Shuffle Private Linear Contextual Bandits

Sayak Ray Chowdhury [*1] **Xingyu Zhou** [*2]

Abstract

Differential privacy (DP) has been recently introduced to linear contextual bandits to formally address the privacy concerns in its associated personalized services to participating users (e.g., recommendations). Prior work largely focus on two trust models of DP – the central model, where a central server is responsible for protecting users' sensitive data, and the (stronger) local model, where information needs to be protected directly on users' side. However, there remains a fundamental gap in the utility achieved by learning algorithms under these two privacy models, e.g., if all users are *unique* within a learning horizon T, $\widetilde{O}(\sqrt{T})$ regret in the central model as compared to $\widetilde{O}(T^{3/4})$ regret in the local model. In this work, we aim to achieve a stronger model of trust than the central model, while suffering a smaller regret than the local model by considering recently popular *shuffle* model of privacy. We propose a general algorithmic framework for linear contextual bandits under the shuffle trust model, where there exists a trusted shuffler – in between users and the central server– that randomly permutes a batch of users data before sending those to the server. We then instantiate this framework with two specific shuffle protocols – one relying on privacy amplification of local mechanisms, and another incorporating a protocol for summing vectors and matrices of bounded norms. We prove that both these instantiations lead to regret guarantees that significantly improve on that of the local model, and can potentially be of the order $\widetilde{O}(T^{3/5})$ if all users are unique. We also verify this regret behavior with simulations on synthetic data. Finally, under the practical scenario of non-unique users, we show that the regret of our shuffle private algorithm scale as $\widetilde{O}(T^{2/3})$, which *matches* what the central model could achieve in this case.

1. Introduction

In the linear contextual bandit problem (Auer, 2003; Chu et al., 2011), a learning agent observes the context information c_t of an user at every round t. The goal is to recommend an action a_t to the user so that the resulting reward y_t is maximized. The mean reward is given by a linear function of an *unknown* parameter vector $\theta^* \in \mathbb{R}^d$, $d \in \mathbb{N}$, i.e.,

$$\mathbb{E}\left[y_t \mid c_t, a_t\right] = \langle \theta^*, \phi(c_t, a_t) \rangle \ ,$$

where $\phi : \mathcal{C} \times \mathcal{X} \to \mathbb{R}^d$ maps a context-action pair to a d-dimensional feature vector, and $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product. The context and action sets $\mathcal{C}$ and $\mathcal{X}$ are arbitrary, and can also possibly be varying with time. An agent's performance over T rounds is typically measured through the cumulative pseudo-regret

$$\mathrm{Reg}(T) = \sum\nolimits_{t=1}^{T} \left[\max_{a \in \mathcal{X}} \langle \theta^*, \phi(c_t, a) \rangle - \langle \theta^*, \phi(c_t, a_t) \rangle \right] ,$$

which is the total loss suffered due to not recommending the actions generating highest possible rewards corresponding to observed contexts. This framework has found applications in many real-life settings such as internet advertisement selection (Abe et al., 2003), article recommendation in web portals (Li et al., 2010), mobile health (Tewari & Murphy, 2017), to name a few. The general applicability of this framework has motivated a line of work (Shariff & Sheffet, 2018; Zheng et al., 2020) studying linear contextual bandit problems under the additional constraint of *differential privacy* (Dwork, 2008), which guarantees that the users' contexts and generated rewards will not be inferred by an adversary during this learning process.

To illustrate the privacy concern in the contextual bandit problem, let us consider a mobile medical application in which an mobile app recommends a tailored treatment plan (i.e., action) to each patient (i.e., user) based on her personal information such as age, weight, height, medical history etc. (i.e., context). Meanwhile, this mobile app's recommendation algorithm also needs to be updated once in a while in a cloud server after collecting data from a batch of patients, including treatment outcomes (i.e., rewards)

[*]Equal contribution [1]Boston University, USA [2]Wayne State University, USA. Correspondence to: Sayak Ray Chowdhury <sayak@bu.edu>, Xingyu Zhou <xingyu.zhou@wayne.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

and contexts, which are often considered to be private and sensitive information. Hence, each patient would like to obtain a personalized and effective treatment plan while guaranteeing their sensitive information remains protected against a potential adversarial attack in this interactive process. Protection of privacy is typically achieved by injecting sufficient noise in users' data (Arora et al., 2014; Xin & Jaakkola, 2014), which results in a loss in utility (i.e., an increase in regret) of the recommended action. Hence, the key question is how to balance utility and privacy carefully.

This has motivated studies of linear contextual bandits under different trust models of differential privacy (i.e., who the user will trust with her sensitive data). On one end of the spectrum lies the *central* model, which guarantees privacy to users who trust the learning agent to store their raw data in the server and use those to update its strategy of recommending actions. Under this trust model, Shariff & Sheffet (2018) has shown that the cumulative regret is $\widetilde{O}(\frac{\sqrt{T}(\log(1/\delta))^{1/4}}{\sqrt{\varepsilon}})$, where ε and δ are privacy parameters with smaller values denoting higher level of protection. Perhaps unsurprisingly, this regret bound – due to the high degree of trust – matches the optimal $\Theta(\sqrt{T})$ scaling for non-private linear contextual bandits (Chu et al., 2011). However, this relatively high trust model is not always feasible since the users may not trust the agent at all. This is captured by the *local* model, where any data sent by the users must already be private, and the agent can only store those randomized data in the server. This is a strictly stronger notion of privacy, and hence, often comes at a price. Under this trust model, Zheng et al. (2020) has shown that the cumulative regret is $\widetilde{O}(\frac{T^{3/4}(\log(1/\delta))^{1/4}}{\sqrt{\varepsilon}})$, which, as expected, is much worse than that in the central model. This naturally leads to the following question:

Can a finer trade-off between privacy and regret in linear contextual bandits be achieved?

Furthermore, in both Shariff & Sheffet (2018) and Zheng et al. (2020), the learning agents update their strategy at every round. This not only puts excessive computational burden on the server (due to T updates each taking at least $O(d^2)$ time and memory) but also could be practically infeasible at times. For example, consider the above mobile health application. The cloud server is often infeasible to update the algorithm deployed in mobile app after interactions with each single user. Rather, a more practical strategy is to update the algorithm after collecting a batch of users' data (e.g., a one-month period of data).

Motivated by these, we consider the linear contextual bandit problem under an intermediate trust model of differential privacy, known as the *shuffle* model (Cheu et al., 2019; Erlingsson et al., 2019) in the hope to attain a finer regret-privacy trade-off, while only using batch updates. In this new trust model, there exists a shuffler between users and the central server which permutes a *batch* of users' randomized data before they are viewed by the server so that it can't distinguish between two users' data. Shuffling thus adds an another layer of protection by decoupling data from the users that sent them. Here, as in the local model, the users don't trust the server. However, it is assumed that they have a certain degree of trust in the shuffler since it can be efficiently implemented using cryptographic primitives (e.g., mixnets) due to its simple operation (Bittau et al., 2017; Apple, 2017). The shuffle model provides the possibility to achieve a stronger privacy guarantee than the central model while suffering a smaller utility loss than the local model. The key intuition behind this is that the additional randomness of the shuffler creates a *privacy blanket* (Balle et al., 2019b) so that each user now needs much less random noise to hide her information in the crowd. Indeed, the shuffle model achieves a better trade-off between utility and privacy as compared to central and local model in several learning problems such as empirical risk minimization (Girgis et al., 2021), stochastic convex optimization (Lowy & Razaviyayn, 2021; Cheu et al., 2021), and standard multi-arm bandits (Tenenbaum et al., 2021). However, little is known about (linear) contextual bandits in the shuffle model due to its intrinsic challenges. That is, in addition to rewards, the contexts are also sensitive information that need to be protected, which not only results in the aforementioned large gap in regret between local and central model[1], but also leads to new challenges in the shuffle model. Against this backdrop, we make the following contributions:

- We design a general algorithmic framework (Algorithm 1) for private linear contextual bandits in the shuffle model. It decomposes the learning process into three black-box components: a local randomizer at each user, an analyzer at the central server and a shuffler in-between. We instantiate the framework with two specific shuffle protocols. The first one directly builds on privacy amplification of existing local mechanisms. The other one utilizes an efficient mechanism for summing vectors with bounded ℓ_2 norms.
- We show that both shuffle protocols provide stronger privacy protection compared to the central model. Furthermore, when all users are *unique*, we prove a regret bound of $\widetilde{O}\left(T^{3/5}\right)$ for both the protocols, which improves over the $\widetilde{O}\left(T^{3/4}\right)$ regret of local model. Hence, we achieve a finer trade-off between regret and privacy. We further perform simulations on synthetic data that corroborate our theoretical results.
- As a practical application of our general framework, we show that under the setting of non-unique or *returning* users, the regret of both our shuffle protocols *matches* the one that the central model would achieve in the

[1] In contrast, for MAB, the problem-independent upper bounds in the local and central model are both $\widetilde{O}(\sqrt{T})$ (Ren et al., 2020a).

same setting. This, along with the fact that both shuffle protocols also offer a certain degree of local privacy, further elaborate usefulness of shuffle model in private linear contextual bandits.

