A Hassle-free Algorithm for Strong Differential Privacy in Federated Learning Systems

H. Brendan McMahan, Zheng Xu, Yanxiang Zhang
Google
Correspondence: {mcmahan, xuzheng, zhangyx}google.com

Abstract

The state-of-the-art for training on-device language models for mobile keyboard applications combines federated learning (FL) with differential privacy (DP) via the DP-Follow-the-Regularized-Leader (DP-FTRL) algorithm. Two variants of DP-FTRL are used in practice, tree aggregation and matrix factorization. However, tree aggregation suffers from significantly suboptimal privacy/utility tradeoffs, while matrix mechanisms require expensive optimization parameterized by hard-to-estimate-in-advance constants, and high runtime memory costs. This paper extends the recently introduced Buffered Linear Toeplitz (BLT) mechanism to multi-participation scenarios. Our BLT-DP-FTRL maintains the ease-of-use advantages of tree aggregation, while essentially matching matrix factorization in terms of utility and privacy. We evaluate BLT-DP-FTRL on the StackOverflow dataset, serving as a re-producible simulation benchmark, and across four on-device language model tasks in a production FL system. Our empirical results highlight the advantages of the BLT mechanism and elevate the practicality and effectiveness of DP in real-world scenarios.

1 Introduction

Language models (LMs) that can predict the next word for input text are a powerful tool for many applications. In mobile keyboard applications, LMs are deployed on device to support various features (e.g., auto correction, smart completion and suggestion, and next word prediction) to improve users' typing experience. On-device LMs are typically small (less than ten million parameters) due to latency requirement and limited on-device resources. Their performance can be significantly improved by training from user data (Hard et al., 2018; Xu et al., 2023); recent work (Wang et al., 2023; Wu et al., 2024) shows the necessity of training on user data to achieve high utility even when we can access large-scale web data and pre-trained large LMs with billions of parameters.

As mobile-keyboard user data can be highly privacy sensitive, differential privacy (DP) (Dwork et al., 2006, 2014) and federated learning (FL) (McMahan et al., 2017a; Kairouz et al., 2019) have emerged as best practices for such models. DP provides a mathematical formulation to upper-bound the memorization of an individual's information in model training. FL minimizes data exposure by aggregating focused model updates from decentralized data stored only on user devices. DP and FL are combined when training on-device language models in production mobile keyboard applications (Xu et al., 2023). Applying DP in a production cross-device FL system is challenging as many DP algorithms require specific pattern of sampling training data to achieve strong privacy-utility trade-off. However, a cross-device FL system has limited control of sampling as clients can only participate in training when local criteria (e.g., charging, idle, and connected to an unmetered network) are satisfied (Bonawitz et al., 2019; Huba et al., 2022). Recently, DP-Follow-the-Regularized-Leader (DP-FTRL) algorithms (Kairouz et al., 2021; Choquette-Choo et al., 2023) have achieved superior privacy-utility trade-off with simpler client participation requirements, and are used in practice in FL systems (Xu et al., 2023; Zhang et al., 2023).

Instead of requiring uniform or Poisson sampling of devices as in previous work (Abadi et al., 2016; McMahan et al., 2017b), DP-FTRL uses minimum separation (min-sep) to characterizes the participation pattern. Min-sep is the smallest number of rounds between the consecutive participation of a client, and smaller min-sep necessitates adding more noise to achieve a desired DP guarantee. Min-sep is enforced in the FL system by implementing a timer on each device so that a device only becomes eligible for training if a certain period of

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 842–865

time (e.g., three days) has passed since their last participation. DP-FTRL algorithms leverage correlated noise mechanisms such as tree aggregation (TREEAGG) (Kairouz et al., 2021) or matrix factorization (MF) (Choquette-Choo et al., 2023) with the client participation pattern in FL. The banded MF (BANDMF) mechanism pre-computes matrices to generate correlated noise from independent noise to achieve stronger DP guarantees than the TREEAGG mechanism. BANDMF is superior when the number of rounds and min-sep can be (accurately) estimated before training to optimize matrices. However, min-sep is only known after training with time-based separation as many system factors may potentially affect training time [1]. Furthermore, BANDMF consumes more memory for noise generation, and hence is used less often than TREEAGG in practice.

In this work, we focus on the challenges of achieving strong DP guarantees in training production LMs in a cross-device FL system. We discuss how we extend the recent theoretical advancements of the Buffered Linear Toeplitz (BLT) mechanism from single participation (Dvijotham et al., 2024) to multi-participation scenarios, and adapt BLT to DP-FTRL. We apply BLT-DP-FTRL to FL in practice, demonstrating its advantages in flexibility, ease of use, and privacy-utility trade-offs. The BLT-DP-FTRL algorithm offers flexibility in handling varying numbers of training rounds, and robustness to a wide range of min separation between user participations. Furthermore, BLT-DP-FTRL simplifies the correlated noise generation process and reduces memory by exploiting the parameterization of the Toeplitz matrices. We empirically evaluate BLT-DP-FTRL on the StackOverflow benchmark dataset and across four on-device LM training tasks in a production FL system. Our BLT-DP-FTRL achieves better privacy-utility trade-off compared to the widely used TREEAGG mechanism, and comparable results compared to the state-of-the-art BANDMF mechanism. BLT-DP-FTRL exhibits desirable robustness properties in practice, offering a practical and effective solution for achieving strong DP in real-world FL systems.

[1] Despite being more complicated in FL systems, it is possible to enforce round-based separation so that a device only becomes eligible for training if min-sep rounds has passed since their last participation. However, it is still challenging to pre-specify min-sep before training due to the dynamics of client availability and population size. If the target min-sep is too large, training might halt because of lacking eligible devices. If the target min-sep is too small, the MF mechanism is not optimal for the correlated noise generation.

2 (BLT-)DP-FTRL for Private Learning

2.1 Background

We use (ϵ, δ)-DP (Dwork et al., 2006, 2014) and ρ-zCDP (zero-Concentrated DP) (Bun and Steinke, 2016) to quantify the privacy protection: smaller ϵ (ρ) correspond to stronger DP guarantees. A formal definition and more discussion are in App. A.1.

FL with DP We apply the generalized Federated Averaging (FedAvg) algorithm (McMahan et al., 2017a; Wang et al., 2021), as shown in Alg. 1 of App. A. FedAvg is the most common algorithm in cross-device FL systems. In a training round t of total n rounds, the server broadcasts a global model y^t to a subset of clients; each client i then updates their local model y_i by SGD, and sends back the model delta; the model deltas are aggregated and used as a pseudo gradient on the server to update the global model. DP is achieved by clipping the l_2 norm of the model delta to control the sensitivity (contribution of each device), and then adding noise to the aggregated deltas on the server.

While our primary focus is federated learning with decentralized data in this paper, Alg. 1 can also be applied in datacenter to achieve user-level DP (Xu et al., 2022; Chua et al., 2024; Charles et al., 2024). When using only one batch of a single sample for gradient computation in the ClientUpdate function and TREEAGG for correlated noise, Alg. 1 coincides with the DP-FTRL algorithm described in (Kairouz et al., 2021). The DP guarantee is determined by noise calibrated to sensitivity, which depends on clip norm noise multiplier σ, the correlated noise mechanism, total number of rounds T, and client participation pattern (min-sep b). Clip norm ζ and clip norm noise multiplier σ are used as algorithmic hyperparameters, similar to independent noise mechanism (e.g., DP-SGD/DP-FedAvg (Abadi et al., 2016; McMahan et al., 2018)). However, instead of directly applying independent Gaussian noise of standard deviation $\sigma\zeta$, correlated noise are generated to privatize model updates.

MF for DP-FTRL DP-FTRL (Kairouz et al., 2021) adds correlated noise to achieve strong privacy-utility trade-offs, observing that privatizing the prefix sum of model updates are essential for privatizing the training process. The intuition of privatizing prefix sum is easier to understand when

server optimizer is SGD, as the iterative process of per-round updates is equivalent to updating with prefix sum, i.e.,

$$y^t = y^{t-1} - \eta_s \Delta^t = y^{-1} - \eta_s \sum_{j=0}^{t} \Delta^j.$$

We can write similar formulation for additional linear operation in optimization, such as momentum in SGD. In practice, it is often easier to get privatized per-round update by (conceptually) subtracting the privatized prefix sum from two consecutive rounds, and use the privatized update $\tilde{\Delta}^t$ in various server optimizers, guaranteed by the post-processing property of DP.

We represent the model updates for n rounds as a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, where each row is the sum of clipped updates (i.e., $\boldsymbol{X}_{t,:} := \sum_{i \in \mathcal{Q}^t} \Delta_i^t \in \mathbb{R}^m$ from Alg. 1), we aim to privatize $\boldsymbol{AX}$, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is a lower triangular matrix of ones, i.e., $\boldsymbol{A}_{i,j} = 1, \forall i \leq j$ and $\boldsymbol{A}_{i,j} = 0, \forall i > j$. Given the privatized prefix sum $\widetilde{AX}$, the privatized model update is $\tilde{\Delta}^t \leftarrow \widetilde{\boldsymbol{AX}}_{t,:} - \widetilde{\boldsymbol{AX}}_{t-1,:}$, and Alg. 1 is privatized because of the post-processing property of DP. Kairouz et al. (2021) adopts the TreeAgg mechanism to privatize $\boldsymbol{AX}$. Recent work suggest a general matrix factorization framework (Choquette-Choo et al., 2023) can be used to achieve even stronger privacy-utility trade-offs, and both TreeAgg-DP-FTRL and the popular DP-SGD algorithm (Abadi et al., 2016) are special cases in the MF-DP-FTRL framework. MF mechanism considers the factorization of $\boldsymbol{A} = \boldsymbol{BC}$ and privatizes $\boldsymbol{CX}$ by adding independent noise $\boldsymbol{Z} \in \mathbb{R}^{n \times m}$ with standard deviation $\sigma\zeta$. We can use the (pseudo-)inverse of the $\boldsymbol{C}$ matrix to generate the correlated noise in the streaming setting (Choquette-Choo et al., 2023),

$$\widetilde{\boldsymbol{AX}} = \boldsymbol{B}(\boldsymbol{CX} + \zeta \boldsymbol{Z}) = \boldsymbol{AX} + \zeta \boldsymbol{C}^{-1}\boldsymbol{Z}$$
$$\Rightarrow \tilde{\Delta}^t = \Delta^t + (\zeta \boldsymbol{C}^{-1}\boldsymbol{Z})_{t,:} \quad (1)$$

Eq. (1) also suggests the alternative interpretation of correlated noise in DP-FTRL: at round t, the noise added in previous rounds can be cancelled when $\boldsymbol{C}^{-1}_{t,:}$ is negative.

TreeAgg can be written in MF form, and the stronger variance reduction variant (TreeAgg-Full) (Honaker, 2015) is equivalent to setting $\boldsymbol{B} = \boldsymbol{AC}^{-1} \in \mathbb{R}^{2^{l-1} \times (2^l - 1)}$ by computing the Moore-Penrose pseudoinverse of $\boldsymbol{C}$ (Denisov et al., 2022). However, the MF-basedTreeAgg-Full does not have a memory-efficient implementation, and consumes nm memory. In this paper, we consider memory-efficient TreeAgg (Kairouz et al., 2021) that is widely used in industry (Xu et al., 2023), and TreeAgg-Full (Denisov et al., 2022) that achieves better privacy-utility trade-off but less memory and computation efficient. BandMF (Choquette-Choo et al., 2023) is the state-of-the-art for FL, which optimizes matrices with estimated min-sep bands. More related work with detailed discussion are in App. A.3.

2.2 BLT Mechanisms in DP-FTRL

We now consider lower-triangular Toeplitz matrix in MF mechanism, i.e., $\boldsymbol{C} := \text{LtToep}(c) \in \mathbb{R}^{n \times n}$ where $\boldsymbol{C}_{i,j} = c_{i-j}, \forall i \leq j$ otherwise $C_{i,j} = 0$. Buffered-linear Toeplitz (BLT) matrices (Dvijotham et al., 2024) parameterize $\boldsymbol{C}$ by $\theta \in (0, 1]^d$ (the "buffer decay" parameters) and non-negative $\omega \in \mathbb{R}_+^d$ (the "output scale" parameters), where the Toeplitz coefficients are given by

$$c_i = \begin{cases} 1 & i = 0 \\ \sum_{j \in [d]} \omega_j \theta_j^{i-1} & i > 0. \end{cases} \quad (2)$$

The BLT(ω, θ) matrices have many useful properties, most importantly for our purposes: (1) Streaming multiplication by $\boldsymbol{C}$ ($\boldsymbol{Z} = \boldsymbol{C}\hat{\boldsymbol{Z}}$ for $\boldsymbol{Z}, \hat{\boldsymbol{Z}} \in \mathbb{R}^{n \times m}$) can be computed efficiently using only $\mathcal{O}(dm)$ memory and $\mathcal{O}(dm)$ time per round t, without fully materializing $\boldsymbol{C}$, $\boldsymbol{Z}$, or $\hat{\boldsymbol{Z}}$. Hence $\boldsymbol{C}$ is referred as a d-buffer BLT. (2) The inverse of a d-buffer BLT ($\boldsymbol{C} = \text{BLT}(\omega, \theta)$) is another d-buffer BLT ($\boldsymbol{C}^{-1} = \text{BLT}(\hat{\omega}, \hat{\theta})$), and we can efficiently compute Toeplitz coefficients of $\boldsymbol{C}^{-1}$ using Eq. (2) applied to $(\hat{\omega}, \hat{\theta})$. We now derive the correlated noise generation schema for $\boldsymbol{C}^{-1}\boldsymbol{Z}$ in Eq. (1) based on the BLT properties. We can first derive the BLT parameters $(\hat{\theta}, \hat{\omega})$ of $\boldsymbol{C}^{-1}$ such that $\boldsymbol{C}^{-1} = \text{BLT}(\hat{\theta}, \hat{\omega})$, and then generate the correlated noise based on $(\hat{\theta}, \hat{\omega})$ in streaming setting. However, we show a simpler alternative that directly uses the BLT parameters (θ, ω) to generate streaming correlated noise $\hat{\boldsymbol{Z}}$.

