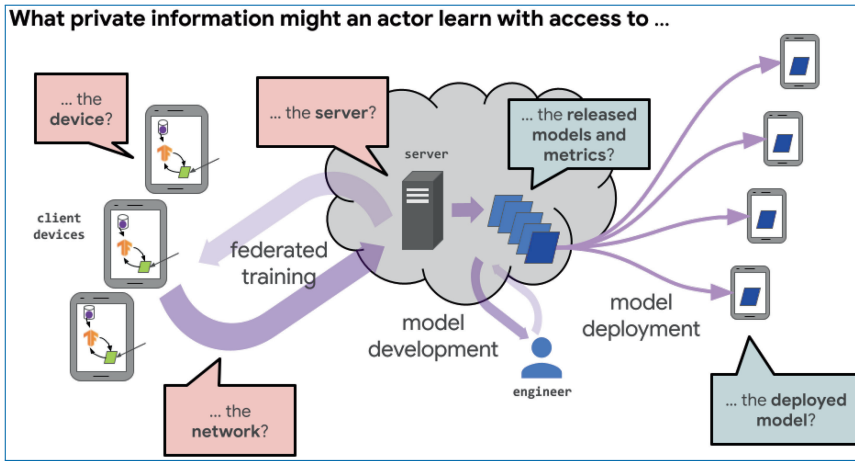

Figure 8.4. Threat models for an end-to-end federated learning system. The goal of the system is to release some models and metrics to the model engineer, and eventually deploy a model to production. Thus, anonymous aggregation is essential for these released outputs of the computation. Data minimization approaches can address potential threats to the device, network, and server, e.g. improving security and minimizing the retention of data and intermediate results.

The number of potentially-malicious parties varies dramatically as information flows through this system. A very small number of parties should have physical or root access to the coordinating server, for example, but nearly anyone might be able to access the final model shipped out to a large fleet of smartphones.

Privacy claims must therefore be assessed for a complete end-to-end system. A guarantee that the final deployed model has not memorized user data may not matter if suitable security precautions are not taken to protect the raw data on device or an intermediate computation state in transit. Other techniques can provide even stronger guarantees.

Figure 8.4 shows threat models for an end-to-end FL system and the role of data minimization and anonymous aggregation. Data minimization addresses potential threats to the device, network, and server by, e.g., improving security and minimizing the retention of data and intermediate results. When models and metrics are released to the model engineer or deployed to production, anonymous aggregation protects individuals' data from parties with access to these released outputs.

8.2 Data Minimization for Federated Aggregation

At several points in a federated computation, the participants expect one another to take the appropriate actions, and only those actions. For example, the server expects the clients to execute their preprocessing step accurately; the clients expect

the server to keep their individual updates a secret until they have been aggregated; both the clients and the server expect that neither the data analyst nor the deployed ML model user will be able to extract an individual's data; and so on.

Privacy-preserving technologies support the structural enforcement of these interparty expectations, preventing participants from deviating even if they happen to be malicious or compromised. In fact, FL systems can be viewed as a kind of privacy-preserving technology in themselves, structurally preventing the server from accessing anything about a client's data that was not included in the update submitted by that client.

Take, for example, the aggregation phase of FL. An idealized system might imagine a completely trusted third party who aggregates the clients' updates and reveals only the final aggregate to the server. In reality, no such mutually trusted third party typically exists to play this role, but various technologies allow an FL system to simulate such a third party under a wide range of conditions.

For example, a server could run the aggregation procedure within a secure enclave—a specially constructed piece of hardware that can not only prove to the clients what code it is running, but also ensure that no one (not even the hardware's owner) can observe or tamper with the execution of that code. Currently, however, the availability of secure enclaves is limited, both in the cloud and on consumer devices, and available enclaves may implement only some of the desired enclave properties (secure measurement, confidentiality, and integrity [Sub+17]). Moreover, even when available and full-featured, secure enclaves may come with additional limitations, including very limited memory or speed; vulnerability to data exposure via side channels (e.g., cache-timing attacks); difficult-to-verify correctness (because of proprietary implementation details); dependence on manufacturer-provided attestation services (and key secrecy); etc.