Related work. Due to the utility gap present between central and local models, a significant body of recent work have focused on the shuffle model (Balle et al., 2019b; Feldman et al., 2020; Ghazi et al., 2019; Balle et al., 2019a). A nice overview of recent work in the shuffle model is presented in Cheu (2021). Regret performance of multi-armed bandit algorithms under central and local trust models have been considered in Mishra & Thakurta (2015); Sajed & Sheffet (2019); Ren et al. (2020a); Chen et al. (2020); Zhou & Tan (2020); Dubey (2021); Tossou & Dimitrakakis (2017), whereas online learning algorithms under full information have appeared in Guha Thakurta & Smith (2013); Agarwal & Singh (2017). Recently, the two models have also been adopted to design differentially private control and reinforcement learning algorithms (Vietri et al., 2020; Garcelon et al., 2020; Chowdhury et al., 2021; Chowdhury & Zhou, 2021). Han et al. (2021) consider linear bandits with stochastic contexts, and show that $\widetilde{O}(\sqrt{T}/\varepsilon)$ regret can be achieved even in the local model. In contrast, in this work, we allow the contexts to be arbitrary and can even be adversarially generated, which pose additional challenges.

Batched linear bandits are studied in Han et al. (2020); Ren et al. (2020b), where the authors show that only $O(\sqrt{T})$ model update is sufficient to achieve corresponding minimax optimal regrets. In the shuffle private model, batched learning not only reduces the model update frequency, but more importantly plays a key role in amplifying privacy via shuffling a batch of users' data. Interestingly, as a by-product, our established generic regret bound also improves over the non-private one in Ren et al. (2020b) in the sense that no restriction is required for the regularizer.

Concurrent and independent work. While preparing this submission, we have noticed that Garcelon et al. (2021) also study linear contextual bandits in the shuffle model. The authors claim that a single fixed batch schedule is not sufficient to obtain a better regret-privacy trade-off in shuffle model. They propose to use separate asynchronous schedules – a fixed batch scheme for the shuffler and an adaptive model update scheme for the server. In contrast, thanks to a tighter analysis, we show that *a single fixed batch schedule* is indeed sufficient to attain the same regret-privacy trade-off in shuffle model. Moreover, we believe, there exists a fundamental gap in their analysis for the adaptive model update, which might make their results ungrounded. We provide a detailed discussion on this in Section 6, which highlights the key difference in dealing with adaptive update in the non-private and the private settings. Finally, in addition to the above differences in theoretical results, our established generic framework enables to design flexible shuffle private protocols for linear contextual bandits that are able to handle a wide range of practically interested privacy budget ε rather than a restricted small value $\varepsilon \ll 1$ in the concurrent work (Garcelon et al., 2021).

2. Privacy in the Shuffle Model

In this section, we introduce the shuffle model, and its corresponding privacy notion called the *shuffle differential privacy*. Before that, we recall definitions of differential privacy under central and local models (Dwork et al., 2014).

2.1. Central and Local Differential Privacy

Throughout, we let $\mathcal{D}$ denote the data universe, and $n \in \mathbb{N}$ the number of (unique) users. Let $D_i \in \mathcal{D}, i = 1, 2, \ldots, n$, denote the data point of user i, and $D_{-i} \in \mathcal{D}^{n-1}$ denote collection of data points of all but the i-th user. Let $\varepsilon > 0$ and $\delta \in (0, 1]$ be given privacy parameters.

Definition 2.1 (Differential Privacy (DP))**.** A mechanism $\mathcal{M}$ satisfies (ε, δ)-DP if for each user $i \in [n]$, each data set $D, D' \in \mathcal{D}^n$, and each event $\mathcal{E}$ in the range of $\mathcal{M}$,

$$\mathbb{P}\left[\mathcal{M}(D_i, D_{-i}) \in \mathcal{E}\right] \leqslant \exp(\varepsilon)\mathbb{P}\left[\mathcal{M}(D'_i, D_{-i}) \in \mathcal{E}\right] + \delta.$$

Definition 2.2 (Local Differential Privacy (LDP))**.** A mechanism $\mathcal{M}$ satisfies (ε, δ)-LDP if for each user $i \in [n]$, each data point $D_i, D'_i \in \mathcal{D}$ and each event $\mathcal{E}$ in the range of $\mathcal{M}$,

$$\mathbb{P}\left[\mathcal{M}(D_i) \in \mathcal{E}\right] \leqslant \exp(\varepsilon)\mathbb{P}\left[\mathcal{M}(D'_i) \in \mathcal{E}\right] + \delta.$$

Roughly speaking, a central DP (or, simply, DP) mechanism ensures that the outputs of the mechanism on two neighbouring data sets (i.e., those differ only on one user) are approximately indistinguishable. In contrast, local DP ensures that the output of the local mechanism for each user is indistinguishable.

2.2. Shuffle Differential Privacy

A (standard) shuffle protocol $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ consists of three parts: (i) a (local) randomizer $\mathcal{R}$, (ii) a shuffler $\mathcal{S}$ and (iii) an analyzer $\mathcal{A}$. For n users, the overall protocol works as follows. Each user i first applies the randomizer on its raw data D_i and then sends the resulting messages $\mathcal{R}(D_i)$ to the shuffler. The shuffler $\mathcal{S}$ permutes messages from all the users uniformly at random and then reports the permuted messages $\mathcal{S}(\mathcal{R}(D_1), \ldots, \mathcal{R}(D_n))$ to the analyzer. Finally, the analyzer $\mathcal{A}$ computes the output using received messages. In this protocol, the users trust the shuffler but not the analyzer. Hence, the privacy objective is to ensure that the outputs of the shuffler on two neighbouring datasets are indistinguishable in the analyzer's view. To this end, define the mechanism $(\mathcal{S} \circ \mathcal{R}^n)(D) := \mathcal{S}(\mathcal{R}(D_1), \ldots, \mathcal{R}(D_n))$, where $D \in \mathcal{D}^n$.

Definition 2.3 (Shuffle differential privacy (SDP)). A protocol $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ for n users satisfies (ε, δ)-SDP if the mechanism $\mathcal{S} \circ \mathcal{R}^n$ satisfies (ε, δ)-DP.

To achieve benefits of the shuffle model in intrinsically adaptive algorithms (e.g., gradient descent, multi-armed bandits etc.), one needs to divide the users into multiple batches, and run a potentially different shuffle protocol on each batch (Cheu et al., 2021; Tenenbaum et al., 2021). This is quite natural since the shuffler needs enough users' data to infuse sufficient randomness so as to amplify the privacy. Moreover, each protocol might depend on the output of the preceding protocols to foster adaptivity. Formally, a general M-batch, $M \in \mathbb{N}$, shuffle protocol $\mathcal{P}$ for n users works as follows. In each batch m, we simply run a standard single-batch shuffle protocol for a subset of n_m users (such that $n = \sum_m n_m$) with randomizer $\mathcal{R}_m$, shuffler $\mathcal{S}$ and analyzer $\mathcal{A}$. To ensure adaptivity, the randomizer $\mathcal{R}_m$ and number of users n_m for the m-th batch could be chosen depending on outputs of the shuffler from all the previous batches, given by $\left\{\mathcal{S}\left(\mathcal{R}_{m'}(D_1), \ldots, \mathcal{R}_{m'}(D_{n_{m'}})\right)\right\}_{m'<m}$. The objective of privacy is same as in the single-batch protocol – the analyzer's view must satisfy DP. However, instead of a single-batch output, one need to protect outputs of all the M batches. To this end, define the (composite) mechanism $\mathcal{M}_{\mathcal{P}} = (\mathcal{S} \circ R_1^{n_1}, \ldots, \mathcal{S} \circ \mathcal{R}_m^{n_m})$, where each individual mechanism $\mathcal{S} \circ \mathcal{R}_m^{n_m}$ operates on n_m users' data, i.e., on datasets from $\mathcal{D}^{n_m}$.

Definition 2.4 (M-batch SDP). An M-batch shuffle protocol $\mathcal{P}$ is (ε, δ)-SDP if the mechanism $\mathcal{M}_{\mathcal{P}}$ is (ε, δ)-DP.

Algorithm 1 Shuffle Private LinUCB

Parameters: Batch size $B \in \mathbb{N}$, regularization $\lambda > 0$, confidence radii $\{\beta_m\}_{m \geqslant 0}$, feature map $\phi : \mathcal{C} \times \mathcal{X} \to \mathbb{R}^d$
Initialize: Batch counter $m = 1$, end-time $t_0 = 0$, batch statistics $V_0 = \lambda I_d$, $u_0 = 0$, parameter estimate $\widehat{\theta}_0 = 0$
for local user $t = 1, 2, \ldots$ **do**
  Observe user's context information $c_t \in \mathcal{C}$
  Choose action $a_t \in \operatorname{argmax}_{a \in \mathcal{X}} \langle \phi(c_t, a), \widehat{\theta}_{m-1} \rangle + \beta_{m-1} \|\phi(c_t, a)\|_{V_{m-1}^{-1}}$
  Observe reward y_t
  # For the local randomizer:
  Send randomized messages $M_{t,1} = R_1(\phi(c_t, a_t) y_t)$ and $M_{t,2} = R_2(\phi(c_t, a_t)\phi(c_t, a_t)^\top)$ to the shuffler
  if $t = mB$ **then**
    # For the shuffler:
    Set batch end-time: $t_m = t$
    Permute all received messages uniformly at random $Y_{m,1} = S_1\left(\{M_{\tau,1}\}_{t_{m-1}+1 \leq \tau \leq t_m}\right)$ and $Y_{m,2} = S_2\left(\{M_{\tau,2}\}_{t_{m-1}+1 \leq \tau \leq t_m}\right)$
    # For the analyzer (server):
    Compute per-batch statistics $\widetilde{u}_m = A_1(Y_{m,1})$ and $\widetilde{V}_m = A_2(Y_{m,2})$ using shuffled messages
    Update overall batch statistics: $u_m = u_{m-1} + \widetilde{u}_m$, $V_m = V_{m-1} + \widetilde{V}_m$
    Compute parameter estimate $\widehat{\theta}_m = V_m^{-1} u_m$
    Send updated models $(\widehat{\theta}_m, V_m)$ to users
    Increase batch counter: $m = m + 1$
  end if
end for

3. A Shuffle Algorithm for Contextual Bandits

In this section, we introduce a general algorithmic framework (Algorithm 1) for linear contextual bandits under the shuffle model. We build on the celebrated LinUCB algorithm (Chu et al., 2011; Abbasi-Yadkori et al., 2011), which is an application of the *optimism in the face of uncertainty* principle to linear bandits. Throughout the paper, we make the following assumptions, which are standard in the literature (Chu et al., 2011; Shariff & Sheffet, 2018).