Applying the parameterization in Eq. (2) to (Dvijotham et al., 2024, Alg 1), and initializing buffers $\boldsymbol{S}_{-1} \leftarrow \boldsymbol{0} \in \mathbb{R}^{d \times m}$, we can efficiently compute $\boldsymbol{Z}_{t,:}$ from $\hat{\boldsymbol{Z}}_{t,:}$ in the streaming setting, $\boldsymbol{Z}_{t,:} = \hat{\boldsymbol{Z}}_{t,:} + \omega^T \boldsymbol{S}_{t-1}$, $\boldsymbol{S}_t = \text{diag}(\theta)\boldsymbol{S}_{t-1} + \boldsymbol{1}_d \hat{\boldsymbol{Z}}_{t,:}$. We rearrange the update equations to get $\hat{\boldsymbol{Z}}$ from

$\boldsymbol{Z}$ and $\boldsymbol{S}$,

$$\hat{\boldsymbol{Z}}_{t,:} = \boldsymbol{Z}_{t,:} - \omega^T \boldsymbol{S}_{t-1},$$
$$\boldsymbol{S}_t = \mathrm{diag}(\theta)\boldsymbol{S}_{t-1} + \mathbf{1}_d \hat{\boldsymbol{Z}}_{t,:}. \tag{3}$$

To efficiently generate correlated noise $\hat{\boldsymbol{Z}}_{t,:}$ at round t, we only need to materialize the independent noise $\boldsymbol{Z}_{t,:} \in \mathbb{R}^{1\times m}$ and use the buffers $\boldsymbol{S}_{t-1} \in \mathbb{R}^{d\times m}$ in Eq. (3). The efficient correlated noise generation parameterized by BLT parameters (θ, ω) for $\boldsymbol{C}^{-1}$ (instead of $\boldsymbol{C}$) did not appear in Dvijotham et al. (2024) and is new to this work. Eq. (3) intuitively shows the noise cancellation view of correlated noise, where the previous noises are tracked in the states decaying with $\theta \in (0, 1]$, and then subtracted in the current round after scaling with ω.

For completeness, we provide the streaming multiplication algorithm of $\boldsymbol{Z} = \boldsymbol{C}\hat{\boldsymbol{Z}}$ and $\hat{\boldsymbol{Z}} = C^{-1}\boldsymbol{Z}$ for $\boldsymbol{C} = \mathrm{BLT}(\theta, \omega)$ in Alg. 2 and Alg. 3, respectively. Alg. 2 is a direct application of (Dvijotham et al., 2024, Alg 1) and only used to derive Alg. 3, which is our streaming algorithm for generating correlated noise with BLTs using only dm memory. Finally, we apply the streaming correlated noise $\hat{\boldsymbol{Z}}$ by BLT from Alg. 3 (using Eq. (3)) in Alg. 1 for the BLT-DP-FTRL algorithm, i.e.,

$$\tilde{\Delta}^t \leftarrow \sum_{i\in\mathcal{Q}^t} \Delta_i^t + \hat{\boldsymbol{Z}}_{t,:}. \tag{4}$$

3 Multi-participation BLTs

We study how to optimize for the BLT parameters $\theta \in \mathbb{R}^d$ and $\omega \in \mathbb{R}^d$ in Eq. (3) for the BLT-DP-FTRL algorithm, and account for DP guarantees. Particularly, we generalize the BLT optimization and DP accounting in (Dvijotham et al., 2024) from single participation to multiple participations

3.1 Sensitivity Under Multiple Participations

We provide additional background about multi-participation sensitivity definition, computation and usage in DP in App. C.2, and only discussing the main results in this section. We further derive a lower bound for sensitivity in App. C.3 used for TreeAgg in simulation experiments in Sec. 4.

Let $\boldsymbol{C} = \mathrm{LtToep}(c) \in \mathbb{R}^{n\times n}$ be a lower-triangular Toeplitz matrix defined by the sequence of Toeplitz coefficients $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{R}^{\mathbb{N}}$ as in Sec. 2.2. We assume $c_i \geq 0$ and $\mathbf{c}$ is non-increasing, and consider the sensitivity of $\boldsymbol{C}$. Let $\mathbf{c}_i = \boldsymbol{C}_{:,i}$ be the ith column of $\boldsymbol{C}$, so $\mathbf{c}_0 = \mathbf{c}$ and generally

$$\mathbf{c}_j = \big(\underbrace{0, 0, \ldots, 0}_{j \text{ zeros}}, c_0, c_1, \ldots c_{n-j-1}\big) \in \mathbb{R}^n.$$

The sensitivity of general Toeplitz matrices with decaying coefficients is recently discussed in Kalinin and Lampert (2024, Thm. 2), which we restate in Thm. 3.1 with our notation. The participation pattern $\pi^\star$ simply puts the k participations as early as possible, with each participation separated by exactly b. This sensitivity computation is important for both DP accounting and optimizing for BLT parameters in Sec. 3.3.

Theorem 3.1. *Given a Toeplitz strategy matrix $\boldsymbol{C} = \mathit{LtToep}(\mathbf{c}) \in \mathbb{R}^{n\times n}$ with $\mathbf{c}$ non-increasing and non-negative. Then, $\mathrm{sens}_{\Pi_b}(\boldsymbol{C})$ can be computed in time $\mathcal{O}(kn)$ as*

$$\mathrm{sens}_{\boldsymbol{N}_\Pi}(\boldsymbol{C}) = \|\boldsymbol{C}u(\pi^\star)\|_2$$

where $\pi^\star$ is given by

$$\pi^\star = (0, b, 2b, \ldots, (k-1)b). \tag{5}$$

3.2 Analytical Utility as Objective

While our end goal is good learning performance (as measured by held-out test set accuracy), we can estimate the utility of a matrix mechanism for DP-FTRL by quantifying the error it introduces into prefix sum estimates. The total noise introduced by the DP mechanism into prefix sum estimates in Eq. (1) will be $\boldsymbol{BZ} = \widetilde{\boldsymbol{AX}} - \boldsymbol{AX}$ where $\boldsymbol{Z} \in \mathbb{R}^{n\times m}$ is IID Gaussian noise with σ determined according to the desired DP guarantee, and $\boldsymbol{B} = \boldsymbol{AC}^{-1}$. We consider two error metrics based on the standard deviation of the total noise added to the prefix sum estimates. The MaxError is the worst-case standard deviation in the estimate of any prefix sum, which can be computed as

$$\mathrm{MaxError}(\boldsymbol{B}) \coloneqq \max_{i\in[n]} \sqrt{\sum_{j\in[n]} \boldsymbol{B}_{i,j}^2};$$

similarly the root-mean-squared error over all iterations $i \in [n]$ is

$$\mathrm{RmsError}(\boldsymbol{B}) \coloneqq \sqrt{\sum_{i\in[n]}\sum_{j\in[n]} \boldsymbol{B}_{i,j}^2 / n}.$$

The standard deviation σ of the noise $\boldsymbol{Z}$ must scale linearly in the sensitivity of $\boldsymbol{C}$ to achieve a target DP guarantee, so our final measures of noise account for this:

$$\mathrm{MaxLoss}(\boldsymbol{B}, \boldsymbol{C}) \coloneqq \mathrm{MaxError}(\boldsymbol{B}) \cdot \mathrm{sens}_\Pi(\boldsymbol{C}) \tag{6}$$
$$\mathrm{RmsLoss}(\boldsymbol{B}, \boldsymbol{C}) \coloneqq \mathrm{RmsError}(\boldsymbol{B}) \cdot \mathrm{sens}_\Pi(\boldsymbol{C}) \tag{7}$$

Eqs. (6) and (7) measure the distribution of noise to approximate the privacy-utility trade-off as the loss, which do not depend on the noise multiplier $\alpha = \sigma / \operatorname{sens}_\Pi(\boldsymbol{C})$ that is directly used in accounting for the DP guarantees, as discussed in (Dvijotham et al., 2024, Introduction). For optimized $\boldsymbol{C}$ in matrix factorization mechanisms (e.g., TREEAGG, BANDMF, and BLT), specific DP guarantees are achieved by scaling α (and corresponding σ in Alg. 1). The total noise on the prefix sum BZ also scales MaxLoss and RmsLoss by α. Hence, without loss of generality, we use $RmsLoss$ and $MaxLoss$ to compare the optimality of different matrix factorization mechanisms, which is equivalent to assuming $\alpha = 1$ that corresponds to $(\epsilon = 5.3, \delta = 10^{-7})$-DP (for example).

Note we deviate from the definitions of (Dvijotham et al., 2024, arXiv v3), in order to distinguish the error introduced by $\boldsymbol{B}$ from the total loss (which is what we optimize and use to compare mechanisms), which incorporates the sensitivity of $\boldsymbol{C}$.

3.3 Optimizing Multi-participation BLTs

For the typical scale of $n < 10^5$ in FL systems, rather than deriving closed forms for sensitivity and error as in Dvijotham et al. (2024), we use an alternative approach that is flexible and simple to implement. Recall the properties of BLTs discussed in Sec. 2.2, we parameterize the optimization of the BLT by the pair $(\theta, \hat{\theta})$. Dvijotham et al. (2024, Lem. 5.2) implies given a pair $(\theta, \hat{\theta})$, there exist unique $(\omega, \hat{\omega})$ such the $\text{BLT}(\theta, \omega)^{-1} = \text{BLT}(\hat{\theta}, \hat{\omega})$, and we can compute ω and $\hat{\omega}$ in time $\mathcal{O}(d^2)$; this result is summarized below as Alg. 5. Thus, given a $(\hat{\theta}, \theta)$, we can efficiently compute the Toeplitz coefficients of $\boldsymbol{C}$ (using Eq. (2) applied to (θ, ω)) and $\boldsymbol{C}^{-1}$ (applying Eq. (2) to $(\hat{\theta}, \hat{\omega})$). From the Toeplitz coefficients of $\boldsymbol{C}$ we can then efficiently compute sensitivity using Thm. 3.1. RmsError can be computed efficiently from the Toepltiz coefficients of $\boldsymbol{C}^{-1}$ following the approach of McKenna (2024, Prop. 3.1), and a simple generalization of this approach applies to to MaxError as well. For completeness we summarize in the following proposition:

Proposition 3.2. *Let $\boldsymbol{C} = \mathit{LtToep}(c) \in \mathbb{R}^{n \times n}$ be a lower-triangular Toeplitz matrix defined by Toeplitz coefficients $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{R}^{\mathbb{N}}$. Then $\boldsymbol{C}^{-1}$ is also a lower-triangular toeplitz matrix; let $\boldsymbol{C}^{-1} = \mathit{LtToep}(\hat{c})$. Then, $\boldsymbol{B} \coloneqq \boldsymbol{A}\boldsymbol{C}^{-1} = \mathit{LtToep}(b)$ where $b_i = \sum_{j=0}^{i-1} \hat{c}_i$. Further, we can compute*

$$\text{MaxError}(\boldsymbol{B}) = \sqrt{\sum_{i \in [n]} b_i^2}$$

and

$$\text{RmsError}(\boldsymbol{B}) = \sqrt{\sum_{i \in [n]} (n-i) b_i^2 / n}.$$

4 Simulation Experiments

Mechanism	Test Accuracy $\epsilon = 2$	Test Accuracy $\epsilon = 8$	RMS Loss	Max Loss
BANDMF (band=342)	23.21	24.86	10.21	8.60
TREEAGG-FULL	22.54	24.47	14.98	12.47
BLT (nbuf=2,k=1)	22.37	24.64	11.80	11.15
BLT (nbuf=5,k=1)	22.40	24.63	11.40	10.87
BLT *(nbuf=2,k=6)	23.09	24.83	10.81	9.34
BLT *(nbuf=3,k=6)	23.13	24.87	10.79	9.33
BLT *(nbuf=4,k=6)	23.13	24.83	10.79	9.33
BLT *(nbuf=5,k=6)	23.07	24.84	10.79	9.33

Table 1: Comparing mechanisms in terms of test-set accuracy on the StackOverflow NWP task. All runs are based on $n = 2052$ rounds of training with $k = 6$ participations and min-sep $b = 342$. BLTs are optimized for MaxLoss. Results are visualized in Fig. 11 in App. G.

We run simulation experiments before applying our BLT-DP-FTRL algorithm in Sec. 2.2 to train production LMs. The BLT parameters (θ, ω) are optimized with our multi-participation approach in Sec. 3. We present private-utility trade-off on StackOverflow benchmark dataset in Sec. 4.1, and MaxLoss, RmsLoss across a range of scenarios in Sec. 4.2. We compare BLTs to both flexible TREEAGG (Kairouz et al., 2021) and state-of-the-art BANDMF (Choquette-Choo et al., 2023) (see Sec. 2 for more discussion). In the simulation experiments, we are *maximally generous* in evaluating TREEAGG mechanisms, considering the memory cost to be $\lceil \log_2 n \rceil$, while calculating RmsError and MaxError using the optimal TREEAGG-FULL (Denisov et al., 2022) without memory-efficient implementation, and use the lower bound of Remark C.1 to account for overly optimistic privacy-utility trade-off. Thus, in all cases we over-estimates the true performance of the binary tree, but nevertheless we show BLTs have superior performance in terms of both error and memory.