Distributed cryptographic protocols for secure multiparty computation can be used collaboratively to simulate a trusted third party without the need for specialized hardware, so long as a sufficiently large number of the participants behave honestly. While secure multiparty computation for arbitrary functions remains computationally prohibitive in most cases, specialized secure aggregation algorithms for vector summation in the federated setting have been developed that provably preserve privacy even against an adversary that observes the server and controls a significant fraction of the clients, while maintaining robustness against clients dropping out of the computation [Bon+17]. Such algorithms are both:

- Communication efficient - $O(\log n + l)$ communication per client, where n is the number of users and l is the vector length, with small constants yielding less than twice the communication of aggregation in the clear for a wide range of practical settings; and

- Computation efficient - $O(\log(2n) + l \log n)$ computation per client [Bel+20].

Cryptographic secure aggregation protocols have been deployed in commercial federated computing systems for years [Bon+19; RM20]. The development of highly efficient algorithms [Bel+20] has been critical for this, enabling the aggregation of updates for models with millions of parameters from thousands of clients per round. This technology is now used in production to train Gboard language models and Android smart selection models [Xu+23; Zha+23; HK23].

Beyond private aggregation, privacy-preserving technologies can be used to secure other parts of an FL system. For example, either secure enclaves or cryptographic techniques (e.g., zero knowledge proofs) can ensure that the server may trust that clients have preprocessed faithfully. Even the model broadcast stage can benefit: For many learning tasks, an individual client may have data relevant to only a small portion of the model; in this case, the client can privately retrieve just that segment of the model for training, again using either secure enclaves or cryptographic techniques (e.g., private information retrieval) to ensure that the server learns nothing about the segment of the model for which the client has relevant training data.

8.3 Data Anonymization for Federated Aggregation

While secure enclaves and private aggregation techniques can strengthen data minimization, they are not designed specifically to produce anonymous aggregates—for example, limiting the influence of a user on the model being trained. Indeed, a growing body of research suggests that the learned model can (in some cases) leak sensitive information [Car+19; Car+20].

The gold-standard approach to data anonymization is DP (differential privacy) [DMNS06]. For a generic procedure that aggregates records in a database, DP requires bounding any record's contribution to the aggregate and then adding an appropriately scaled random perturbation. For example, as discussed in Section 7.4 of Chapter 7, in DP-SGD (differentially private stochastic gradient descent) you clip the ℓ_2 norm of the gradients, aggregate the clipped gradients, and add Gaussian noise in each training round [SCS13; BST14; Aba+16].

Differentially private algorithms are necessarily randomized, and hence you can consider the distribution of models produced by an algorithm on a particular dataset. Intuitively, differential privacy says this distribution over models is similar when the algorithm is run on input datasets that differ by a single record. Formally, DP is commonly quantified by privacy loss parameters (ε, δ), where a

smaller (ε, δ) pair corresponds to increased privacy. A randomized algorithm A is (ε, δ)-differentially private if for all possible outputs (e.g., models) m, and for all datasets D and D' that differ in at most one record:

$$\Pr\left(A(D) = m\right) \leq e^{\varepsilon} \Pr\left(A(D') = m\right) + \delta. \tag{8.1}$$

This goes beyond simply bounding the sensitivity of the model to each record by adding noise proportional to any record's influence, therefore ensuring sufficient randomness to mask any one record's contribution to the output. For a review of Differential Privacy and its properties, please see Chapter 1.

8.3.1 Privacy Units

In the context of cross-device FL, a record is typically defined as all the training examples of a single user/client [MRTZ18]. This notion of DP is referred to as user-level DP and is stronger than example-level DP where a record corresponds to a single training example, because in general one user may contribute many training examples. Even in centralized settings, FL algorithms are well suited for training with user-level DP guarantees, because they compute a single update to the model from all of a user's data, making it much easier to bound each user's total influence on the model update (and hence final model).

In the context of cross-silo FL, the unit of privacy can take on a different meaning. For example, it is possible to define a record as all the examples on a data silo if the participating institutions want to ensure that an adversary who has access to the model iterates or final model cannot determine whether or not a particular institution's dataset was used in the training of that model. User-level DP can still be meaningful in cross-silo settings where each silo holds data for multiple users. Enforcing user-level privacy, however, may be more challenging if multiple institutions have records from the same user.