Assumption 3.1 (Boundedness). The rewards are bounded for all t, i.e., $y_t \in [0, 1]$. Moreover, the parameter vector and the features have bounded norm, i.e., $\|\theta^*\|_2 \leqslant 1$ and $\sup_{c,a} \|\phi(c, a)\|_2 \leqslant 1$.[2]

3.1. Algorithm: Shuffle Private LinUCB

Our shuffle algorithm for contextual bandits consist of batches with a fixed size B, i.e., we have total $M = T/B$ batches.[3] The central idea is to construct, for each batch m, a d-dimensional ellipsoid $\mathcal{E}_m$ with centre $\widehat{\theta}_m$, shape matrix V_m and radius β_m so that it contains the unknown parameter θ^* with high probability. Moreover, the ellipsoids are designed while keeping the privacy setting in mind. They depend on the randomizer, shuffler and analyzer employed in the shuffle protocol based on required privacy levels ε, δ. The personal data of user t in batch m is given by the feature vector $\phi(c_t, a_t)$ and reward y_t, where the action a_t is selected given the context c_t as

$$a_t \in \operatorname*{argmax}_{a \in \mathcal{X}} \{\langle \phi(c_t, a), \widehat{\theta}_{m-1} \rangle + \beta_{m-1} \|\phi(c_t, a)\|_{V_{m-1}^{-1}}\}.$$

We consider a fixed randomizer across all the batches given by two functions R_1 and R_2 that locally operate on the vectors $\phi(c_t, a_t) y_t$ and matrices $\phi(c_t, a_t)\phi(c_t, a_t)^\top$, respectively. Similarly, we have shuffler functions S_1 and S_2 operating on batches (of size B) of those respective randomized messages. Finally, the analyzer functions A_1 and A_2 receive permuted messages from S_1 and S_2, and output, for each batch m', an aggregate vector $\widetilde{u}_{m'}$ and matrix $\widetilde{V}_{m'}$, respectively. The central server uses this aggregate batch statistics to construct the ellipsoid: $V_m = \lambda I_d + \sum_{m'=1}^{m} \widetilde{V}_{m'}$

[2] All terms are assumed to be bounded by one via normalization.
[3] We assume, wlog, total number of rounds T is multiple of B.

and $\widehat{\theta}_m = V_m^{-1} \sum_{m'=1}^{M} \widetilde{u}_{m'}$. For a given confidence level $\alpha \in (0,1]$, the radius of the ellipsoid is set as $\beta_m = O\left(\sqrt{2\log\left(\frac{2}{\alpha}\right) + d\log\left(1+\frac{t_m}{d\lambda}\right)} + \sqrt{\lambda}\right)$, where t_m is the time when batch m ends. The regularizer λ and thus, in turn, the confidence radius β_m typically depend on the total noise infused in the shuffle protocol. On a high level, these randomizer, shuffler and analyzer functions together provide suitable random perturbations to the Gram matrices and feature-reward vectors based on the privacy budget ε, δ, and in turn, they affect the regret performance via the noise levels of these perturbations. Next, we turn to discuss specific choices of these functions, and the associated performance guarantees of Algorithm 1 under those choices.

3.2. Achieving SDP via LDP Amplification

In this section, we show that our general framework (Algorithm 1) enables us to directly utilize existing LDP mechanisms for linear contextual bandits to achieve a finer utility-privacy trade-off. The key idea here is to leverage the explicit privacy amplification property of the shuffle protocol (Feldman et al., 2020). Roughly, the privacy guarantee can be amplified by a factor of $\sqrt{B}$ by randomly permuting the output of an LDP mechanism independently operating on a batch of B different users. In other words, the same level of privacy can be achieved for each user by adding a $\sqrt{B}$ factor *less* noise in the presence of shuffler, yielding a better utility. Specifically, we instantiate Algorithm 1 with the shuffle protocol $\mathcal{P}_{\text{Amp}} = (\mathcal{R}_{\text{Amp}}, \mathcal{S}_{\text{Amp}}, \mathcal{A}_{\text{Amp}})$, where we employ standard Gaussian mechanism (Dwork et al., 2014) as randomizer functions. Essentially, we inject independent Gaussian perturbation to each entry of the vector $\phi(c_t, a_t) y_t$ and the matrix $\phi(c_t, a_t)\phi(c_t, a_t)^\top$ with variances σ_1^2 and σ_2^2, respectively. We make sure the noisy matrix is symmetric by perturbing upper diagonal entries, and copying those to the lower terms. The noise variances are properly tuned depending on the sensitivity of these elements to achieve desired level of privacy. In this case, the shuffler functions simply permute its data uniformly at random, and the job of the analyzer is to simply add its received data (i.e., vectors or matrices). We defer further details on the protocol $\mathcal{P}_{\text{Amp}}$ to Appendix B and focus on performance guarantees first.

Theorem 3.2 (Performance under LDP amplification)**.** *Fix time horizon $T \in \mathbb{N}$, batch size $B \in [T]$, confidence level $\alpha \in (0,1]$, privacy budgets $\delta \in (0,1]$, $\varepsilon \in (0, \sqrt{\frac{\log(2/\delta)}{B}}]$. Then, Algorithm 1 instantiated using shuffle protocol $\mathcal{P}_{Amp}$ with noise $\sigma_1 = \sigma_2 = \frac{4\sqrt{2\log(2.5B/\delta)\log(2/\delta)}}{\varepsilon\sqrt{B}}$, and regularizer $\lambda = \Theta(\sqrt{T}\sigma_1(\sqrt{d} + \sqrt{\log(T/B\alpha)}))$, enjoys the regret*

$$Reg(T) = O\left(dB\log T + \frac{\log^{1/2}(B/\delta)}{\varepsilon^{1/2}B^{1/4}} d^{3/4}T^{3/4}\log^2(T/\alpha)\right),$$

with probability at least $1-\alpha$. Moreover, it satisfies $O(\varepsilon, \delta)$-shuffle differential privacy (SDP).

Corollary 3.3. *Setting batch size $B = O(T^{3/5})$ in Algorithm 1, we can achieve regret $\widetilde{O}\left(\frac{T^{3/5}}{\sqrt{\varepsilon}}\log^{1/2}(T/\delta)\right)$.*[4]

Comparsion with central and local DP models. At this point, we turn to compare the regret of our Shuffle Private LinUCB algorithm to that of LinUCB under central model with JDP guarantee[5] (Shariff & Sheffet, 2018) and local model with LDP (Zheng et al., 2020) guarantee. As mentioned before, LinUCB achieves $\tilde{O}\left(\sqrt{\frac{T}{\varepsilon}}\right)$ and $\tilde{O}\left(\frac{T^{3/4}}{\sqrt{\varepsilon}}\right)$ regret under JDP and LDP guarantees, respectively. As seen in Corollary 3.3, our regret bound in the shuffle trust model lies perfectly in between these two extremes. Importantly, it improves over the $T^{3/4}$ scaling in the (stronger) local trust model, achieving a better trade-off between regret and privacy. However, it couldn't achieve the optimal $\sqrt{T}$ scaling in the (weaker) central trust model. It remains an open question whether $\sqrt{T}$ regret can be achieved under any notion of privacy stronger than the central model.

Remark 3.4. Our shuffle protocol $\mathcal{P}_{\text{Amp}}$, by design, provides a certain level of local privacy to each user. Specifically, for batch size B, Algorithm 1 is $O(\varepsilon\sqrt{B/\log(2/\delta)}, \delta/B)$-LDP. Furthermore, since shuffe model ensures a higher level of trust than the central model, Algorithm 1 is also $O(\varepsilon, \delta)$-JDP. See Appendix B for details.

Apart from achieving a refined utility-privacy trade-off, the above shuffle protocol $\mathcal{P}_{\text{Amp}}$ requires minimum modifications over existing LDP mechanisms. However, the privacy guarantee in Theorem 3.2 holds only for small privacy budget ε particularly when the batch size B is large, which could potentially limit its application in some practical scenarios (e.g., when ε is around 1 or larger (Apple, 2017)). Moreover, $\mathcal{P}_{\text{Amp}}$ needs to communicate and shuffle *real* vectors and matrices, which are often difficult to encode on finite computers in practice (Canonne et al., 2020; Kairouz et al., 2021) and a naive use of finite precision approximation may lead to a possible failure of privacy protection (Mironov, 2012). To overcome these limitations of $\mathcal{P}_{\text{Amp}}$, we introduce a different instantiation of Algorithm 1 in the next section.

3.3. Achieving SDP via Vector Summation

We instantiate Algorithm 1 with the shuffle protocol $\mathcal{P}_{\text{Vec}} = (\mathcal{R}_{\text{Vec}}, \mathcal{S}_{\text{Vec}}, \mathcal{A}_{\text{Vec}})$, where we rely on a particularly efficient and accurate mechanism for summing vectors with bounded ℓ_2 norms (Cheu et al., 2021). First, the local randomizer $\mathcal{R}_{\text{Vec}}$ adopts a one-dimensional randomizer that operates

[4] Note that with a careful choice of B (depending on privacy parameters ε, δ), we can have a better regret dependence on ε, δ. See Corollary B.2 for details.