4.1 StackOverflow NWP Benchmark

We follow Choquette-Choo et al. (2023) for StackOverflow next word prediction (NWP) experiments, including all hyperparameter tuning, and vary only the DP mechanism to compare BANDMF,

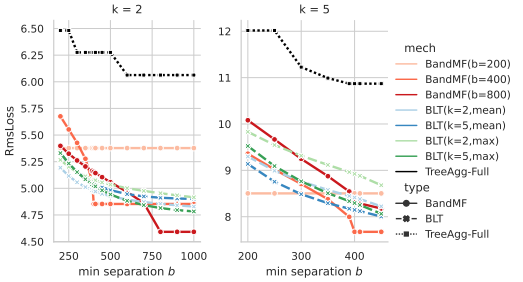


Figure 1: Comparison of mechanisms in terms of prefix-sum root-mean-squared error at a fixed privacy level. BLTs were optimized for either RmsLoss ("mean") or MaxLoss ("max"), and for either $k = 2$ or $k = 5$ participations at min-separation $b = 400$. BandMF matrices were optimized for $b \in \{200, 400, 800\}$ (the BandMF optimization does not depend on k, and previous work optimizes for RmsLoss). We also include TreeAgg-Full using the optimal ("full Honaker") decoding (for which a memory-efficient noise generation algorithm is unknown). **We observe that all the BLTs perform competitively with BandMF, and can outperform BandMF** when the min-separation differs significantly from the number of bands. For example, with $k = 2$ participations (left panel) our BLT($k = 2$,mean) (light blue) BLT outperforms BandMF($b = 400$) when min separation is less than 390 or greater than 700.

TreeAgg-Full, and BLTs. Results are summarized in Tab. 1. BandMF still achieves the highest performance, as in this scenario we train for a fixed known number of rounds, with an exactly known max participations $k = 6$ and min-separation $b = 342$. TreeAgg-Full and BLTs optimized for only $k = 1$ participation (Dvijotham et al., 2024) are noticeably worse, but our multi-participation-optimized BLTs are very competitive with BandMF with only 2 or 3 buffers (with a $171\times$ and $114\times$ reduction in runtime memory overhead). In the relatively large signal-to-noise ratio regime ($\epsilon = 8$), BLT* achieves comparable or even better learning accuracy though the RmsLoss (MaxLoss) is slightly worse.

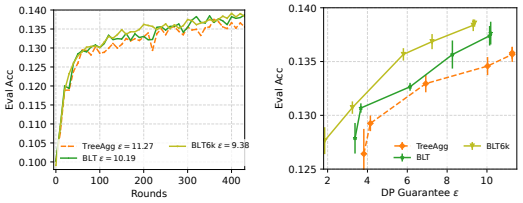
4.2 RmsLoss and MaxLoss Experiments

Comparing BLT to TreeAgg-Full and BandMF We further show that BLT is better than TreeAgg-Full, and more flexible than BandMF by computing RmsLoss (MaxLoss) in a wide range of scenarios. Because the BandMF mechanisms are optimized for RmsLoss, we compare on this metric in Fig. 1. However, in both our StackOverflow and Gboard experiments described subsequently, we deploy BLT mechanisms optimized for MaxError following (Dvijotham et al., 2024). For completeness, we provide Fig. 12 in App. G that compares the mechanisms on MaxLoss. We observe that *all the* BLTs *perform competitively with* BandMF*, and can outperform* BandMF when the min-sep differs significantly from the number of bands. For example, in Fig. 1, with $k = 2$ participations (left panel) our BLT($k = 2$,mean) (light blue) BLT outperforms BandMF($b = 400$) when min separation is less than 390 or greater than 700.

We provide more results on the robustness of BLTs to min-sep b, number of rounds n, and comparing with BandToep in App. E.

5 Production LMs for Mobile Keyboard

Production Setting Our BLT-DP-FTRL algorithm is flexible in min-sep (shown in Sec. 4.2), achieves competitive privacy-utility performance for relatively large signal-to-noise ratio (shown in Sec. 4.1), and saves computation and memory cost (shown in Tab. 3), which motivates the usage in production FL systems. We follow (Xu et al., 2023) for the production setting, and provide additional details including the configuration for baselines in App. G.1.



(a) NWP accuracy (b) Privacy-utility trade-off

Figure 2: NWP evaluation accuracy and the dervied privacy-utility trade-off curves for training the Portuguese LM in Portugal (pt-PT) with DP-FTRL in a FL system. Additional curves for es-ES, id-ID, and pt-BR are provided in Figs. 13 to 15. BLTs achieve better privacy-utility trade-off.

Main Results We summarize the privacy and utility results for es-ES, id-ID, pt-BR, and pt-PT LMs in Tab. 2, and show the privacy-utility curves for training pt-PT in Fig. 2. We provide additional curves for other LMs in Figs. 13 to 15 inApp. G, and discuss the observations here. (1) *Most of the models achieve DP guarantee of* $\epsilon < 10$ *with exception of* $\epsilon \sim 10$ *for pt-PT due to the challenge of small population; the pt-BR model trained with* BLT-*16.1 achieves* $\epsilon < 1$ *at round 2000.* DP guarantees of $\epsilon < 10$ is commonly used for machine learning, and $\epsilon < 1$ are considered strong guarantees (Ponomareva et al., 2023). To achieve single-digit DP guarantees in practice without sacrificing the utility, the production LMs are trained with large number of clients per round (known as report goal in FL systems). We use typical report goal 6.5K (Xu et al., 2023) for es-ES, id-ID and pt-BR,

LM	Rnds	Utility			Privacy		
		NWP(%)	WMR(-%)	WPM(+%)	Mech-σ/MinS/MaxP	zCDP	DP-ϵ
es-ES	1280	14.07 ± 0.06	-	-	BANDMF-1.411 / 296 / 4	0.29	4.82
		13.98 ± 0.11	0.38	0.13	BLT-7.379 / 300 / 4	0.16	3.46
id-ID	2350	5.80 ± 0.10	-	-	TREEAGG-7 / 437 / 5	0.94	9.29
		5.87 ± 0.04	0.09	0.07	BLT-7.379 / 447 / 5	0.20	3.93
pt-BR	2000	13.77 ± 0.36	-	-	BANDMF-4.523 / 2001 / 1	2.45e-2	1.32
		13.86 ± 0.25	-0.04	0.04	BLT-8.681 / 2001 / 1	2.23e-2	1.25
		13.96 ± 0.18	-0.13	0.18	BLT-16.1 / 1181 / 2	1.40e-2	0.98
pt-PT	430	13.58 ± 0.06	-	-	TREEAGG-7 / 91 / 4	1.33	11.27
	430	13.76 ± 0.11	0.38	-0.24	BLT-3.12 / 92 / 4	1.11	10.19
	320	13.66 ± 0.04	0.07	-0.12	BLT-5.5 / 49 / 6	0.75	8.19

Table 2: Privacy and utility of production LMs. Utility are measured by NWP accuracy averaged between $r \pm 50$ rounds for round r ($r \pm 10$ for pt-PT), and the relative WMR decrease and WPM increase in A/B test; privacy shows the key parameters and corresponding DP guarantees, and smaller DP guarantees represent stronger protection; DP-ϵ is accounted for small $\delta = 10^{-10}$; estimated population sizes are es-ES (4.21M), id-ID (8.9M), pt-BR (16.6M), and pt-PT (0.83M). We run additional experiments on pt-BR and pt-PT with larger noise multipliers linearly scales with larger report goal for the BLT mechanism.

and use a smaller report goal 3K for pt-PT with a smaller population. We additionally run BLT-DP-FTRL with larger report goal and linearly scale up the noise multiplier to keep the signal-to-noise ratio for utility: BLT-16.1 with report goal 12K for pt-BR and BLT-5.5 with report goal 6K for pt-PT. The resulting min-seps in pt-BR and pt-PT almost halved when the report goals are increased. We use noise multiplier 7 for TREEAGG, which is determined by StackOverflow simulation experiments (Xu et al., 2023). (2) BLT *achieves better privacy-utility trade-off compared to* TREEAGG *and* BANDMF. BLT achieves comparable and slightly better NWP accuracy for the models in Tab. 2 and Fig. 2, and stronger DP guarantees. The model performance are further verified by A/B test in the production application comparing BLT models to baseline models for WMR and WPM, where we target on improved or neutral utilities. The advantage of BLT in privacy-utility trade-off is clearly demonstrated in Fig. 15, and BLT is better than not only TREEAGG, but also BANDMF across the production LM training. The practical min-sep can be quite different from the estimated min-seps for optimizing BANDMF and BLT matrices, e.g., $\sim$300 compared to 400 for es-ES, and 2000+ compared to 1000 for pt-BR. As BLT is more flexible on min-sep estimation, the challenge of reliably estimating min-sep resulting in BLT achieving even stronger privacy-utility trade-offs than BANDMF in the production LMs training.

Extrapolation We extrapolate the results for production setting by assuming linearly increase report goal and noise multiplier, and changing min-sep will not change the utility, and hence we can study the effect on DP without actually training the model. We provide results and detailed discussion in App. G.2, which further demonstrate the advantages of BLT-DP-FTRL.

6 Concluding Remarks

This work addresses the critical challenge of achieving strong DP in FL for on-device LMs. We have successfully extended the BLT mechanism to multi-participation scenarios and integrated it into the DP-FTRL framework. Our BLT-DP-FTRL algorithm demonstrates superior privacy-utility trade-offs compared to the widely-used TREEAGG mechanism while maintaining its ease of use. Furthermore, it rivals the state-of-the-art BANDMF mechanism in performance, yet without the associated complexities and high memory costs. Through extensive empirical evaluations on both a benchmark dataset and real-world production tasks, we have showcased the practicality and effectiveness of BLT-DP-FTRL, paving the way for its broader adoption.

The empirical results in this paper primarily focus on the cross-device FL setting where privacy amplification by sampling is challenging in practice. The discussions (e.g. Tab. 3) can also be applied to centralized setting for user-level DP or example-level DP. In centralized setting, BANDMF (Choquette-Choo et al., 2023) with amplification can achieve better privacy-utility trade-off measured by RmsLoss among the mentioned mechanisms, when number of rounds n and model dimension m is not too large for optimizing and applying the mechanism. When n and m are large, BLT and BANDTOEP (McKenna, 2024) (similarly, BANDFHU (Kalinin and Lampert, 2024)) can both be applied, where BLT has less optimization cost for very large n (shown in Fig. 3), while BANDTOEP can apply existing amplification by sampling.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Joel Daniel Andersson and Rasmus Pagh. 2024. A smooth binary mechanism for efficient private continual observation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NeurIPS, Red Hook, NY, USA. Curran Associates Inc.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer.

Zachary Charles, Arun Ganesh, Ryan McKenna, H Brendan McMahan, Nicole Mitchell, Krishna Pillutla, and Keith Rush. 2024. Fine-tuning large language models with user-level differential privacy. *arXiv preprint arXiv:2407.07737*.

Christopher A. Choquette-Choo, Arun Ganesh, Ryan McKenna, H. Brendan McMahan, Keith Rush, Abhradeep Thakurta, and Zheng Xu. 2023. (Amplified) Banded Matrix Factorization: A unified approach to private training. In *NeurIPS*.

Christopher A. Choquette-Choo, Hugh Brendan McMahan, J. Keith Rush, and Abhradeep Guha Thakurta. 2023. Multi-epoch matrix factorization mechanisms for private machine learning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 5924–5963. PMLR.

Lynn Chua, Badih Ghazi, Yangsibo Huang, Pritish Kamath, Daogao Liu, Pasin Manurangsi, Amer Sinha, and Chiyuan Zhang. 2024. Mind the privacy unit! user-level differential privacy for language model fine-tuning. *arXiv preprint arXiv:2406.14322*.

Sergey Denisov, Brendan McMahan, Keith Rush, Adam Smith, and Abhradeep Guha Thakurta. 2022. Improved differential privacy for sgd via optimal private linear operators on adaptive streams. *arXiv preprint*.

Vadym Doroshenko, Badih Ghazi, Pritish Kamath, Ravi Kumar, and Pasin Manurangsi. 2022. Connect the dots: Tighter discrete approximations of privacy loss distributions. *arXiv preprint arXiv:2207.04380*.

DP Team. 2022. Google's differential privacy libraries. https://github.com/google/differential-privacy.

Krishnamurthy Dvijotham, H Brendan McMahan, Krishna Pillutla, Thomas Steinke, Abhradeep Thakurta, et al. 2024. Efficient and near-optimal noise generation for streaming differential privacy. *arXiv preprint arXiv:2404.16706*.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

Hendrik Fichtenberger, Monika Henzinger, and Jalaj Upadhyay. 2022. Constant matters: Fine-grained complexity of differentially private continual observation. *arXiv preprint arXiv:2202.11205*.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

James Honaker. 2015. Efficient use of differentially private binary trees. *Theory and Practice of Differential Privacy (TPDP 2015), London, UK*.

Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, et al. 2022. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems*, 4:814–832.

Peter Kairouz, Brendan Mcmahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. 2021. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning (ICML)*, pages 5213–5225.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kaylee Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal,

Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2019. Advances and open problems in federated learning. *CoRR*, abs/1912.04977.

Nikita Kalinin and Christoph Lampert. 2024. Banded square root matrix factorization for differentially private model training. *Preprint*, arXiv:2405.13763.

Ryan McKenna. 2024. Scaling up the banded matrix factorization mechanism for differentially private ml. *Preprint*, arXiv:2405.15913.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017a. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, pages 1273–1282. PMLR.

Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning differentially private recurrent language models. In *International Conference on Learning Representations (ICLR)*.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017b. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*.

Natalia Ponomareva, Hussein Hazimeh, Alex Kurakin, Zheng Xu, Carson Denison, H. Brendan McMahan, Sergei Vassilvitskii, Steve Chien, and Abhradeep Thakurta. 2023. How to dp-fy ml: A practical guide to machine learning with differential privacy. *Preprint*, arXiv:2303.00654.

Boxin Wang, Yibo Jacky Zhang, Yuan Cao, Bo Li, H Brendan McMahan, Sewoong Oh, Zheng Xu, and Manzil Zaheer. 2023. Can public large language models help private cross-device federated learning? *arXiv preprint arXiv:2305.12132*.

Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Blaise Aguera y Arcas, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, et al. 2021. A field guide to federated optimization. *arXiv:2107.06917*.

Shanshan Wu, Zheng Xu, Yanxiang Zhang, Yuanbo Zhang, and Daniel Ramage. 2024. Prompt public large language models to synthesize data for private on-device applications. *arXiv preprint arXiv:2404.04360*.

Zheng Xu, Maxwell Collins, Yuxiao Wang, Liviu Panait, Sewoong Oh, Sean Augenstein, Ting Liu, Florian Schroff, and H Brendan McMahan. 2022. Learning to generate image embeddings with user-level differential privacy. *arXiv preprint arXiv:2211.10844*.

Zheng Xu and Yanxiang Zhang. 2024. Advances in private training for production on-device language models.

Zheng Xu, Yanxiang Zhang, Galen Andrew, Christopher A Choquette-Choo, Peter Kairouz, H Brendan McMahan, Jesse Rosenstock, and Yuanbo Zhang. 2023. Federated learning of gboard language models with differential privacy. *ACL Industry*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Yuanbo Zhang, Daniel Ramage, Zheng Xu, Yanxiang Zhang, Shumin Zhai, and Peter Kairouz. 2023. Private federated learning in gboard. *arXiv preprint arXiv:2306.14793*.

A Additional Background on Federated Learning (FL) with Differential Privacy (DP)

A.1 DP Formulation

We present the definition of (ϵ, δ)-DP (Dwork et al., 2006, 2014) to quantify the privacy protection.

Definition A.1 ((ϵ, δ)-Differential Privacy)**.** *A randomized algorithm $\mathcal{M}$ satisfies (ϵ, δ)-DP for $\mathbb{D}$ if for any two neighboring datasets $\mathbb{D}$, $\mathbb{D}'$ and for all $\mathcal{S} \subset Range(\mathcal{M})$:*

$$\Pr[\mathcal{M}(\mathbb{D}) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{M}(\mathbb{D}') \in \mathcal{S}] + \delta.$$

Smaller (ϵ, δ) values suggest stronger DP guarantees, and we often measure ϵ at a fixed small $\delta = 10^{-10}$. DP-FTRL also uses an alternative definition, ρ-zCDP (zero-Concentrated DP) (Bun and Steinke, 2016) designed for Gaussian mechanism, and smaller ρ suggests stronger DP guarantees. We use PLD (privacy Loss Distributions) accounting (Doroshenko et al., 2022; DP Team, 2022) to convert ρ-zCDP to (ϵ, δ) DP. When applying DP in FL, the neighboring datasets $\mathbb{D}$, $\mathbb{D}'$ are defined by zeroing out the contribution of all data on a user device. More discussions on neighboring dataset for the streaming setting in learning, and connection to DP guarantees are provided in Sec. 3.1.

A.2 DP-FTRL for DP FL algorithm

Algorithm 1 FedAvg (McMahan et al., 2018) with DP-FTRL (Kairouz et al., 2021) for DP FL

input : clients per round m, learning rate on client η_c and on server η_s, momentum $\beta = 0.9$, total number of rounds T, clip norm ζ, clip norm noise multiplier σ,

Initialize model y^{-1} with pretraining
Initialize server optimizer state $\mathcal{P}$
Initialize correlated noise state $\mathcal{S}$ with $\sigma\zeta$
for each round $t = 0, 1, 2, \ldots, n-1$ **do**
  $\mathcal{Q}^t \leftarrow$ (at least m users that did not participate in the previous b rounds)
  for each user $i \in \mathcal{Q}^t$ **in parallel do**
    $\Delta_i^t \leftarrow \text{ClientUpdate}(i, y^{t-1})$
  $\tilde{\Delta}^t, \mathcal{S} \leftarrow \text{AddCorrNoise}(\mathcal{S}, \sum_{i \in \mathcal{Q}^t} \Delta_i^t)$
  $y^t, \mathcal{P} \leftarrow \text{ServerOpt}(y^{t-1}, \frac{1}{m}\tilde{\Delta}^t, \eta_s, \beta, \mathcal{P})$

function ClientUpdate(i, x_i)
  $\mathcal{G} \leftarrow$ (batches of user i's local data)
  for batch $g \in \mathcal{G}$ **do**
    $y_i \leftarrow y_i - \eta_c \nabla \ell(y_i; g)$
  $\Delta \leftarrow y_i - y_i^{(0)}$
  $\Delta' \leftarrow \Delta \cdot \min\left(1, \frac{\zeta}{||\Delta||}\right)$
  return Δ'

A.3 TreeAgg and BandMF in DP-FTRL

TreeAgg can be written in MF form by recursively defining $\boldsymbol{C}^l \in \{0,1\}^{(2^l-1)\times 2^{l-1}}$ $l = \lceil \log_2 n \rceil$, as $\boldsymbol{C}^1 = [1]$, $\boldsymbol{C}^l = [[\boldsymbol{C}^{l-1}, \boldsymbol{0}], [\boldsymbol{0}, \boldsymbol{C}^{l-1}], [\boldsymbol{1}, \boldsymbol{1}]]$, where each row $\boldsymbol{C}_{i,:}$ represents a node in the binary tree and the ones in $\boldsymbol{C}_{i,:}$ represent the leaf nodes for a subtree. After adding noise $\boldsymbol{Z}$ to every tree node, vanilla TreeAgg uses matrix $\boldsymbol{B}$ to selects and aggregates tree nodes to privatize the prefix sum, i.e., $\boldsymbol{B} \in \{0,1\}^{2^{l-1}\times(2^l-1)}$ has $\boldsymbol{B}_{i,j} = 1, \forall j = 2^{k+1} - 1, k \in \kappa, i = \sum_{k\in\kappa} 2^k$, otherwise $\boldsymbol{B}_{i,j} = 0$. Several schemes improve vanilla binary TreeAgg for prefix sums appear in the literature. Kairouz et al. (2021) efficiently implemented TreeAgg with partial variance reduction (Honaker, 2015), which leverages the recursive structure of C and only needs $\lceil \log_2 n \rceil m$ memory to generate correlated noise. The full variance reduction trick (Honaker, 2015) can further improve the performance and is equivalent to setting $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}^{-1} \in \mathbb{R}^{2^{l-1}\times(2^l-1)}$ by computing the Moore-Penrose pseudoinverse of $\boldsymbol{C}$ (Denisov et al., 2022) (we use an abuse of notation $\boldsymbol{C}^{-1}$ for pseudoinverse). However, the full variance reduction TreeAgg (TreeAgg-Full) does not have a memory-efficient implementation, and consumes nm memory. Another variant (Andersson and Pagh, 2024) is more memory-efficient, but achieves suboptimal performance compared to MF approaches (Fichtenberger et al., 2022). In this paper, we primarily consider

TREEAGG (Kairouz et al., 2021) that is widely used in industry (Xu et al., 2023), and TREEAGG-FULL (Denisov et al., 2022) that achieves better privacy-utility trade-off but less memory and computation efficient, to represent the tree aggregation mechanisms.

BANDMF Choquette-Choo et al. (2023) exploits the banded structure, i.e., $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ where $\boldsymbol{C}_{i,j} = 0, \forall |i-j| \geq \hat{b}$, to simplify the optimization and privacy accounting for MF mechanisms. BANDMF successfully applied MF mechanisms to the FL system for the first time. When fixing all the other configurations for training a production LM, BANDMF improved the DP guarantee from $\rho = 0.52$-zCDP by TREEAGG to $\rho = 0.24$-zCDP. However, BANDMF has to estimate the band size $\hat{b}$ and total rounds n for optimizing matrices before training, and the performance quickly drops when the actual min-sep b in FL training is smaller than $\hat{b}$, or the training round is more than n. BANDMF improves memory usage of MF from $n \times m$ to $\hat{b} \times m$ for correlated noise, but the typical value of min-sep b in FL is still hundreds to thousands for strong DP guarantees. More recently, BANDFHU (Kalinin and Lampert, 2024) and BANDTOEP (McKenna, 2024) optimize banded Toeplitz matrices for larger n and exploit Toeplitz structure for computation efficiency, but they have not been shown to outperform BANDMF in the FL setting.

B Stream Multiplication by BLT matrices C and C^{-1}

Algorithm 2 Stream Mult. by BLT(θ, ω) (Dvijotham et al., 2024)

Input:
Input stream $\hat{\boldsymbol{Z}} \in \mathbb{R}^{n \times m}$
$\theta \in \mathbb{R}^d, \omega \in \mathbb{R}^d$ for $\boldsymbol{C} = \text{BLT}(\theta, \omega)$
Output:
The rows $\boldsymbol{Z}_{t,:}$ of $\boldsymbol{Z} = \boldsymbol{C}\hat{\boldsymbol{Z}}$

Initialize buffers $\boldsymbol{S}_{-1} \leftarrow \boldsymbol{0} \in \mathbb{R}^{d \times m}$
for $t = 0, \ldots, n-1$ **do**
$\boldsymbol{Z}_{t,:} = \hat{\boldsymbol{Z}}_{t,:} + \omega^T \boldsymbol{S}_{t-1}$
▷ Decay each buffer by θ and add $\hat{\boldsymbol{Z}}_{t,:}$ to each
$\boldsymbol{S}_t = \text{diag}(\theta)\boldsymbol{S}_{t-1} + \boldsymbol{1}_d \hat{\boldsymbol{Z}}_{t,:}$
Output $\boldsymbol{Z}_{t,:}$

Algorithm 3 Stream Mult. by BLT$^{-1}(\theta, \omega)$

Input:
Input stream $\boldsymbol{Z} \in \mathbb{R}^{n \times m}$
$\theta \in \mathbb{R}^d, \omega \in \mathbb{R}^d$ for $\boldsymbol{C} = \text{BLT}(\theta, \omega)$
Output:
The rows $\hat{\boldsymbol{Z}}_{t,:}$ of $\hat{\boldsymbol{Z}} = \boldsymbol{C}^{-1}\boldsymbol{Z}$

Initialize buffers $\boldsymbol{S}_{-1} \leftarrow \boldsymbol{0} \in \mathbb{R}^{d \times m}$
for $t = 0, \ldots, n-1$ **do**
$\hat{\boldsymbol{Z}}_{t,:} = \boldsymbol{Z}_{t,:} - \omega^T \boldsymbol{S}_{t-1}$
▷ The buffer update is the same as Alg. 2
$\boldsymbol{S}_t = \text{diag}(\theta)\boldsymbol{S}_{t-1} + \boldsymbol{1}_d \hat{\boldsymbol{Z}}_{t,:}$
Output $\hat{\boldsymbol{Z}}_{t,:}$

C More discussion on BLT-DP-FTRL

C.1 Comparing DP-FTRL Mechanisms

We summarize DP-FTRL mechanisms in Tab. 3 in App. C and show the advantages of BLT-DP-FTRL. Our BLT mechanism can optimize either MaxLoss or RmsLoss for generating correlated noise (detailed in Sec. 3.3 following (Dvijotham et al., 2024)), while previous MF mechanisms in practice primarily consider RmsLoss (Choquette-Choo et al., 2023; McKenna, 2024). It is possible to extend the previous mechanisms to use MaxLoss, while it is still an open problem which loss format corresponds better with learning performance when running with the DP-FTRL algorithms. TREEAGG, especially TREEAGG-FULL, is equivalent to considering RmsLoss though the mechanism is predefined without explicit optimization cost; and we present the lower memory overhead ($\lceil \log_2(n) \rceil \times m$) for TREEAGG while TREEAGG-FULL without an efficient algorithm yet actually needs .

BLTs achieve better privacy-utility trade-offs than TREEAGG-FULL in simulation benchmark experiments (see Sec. 4), and clearly outperforms TREEAGG in production cross-device FL experiments (see Sec. 5), as lower noise is added in BLTs. While BANDMF (Choquette-Choo et al., 2023) can add lowest noise (measured by RmsLoss in Fig. 1), BLTs have lower mechanism optimization cost and memory overhead. Moreover, Secs. 4 and 5 show the learning performance of BLTs are often comparable with

Mech	Loss	Mech. opt. cost	Memory overhead	Noise Added	(n, b)-fragility
BLT (ours)	MaxLoss/ RmsLoss	Low ($\mathcal{O}(n)$)	Low ($\sim 4 \times m$)	Low	Low
BandMF	RmsLoss	High ($\mathcal{O}(n^2)$)	High ($\hat{b} \times m$)	Lowest	Med
BandToep	RmsLoss	Low ($\mathcal{O}(n)$)	High ($\hat{b} \times m$)	Low	Med
TreeAgg	RmsLoss*	Lowest (predefined)	Med ($\lceil \log_2(n) \rceil \times m$)	High	Low

Table 3: Summary of mechanisms considered, evaluated in terms of: *mechanism optimization cost*, how expensive is it to compute the mechanism $\boldsymbol{C}$; $\mathcal{O}(\cdot)$ gives the cost of a single gradient calculation. The next two columns relate to the deployment of the mechanism in a ML training system: *memory overhead* is the additional state (as a multiple of the model dimension m) that the server needs to maintain; the per-round runtime cost is also proportional to this value. The *noise added* is categorized subjectively, for details see examples in Fig. 1. (n, b)-fragility reflects the degree to which the mechanisms performance degrades when total number of rounds n and min-sep b are poorly estimated, see discussion on quantitative results in Figs. 4 to 7. The conclusion: **BLTs perform well on all aspects.**

BandMF under the same privacy guarantee in practical settings, though BLTs' RmsLoss is slightly worse. The memory overhead of BLTs is $d \times m$ where we empirically observe that buffer size $d = 4$ achieves low losses and further increasing d does not further improve in our empirical settings of total rounds n and min-sep b. The BLT memory overhead of $d \sim 4$ is smaller than TreeAgg where $\lceil \log_2(n) \rceil \sim 11$, and much smaller than typical $\hat{b} \sim X00$ for BandMF and BandToep. BandToep (McKenna, 2024) suggested small $\hat{b}$ is preferred when using amplification by sampling in the many participation settings; however, sampling is generally not possible in practical cross-device FL systems.