8.3.2 The Differentially Private Federated Averaging (DP-FedAvg) Algorithm

The easiest way to train federated models with user-level DP is to extend the Federated Averaging (FedAvg) algorithm in the following ways:

- When on-device training is completed, the model update is clipped to bound the ℓ_2 sensitivity of the update.
- Once the server has aggregated all the clipped model updates, it adds Gaussian noise.

Algorithm 7 A Single Round of Differentially Private Federated Averaging (DP-FedAvg)

Parameters: Model update x_i for each client i; ℓ_2 clip norm $c > 0$; Target noise variance $\sigma^2 > 0$

function ClientProcedure(x_i, c)
  return $\hat{x}_i = \min(1, c/\|x_i\|_2) \cdot x_i$ ▷ Clip model update locally
end function

function ServerProcedure($\hat{x}_1, \cdots, \hat{x}_n, \sigma^2$)
  return $\bar{x} = \frac{1}{n}\sum_i \hat{x}_i + \mathcal{N}(0, \sigma^2 I)$ ▷ Add Gaussian noise on the server
end function

This is shown in Algorithm 7. We note that Algorithm 7 uses a fixed ℓ_2 clip norm of c. Observe that for a fixed target ε, σ (the noise's standard deviation) has to scale linearly with c. On the one hand, choosing a small c implies a smaller σ but leads to (potentially) higher bias as it may lead to more frequently clipping of model updates. On the other hand, choosing a large c implies a larger σ, increasing the variance of per-round gradient estimate. Thus, there is no good a priori setting of the clipping norm across tasks and learning settings: the model update norm distribution depends on the model architecture and loss, the amount of data on each device, the client learning rate, and possibly various other parameters.

To resolve this issue, [ATMR21] proposes a method wherein instead of a fixed clipping norm, one clips to a value at a specified quantile of the model update norm distribution, where the value at the quantile is itself estimated online, with differential privacy. The method tracks the quantile closely, uses a negligible amount of privacy budget, is compatible with other federated learning technologies such as compression and secure aggregation, and has a straightforward joint DP analysis with DP-FedAvg. Experiments demonstrate that adaptive clipping to the median update norm works well across a range of realistic federated learning tasks, sometimes outperforming even the best fixed clip chosen in hindsight, and without the need to tune any clipping hyperparameter. The implementation details can be found in Algorithm 1 of [ATMR21].

8.3.3 Formal Privacy Guarantees for Cross-Device FL

While it is easy to add DP to the FedAvg algorithm, providing a formal (ε, δ) is more complex and requires mathematical care. Providing formal (ε, δ) guarantees in the context of cross-device FL systems can be particularly challenging because the set of all eligible users is dynamic and not known in advance, and the participating

users may drop out at any point in the protocol [Bon+19; Bal+20]. To address this challenge, Kairouz et al. [Kai+21] propose a concrete approach to overcome these challenges based on the DP-FTRL ("DP-Follow-The-Regularized-Leader") algorithm, described in detail in Section 6.3.2. This method was recently used to train and launch a federated language model with a rigorous DP guarantee [MT21]. In fact, this approach, assuming an honest server for noise addition, has been used to train and launch over thirty Gboard language models with meaningful, formal (ε, δ)-DP guarantees, with ε values in the range of $[0.994, 13.69]$ for $\delta = 10^{-10}$ [Xu+23].

The primary difference between DP-FTRL and DP-SGD methods is that DP-FTRL uses correlated noise instead of independent noise in each training round. For an in-depth treatment of this topic, we direct the review to the comprehensive monograph by Pillutla et al. [Pil+25].

8.3.4 Distributing Trust in Differentially Private Federated Learning

Over the past decade, an extensive set of techniques has been developed for differentially private data analysis, particularly for the central or trusted-aggregator setting, where the raw (or minimized) data is collected by a trusted service provider that implements the DP algorithm. Another area of significant interest is the local model of DP [Kas+08], where the data is perturbed on the client side before it is collected by a service provider (see also Chapter 2 of this book for an extensive discussion about Local DP). Local DP avoids the need for a fully trusted aggregator, but it is now well established that local DP leads to a steep hit in accuracy.