[5] JDP, or, joint differential privacy, is a notion of privacy under central trust model specific to contextual bandits. See Appendix D.

independently on each entry of the vector $\phi(c_t, a_t)y_t$ and the matrix $\phi(c_t, a_t)\phi(c_t, a_t)^\top$, respectively. This adopted one-dimensional randomizer transmits only bits $(0/1)$ via a fixed-point encoding scheme (Cheu et al., 2019), and ensures privacy by injecting binomial noise. In particular, given any entry $x \in [0, 1]$, it is first encoded as $\widehat{x} = \bar{x} + \gamma_1$, using an accuracy parameter $g \in \mathbb{N}$, where $\bar{x} = \lfloor xg \rfloor$ and $\gamma_1 \sim \texttt{Ber}(xg - \bar{x})$. Then a binomial noise is generated, $\gamma_2 \sim \texttt{Bin}(b, p)$, where parameters $b \in \mathbb{N}, p \in (0, 1)$ control the privacy noise. The output of the one-dimensional randomizer is simply a collection of total $g + b$ bits, in which $\widehat{x} + \gamma_2$ bits are 1 and the rest are 0. Combining the outputs of the one-dimensional randomizer for each entry of vector $\phi(c_t, a_t)y_t$ and matrix $\phi(c_t, a_t)\phi(c_t, a_t)^\top$, yield final outputs of randomizer. The shuffler functions in $\mathcal{S}_{\text{Vec}}$ simply permutes all the received bits uniformly at random. The job of the analyzer $\mathcal{A}_{\text{Vec}}$ is to add the received bits for each entry, and remove the bias introduced due to encoding and binomial noise. This is possible since bits are already labeled entry-wise when leaving $\mathcal{R}_{\text{Vec}}$. The constants g, b, p are left as tunable parameters of $\mathcal{P}_{\text{Vec}}$, and need to be set properly depending on the desired level of privacy. The detailed implementation of this scheme is deferred to Appendix C. The following theorem states the performance guarantees of Algorithm 1 instantiated with $\mathcal{P}_{\text{Vec}}$.

Theorem 3.5 (Performance under vector sum)**.** *Fix batch size $B \in [T]$, privacy budgets $\varepsilon \in (0, 15]$, $\delta \in (0, 1/2)$. Then, Algorithm 1 instantiated with $\mathcal{P}_{\text{Vec}}$ with parameters $p = 1/4$, $g = \max\{2\sqrt{B}, d, 4\}$ and $b = \frac{C \cdot g^2 \cdot \log^2\left(d^2/\delta\right)}{\varepsilon^2 B}$ is (ε, δ)-SDP, where $C \gg 1$ is some sufficiently large constant. Furthermore, for any $\alpha \in (0, 1]$, setting $\lambda = \Theta\left(\frac{\log(d^2/\delta)\sqrt{T}}{\varepsilon\sqrt{B}}(\sqrt{d} + \sqrt{\log(T/B\alpha)})\right)$, it enjoys the regret*

$$\textit{Reg}(T) = O\left(dB \log T + \frac{\log^{1/2}(d^2/\delta)}{\varepsilon^{1/2} B^{1/4}} d^{3/4} T^{3/4} \log^2(T/\alpha)\right),$$

with probability at least $1 - \alpha$.

Remark 3.6. Similar to Corollary 3.3, an $\widetilde{O}\left(\frac{T^{3/5}}{\sqrt{\varepsilon}}\right)$ regret can also be achieved in this case by setting $B = O(T^{3/5})$, but the dependence on δ is now: $\log^{1/2}(d^2/\delta)$ as compared to $\log^{1/2}(T/\delta)$. Moreover, in contrast to Theorem 3.2, the guarantees hold for a wide range of ε, making $\mathcal{P}_{\text{Vec}}$ better suitable for practical purposes (Apple, 2017). Finally, as before, if B also depends on privacy parameters, the dependence on ε, δ can be improved, see Corollary C.2.

Remark 3.7. $\mathcal{P}_{\text{Vec}}$ can also be regarded as privacy amplification of Binomial mechanism (rather than Gaussian mechanism in $\mathcal{P}_{\text{Amp}}$), which is the reason that it also offers a certain degree, $O(\varepsilon\sqrt{B}, \delta)$, to be precise, of LDP guarantee.

Remark 3.8. Both shuffle protocols, $\mathcal{P}_{\text{Amp}}$ and $\mathcal{P}_{\text{Vec}}$, in fact, can be tuned to satisfy (ε, δ)-LDP by sacrificing on regret performance. See Corollaries B.3 and C.3 for details.

3.4. Key Techniques: Overview

In this section, we provide a generic template of regret bound for linear contextual bandits under the shuffle model of privacy. To this end, we need following notations to discuss the effect of noise added by shuffle protocol, in the learning process. Let $n_m = \widetilde{u}_m - \sum_{t=t_{m-1}+1}^{t_m} \phi(c_t, a_t)y_t$ and $N_m = \widetilde{V}_m - \sum_{t=t_{m-1}+1}^{t_m} \phi(c_t, a_t)\phi(c_t, a_t)^\top$ denote the total noise added during batch m in the feature-reward vector, and in the Gram-matrix, respectively. Furthermore, assume that there exist constants $\widetilde{\sigma}_1$ and $\widetilde{\sigma}_2$ such that for each batch m, (i) $\sum_{m'=1}^{m} n_{m'}$ is a random vector whose entries are independent, mean zero, sub-Gaussian with variance at most $\widetilde{\sigma}_1^2$, and (ii) $\sum_{m'=1}^{m} N_{m'}$ is a random symmetric sub-Gaussian matrix whose entries on and above the diagonal are independent with variance at most $\widetilde{\sigma}_2^2$. Let $\sigma^2 = \max\{\widetilde{\sigma}_1^2, \widetilde{\sigma}_2^2\}$. Then, we have the following result.

Lemma 3.9 (Informal)**.** *With the choice of $\lambda \approx \sigma(\sqrt{d} + \sqrt{\log(T/(B\alpha))})$, the regret of Algorithm 1 satisfies*

$$\textit{Reg}(T) = \widetilde{O}\left(dB + d\sqrt{T} + \sqrt{\sigma T} d^{3/4}\right) \textit{ with high probability.}$$

With the above result, one only needs to determine the noise variance σ^2 under different privacy protocols. We illustrate this with the shuffle protocols introduced in previous sections. First, note that since we assume unique users, Algorithm 1 is SDP if each batch is SDP. Now, for the LDP amplification protocol $\mathcal{P}_{\text{Amp}}$, in order to guarantee SDP for each batch with sufficiently small privacy loss ε, it suffices to work with an LDP mechanism with loss $\varepsilon\sqrt{B}$ by virtue of amplification.[6] We ensure this by choosing Gaussian mechanism with noise variance $O(1/(\varepsilon^2 B))$. Hence, the total noise variance added by $\mathcal{P}_{\text{Amp}}$ is $\sigma^2 \approx O(\frac{T}{\varepsilon^2 B})$. Thus, by Lemma 3.9, we obtain the result in Theorem 3.2. Similarly, for the vector sum protocol $\mathcal{P}_{\text{Vec}}$, we ensure $\mathcal{P}_{\text{Vec}}$ to be SDP by properly setting parameters g, b, p. Moreover, the analyzer's outputs are unbiased estimates of the sum of the non-private vectors (matrices) within that batch, and the entry-wise private noise is sub-Gaussian with variance of $O(\frac{1}{\varepsilon^2})$. Thus, the total noise variance added by $\mathcal{P}_{\text{Vec}}$ is $\sigma^2 \approx O(\frac{T}{\varepsilon^2 B})$, and hence, by Lemma 3.9, we have the result in Theorem 3.5.

Remark 3.10. Lemma 3.9, in fact, can serve as a general template of regret for private linear contextual bandit algorithms. For example, for the local model (Zheng et al., 2020), $B = 1$ and $\sigma^2 \approx \frac{T}{\varepsilon^2}$, yielding $\widetilde{O}\left(\frac{T^{3/4}}{\sqrt{\varepsilon}}\right)$ regret. Similarly, for the central model (Shariff & Sheffet, 2018), $B = 1$ and $\sigma^2 \approx \frac{\log T}{\varepsilon^2}$, which yields $\widetilde{O}\left(\frac{T^{1/2}}{\sqrt{\varepsilon}}\right)$ regret.

[6] We provide intuition without worrying about the details related to δ-dependent terms. Refer to Appendix A for formal proofs.

4. Regret Performance under Returning Users

Similar to existing work on differentially private bandits, in previous sections, we have assumed that all participating users are unique, i.e., each user participates in the protocol only at one round. A more practical scenario is that an user can contribute with her data at multiple rounds. For example, consider the mobile medical application described in the introduction. The cloud server can collect one particular user's data during multiple batches to track the effectiveness of its treatment plan over a period, and hence, use same user's data multiple times to update its recommendation algorithm. Motivated by this, we provide privacy and regret guarantees of Algorithm 1 under the setting of *returning users* in linear contextual bandits. We first define the setting of returning users that we consider in this section, and then state the performance guarantee for Algorithm 1.

Assumption 4.1 (Returning Users)**.** For a given time horizon $T \in \mathbb{N}$ and batch size $B \in [T]$, any user can participate in *all* $M = T/B$ batches, but within each batch $m \in [M]$, she only contributes once.

In addition to the above motivating example, this assumption also captures many practical adaptive learning scenarios such as clinical trials and product recommendations, in which each trial (batch) involves a group of unique people, but the same person may participate in multiple trials (Ren et al., 2020b; Schwartz et al., 2017).