As shown in Figs. 4 to 7, BLT is also more robust than banded MF mechanisms when number of total rounds n and min-sep b are not accurately estimated. Specifically, it is unclear how to run Banded MF mechanisms beyond the estimated n after optimizing the $\boldsymbol{C} \in \mathbb{R}^{n,n}$ matrix for correlated noise. Optimizing $\boldsymbol{C} \in \mathbb{R}^{n,n}$ for a much larger n and truncating it to the actual number of training rounds can achieve good privacy-utility trade-offs, but encounter non-trivial mechanism optimization cost. BandMF performance degrades fast when the actual min-sep b is smaller than the estimated band $\hat{b}$, but the stronger DP guarantees are generally achieved when $\hat{b}$ is large. Hence the tricky task of estimating min-sep b is more important for BandMF. In general, BLT-DP-FTRL is competitive for achieving state-of-the-art privacy-utility trade-off (compared to BandMF), while maintains ease to use in practice (compared to TreeAgg).

C.2 Background on Multi-participation Sensitivity

Adjacent Data Streams and Privacy Accounting We assume users (FL clients) participate in training according to a *participation schema* $\Pi \subset \text{Powerset}([n])$, where each *participation pattern* $\pi \in \Pi$ (and so $\pi \subseteq [n]$) indicates a set of indexes of steps in which a single user might participate. Each Π results in a adjacency relation $\boldsymbol{N}_\Pi$ on data streams: two data streams $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ are adjacent, that is $(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \in \boldsymbol{N}$, if there exists a $\pi \in \Pi$ such that $\boldsymbol{x}_t = \tilde{\boldsymbol{x}}_t$ for $t \notin \pi$, and $\|\boldsymbol{x}_t - \tilde{\boldsymbol{x}}_t\|_2 \leq \zeta$ for $t \in \pi$. In FL for user-level DP (Alg. 1), $\boldsymbol{x}_t := \sum_i \Delta_i^t$ is a sum over per-user model gradients each subject to an L2-norm bound ζ, and two streaming datasets are adjacent if one can be formed from the other by "zeroing out" all the gradient contributions from any one user following Defn. 1.1 of Kairouz et al. (2021). Under this adjacent relationship, the DP guarantees of MF mechanism in DP-FTRL can be accounted for the release of $\boldsymbol{C}\boldsymbol{X} + \zeta \boldsymbol{Z}$ according Eq. (1), computing the sensitivity of $\boldsymbol{C}\boldsymbol{X}$ to calibrate with the Gaussian noise $\boldsymbol{Z}$ of zero mean and σ standard deviation (Choquette-Choo et al., 2023).

Multi-participation Sensitivity We consider b-min-sep-participation, where the distance between any two participations is *at least* b and there are at most k total participations, formally

$$\Pi_{b,k} = \{\pi \subseteq [n] \mid |\pi| < k, \{i, j\} \subseteq \pi, i \neq j \Rightarrow |i - j| \geq b\}.$$

This is motivated not only by the applicability to federated learning (as discussed by Choquette-Choo et al. (2023), which also formalized this schema), but also because (implicitly) it is the participation schema under which TREEAGG was extended to multiple participations by Kairouz et al. (2021).

Let $\mathfrak{D} := \{\boldsymbol{x} - \tilde{\boldsymbol{x}} \mid (\boldsymbol{x}, \tilde{\boldsymbol{x}}) \in \boldsymbol{N}\}$ represent the set of all possible differences between adjacent $\boldsymbol{x}, \tilde{\boldsymbol{x}}$. Then, the L2 sensitivity of $\boldsymbol{C}$ under $\boldsymbol{N}$ is given by

$$\text{sens}_{\boldsymbol{N}}(\boldsymbol{C}) = \sup_{(\boldsymbol{x},\tilde{\boldsymbol{x}})\in\boldsymbol{N}} \|\boldsymbol{C}\boldsymbol{x} - \boldsymbol{C}\tilde{\boldsymbol{x}}\|_F = \sup_{\boldsymbol{u}\in\mathfrak{D}} \|\boldsymbol{C}\boldsymbol{u}\|_F. \tag{8}$$

In this work, we only consider $\boldsymbol{C} \geq 0$ (elementwise), and so the supremum over $\boldsymbol{u}$ in Eq. (8) will always be achieved by some $\boldsymbol{u} \geq 0$ (observe each non-zero entry $\boldsymbol{u}_i \in \mathbb{R}^m$ can be chosen arbitrarily from the unit ball of radius ζ). The non-negativity also implies $\boldsymbol{C}^\top \boldsymbol{C} \geq 0$, and hence following Corollary 2.1 of Choquette-Choo et al. (2023), we have

$$\text{sens}_{\boldsymbol{N}_\Pi}(\boldsymbol{C}) = \zeta \max_{\pi\in\Pi} \|\boldsymbol{C}u(\pi)\|_2 \qquad \text{when} \qquad \boldsymbol{C} \geq 0. \tag{9}$$

where $u(\pi) \in \{0,1\}^n$ is given by $u(\pi)_i = 1$ if $i \in \pi$ and 0 otherwise. Note that ζ simply introduces a linear scaling, and so we can take $\zeta = 1$ w.l.o.g. both when optimizing mechanisms and when computing MaxLoss and RmsLoss.

C.3 A Sensitivity Lower Bound

A Sensitivity Lower Bound Inspired by Thm. 3.1, we state a sensitivity lower bound for general matrix in Remark C.1. An overly optimistic (instead of commonly worst-case) DP guarantees can be computed for MF mechanism with sensitivity in Remark C.1. We *only* use Remark C.1 for the privacy accounting of baseline binary tree mechanisms in simulation experiments in Sec. 4 as the dynamic programming accounting in (Kairouz et al., 2021) is computationally expensive. In practice we find the lower-bound of Remark C.1 is tight for the binary tree matrices we consider; proving this is an interesting open problem.

Remark C.1. *Letting $\pi^\star \in \Pi$ as in Eq. (5), for any mechanism $\boldsymbol{C}$,*

$$\text{sens}_\Pi(\boldsymbol{C}) \geq \|\boldsymbol{C}u(\pi^\star)\|_2 \tag{10}$$

is a lower-bound on sensitivity (the actual sensitivity might be higher). While Kairouz et al. (2021) introduced a dynamic program for computing binary-tree sensitivity, it requires some work to extend it to the tree completion trick, and in practice it is expensive to compute. Hence, when evaluating TREEAGG *approaches, for simplicity we use the lower bound of Eq. (10), which can be computed immediately for the tree-completion matrices $\boldsymbol{C}$ used when n is not a power of 2.*

D Optimizing for BLT Matrices

Combining these elements gives us an efficient and differentiable algorithm for computing MaxLoss and RmsLoss. Complete pseudo-code for the differentiable loss calculation is given in Alg. 4. Following Dvijotham et al. (2024), we use auto differentiation and L-BFGS optimizer in JAX (Bradbury et al., 2018) to optimize $(\theta, \hat{\theta})$ for the BLT-DP-FTRL algorithm, and then extract $\text{BLT}(\theta, \omega)$ for noise generation. Similar to (Dvijotham et al., 2024), we introduce log-barrier penalties w/ strength 10^{-7} to keep $\omega > 0$, $\theta > 0$ and $\theta < 1$ (which is necessary to ensure the Toeplitz coefficients of $\boldsymbol{C}$ are decreasing to satisfy Thm. 3.1). For high precision optimization, we use double precision in JAX on CPUs and GPUs. We observe that increasing buffer size d does not necessarily reduce the loss due to numerical stability and optimization challenges, and different BLT parameter (θ, ω) may be achieved in different optimization runs. We also highlight that the different BLT parameters (θ, ω) can generate similar Toeplitz coefficients for $\boldsymbol{C}$, which suggests a smaller d might help mitigate the optimization challenge from overparametrization.

The primary motivation for utilizing the $(\theta, \hat{\theta})$ parameterization in Alg. 4 is computational efficiency. Tab. 4 compares the time to compute n Toepltiz coefficients for $\boldsymbol{C}^{-1}$ given either $\text{BLT}(\theta, \omega)$ or given $(\theta, \hat{\theta})$. In the first caes ("brute force"), we construct the Toeplitz coefficients of $\boldsymbol{C}$ using Eq. (2), and then solve a linear system (using `jax.lax.scan` and the property that the inverse of a lower triangular Toeplitz

Algorithm 4 Differentiable Loss for BLTs

Inputs:
Pair of buffer-decay parameters $(\theta, \hat{\theta})$ with d buffers each $(\theta, \hat{\theta} \in [0,1]^d)$.
num rounds n, min-separation b, max participations k
Penalty strength λ, set to zero for loss calculation, or $\lambda = 10^{-7}$ to stabilize optimization
Outputs:
Either $\mathrm{MaxLoss}(\boldsymbol{A}\boldsymbol{C}^{-1}, \boldsymbol{C})$ or $\mathrm{RmsLoss}(\boldsymbol{A}\boldsymbol{C}^{-1}, \boldsymbol{C})$.

▷ Use Alg. 5 to calculate the unique ω and $\hat{\omega}$ such that $\boldsymbol{C} = \mathrm{BLT}(\theta, \omega)$ and $\boldsymbol{C}^{-1} = \mathrm{BLT}(\hat{\theta}, \hat{\omega})$
$\omega = \texttt{calc\_output\_scale}(\theta, \hat{\theta})$
$\hat{\omega} = \texttt{calc\_output\_scale}(\hat{\theta}, \theta)$

▷ Compute $\texttt{sens} = \|\boldsymbol{C}\pi^\star\|_2$ where $\boldsymbol{C} = \mathrm{LtToep}(c)$
Compute $c \in \mathbb{R}^n$ where $c_0 = 1$ and $c_i = \sum_{j\in[d]} \omega_j \theta_j^{i-1}$ for $i \in \{1, \ldots, n\}$.
$\bar{c} = 0 \in \mathbb{R}^n$ ▷ Holds the sum of columns of $\boldsymbol{C}$
for $i \in [k]$ **do**
  $\bar{c}[b \cdot i :] \mathrel{+}= c[0 : n - b \cdot i]$ ▷ numpy-like semantics
$\texttt{sens} = \|\bar{c}\|_2$ ▷ Because $\bar{c} = \boldsymbol{C}\pi^\star$.

▷ Compute $\mathrm{Errror}(\boldsymbol{A}\boldsymbol{C}^{-1})$ where $\boldsymbol{C}^{-1} = \mathrm{LtToep}(\hat{c})$.
Compute $\hat{c} \in \mathbb{R}^n$ where $\hat{c}_0 = 1$ and $\hat{c}_i = \sum_{j\in[d]} \hat{\omega}_j \hat{\theta}_j^{i-1}$ for $i \in \{1, \ldots, n\}$.
Compute $b \in \mathbb{R}^n$ by $b_i = \sum_{j=0}^{i} \hat{c}_i$ ▷ So $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}^{-1} = \mathrm{LtToep}(b)$.

$$\texttt{err} = \begin{cases} \sqrt{\sum_{i\in[n]} b_i^2} & \text{for MaxError} \\ \sqrt{\sum_{i\in[n]} (n-i) b_i^2 / n} & \text{for RmsError.} \end{cases}$$

▷ Log-barrier penalties to keep $\theta > 0$, $\theta < 1$, and $\omega > 0$ for numerical stability when optimizing
$\texttt{penalty} = \lambda(-\log(\theta) - \log(1-\theta) - \log(\omega))$

Return $\texttt{loss} = \texttt{err} \cdot \texttt{sens} + \texttt{penalty}$

Algorithm 5 `calc_output_scale` (Lemma 5.2 of Dvijotham et al. (2024))

Input:
Pair of buffer-decay parameters $(\theta, \hat{\theta})$ with d buffers each $(\theta, \hat{\theta} \in [0,1]^d)$.
Output:
The unique ω s.t. $\boldsymbol{C} = \mathrm{BLT}(\theta, \omega)$ has a BLT inverse with buffer-decay $\hat{\theta}$ ($\boldsymbol{C}^{-1} = \mathrm{BLT}(\hat{\theta}, \cdot)$).

$p(x) = \prod_{i\in[d]}(1 - \theta_i x)$
$q(x) = \prod_{i\in[d]}(1 - \hat{\theta}_i x)$
$f(x) = (p(x) - q(x))/x$ ▷ Polynomial division gives f, a polynomial of degree $d-1$
$z = \prod_{i\in[d]} -\theta_i$
$w_i = \left(\prod_{j\neq i}(\theta_i^{-1} - \theta_j^{-1})\right)^{-1}$ for $i \in [d]$
Define ω by $\omega_i = f(\theta_i^{-1})\frac{-\theta_i w_i}{z}$ for $i \in [d]$
Return ω

n	brute force	via Alg. 5	speedup
2000	0.021	3.8e-5	550×
20000	0.258	3.6e-5	7176×
200000	4.772	4.5e-5	104884×

Table 4: Seconds to compute n Toeplitz coefficients of $\boldsymbol{C}^{-1}$. JAX just-in-time (JIT) compilation is *not* included for either approach.