To recover the utility of central DP without having to rely on a fully trusted central server, a set of approaches, often referred to as distributed DP, can be used [Bit+17; KLS21; AKL21]. The goal is to render the output differentially private before it becomes visible (in plaintext) to the server. Under distributed DP, clients first compute minimal application-specific reports, perturb these slightly with random noise, and then execute a private aggregation protocol. The server then has access only to the output of the private aggregation protocol. The noise added by individual clients is typically insufficient for a meaningful local DP guarantee on its own. After private aggregation, however, the output of the private aggregation protocol provides a stronger DP guarantee based on the total sum of noise added across all clients. This applies even to someone with access to the server under the security assumptions necessary for the private aggregation protocol. This approach has been used in production to train smart text selection models in Android [HK23], although achieving strong formal DP guarantees can be challenging in practice due to difficulties with techniques like privacy amplification via

sampling in large-scale cross-device federated systems. A very nascent line of work explores how to physically distribute DP-FTRL based algorithms across participating devices (see [Bal+24]), but more work is needed to make these approaches feasible in practice at scale.

For the sake of completeness, we briefly summarize a distributed DP via secure aggregation protocol based on the distributed discrete Gaussian mechanism in Algorithm 8. The implementation details can be found in [KLS21].

Algorithm 8 A single round of the Distributed Discrete Gaussian Mechanism

Parameters: Model update $x_i \in \mathbb{R}^d$ for each client i; ℓ_2 clip norm $c > 0$; Target noise variance $\sigma^2 > 0$; Secure Aggregation's modulus $M \in \mathbb{N}$; A random rotation matrix U_{rotate}; A large scaling parameter s

function ClientProcedure($x_i, c, M, \sigma^2, U_{\text{rotate}}, s$)
  Clip and scale x_i so that $||x_i'||_2 < s \cdot c$
  Randomly rotate vector: $x_i'' = U_{\text{rotate}} \cdot x_i'$
  Stochastically round x_i'' to obtain $x_i''' \in \mathbb{Z}^d$
  Compute $Z_i = x_i''' + \mathcal{N}_{\mathbb{Z}}(0, \sigma^2) \mod M$, where $\mathcal{N}_{\mathbb{Z}}$ is the discrete Gaussian noise
  return $Z_i \in \mathbb{Z}_M^d$
end function

$S = \sum_i Z_i \mod M$: the output of the Secure Aggregation protocol

function ServerProcedure(S, U_{rotate}, s)
  return $(1/s) U_{\text{rotate}}^T S$ ▷ Unscale and unrotate the output of Secure Aggregation
end function

Significant practical challenges arise when physically distributing the DP mechanism and employing private aggregation to limit an honest-but-curious server's view to a DP aggregate in each round. First, the required multi-round cryptographic communications are computationally intensive and introduce high network overhead, which can account for the vast majority of training time. Further, these systems are fragile and highly susceptible to client dropouts, a common occurrence in real-world federated settings. When a client drops out, its noise contribution is lost, which can compromise the formal DP guarantee by making the final aggregate less noisy than required by the privacy budget. Second, the approach presented in Algorithm 8 distributes DP mechanisms that add independent Gaussian noise in every round (e.g DP-FedAvg), but it is not capable of distributing DP mechanisms that add correlated Gaussian noise across training rounds (e.g. DP-FTRL), which

is necessary for achieving formal DP guarantees in the context of cross-device FL. Third, this approach is susceptible to Sibyl attacks since a malicious could (in theory) control all but one of the devices participating in a training round, reducing the privacy guarantee substantially.

To address these challenges, a new approach, called Confidential Federated Computations (CFC), was recently introduced [Eic+24; VR25]. CFC leverages hardware-based Trusted Execution Environments (TEEs), which create a secure, isolated enclave on the server where data can be processed in a confidential and tamper-proof manner, inaccessible even to the server operator. This architecture makes it possible to deploy high-utility DP algorithms that use correlated noise across training rounds without the need for a trusted server, closing the utility gap while providing strong, verifiable privacy.

The framework's verifiability is built on a "chain of trust" that begins on the user's device. Before uploading, data is encrypted and cryptographically bound to a specific, publicly auditable "Access Policy" that dictates exactly which computations are permitted to process it. A TEE-hosted service called the "Ledger" acts as a keymaster, only releasing decryption keys to other TEEs that use cryptographic attestation to prove they are running an authorized, open-source computation from the policy.