Theorem 4.2 (Performance guarantees (informal))**.** *Under Assumption 4.1, we obtain the following results for $\mathcal{P}_{Amp}$ and $\mathcal{P}_{Vec}$, respectively.*

(i) For any $\varepsilon \leqslant \frac{2}{B}\log(2/\delta)\sqrt{2T}$ and $\delta \in (0,1]$, Algorithm 1 instantiated using $\mathcal{P}_{Amp}$ with noise levels $\sigma_1 = \sigma_2 = \frac{16\log(2/\delta)\sqrt{T(\log(5T/\delta))}}{\varepsilon B}$ is $O(\varepsilon,\delta)$-SDP, and enjoys, with high-probability, the regret bound

$$Reg(T) = \widetilde{O}\left(\frac{dT}{M} + \sqrt{\frac{MT}{\varepsilon}} d^{3/4}\log^{3/4}(T/\delta)\right).$$

(ii) For any $\varepsilon \leqslant 15$, $\delta \in (0,1/2)$, there exist choices of parameters $g, b \in \mathbb{N}$, $p \in (0,1/2)$ depending on B, ε, δ such that Algorithm 1 instantiated using $\mathcal{P}_{Vec}$ is (ε,δ)-SDP, and, enjoys, with high probability, the regret bound

$$Reg(T) = \widetilde{O}\left(\frac{dT}{M} + \sqrt{\frac{MT}{\varepsilon}} d^{3/4}\log^{3/4}(d^2 M/\delta)\right).$$

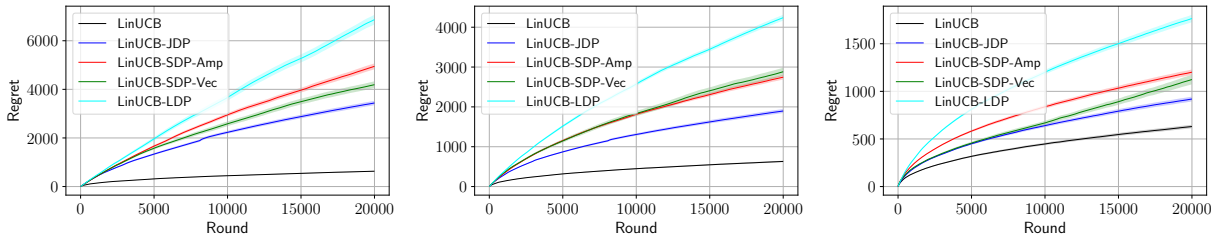
Proof sketch. In contrast to Section 3 for unique users, where (ε,δ)-SDP guarantee for Algorithm 1 can be established by showing each batch is (ε,δ)-SDP, we now need to guarantee that outputs of all the batches together have a total privacy loss of (ε,δ). This is due to the fact that now each batch can potentially operate on same set of users, and hence, one need to use advanced decomposition to calculate the total privacy loss. This leads to scaling up the noise variance by a multiplicative factor of $O(M)$ at each batch, which eventually leads to the above bound (the additional M factor in δ also comes from advance composition). □

Interestingly, the privacy (ε,δ)-dependent term in above regret bounds match the one that can be achieved in the *user-level* central trust model that handles returning users. Note that, since existing work in the central model of privacy (i.e., under JDP guarantee) assume unique users (Shariff & Sheffet, 2018), we first generalize it to handle returning users. This can be viewed as the same form of generalization from *event-level* DP to *user-level* DP under continual observation, where the adjacent relation between two data streams changes from the flip of one single round to the flip of multiple rounds associated with a single user (Dwork et al., 2010). See Appendix D for formal definitions of event-level and user-level joint differential privacy (JDP). As in standard notion of DP, one straightforward approach for converting event-level JDP to user-level JDP is to use group privacy (Dwork et al., 2014). However, this black-box approach would blow up the terms dependent on δ. To overcome this, we propose a simple modification of original (event-level) algorithm in Shariff & Sheffet (2018) so that it can handle returning users. In particular, user-level JDP can be achieved by scaling up the noise variance by a multiplicative factor of M_0^2, if any user participates in at most M_0 rounds. This follows from the fact that flipping one user now would change the ℓ_2 sensitivity of the expanded binary-tree nodes from $O(\sqrt{\log T})$ to $O(M_0\sqrt{\log T})$. Note that we use M_0 to distinguish from the number of batches M since there is no batch concept in standard central model. This modified version enjoys the following regret guarantee.

Proposition 4.3. *If any user participates in at most M_0 rounds, the algorithm in Shariff & Sheffet (2018), with the above modification to handle user-level privacy, achieves the high-probability regret bound*

$$Reg(T) = \widetilde{O}\left(d\sqrt{T} + \sqrt{\frac{M_0 T}{\varepsilon}} d^{3/4}\log^{1/4}(1/\delta)\right).$$

Remark 4.4. Comparing Theorem 4.2 and Proposition 4.3, we observe that the cost of privacy in the shuffle model is essentially same (upto a log factor) as in the central model under the setting of returning users. In particular, if $M = M_0 = T^{1/3}$ rounds (i.e., the same number of possible returning rounds for any user), the regret is $\widetilde{O}\left(\frac{T^{2/3}}{\sqrt{\varepsilon}}\right)$ in both shuffle and user-level central trust models. See Appendix E for complete proofs and more details.



(a) $\varepsilon = 0.2$ (b) $\varepsilon = 1$ (c) $\varepsilon = 10$

Figure 1: Comparison of cumulative regret for LinUCB (non-private), LinUCB-JDP (central model), LinUCB-SDP (shuffle model) and LinUCB-LDP (local model) with varying privacy level $\varepsilon = 0.2$ (a), $\varepsilon = 1$ (b) and $\varepsilon = 10$ (c). For $\varepsilon = 0.2$ (higher privacy level), gap between private and non-private regret is higher as compared to $\varepsilon = 10$ (lower privacy level). In all cases, regret of LinUCB-SDP lies perfectly in between LinUCB-JDP and LinUCB-LDP, achieving finer regret-privacy trade-off.

5. Simulation Results

In this section, we empirically evaluate the regret performance of Algorithm 1 (under shuffle model), which we abbreviate as LinUCB-SDP-Amp and LinUCB-SDP-Vec when instantiated with $\mathcal{P}_{\text{Amp}}$ and $\mathcal{P}_{\text{Vec}}$, respectively. We compare them with the algorithms of Shariff & Sheffet (2018) and Zheng et al. (2020) under central and local models, which we call LinUCB-JDP and LinUCB-LDP, respectively. We benchmark these against the non-private algorithm of Abbasi-Yadkori et al. (2011), henceforth referred as LinUCB. For all the experiments, we consider 100 arms, set $T = 20000$ rounds, and average our results over 50 randomly generated bandit instances. Each instance is characterized by an (unknown) parameter θ^* and feature vectors of dimension $d = 5$. To ensure boundedness, similar to Vaswani et al. (2020), we generate each θ^* and feature vectors by sampling a $(d-1)$-dimensional vectors of norm $1/\sqrt{2}$ uniformly at random, and append it with a $1/\sqrt{2}$ entry. We consider Bernoulli $\{0, 1\}$ rewards. We fix $\delta = 0.1$ and plot the results for varying privacy level $\varepsilon \in \{0.2, 1, 10\}$ in Figure 1. We use Batchsize $B = 20$ for LinUCB-SDP. We postpone the results for $d = 10, 15$ to Appendix G.

From Figure 1, we observe that the regret performance of LinUCB-SDP (under both shuffle protocols $\mathcal{P}_{\text{Amp}}$ and $\mathcal{P}_{\text{Vec}}$) is indeed better than LinUCB-LDP. In addition, it is not surprising that LinUCB-SDP incurs a larger regret than LinUCB-JDP. Moreover, the regret performance of LinUCB-SDP (in fact for any private algorithm) comes closer to that of LinUCB as ε increases, i.e, as the privacy guarantee becomes weaker. The experimental findings are consistent with our theoretical results.[7]

[7]Code is available at https://github.com/sayakrc/Differentially-Private-Bandits.

6. Concluding Remarks

We conclude by discussing some important theoretical and practical aspects about shuffle protocols, and in general, about privacy in linear contextual bandits.

Communications. In the protocol $\mathcal{P}_{\text{Amp}}$, each participating user at each round need to send one d-dimensional real vector and one $d \times d$ real matrix. On the other hand, the protocol $\mathcal{P}_{\text{Vec}}$ only communicates $0/1$ bits. In particular, each participating user at each round sends out a total of $O(d^2(g+b))$ bits, where $g+b \approx \sqrt{B}+\log(1/\delta)/\varepsilon^2$. Hence, $\mathcal{P}_{\text{Vec}}$ might be more feasible in practice than $\mathcal{P}_{\text{Amp}}$.

Batched algorithms for local and central models. Existing work on differentially private linear contextual bandits under both local and central models perform sequential update, i.e., the model estimates are updated after each round. As mentioned before, this may not be feasible in practice due to computational load. Fortunately, our proposed algorithm (Algorithm 1) along with its generic regret bound (Lemma 3.9) also offers a simple way to design and analyze private algorithms for local and central models with batched update. In particular, we show that it suffices to update after every $B = \widetilde{O}(T^{3/4})$ rounds to achieve the same privacy-regret trade-off as in the sequential local model and every $B = \widetilde{O}(\sqrt{T})$ to match the sequential central model. See Appendix F for the details.