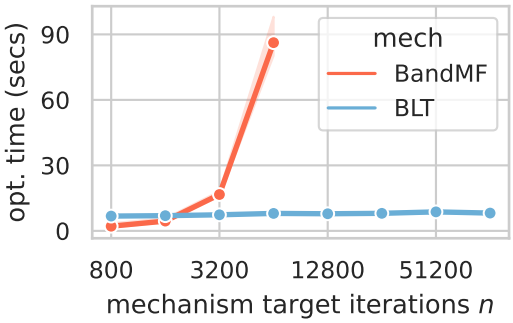


Figure 3: Total wall clock optimization time including JIT compilation on a V100 GPU for BandMF and BLTs, fixing $b = 400$ and varying n. The average over 3 runs is reported.

matrix is also lower triangular Toeplitz) to compute the coefficients of $\boldsymbol{C}^{-1}$). In the second case, we use Alg. 5 to compute $(\omega, \hat{\omega})$, and then compute the Toeplitz coefficients by applying Eq. (2) to BLT$(\hat{\theta}, \hat{\omega})$. The comparison uses a V100 GPU and a compiled `jax` implementation, and is repeated many times with an average is given. The second approach can be fully vectorized, and is orders of magnitude faster. This is critical because this is the primary computational step in computing the RmsLoss or MaxLoss and occurs in the inner loop of the mechanism optimization procedure: the net result is mechanism optimization is substantially faster than for BandMF, and scales to much larger n, see Fig. 3. Alg. 4 does incur more `jax` just-in-time (JIT) compilation overhead compared to BandMF optimization, which accounts BLT optimization being slightly slower for small n.

E More RmsLoss and MaxLoss Experiments

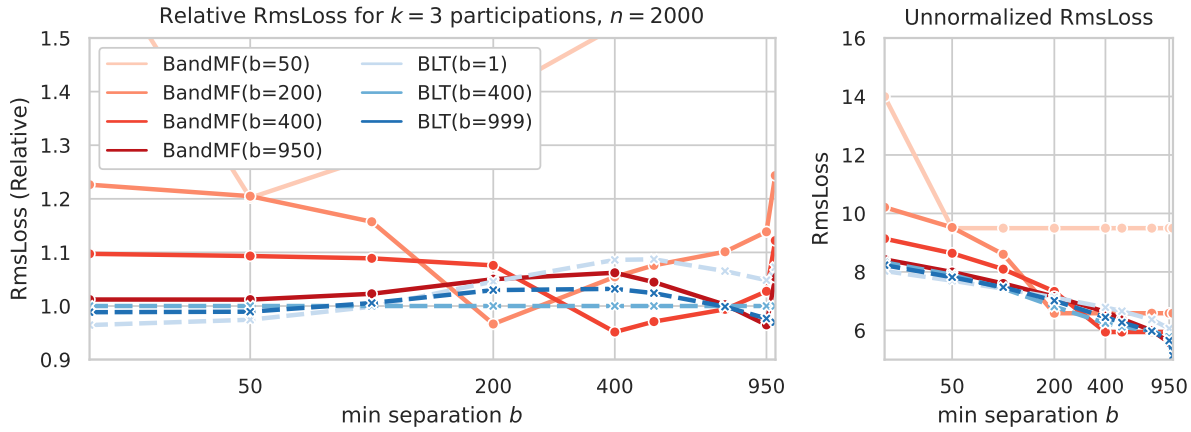


Figure 4: Comparison of BandMF and BLTs with $n = 2000$ and $k = 3$ participations, varying the min-separation b. The BLTs were optimized for RmsLoss. The x-axis is shown on a log-scale. The left panel gives loss relative to the BLT$(b = 400)$ mechanism, while the right panel gives the same data on an unnormalized y-axis scale.

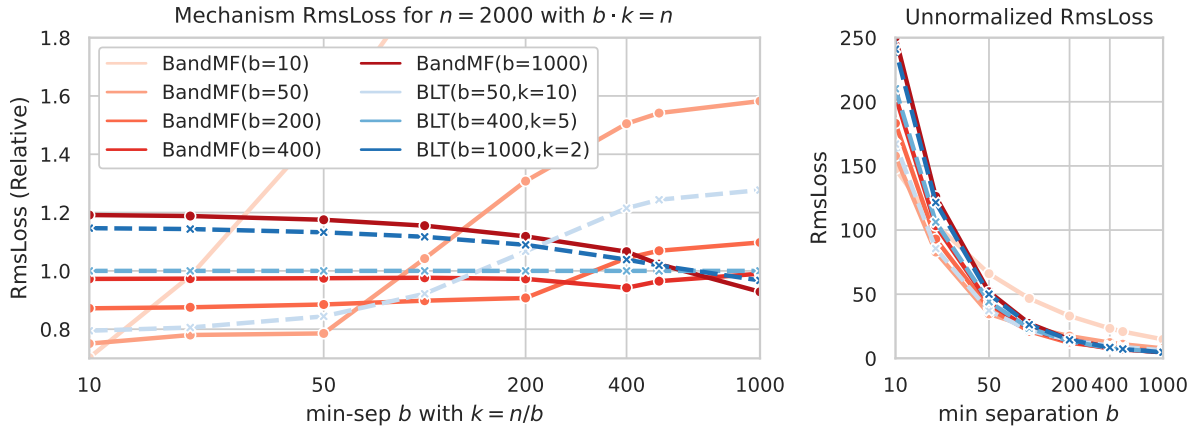


Figure 5: Comparison of BANDMF and BLTS with $n = 2000$, varying b such that $k = n/b$ is an integer. The BLTS were optimized for RmsLoss. The x-axis is shown on a log-scale. The left panel gives loss relative to the BLT($b = 400$) mechanism, while the right panel gives the same data on an unnormalized y-axis scale.

Robustness to Min-sep b Fig. 4 compares BLT to the strong baseline BANDMF, fixing $n = 2000$ and $k = 3$ participations and varying the min-separation b. We make three primary observations: (1) The optimization of the BANDMF mechanisms implicitly assumes $b \sim \hat{b}$, and as expected, near this regime BANDMF in fact (slightly) outperforms the BLTS. (2) However, when $b \ll n/k$, the BLTS perform better. (3) Interestingly, BANDMF performs significantly worse than the BLTS at $b = 999$. In this case, the only participation patterns that actually have 3 participations are e.g. $\{0, 999, 1998\}$, $\{0, 1000, 1999\}$ — importantly, only the columns $\{0, 1, 999, 1000, 1998, 1999\}$ can ever occur in a 3-participation pattern. Because the columns of $\boldsymbol{C}$ for BANDMF all have the same column norm, this fact cannot be exploited. However, because of their Toeplitz structure, columns 1998 and 1999 have smaller norms than other columns, and that is beneficial in this situation.

The setting where k is fixed and we vary b includes situations that should generally be avoided in practice. For example, if we had $k = 3$, $b = 10$, and $n = 2000$, this indicates we had enough data we should have been able to achieve $b = n/k \approx 667$, and so $b = 10$ indicates a highly suboptimal data ordering. Similarly, if we had $k = 3$, $b = 999$, and $n = 2000$, then we would have been better off stopping at $n = 1998$, which would have ensured only $k = 2$ participations and significantly decreased sensitivity (at presumably a small cost in learning performance compared to training for $n = 2000$ iterations).

Fig. 5 shows the contrasting scenario (indicating an essentially optimal data ordering, general not possible in federated learning) that occurs when we fix $n = 2000$, and choose b that exactly divide n so that we can take $k = n/b$ exactly. Fig. 5 considers the worst-case max participation k for given min-sep b and total rounds n, and achieves generally larger RmsLoss. When $b \leq \hat{b}$, BANDMF slightly outperforms BLTS, but BANDMF degrade more rapidly for $b \leq \hat{b}$. In general, the curves of BLTS are more smoother across different min-sep b in both Fig. 4 and Fig. 5.

Robustness to Total Rounds n Fig. 6 considers varying the number of steps of the mechanism actually executed for mechanisms optimzied for different n. BANDMF mechanisms can only be used up to the n they are optimized for, but BLTS naturally extend to any n. This figure demonstrates that again BLTS are not particularly sensitive to the n for which they are optimized. For this figure, the maximum number of participations is chosen to be the largest allowed given n and $b = 400$ (i.e., $k = n/d$), leading to the stairstep behavior of the unnormalized RmsLoss in Fig. 6 (Right). BANDMF optimizing for large n performs well when the actual number of iterations executed is small, but optimizing for large n encounters nontrivial as discussed in Tab. 3 and Fig. 3. Finally, these results show that only $d = 2$ buffers is sufficient for good performance, or a $200\times$ memory savings compared to BANDMF with $b = 400$ bands.

Comparing BLT to BANDTOEP and BANDFHU Finally, we compare BLT-DP-FTRL to several other more recent mechanisms:

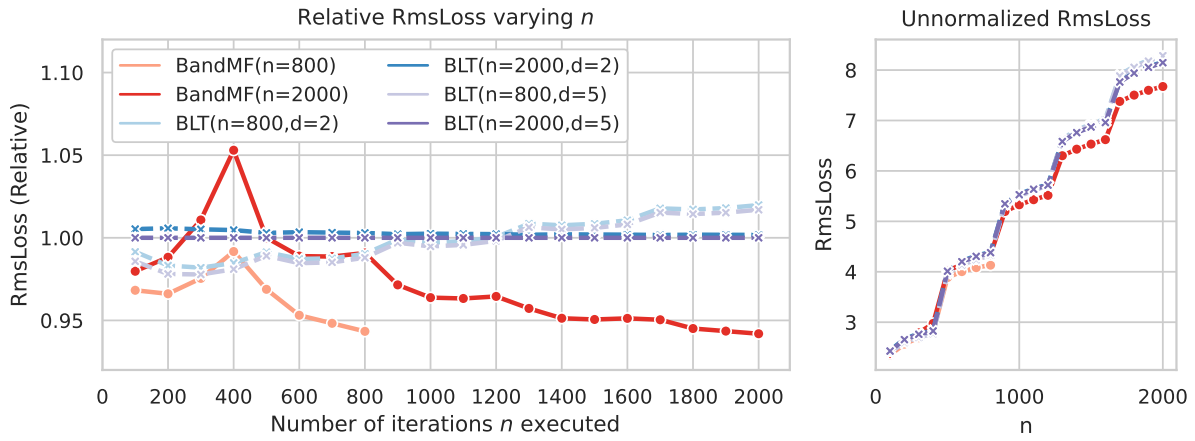


Figure 6: Comparison of BANDMF and BLTS optimized for $n = 800$ and $n = 2000$, and evaluated for n up to the optimization target (for BANDMF) and over $[0, 2000]$ for the BLTS. All mechanisms were optimized for min-separation (bands) $b = 400$. BLTS with $d = 2$ and $d = 5$ perform almost equivalently; $d = 1$ (not shown),is not sufficient with relative RmsLoss > 1.07.

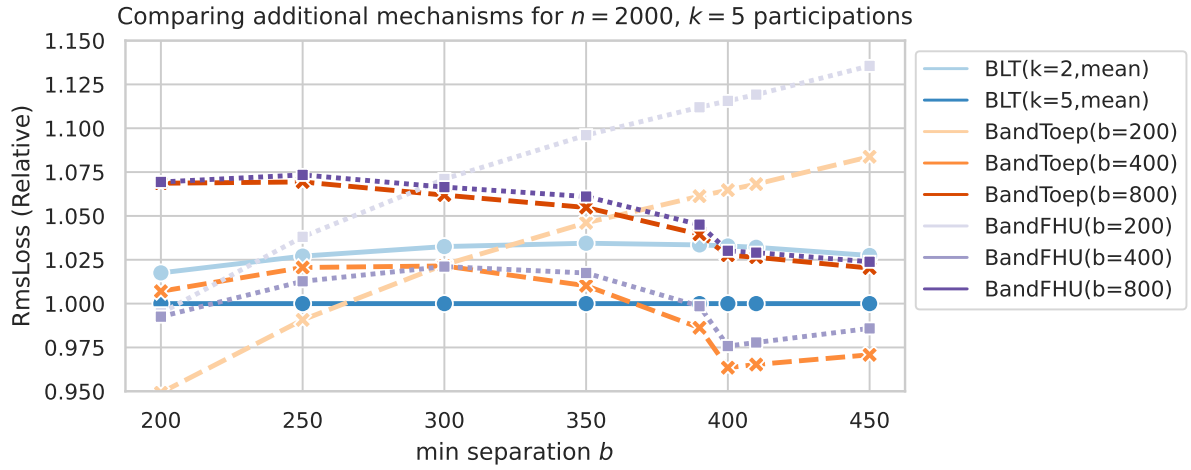


Figure 7: Comparison of mechanisms for $n = 2000$ and $k = 5$ participations, optimized for min-separation $b = 400$ and compared for different actual values of min-separation. The setting is comparable to that of Fig. 1, so only relative lines are given.

- BANDTOEP (McKenna, 2024), which optimizes banded Toeplitz matrices $\boldsymbol{C}$ for b-min-sep-participation under RmsLoss. The primary advantage of this mechanism compared to BANDMF is that BANDTOEP matrices can be optimized for much larger n. However, the runtime is the same as BANDMF, and the optimization is still slower than the optimization of BLTS.
- BANDFHU (Kalinin and Lampert, 2024), which uses prefixes of the optimal-for-single-participation MaxLoss coefficients of Fichtenberger et al. (2022) to form banded Toeplitz matrices. These will likely be worse than BANDTOEP (which is specifically optimized for multiple participations), but require no mechanism optimization.

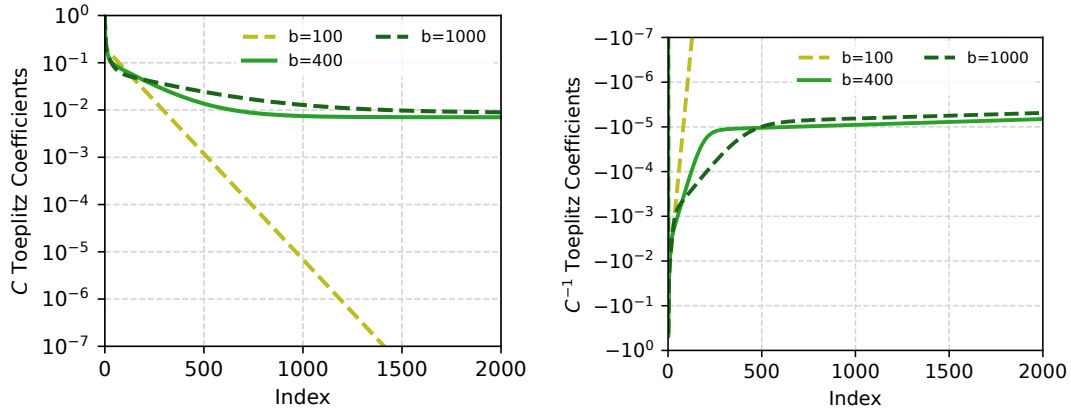
Fig. 7 shows that BLTS are comparable or better to both of these approaches.