A primary early application of this system is Confidential Federated Analytics, which was deployed to improve Google Keyboard (Gboard). In particular, it was used to discover new, frequently typed out-of-vocabulary Indonesian words. In the past, this task relied on locally DP protocols like TrieHH [Zhu+20], which struggled to detect rare words or work well in low-volume languages due to high noise. With CFC, user devices submit encrypted local data to a server-side TEE that runs a verified DP histogram algorithm, enabling Google to extract useful aggregate insights with far lower noise and tighter privacy guarantees ($\varepsilon = \ln 3$ per week, per device). The system successfully uncovered 3,600 missing words in just two days, showcasing both improved utility and user trust through verifiability and hardware-enforced privacy. The application of CFC in the context of training a model on federated data is widely believed to be possible but has not happened yet.

8.3.5 Complementary Privacy Auditing Empirical Techniques

For an algorithm to provide a formal user-level DP guarantee, it must not only bound the sensitivity of the model to each user's data, but also add noise proportional to that sensitivity. While the addition of sufficient random noise is required to ensure a small enough ε for the DP definition itself to offer a strong guarantee, empirically it has been observed that limiting sensitivity even with small amounts of noise (or no noise at all) can significantly reduce memorization [Ram+20]. This gap

is to be expected, as DP assumes a "worst-case adversary" with infinite computation and access to arbitrary side information. These assumptions are often unrealistic in practice. Thus, there are substantial advantages to training using a DP algorithm that limits each user's influence, even if the explicit random noise introduced into the training process is not enough to ensure a small ε formally. Nevertheless, designing practical FL and FA algorithms that achieve small ε guarantees is an important area of ongoing research.

Model auditing techniques can be used to further quantify the advantages of training with DP [Car+19; Car+20; Ram+20]. These techniques are empirical in nature and can be applied during or after training. They broadly include techniques that quantify how much a model overlearns (or memorizes) unique or rare training examples and techniques that quantify to what extent it is possible to infer whether or not a user's examples were used during training. These auditing techniques are useful even when a large ε is used, as they can quantify the gap between DP's worst-case adversaries and realistic ones with limited computational power and side information. They can also serve as a complementary technology for pressure-testing DP implementations: unlike the formal mathematical statements of DP, these auditing techniques are applied to complete end-to-end systems, potentially catching software bugs or mis-chosen parameters.

Some additional discussion regarding auditing the information leakage of a system is presented in Chapter 16.

8.4 Federated Analytics

The focus of this chapter so far has primarily been on FL. Beyond learning ML models, data analysts are often interested in applying data science methods to the analysis of raw data that is stored locally on users' devices. For example, analysts may be interested in learning aggregate model metrics, popular trends and activities, or geospatial location heatmaps. All of this can be done using FA [RM20]. Similar to FL, FA works by running local computations over each device's data and making only the aggregated results available to product engineers. Unlike FL, however, FA aims to support basic data science needs, such as counts, averages, histograms, quantiles, and other SQL-like queries. Two prominent examples of FA in practice are: (1) its use in Google Health Studies to power privacy-preserving health research [Goo20a], and (2) its use for environmental studies to provide cities with critical information about transportation-related greenhouse gas emissions, derived from aggregated and anonymized Google Maps Timeline [Bia+24].

Consider an application where an analyst wants to use FA to learn the 10 most frequently played songs in a music library shared by many users. The federated and

privacy techniques discussed above can be used to perform this task. For example, clients can encode which songs they have listened to into a binary vector of length equal to the size of the library and use distributed DP to ensure that the server sees only a differentially private sum of these vectors, giving a DP histogram of how many users have played each song. As this example illustrates, however, FA tasks can differ from FL ones in several ways:

1. FA algorithms are often noninteractive and involve rounds with a large number of clients. In other words, unlike FL applications, there are no diminishing returns from having more clients in a round. Therefore, applying DP is less challenging in FA since each round can contain a large number of clients, and fewer rounds are needed.
2. There is no need for the same clients to participate again in later rounds. In fact, clients that participate again may bias the results of the algorithm. Therefore an FA task is best served by an infrastructure that limits the number of times any individual can participate.
3. FA tasks are typically sparse, making efficient private sparse aggregation a particularly important topic; many open research questions exist in this space.

It is worth noting that while limiting client participation and sparse aggregation are particularly relevant to FA, they have applications for FL problems as well.