Adaptive model update. One might wonder whether we can further reduce the update frequency to $O(\log T)$ via an adaptive model update schedule based on the standard determinant trick (Lemma 12 of Abbasi-Yadkori et al. (2011)). In this approach, the key step is to establish that $\|\phi(c,a)\|_{V_{\tau_t}^{-1}} \leqslant \eta \|\phi(c,a)\|_{V_t^{-1}}$, where $\tau_t < t$ is the most recent model update time before t. To this end, if one uses the determinant trick, one can obtain that

$$\|\phi(c,a)\|_{V_{\tau_t}^{-1}} \leqslant \sqrt{\frac{\det(V_t)}{\det(V_{\tau_t})}}\, \|\phi(c,a)\|_{V_t^{-1}},$$

if the condition $V_t \succeq V_{\tau_t}$ *holds*. Note that this is true in

the non-private setting. However, this does not necessarily hold in private settings due to the added noise, which, to the best of our knowledge, is the key analytical gap in the current proof of the main result (Theorem 10) in Garcelon et al. (2021). As we can see, this issue exists in all three trust models when one needs to use the *noisy* design matrix to determine the update frequency via the determinant trick.

Close the gap. For the case of unique users, we have shown that shuffle model enables us to achieve a regret $\tilde{O}(T^{3/5})$, which is between local model regret $\tilde{O}(T^{3/4})$ and central model regret $\tilde{O}(\sqrt{T})$. One important question is how to further close the gap. To this end, one might first need to have a tight lower bound for the regret under the local model. To the best of our knowledge, Liao et al. (2021) presents the first lower bound under local model $\Omega(\frac{1}{\min\{2,e^{\varepsilon}\}(e^{\varepsilon}-1)}d\sqrt{T})$. It is unclear to us whether the upper bound and lower bound are tight for the local model[8].

Advanced shuffle protocols. We use vector summation protocol in Cheu et al. (2021) rather than advanced shuffle protocols (in terms of communication cost) (e.g., Ghazi et al. (2020); Balle et al. (2020)) because it is simple to implement and suffices to deliver our main message. Another key technical reason is that the random noise introduced by these advanced shuffle protocols is (discrete) Laplace, which at best exhibits sub-exponential tail behavior. Now, in contextual bandits, one need to protect both rewards and feature vectors, and a principled way to achieve privacy is to inject suitable random noise to the matrices $\phi(\cdot)\phi(\cdot)^\top$ and vectors $\phi(\cdot)y$ generated by rewards and features. This raises the need to control respective norms of random vectors and random symmetric matrices with i.i.d. sub-exponential entries in order to achieve a meaningful utility/regret bound. In contrast, our current adopted shuffle protocol from Cheu et al results in random vectors and matrices with Binomial entries, which have sub-Gaussian tails. Hence, the respective norms can be controlled using standard results from random matrix theory. However, we are not aware of non-trivial bounds on matrix norms with sub-exponential entries (i.e., without a simple union bound over co-ordinates), which abstains us from using advanced shuffle protocols.

Pure DP in the shuffle model. *First,* let us highlight the reason why we focus only on approximate DP. Existing linear contextual bandit algorithms under central and local privacy models consider only approximate DP, and our main motivation in this work is to close the gap in regret achieved under these two models via shuffling. *Second,* almost all the existing shuffle protocols can only guarantee approximate DP. To the best of knowledge, the very recent work (Cheu & Yan, 2021) develops the *first* shuffle protocol that is able to achieve pure DP with an error of $O(1/\varepsilon)$. However, since this new protocol also relies on discrete Laplace noise, the same challenge of controlling norms of vectors and matrices with sub-exponential entries in contextual bandits still remain. At mentioned before, to the best of our knowledge, it remains open to derive *non-trivial* (i.e., without a simple union bound over co-ordinates) concentration bounds on the norm for Laplace random matrices. *That being said*, if there indeed exist non-trivial concentrations for sub-exponential or Laplace matrices, our derived result (i.e., Lemma A.2) can also be used to derive meaningful regret bounds under this new protocol.

Regret in low-privacy regime: As in previous works, we focus on the high-privacy regime. One may wonder if it is possible to potentially achieve $O(\sqrt{T}+\sqrt{T}/\varepsilon)$ regret after amplification in the low-privacy regime (i.e., $\varepsilon \approx \sqrt{\frac{e^{\varepsilon_0}}{n}}$, by Lemma B.4). For example, this is possible if one can replace $\varepsilon_0^{-\frac{1}{2}}$ by $e^{-\frac{\varepsilon_0}{2}}$ in the second term of the regret bound in Proposition F.1 However, this essentially amounts to requiring *a mechanism that guarantees* $(\varepsilon_0, \delta_0)$*-LDP for* $\varepsilon_0 > 1$ *by injecting (sub)-Gaussian noise with standard deviation* $\sigma = O(e^{-\varepsilon_0})$. To the best of our knowledge, we are unaware of any such mechanism. The standard Gaussian mechanism fails to satisfy this since it needs $\sigma = \Theta(1/\sqrt{\varepsilon_0})$ to achieve $(\varepsilon_0, \delta_0)$-DP in low-privacy regime. For pure DP, the state-of-the-art staircase mechanism (Geng et al., 2015), requires a noise with $\sigma = O(e^{-\varepsilon_0/2})$ rather than $\sigma = O(e^{-\varepsilon_0})$. This again can't guarantee $\sqrt{T}/\varepsilon$ regret *even if* one manages to apply it in our context (with proper matrix-norm bounds). *That being said*, if there indeed exists the above required mechanism, our results can be used to establish the desired $\sqrt{T}/\varepsilon$ regret in low-privacy regime.

Future work. One immediate future research direction is to address the above adaptive model update in the private settings. We also believe our framework can be generalized to design shuffle private algorithms for reinforcement learning with linear function approximation (e.g., linear mixture Markov decision processes (MDPs)) to achieve finer trade-off between the local model (Liao et al., 2021) and the central model (Zhou, 2022).

[8] Note that Zheng et al. (2020) conjecture that the lower bound for the local model is $\Omega(T^{3/4})$, see Appendix G therein.

Acknowledgements

We thank anonymous reviewers for their useful comments, which helped preparing the final version. XZ would like to thank Albert Cheu for insightful discussions on shuffle protocols. SRC is grateful to a CISE Postdoctoral fellowship of Boston University.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. In *Advances in*

Neural Information Processing Systems, pp. 2312–2320, 2011.

Abe, N., Biermann, A. W., and Long, P. M. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37(4):263–293, 2003.

Agarwal, N. and Singh, K. The price of differential privacy for online learning. In *International Conference on Machine Learning*, pp. 32–40. PMLR, 2017.

Apple. Learning with privacy at scale. 2017. URL https://machinelearning.apple.com/research/learning-with-privacy-at-scale.

Arora, S., Yttri, J., and Nilsen, W. Privacy and security in mobile health (mhealth) research. *Alcohol research: current reviews*, 36(1):143, 2014.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, March 2003. ISSN 1532-4435.

Balle, B., Bell, J., Gascon, A., and Nissim, K. Differentially private summation with multi-message shuffling. *arXiv preprint arXiv:1906.09116*, 2019a.

Balle, B., Bell, J., Gascón, A., and Nissim, K. The privacy blanket of the shuffle model. In *Annual International Cryptology Conference*, pp. 638–667. Springer, 2019b.

Balle, B., Bell, J., Gascón, A., and Nissim, K. Private summation in the multi-message shuffle model. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 657–676, 2020.

Bittau, A., Erlingsson, Ú., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J., and Seefeld, B. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 441–459, 2017.

Canonne, C. L., Kamath, G., and Steinke, T. The discrete gaussian for differential privacy. In *NeurIPS*, 2020.

Chan, T. H., Shi, E., and Song, D. Private and continual release of statistics. In *International Colloquium on Automata, Languages, and Programming*, pp. 405–417. Springer, 2010.

Chen, X., Zheng, K., Zhou, Z., Yang, Y., Chen, W., and Wang, L. (locally) differentially private combinatorial semi-bandits. In *International Conference on Machine Learning*, pp. 1757–1767. PMLR, 2020.

Cheu, A. Differential privacy in the shuffle model: A survey of separations. *arXiv preprint arXiv:2107.11839*, 2021.

Cheu, A. and Yan, C. Pure differential privacy from secure intermediaries. *arXiv preprint arXiv:2112.10032*, 2021.

Cheu, A., Smith, A., Ullman, J., Zeber, D., and Zhilyaev, M. Distributed differential privacy via shuffling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 375–403. Springer, 2019.

Cheu, A., Joseph, M., Mao, J., and Peng, B. Shuffle private stochastic convex optimization. *arXiv preprint arXiv:2106.09805*, 2021.

Chowdhury, S. R. and Zhou, X. Differentially private regret minimization in episodic markov decision processes. *arXiv preprint arXiv:2112.10599*, 2021.

Chowdhury, S. R., Zhou, X., and Shroff, N. Adaptive control of differentially private linear quadratic systems. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 485–490. IEEE, 2021.

Chu, W., Li, L., Reyzin, L., and Schapire, R. E. Contextual bandits with linear payoff functions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pp. 208–214, 2011.

Dubey, A. No-regret algorithms for private gaussian process bandit optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 2062–2070. PMLR, 2021.

Dwork, C. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pp. 1–19. Springer, 2008.

Dwork, C., Naor, M., Pitassi, T., and Rothblum, G. N. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 715–724, 2010.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., and Thakurta, A. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2468–2479. SIAM, 2019.

Feldman, V., McMillan, A., and Talwar, K. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. *arXiv preprint arXiv:2012.12803*, 2020.

Garcelon, E., Perchet, V., Pike-Burke, C., and Pirotta, M. Local differentially private regret minimization in reinforcement learning. *arXiv preprint arXiv:2010.07778*, 2020.

Garcelon, E., Chaudhuri, K., Perchet, V., and Pirotta, M. Privacy amplification via shuffling for linear contextual bandits. *arXiv preprint arXiv:2112.06008*, 2021.