F BLT Parameters for Production Training

We provide the BLT* parameters we generated and used in training production LMs with DP FL in Sec. 5. The BLT* matrices are optimized for three min-sep settings $b = (100, 400, 1000)$ and each BLT is parameterized by 8 values for buffer size $d = 4$, i.e., buffer decay $\theta \in \mathbb{R}^d$ and output scale $\omega \in \mathbb{R}^d$.

- min-sep $b = 100$, total rounds $n = 2000$, max participation $k = 10$ $\Rightarrow$,
  $\theta = (0.989739971007307, 0.7352001759538236, 0.16776199983448145, 0.1677619998016191)$,
  $\omega = (0.20502892852480875, 0.23357939425278557, 0.03479503245420878, 0.03479509876050538)$.
- min-sep $b = 400$, total rounds $n = 4000$, max participation $k = 5$ $\Rightarrow$,
  $\theta = (0.9999999999921251, 0.9944453083640997, 0.8985923474607591, 0.4912001418098778)$,
  $\omega = (0.0070314825502323835, 0.10613806907600574, 0.1898159060327625, 0.1966594748073734)$.
- min-sep $b = 1000$, total rounds $n = 4000$, max participation $k = 2$ $\Rightarrow$,
  $\theta = (0.9999999999983397, 0.9973412136664378, 0.9584629472313878, 0.6581796870749317)$,
  $\omega = (0.008657392263671862, 0.05890891298180163, 0.14548176930698697, 0.2770117005326523)$.

Fig. 8 visualizes the corresponding Toeplitz coefficients for $\boldsymbol{C}$ to compute sensitivity and $\boldsymbol{C}^{-1}$ for generating correlated noise. The coefficients of $\mathrm{BLT}(\theta, \omega)$ for $b = 100$ decaying faster than $b = 400$ and $b = 1000$.



(a) (b)

Figure 8: Toeplitz coefficients $\{c_0, \ldots, c_n\}$ for $\boldsymbol{C} = \mathrm{LtToep}(c)$ and $\{\hat{c}_0, \ldots, \hat{c}_n\}$ for $\boldsymbol{C}^{-1} = \mathrm{LtToep}(\hat{c})$; For $\mathrm{BLT}(\theta, \omega)$, c can be computed by Eq. (2), and there are many ways to derive corresponding $\hat{c}$ (e.g., set $\boldsymbol{Z} = \boldsymbol{I}$ in Alg. 3).

G Additional Simulation and Production Results

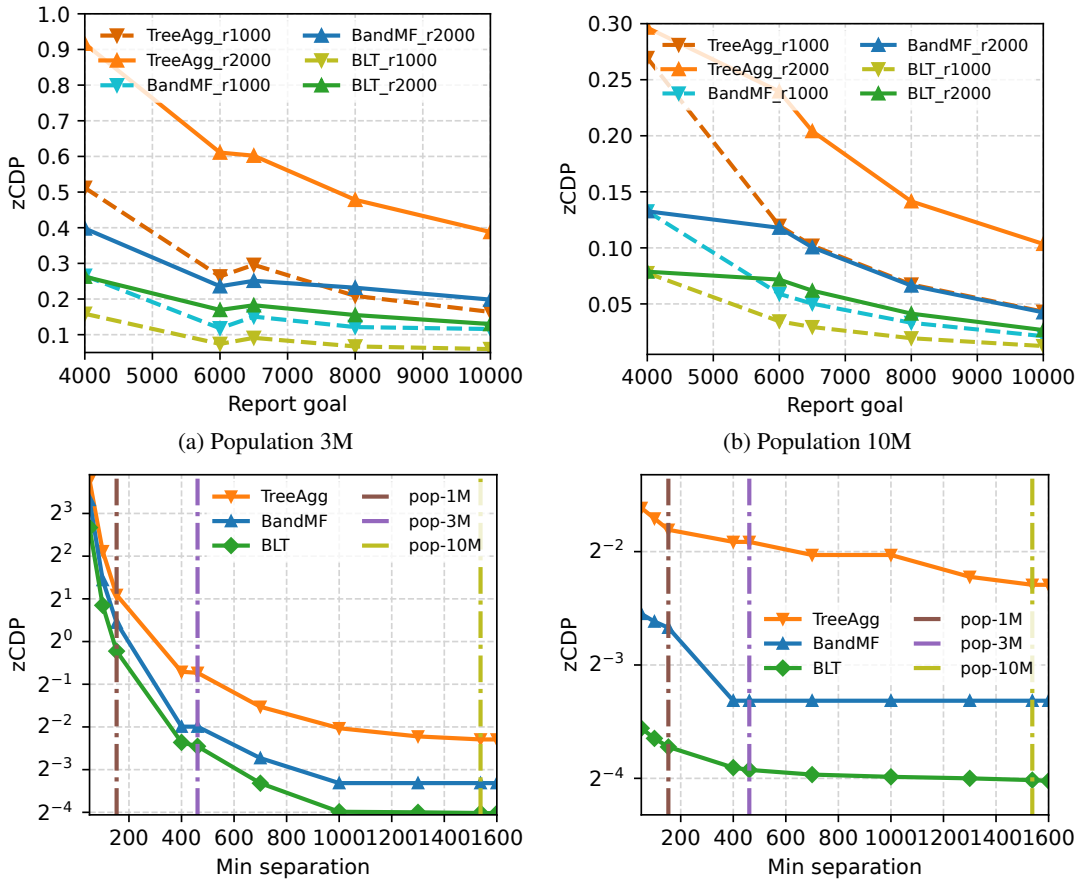
G.1 Production Setting for Mobile Keyboard LMs

Following Hard et al. (2018); Xu et al. (2023), we train one-layer LSTM LMs of $\sim 6.4M$ parameters for mobile keyboard applications. These LMs are deployed on device to predict words during decoding time to facilitate user typing. We use next word prediction (NWP) accuracy on a hold-out set of devices to track the training progress, and also conduct A/B test in production, where following two metrics are reported: (1) Word Modified Ratio (WMR), the ratio of words being modified during typing or after committed; improvement is shown by reduction; (2) Word Per Minute (WPM): the number of committed words per minute. LMs are trained for different language-locale combination in different populations. We study Spanish in Spain (es-ES), Indonesian in Indonesia (id-ID), Portuguese in Brazil (pt-BR) and Portuguese in Portugal (pt-PT). LMs are pre-trained on public multilingual C4 dataset (Xue et al., 2020) before private training with FL and DP.

Algorithm Setting We compare our BLT-DP-FTRL algorithm with TreeAgg (Kairouz et al., 2021; Xu et al., 2023) and BandMF (Choquette-Choo et al., 2023) discussed in Sec. 2. As far as we know, these two are the only DP algorithms actively used to train LMs in a production FL system. We follow the system and parameter configurations in (Xu et al., 2023; Choquette-Choo et al., 2023; Xu and Zhang, 2024) for baselines, and compare to TreeAgg for pt-PT and id-ID, and BandMF for es-ES and pt-BR. However, we highlight it is challenging to exactly reproduce the settings, especially for the min-sep parameter. The BandMF algorithm are optimized for total round $n = 2000$, band $\hat{b} = 400$ for es-ES, and

$\hat{b} = 1000$ for pt-BR. We optimize BLT for total round $n = 4000$[2], estimated min-sep $b = 100$ for pt-PT, $b = 400$ for es-ES and id-ID, and $b = 1000$ for pt-BR. We use BLT* for multi-participation and estimate the max-par based on n/b. For these n, b settings, only $d = 4$ buffers can achieve near-optimal loss in optimization, and BLT* matrices are parameterized by only 8 numbers (these parameters are provided in App. F). Though different configures are used for populations with different sizes, the BLT parameters $(\theta, \omega) \in \mathbb{R}^8$ optimized for $b = 400$ can achieve competitive results for a wide range of min-seps, which can be a reasonable default for BLT-DP-FTRL. As discussed in Tab. 3, BLT is more memory-efficient than both TREEAGG and BANDMF. In the simulation results Tab. 1, BLT is also better than TREEAGG for privacy-utility trade-off, and comparable with BANDMF. The results in production further show that the flexibility of BLT makes it easier to use in practice, and achieve better results than both TREEAGG and BANDMF.

G.2 Extrapolation Results for Production Setting



(a) Population 3M

(b) Population 10M

(c) Report goal 6500, worst-case max-par

(d) Report goal 6500, optimistic max-par=2

Figure 9: The effect of population size, number of rounds, report goals, and min-seps on DP-FTRL privacy guarantees. The results are extrapolate from the setting for es-ES and id-ID (BANDMF and BLT matrices are optimized for min-sep=400) based on the hypothesis that linearly scale noise multiplier and report goal, or only change min-sep will not affect the model utility. The For a fixed number of rounds to achieve utility target, increasing report goal and min-sep can achieve stronger guarantees measured by smaller zCDP. The optimal min-sep is capped by population size for a fixed report goal, and BLT provides better guarantees, and smoother transition across different min-seps.

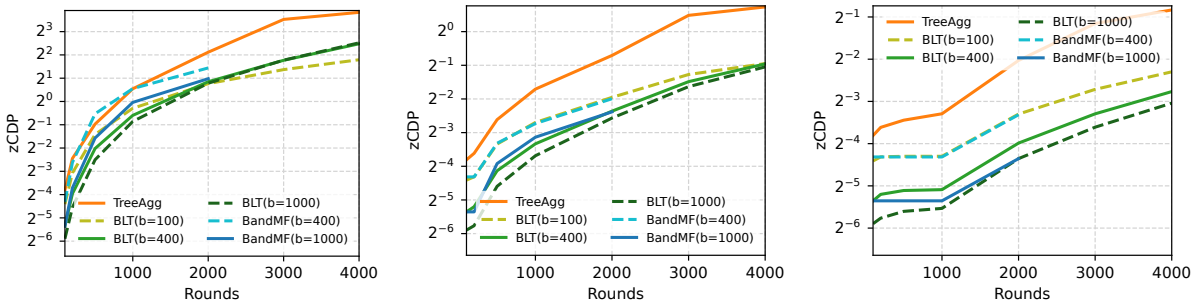
Extrapolation We extrapolate the results for production setting by using a common hypothesis: linearly increase report goal and noise multiplier will not change the utility of the model as the signal-to-noise ratio is maintained. In addition, we assume only changing min-sep will not change the utility because of

[2]As the target round is usually less than 2000, $n = 4000$ for BLT is less favorable compared to $n = 2000$ used for BANDMF. BLT is robust to the target round n, and achieves stronger results with an inferior n.

the signal-to-noise ratio. The hypothesis has been verified in previous work (Kairouz et al., 2021) and the large report goal experiments for pt-BR and pt-PT in Tab. 2 and Fig. 13. Hence we can study the effect on DP without actually training the model, similar to Sec. 4.2 for simulation.

We vary the report goal, and the min-sep is optimistically estimated by *min-sep*=$\lfloor$*population-size/report-goal*$\rfloor$, and *max-par*=$\lceil$*total-rounds/min-sep*$\rceil$ is used unless otherwise specified. We discuss the extrapolation results in Fig. 9, where utility is the same based on the hypothesis. (1) BLT achieves better DP guarantees because of its robustness to min-seps and total rounds. (2) We observe that using larger report goal and optimizing for the largest possible min-sep achieves better results than using smaller report goals and larger corresponding min-sep, similar to observation for TREEAGG in (Xu et al., 2023). (3) Fig. 9b shows training more rounds does not necessarily increasing DP guarantees when min-sep is large. (4) The gap of BLT and BANDMF is small when min-sep is accurately estimated. In the regime of relatively high signal-to-noise ratio (large noise multiplier for limited computation resources), BLT is competitive in a wide range of different configurations. Hence BLT is easier to use in production FL systems compared to BANDMF, and also saves memory during training.

Finally, in Fig. 10, we extrapolate the DP guarantee results by varying the number of total rounds n with the noise multiplier for the fixed report goal 6500, fixed min separation $b = 100, 400, 1000$, and corresponding max participation $k = n/b$. The TREEAGG, BLT and BANDMF mechanisms used in production are compared. Instead of using RmsLoss or MaxLoss to measure privacy-utility trade-offs in Figs. 1 and 6, here we fix utility based on empirical utility of the production training and the signal-to-noise-ratio hypothesis, and compare the DP guarantees. As mentioned before, using BANDMF beyond the optimized matrices for $n = 2000$ has not been studied before, and hence we only extrapolate BANDMF up to $n = 2000$ rounds. TREEAGG and BLT can run arbitrary number of rounds, and BLTs achieve stronger DP guarantees than TREEAGG. In practice, we can use one of the BLTs as a default mechanism across different settings, and perform on-the-fly optimization for given customized setting.



(a) Min-sep b=100 (b) Min-sep b=400 (c) Min-sep b=1000

Figure 10: Extrapolate by varying number of rounds n for the TREEAGG, BLT and BANDMF mechanisms used in production. Use the noise multiplier for the fixed report goal 6500; fix min separation $b = 100, 400, 1000$, respectively; worst-case max participation is varied assuming fixed population size, i.e., $k = n/b$. The utility of different mechanisms at a specific round (x-axis value) are assumed to be similar due to the signal-to-noise ratio hypothesis, and we can compare the corresponding zCDP guarantees (y-axis value).