8.5 Concluding Remarks

We are optimistic that FL will continue to expand, both as a research field and as a set of practical tools and software systems that allow applications by more people to more types of data and problem domains.

Despite this progress, significant challenges remain. The first is scaling to the enormous size of modern foundation models, whose multi-billion parameter counts are orders of magnitude larger than what current cross-device FL systems can handle due to on-device compute, memory, and network constraints. Second, providing strong, externally verifiable privacy guarantees for server-side computations remains a difficult open problem, particularly in defending against a malicious service provider. Finally, the operational complexity of coordinating training across millions of heterogeneous and unreliable devices creates persistent system-level hurdles that can hinder broader adoption.

A promising path forward, which embodies the updated, property-centric definition of FL, involves leveraging confidential cloud computing. Recent proposals outline a paradigm of Confidential Federated Computations [Eic+24], where clients encrypt their data before upload. This data can then only be processed inside a

server-side Trusted Execution Environment (TEE) running a verifiably open-source workload. A trusted ledger service ensures that decryption keys are only released to approved, privacy-preserving workloads (e.g., a workload that provably implements a DP aggregation algorithm). This approach could resolve the scalability bottleneck by moving heavy computation off-device, enabling federated training of large models, while simultaneously strengthening privacy by providing verifiable, end-to-end guarantees about how data is processed, regardless of its location. This evolution represents an exciting next chapter for the field, aiming to achieve stronger privacy with greater flexibility and scale.

For those interested in learning more about active research directions, the recent vision paper "Federated Learning in Practice: Reflections and Projections" discusses the evolution of FL, latest advances and deployments, lingering and emerging challenges, and the future of federated technologies [Dal+24]. The relatively established monograph "Advances and Open Problems in Federated Learning" provides a broad survey, with coverage of important topics not covered in this chapter, including personalization, robustness, fairness, and systems challenges [Kai+19]. If you are interested in a more hands-on introduction to FL, such as trying out algorithms in a simulation environment on either your own data or standard data sets, the Google Parfait GitHub library is a great place to start. It contains many examples that can be executed and modified on the fly in the browser using Google Colab.

Acknowledgements

The authors would like to thank Alex Ingerman and Marco Gruteser for helpful feedback on earlier drafts of this chapter, as well as the many people at Google who have helped develop these ideas and bring them to practice.

References

[Aba+16] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. "Deep Learning with Differential Privacy". In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 308–318. ISBN: 9781450341394. URL: https://doi.org/10.1145/2976749.2978318 (cit. on p. 301).

[AKL21] N. Agarwal, P. Kairouz, and Z. Liu. "The skellam mechanism for differentially private federated learning". In: Advances in Neural Information Processing Systems 34 (2021) (cit. on p. 304).

[App23] Apple. Learning Iconic Scenes with Differential Privacy. https://machinelearning.apple.com/research/scenes-differential-privacy. 2023 (cit. on p. 294).

[App25] Apple. Understanding Aggregate Trends for Apple Intelligence Using Differential Privacy. https://machinelearning.apple.com/research/differential-privacy-aggregate-trends. 2025 (cit. on p. 294).

[ATMR21] G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy. "Differentially private learning with adaptive clipping". In: Advances in Neural Information Processing Systems 34 (2021) (cit. on p. 303).

[Bal+20] B. Balle, P. Kairouz, H. B. McMahan, O. Thakkar, and A. Thakurta. "Privacy amplification via random check-ins". In: NeurIPS. 2020 (cit. on p. 304).

[Bal+24] M. Ball, J. Bell-Clark, A. Gascon, P. Kairouz, S. Oh, and Z. Xie. "Secure Stateful Aggregation: A Practical Protocol with Applications in Differentially-Private Federated Learning". In: arXiv preprint arXiv:2410.11368 (2024) (cit. on p. 305).

[Bel+20] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. "Secure single-server aggregation with (poly) logarithmic overhead". In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020, pp. 1253–1269 (cit. on p. 301).

[Bia+24] C. Bian, A. Cheu, S. Chiknavaryan, Z. Gong, M. Gruteser, O. Guinan, Y. Guzman, P. Kairouz, A. Lagzdin, R. McKenna, et al. "Mayfly: Private Aggregate Insights from Ephemeral Streams of On-Device User Data". In: arXiv preprint arXiv:2412.07962 (2024) (cit. on p. 307).