Geng, Q., Kairouz, P., Oh, S., and Viswanath, P. The staircase mechanism in differential privacy. *IEEE Journal of Selected Topics in Signal Processing*, 9(7):1176–1184, 2015.

Ghazi, B., Golowich, N., Kumar, R., Pagh, R., and Velingker, A. On the power of multiple anonymous messages. *arXiv preprint arXiv:1908.11358*, 2019.

Ghazi, B., Kumar, R., Manurangsi, P., and Pagh, R. Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead. In *International Conference on Machine Learning*, pp. 3505–3514. PMLR, 2020.

Girgis, A., Data, D., Diggavi, S., Kairouz, P., and Suresh, A. T. Shuffled model of differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2521–2529. PMLR, 2021.

Guha Thakurta, A. and Smith, A. (nearly) optimal algorithms for private online learning in full-information and bandit settings. *Advances in Neural Information Processing Systems*, 26:2733–2741, 2013.

Han, Y., Zhou, Z., Zhou, Z., Blanchet, J., Glynn, P. W., and Ye, Y. Sequential batch learning in finite-action linear contextual bandits. *arXiv preprint arXiv:2004.06321*, 2020.

Han, Y., Liang, Z., Wang, Y., and Zhang, J. Generalized linear bandits with local differential privacy. *arXiv preprint arXiv:2106.03365*, 2021.

Hsu, J., Huang, Z., Roth, A., Roughgarden, T., and Wu, Z. S. Private matchings and allocations. *SIAM Journal on Computing*, 45(6):1953–1984, 2016.

Kairouz, P., Liu, Z., and Steinke, T. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *NeurIPS*, 2021.

Kearns, M., Pai, M., Roth, A., and Ullman, J. Mechanism design in large games: Incentives and privacy. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pp. 403–410, 2014.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

Liao, C., He, J., and Gu, Q. Locally differentially private reinforcement learning for linear mixture markov decision processes. *arXiv preprint arXiv:2110.10133*, 2021.

Lowy, A. and Razaviyayn, M. Private federated learning without a trusted server: Optimal algorithms for convex losses. *arXiv preprint arXiv:2106.09779*, 2021.

Mironov, I. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 650–661, 2012.

Mishra, N. and Thakurta, A. (nearly) optimal differentially private stochastic multi-arm bandits. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pp. 592–601, 2015.

Ren, W., Zhou, X., Liu, J., and Shroff, N. B. Multi-armed bandits with local differential privacy. *arXiv preprint arXiv:2007.03121*, 2020a.

Ren, Z., Zhou, Z., and Kalagnanam, J. R. Batched learning in generalized linear contextual bandits with general decision sets. *IEEE Control Systems Letters*, 2020b.

Sajed, T. and Sheffet, O. An optimal private stochastic-mab algorithm based on optimal private stopping rule. In *International Conference on Machine Learning*, pp. 5579–5588. PMLR, 2019.

Schwartz, E. M., Bradlow, E. T., and Fader, P. S. Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science*, 36(4): 500–522, 2017.

Shariff, R. and Sheffet, O. Differentially private contextual linear bandits. *Advances in Neural Information Processing Systems*, 31:4296–4306, 2018.

Tenenbaum, J., Kaplan, H., Mansour, Y., and Stemmer, U. Differentially private multi-armed bandits in the shuffle model. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=P0AeY-efPEx.

Tewari, A. and Murphy, S. A. From ads to interventions: Contextual bandits in mobile health. In *Mobile Health*, pp. 495–517. Springer, 2017.

Tossou, A. and Dimitrakakis, C. Achieving privacy in the adversarial multi-armed bandit. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Vaswani, S., Mehrabian, A., Durand, A., and Kveton, B. Old dog learns new tricks: Randomized ucb for bandit problems. In *International Conference on Artificial Intelligence and Statistics*, pp. 1988–1998. PMLR, 2020.

Vershynin, R. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Vietri, G., Balle, B., Krishnamurthy, A., and Wu, S. Private reinforcement learning with pac and regret guarantees. In *International Conference on Machine Learning*, pp. 9754–9764. PMLR, 2020.

Wang, T., Zhou, D., and Gu, Q. Provably efficient reinforcement learning with linear function approximation under adaptivity constraints. *arXiv preprint arXiv:2101.02195*, 2021.

Xin, Y. and Jaakkola, T. Controlling privacy in recommender systems. Neural Information Processing Systems, 2014.

Zheng, K., Cai, T., Huang, W., Li, Z., and Wang, L. Locally differentially private (contextual) bandits learning. In *NeurIPS*, 2020.

Zhou, X. Differentially private reinforcement learning with linear function approximation. *arXiv preprint arXiv:2201.07052*, 2022.

Zhou, X. and Tan, J. Local differential privacy for bayesian optimization. *arXiv preprint arXiv:2010.06709*, 2020.

A. A Unified Regret Analysis Under Differential Privacy

In this section, we will formally state Lemma 3.9, i.e., the generic regret of Algorithm 1 under sub-Gaussian private noise and then present its proof.

Let's first recall the following notations. For each batch $m \in [M]$, let $N_m := \widetilde{V}_m - \sum_{t=t_{m-1}+1}^{t_m} \phi(c_t, a_t)\phi(c_t, a_t)^\top$ denote the additional noise injected into the non-private Gram-matrix and similarly let $n_m := \widetilde{u}_m - \sum_{t=t_{m-1}+1}^{t_m} \phi(c_t, a_t) y_t$ denote the additional noise injected into the non-private feature-reward vector. Then, we let $H_m := \lambda I_d + \sum_{i=1}^{m} N_i$ to denote the *total* noise in the first m batches plus the regularizer, and similarly let $h_m := \sum_{i=1}^{m} n_i$.

Assumption A.1 (Regularity)**.** For any $\alpha \in (0, 1]$, H_m is positive definite and there exist constants $\lambda_{\max}$, $\lambda_{\min}$ and ν depending on α, such that with probability at least $1 - \alpha$, for all $m \in [M]$

$$\|H_m\| \leq \lambda_{\max}, \quad \left\|H_m^{-1}\right\| \leq 1/\lambda_{\min}, \quad \|h_m\|_{H_m^{-1}} \leq \nu.$$

With the above regularity assumption and the boundedness in Assumption 3.1, we fist establish the following general regret bound of Algorithm 1, which can be viewed as a direct generalization of the results in (Shariff & Sheffet, 2018) to the batched case.

Lemma A.2. *Let Assumptions A.1 and 3.1 hold. Fix any $\alpha \in (0, 1]$, with probability at least $1 - \alpha$, the regret of Algorithm 1 satisfies*

$$\mathit{Reg}(T) \leq \frac{dB}{\log 2} \log\left(1 + \frac{T}{d\lambda_{\min}}\right) + 8\beta_M \sqrt{dT \log\left(1 + \frac{T}{d\lambda_{\min}}\right)},$$

where

$$\beta_M := \sqrt{2\log\left(\frac{2}{\alpha}\right) + d\log\left(1 + \frac{T}{d\lambda_{\min}}\right)} + \sqrt{\lambda_{\max}} + \nu.$$

In fact, Lemma 3.9 in the main paper is a simple application of Lemma A.2 by considering the following assumption.

Assumption A.3 (sub-Gaussian private noise)**.** There exist constants $\widetilde{\sigma}_1$ and $\widetilde{\sigma}_2$ such that for all $m \in [M]$: (i) $\sum_{m'=1}^{m} n_{m'}$ is a random vector whose entries are independent, mean zero, sub-Gaussian with variance at most $\widetilde{\sigma}_1^2$, and (ii) $\sum_{m'=1}^{m} N_{m'}$ is a random symmetric matrix whose entries on and above the diagonal are independent sub-Gaussian random variables with variance at most $\widetilde{\sigma}_2^2$. Let $\sigma^2 = \max\{\widetilde{\sigma}_1^2, \widetilde{\sigma}_2^2\}$.

Now, we are well-prepared to formally state Lemma 3.9 in the main paper.

Lemma A.4 (Formal statement of Lemma 3.9)**.** *Let Assumptions A.3 and 3.1 hold. Fix time horizon $T \in \mathbb{N}$, batch size $B \in [T]$, confidence level $\alpha \in (0, 1]$. Set $\lambda = \Theta(\max\{1, \sigma(\sqrt{d} + \sqrt{\log(T/(B\alpha))}\})$ and $\beta_m = \sqrt{2\log\left(\frac{2}{\alpha}\right) + d\log\left(1 + \frac{T}{d\lambda}\right)} + \sqrt{\lambda}$. Then, Algorithm 1 achieves regret*

$$\mathit{Reg}(T) = O\left(dB\log T + d\sqrt{T}\log(T/\alpha)\right) + O\left(\sqrt{\sigma T} d^{3/4} \log T \log(T/\alpha)\right)$$

with probability at least $1 - \alpha$.

Remark A.5. The above lemma also presents a regret bound for non-private batched LCB when $\sigma = 0$. Note that in this case, our regret bound is achieved with a *dimension-independent* regularizer, in contrast to the necessary condition on $\lambda = \tilde{\Theta}(d)$ as required in (Ren et al., 2020b) to attain the optimal regret.

A.1. Proofs

In this section, we present proofs for Lemma A.2 and Lemma A.4 above, respectively.