G.3 Additional Plots

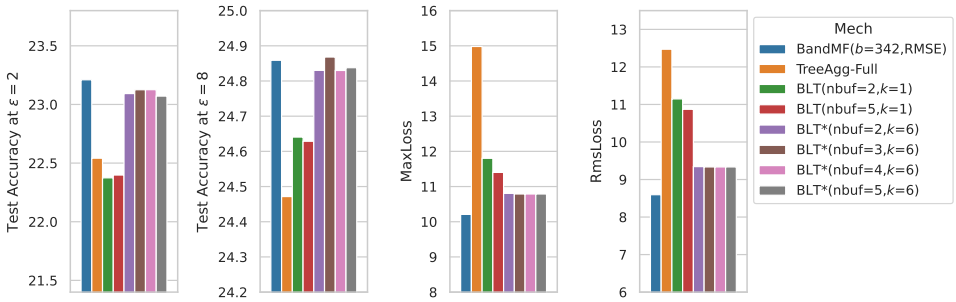


Figure 11: Visualizing the results in Tab. 1.

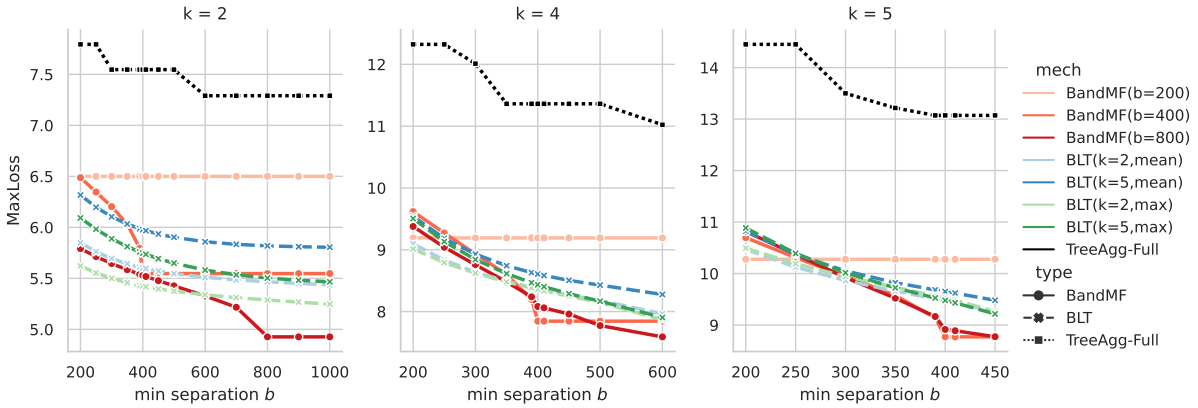
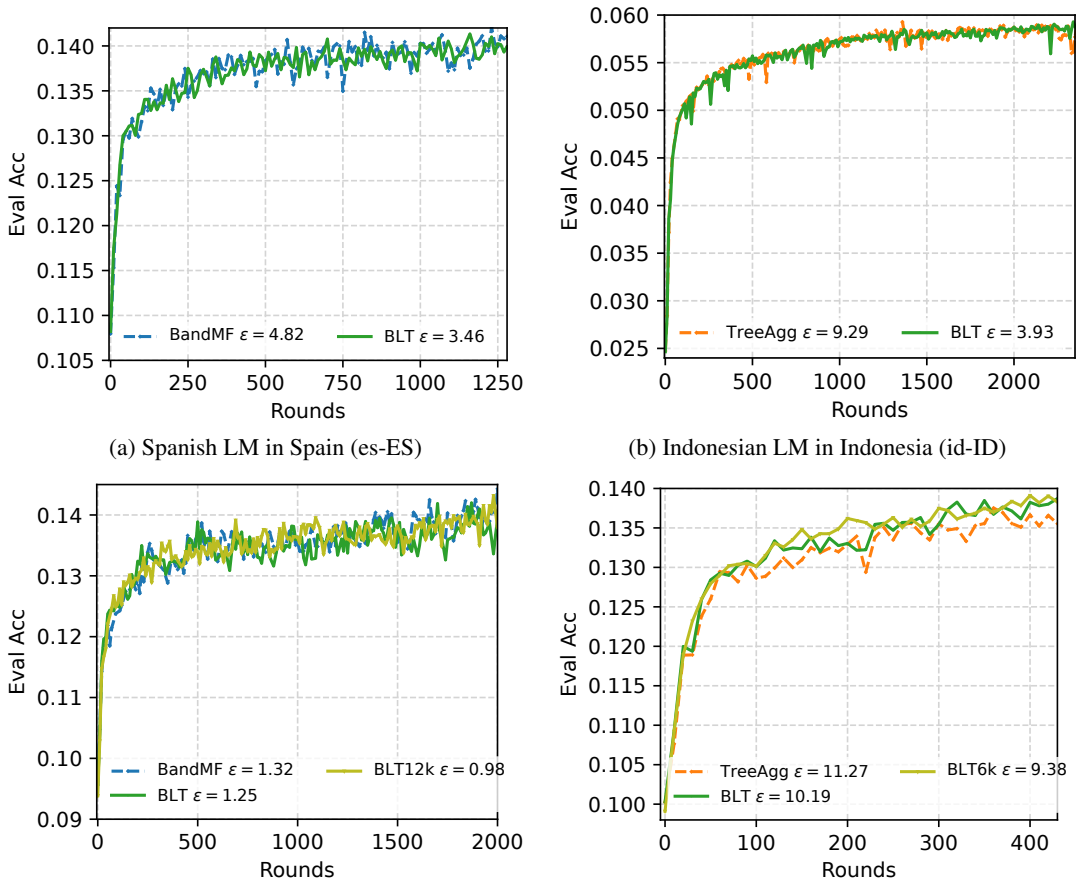


Figure 12: The same mechanisms from Fig. 1, but compared on MaxLoss instead of RmsLoss.



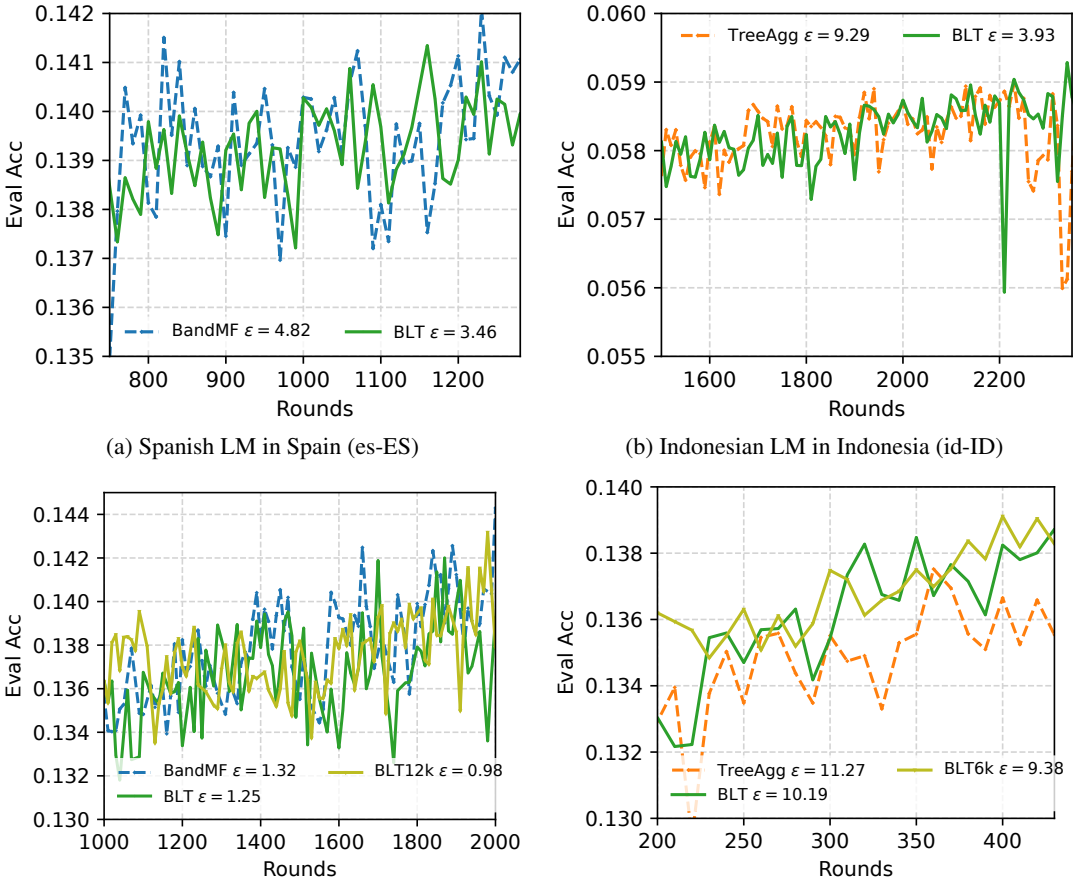
(a) Spanish LM in Spain (es-ES)

(b) Indonesian LM in Indonesia (id-ID)

(c) Portuguese LM in Brazil (pt-BR)

(d) Portuguese LM in Portugal (pt-PT)

Figure 13: The NWP evaluation accuracy curves for training LMs with DP-FTRL in FL. BLT achieves comparable NWP accuracy and slightly better privacy guarantees (at the last round) compared to BANDMF for es-ES and pt-BR; much better DP guarantees, and/or better utility compared to TREEAGG for id-ID and pt-BR.



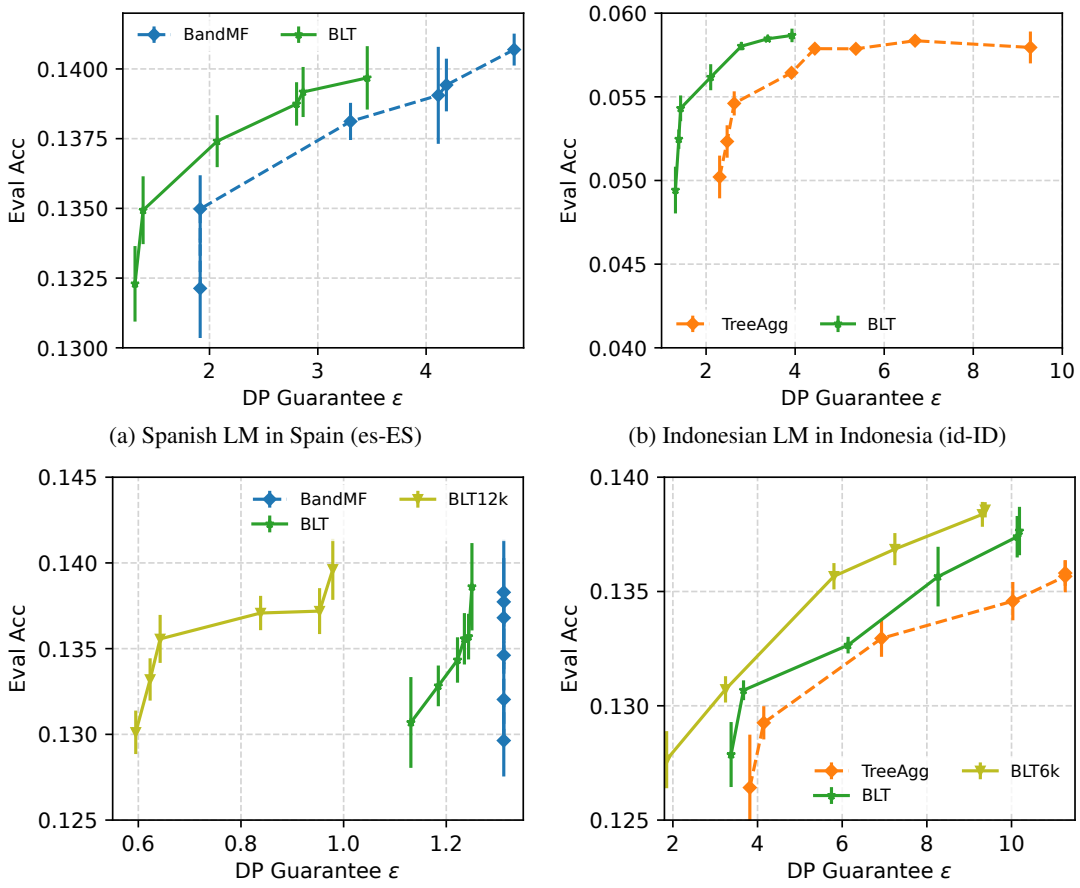
(a) Spanish LM in Spain (es-ES)

(b) Indonesian LM in Indonesia (id-ID)

(c) Portuguese LM in Brazil (pt-BR)

(d) Portuguese LM in Portugal (pt-PT)

Figure 14: The NWP evaluation accuracy curves for training LMs with DP-FTRL in FL. Zoom in the latter stage of training for the curves in Fig. 13. The NWP accuracy increases fast in the first 200 rounds in DP FL training, and the accuracy changes within the range of 0.01 when zooming in the later stage. The oscillation is because of the stochasticity in forming subsets of devices in both training and evaluation per round. The average NWP accuracy from nearby rounds is reported in Tab. 2 to reduce the variance.



(a) Spanish LM in Spain (es-ES)

(b) Indonesian LM in Indonesia (id-ID)

(c) Portuguese LM in Brazil (pt-BR)

(d) Portuguese LM in Portugal (pt-PT)

Figure 15: The privacy-utility trade-off curves derived from Fig. 13. For each selected round r, we compute the mean and standard deviation (shown as vertical bars) for accuracy from the rounds in the range of $r \pm 50$ ($r \pm 10$ for pt-PT), and also accounting the DP guarantees. BLTs show better privacy-utility trade-off as their curves are closer to the top left (small DP guarantees and large NWP accuracy).