[Bit+17] A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld. "Prochlo: Strong Privacy for Analytics in the Crowd". In: Proceedings of the Symposium on Operating Systems Principles (SOSP). 2017, pp. 441–459. URL: https://arxiv.org/abs/1710.00901 (cit. on p. 304).

[Bon+17] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. "Practical secure aggregation for privacy-preserving machine learning". In: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017, pp. 1175–1191 (cit. on p. 300).

[Bon+19] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, et al. "Towards federated learning at scale: System design". In: arXiv preprint arXiv:1902.01046 (2019) (cit. on pp. 293, 297, 301, 304).

[BST14] R. Bassily, A. Smith, and A. Thakurta. "Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds". In: Proc. of the 2014 IEEE 55th Annual Symp. on Foundations of Computer Science (FOCS). 2014, pp. 464–473 (cit. on p. 301).

[Car+19] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. "The secret sharer: Evaluating and testing unintended memorization in neural networks". In: 28th {USENIX} Security Symposium ({USENIX} Security 19). 2019, pp. 267–284 (cit. on pp. 289, 301, 307).

[Car+20] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. "Extracting training data from large language models". In: arXiv preprint arXiv:2012.07805 (2020) (cit. on pp. 289, 301, 307).

[Dal+24] K. Daly, H. Eichner, P. Kairouz, H. B. McMahan, D. Ramage, and Z. Xu. "Federated learning in practice: reflections and projections". In: 2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA). IEEE. 2024, pp. 148–156 (cit. on pp. 287, 291, 297, 309).

[DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith. "Calibrating Noise to Sensitivity in Private Data Analysis". In: IACR Theory of Cryptography Conference (TCC), New York, New York. Vol. 3876. Lecture Notes in Computer Science. Springer-Verlag, 2006, pp. 265–284 (cit. on p. 301).

[Eic+24] H. Eichner, D. Ramage, K. Bonawitz, D. Huba, T. Santoro, B. McLarnon, T. Van Overveldt, N. Fallen, P. Kairouz, A. Cheu, et al. "Confidential Federated Computations". In: arXiv preprint arXiv:2404.10764 (2024) (cit. on pp. 297, 306, 308).

[Goo20a] Google. Advancing health research with Google Health Studies. https://blog.google/technology/health/google-health-studies-app/. 2020 (cit. on p. 307).

[Goo20b] Google. Your chats stay private while Messages improves suggestions. https://support.google.com/messages/answer/9327902. 2020 (cit. on p. 294).

[Har+18] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. "Federated learning for mobile keyboard prediction". In: arXiv preprint arXiv:1811.03604 (2018) (cit. on p. 294).

[Har+22] A. Hard, K. Partridge, N. Chen, S. Augenstein, A. Shah, H. J. Park, A. Park, S. Ng, J. Nguyen, I. L. Moreno, et al. "Production federated keyword spotting via distillation, filtering, and joint federated-centralized training". In: arXiv preprint arXiv:2204.06322 (2022) (cit. on p. 294).

[HK23] F. Hartmann and P. Kairouz. Distributed differential privacy for federated learning. https://research.google/blog/distributed-differential-privacy-for-federated-learning. 2023 (cit. on pp. 294, 301, 304).

[Hub+22] D. Huba, J. Nguyen, K. Malik, R. Zhu, M. Rabbat, A. Yousefpour, C.-J. Wu, H. Zhan, P. Ustinov, H. Srinivas, et al. "Papaya: Practical, private, and scalable federated learning". In: Proceedings of Machine Learning and Systems 4 (2022), pp. 814–832 (cit. on pp. 294, 297).

[Kai+19] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. "Advances and open problems in federated learning". In: arXiv preprint arXiv:1912.04977 (2019) (cit. on pp. 287, 293, 294, 309).

[Kai+21] P. Kairouz, B. Mcmahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu. "Practical and Private (Deep) Learning Without Sampling or Shuffling". In: Proceedings of the 38th International Conference on Machine Learning. 2021, pp. 5213–5225 (cit. on p. 304).

[Kas+08] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith. "What Can We Learn Privately?" In: 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008. IEEE Computer Society, 2008, pp. 531–540. URL: https://doi.org/10.1109/FOCS.2008.27 (cit. on p. 304).

[KLS21] P. Kairouz, Z. Liu, and T. Steinke. "The distributed discrete gaussian mechanism for federated learning with secure aggregation". In: arXiv preprint arXiv:2102.06387 (2021) (cit. on pp. 304, 305).

[Mar+22] E. Marchiori, S. de Haas, S. Volnov, R. Falcon, R. Pinto, and M. Zamarato. "Android Private Compute Core Architecture". In: arXiv preprint arXiv:2209.10317 (2022) (cit. on p. 291).

[McM+17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. "Communication-efficient learning of deep networks from decentralized data". In: Artificial intelligence and statistics. PMLR. 2017, pp. 1273–1282 (cit. on pp. 287, 295).

[MR17] B. McMahan and D. Ramage. Federated Learning: Collaborative Machine Learning without Centralized Training Data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. 2017 (cit. on p. 287).

[MRTZ18] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. "Learning Differentially Private Recurrent Language Models". In: International Conference on Learning Representations (ICLR). 2018 (cit. on p. 302).

[MT21] B. McMahan and A. Thakurta. Federated Learning with Formal Differential Privacy Guarantees. https://ai.googleblog.com/2022/02/federated-learning-with-formal.html. 2021 (cit. on p. 304).

[Pau+21] M. Paulik, M. Seigel, H. Mason, D. Telaar, J. Kluivers, R. van Dalen, C. W. Lau, L. Carlson, F. Granqvist, C. Vandevelde, et al. "Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications". In: arXiv preprint arXiv:2102.08503 (2021) (cit. on pp. 293, 297).

[Pil+25] K. Pillutla, J. Upadhyay, C. A. Choquette-Choo, K. Dvijotham, A. Ganesh, M. Henzinger, J. Katz, R. McKenna, H. B. McMahan, K. Rush, et al. "Correlated Noise Mechanisms for Differentially Private Learning". In: arXiv preprint arXiv:2506.08201 (2025) (cit. on p. 304).

[Ram+20] S. Ramaswamy, O. Thakkar, R. Mathews, G. Andrew, H. B. McMahan, and F. Beaufays. "Training production language models without memorizing user data". In: arXiv preprint arXiv:2009.10031 (2020) (cit. on pp. 306, 307).

[RM20] D. Ramage and S. Mazzocchi. Federated Analytics: Collaborative Data Science without Data Collection. https://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html. 2020 (cit. on pp. 288, 301, 307).

[SCS13] S. Song, K. Chaudhuri, and A. D. Sarwate. "Stochastic gradient descent with differentially private updates". In: 2013 IEEE Global Conference on Signal and Information Processing. IEEE. 2013, pp. 245–248 (cit. on p. 301).

[Sto+22] B. Stojkovic, J. Woodbridge, Z. Fang, J. Cai, A. Petrov, S. Iyer, D. Huang, P. Yau, A. S. Kumar, H. Jawa, et al. "Applied federated learning: Architectural design for robust and efficient learning in privacy aware settings". In: arXiv preprint arXiv:2206.00807 (2022) (cit. on p. 294).

[Sub+17] P. Subramanyan, R. Sinha, I. Lebedev, S. Devadas, and S. A. Seshia. "A formal foundation for secure remote execution of enclaves". In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017, pp. 2435–2450 (cit. on p. 300).

[VR25] T. Van Overveldt and D. Ramage. Discovering new words with confidential federated analytics. https://research.google/blog/discovering-new-words-with-confidential-federated-analytics. 2025 (cit. on p. 306).

[Xu+23] Z. Xu, Y. Zhang, G. Andrew, C. Choquette, P. Kairouz, B. McMahan, J. Rosenstock, and Y. Zhang. Federated Learning of Gboard Language Models with Differential Privacy. 2023 (cit. on pp. 294, 301, 304).

[Zha+23] Y. Zhang, D. Ramage, Z. Xu, Y. Zhang, S. Zhai, and P. Kairouz. "Private Federated Learning in Gboard". In: arXiv preprint arXiv:2306.14793 (2023) (cit. on p. 301).

[Zhu+20] W. Zhu, P. Kairouz, B. McMahan, H. Sun, and W. Li. "Federated heavy hitters discovery with differential privacy". In: International Conference on Artificial Intelligence and Statistics. PMLR. 2020, pp. 3837–3847 (cit. on p. 306).