Proof of Lemma A.2. Let $\mathcal{E}$ be the event given in Assumption A.1, which holds with probability at least $1 - \alpha$ under Assumption A.1. In the following, we condition on the event $\mathcal{E}$. We first show that $\widehat{\theta}_m$ concentrates around the true parameter

θ^* with a properly chosen confidence radius β_m for all $m \in [M]$. To this end, note that

$$\begin{aligned}\widehat{\theta}_m &= V_m^{-1} u_m \\ &= \left(\sum_{t=1}^{t_m} \phi(c_t, a_t)\phi(c_t, a_t)^\top + \lambda I_d + \sum_{i=1}^{m} N_m\right)^{-1} \left(\sum_{t=1}^{t_m} \phi(c_t, a_t) y_t + \sum_{i=1}^{m} n_m\right) \\ &= \left(\sum_{t=1}^{t_m} \phi(c_t, a_t)\phi(c_t, a_t)^\top + H_m\right)^{-1} \left(\sum_{t=1}^{t_m} \phi(c_t, a_t) y_t + h_m\right).\end{aligned}$$

By the linear reward function $y_t = \langle \phi(c_t, a_t), \theta^* \rangle + \eta_t$ and elementary algebra, we have

$$\theta^* - \widehat{\theta}_m = V_m^{-1} \left(H_m \theta^* - \sum_{t=1}^{t_m} \phi(c_t, a_t)\eta_t - h_m\right).$$

Thus, multiplying both sides by $V_m^{1/2}$, yields

$$\begin{aligned}\left\|\theta^* - \widehat{\theta}_m\right\|_{V_m} &\le \left\|\sum_{t=1}^{t_m} \phi(c_t, a_t)\eta_t\right\|_{V_m^{-1}} + \|H_m\theta^*\|_{V_m^{-1}} + \|h_m\|_{V_m^{-1}} \\ &\overset{(a)}{\le} \left\|\sum_{t=1}^{t_m} \phi(c_t, a_t)\eta_t\right\|_{(G_m + \lambda_{\min} I)^{-1}} + \|\theta^*\|_{H_m} + \|h_m\|_{H_m^{-1}},\end{aligned}$$

where (a) holds by $V_m \succeq H_m$ and $V_m \succeq G_m + \lambda_{\min} I$ with $G_m := \sum_{t=1}^{t_m} \phi(c_t, a_t)\phi(c_t, a_t)^\top$ under event $\mathcal{E}$. Further, by the boundedness condition of θ^* and event $\mathcal{E}$, $\|\theta^*\|_{H_m} \le \sqrt{\lambda_{\max}}$ and $\|h_m\|_{H_m^{-1}} \le \nu$. For the remaining first term, we can use self-normalized inequality (cf. Theorem 1 in (Abbasi-Yadkori et al., 2011)) with the filtration $\mathcal{F}_t = \sigma(c_1, a_1, y_1, \ldots, c_t, a_t, y_t, c_{t+1}, a_{t+1})$. In particular, we have with probability at least $1 - \alpha$, for all $m \in [M]$

$$\left\|\sum_{t=1}^{t_m} \phi(c_t, a_t)\eta_t\right\|_{(G_m + \lambda_{\min} I)^{-1}} \le \sqrt{2\log\left(\frac{1}{\alpha}\right) + \log\left(\frac{\det(G_m + \lambda_{\min} I)}{\det(\lambda_{\min} I)}\right)}. \tag{1}$$

Now, using the trace-determinant lemma (cf. Lemma 10 in (Abbasi-Yadkori et al., 2011)) and the boundedness condition on $\|\phi(c, a)\|$, we have

$$\det(G_m + \lambda_{\min} I) \le \left(\lambda_{\min} + \frac{t_m}{d}\right)^d.$$

Putting everything together, we have with probability at least $1 - 2\alpha$, for all $m \in [M]$, $\left\|\theta^* - \widehat{\theta}_m\right\|_{V_m} \le \beta_m$, where

$$\beta_m := \sqrt{2\log\left(\frac{1}{\alpha}\right) + d\log\left(1 + \frac{t_m}{d\lambda_{\min}}\right)} + \sqrt{\lambda_{\max}} + \nu.$$

With the above concentration result and our OFUL-type algorithm, the regret can be upper bounded as follows.

$$\begin{aligned}\mathcal{R}(T) &= \sum_{m=1}^{M} \left[2\beta_{m-1} \sum_{t=t_{m-1}+1}^{t_m} \left(\|\phi(c_t, a_t)\|_{V_{m-1}^{-1}}\right)\right] \\ &\le \sum_{m=1}^{M} \left[2\beta_M \sum_{t=t_{m-1}+1}^{t_m} \left(\|\phi(c_t, a_t)\|_{(G_{m-1} + \lambda_{\min} I)^{-1}}\right)\right]\end{aligned} \tag{2}$$

At this moment, we note that the standard elliptical potential lemma (cf. Lemma 11 in (Abbasi-Yadkori et al., 2011)) cannot be applied to our batch setting due to the delay of G_m.

To handle this, inspired by (Wang et al., 2021), we let $\widehat{V}_k := \sum_{t=1}^{k} \phi(c_t, a_t)\phi(c_t, a_t)^\top + \lambda_{\min} I_d$, that is, a (virtual) design matrix at the end of time k. Hence, we have $G_{m-1} + \lambda_{\min} I_d = \widehat{V}_{t_{(m-1)}}$. Moreover, for any $t_{m-1} < t \le t_m$, let $m_t = t_{m-1}$, that is, mapping t to the starting time of the batch that includes t. Finally, let $\Gamma_i(\cdot, \cdot) := \beta_M \cdot \|\phi(\cdot, \cdot)\|_{\widehat{V}_i^{-1}}$.

With above notations, the bound in (2) can be rewritten as follows.

$$\begin{aligned} R(T) &\le \sum_{m=1}^{M} \left[2\beta_M \sum_{t=t_{m-1}+1}^{t_m} \left(\|\phi(c_t, a_t)\|_{(G_{m-1}+\lambda_{\min})^{-1}} \right) \right] \\ &= \sum_{t=1}^{T} 2\Gamma_{m_t}(c_t, a_t) \end{aligned}$$

In the sequential case (i.e., $B = 1$), we always have $m_t = t - 1$. Thus, the key is to bound the difference between $\Gamma_{m_t}(c_t, a_t)$ and $\Gamma_{t-1}(c_t, a_t)$. To this end, we have the following claim, which will be proved at the end.

Claim A.6. *Define the set Ψ as follows*

$$\Psi = \{t \in [T] : \Gamma_{m_t}(c_t, a_t)/\Gamma_{t-1}(c_t, a_t) > 2\}.$$

Then, we have

$$|\Psi| \le \frac{dT}{2M \log 2} \log\left(1 + \frac{T}{d\lambda_{\min}}\right).$$

According to Claim A.6, we can decompose regret as follows.

$$\begin{aligned} R(T) &\overset{(a)}{\le} \sum_{t=1}^{T} \min\{2\Gamma_{m_t}(c_t, a_t), 1\} \\ &= \sum_{t\in\Psi} \min\{2\Gamma_{m_t}(c_t, a_t), 1\} + \sum_{t\notin\Psi} \min\{2\Gamma_{m_t}(c_t, a_t), 1\} \\ &\overset{(b)}{\le} |\Psi| + \sum_{t\notin\Psi} \min\{4\Gamma_{t-1}(c_t, a_t), 1\} \\ &\overset{(c)}{\le} |\Psi| + \sum_{t=1}^{T} 4\beta_M \min\{\|\phi(c_t, a_t)\|_{\widehat{V}_{t-1}^{-1}}, 1\} \\ &\overset{(d)}{\le} \frac{dT}{2M \log 2} \log\left(1 + \frac{T}{d\lambda_{\min}}\right) + 8\beta_M \sqrt{dT \log\left(1 + \frac{T}{d\lambda_{\min}}\right)} \end{aligned}$$

where (a) holds by the boundedness of reward; (b) holds by definition of Ψ; (c) holds by the fact that $\beta_M \ge 1$; (d) follows from Claim A.6 and standard argument for linear bandit, i.e., Cauchy-Schwartz and standard elliptical potential lemma (cf. Lemma 11 in (Abbasi-Yadkori et al., 2011)). Hence, we have finished the proof of Lemma A.2.

Finally, we give the proof of Claim A.6.

For any $t \in \Psi$, suppose $t_{m-1} < t \le t_m$ for some m. Then, we have $m_t = t_{(m-1)}$ and

$$\log\det(\widehat{V}_{t_m}) - \log\det(\widehat{V}_{t_{(m-1)}}) \overset{(a)}{\ge} \log\det(\widehat{V}_{t-1}) - \log\det(\widehat{V}_{m_t}) \overset{(b)}{\ge} 2\log(\Gamma_{m_t}(c_t, a_t)/\Gamma_{t-1}(c_t, a_t)) > 2\log 2,$$

where (a) holds by the fact $\widehat{V}_{t_m} \succeq \widehat{V}_{t-1}$; (b) holds by Lemma 12 in (Abbasi-Yadkori et al., 2011), that is, for two positive definite matrices $A, B \in \mathbb{R}^{d\times d}$ satisfying $A \succeq B$, then for any $x \in \mathbb{R}^d$, $\|x\|_A \le \|x\|_B \cdot \sqrt{\det(A)/\det(B)}$. Note that here we also use $\det(A) = 1/\det(A^{-1})$ for any matrix;

Therefore, if we let $\widehat{\Psi} := \{m \in [M] : \log\det(\widehat{V}_{t_m}) - \log\det(\widehat{V}_{t_{(m-1)}}) > 2\log 2\}$, then we have $|\Psi| \le (T/M)|\widehat{\Psi}|$. Thus, we only need to bound $|\widehat{\Psi}|$. Note that for each m, $\log\det(\widehat{V}_{t_m}) - \log\det(\widehat{V}_{t_{(m-1)}}) \ge 0$, and hence

$$2\log 2 \cdot |\widehat{\Psi}| \le \sum_{m\in\widehat{\Psi}} \log\det(\widehat{V}_{t_m}) - \log\det(\widehat{V}_{t_{(m-1)}}) \le \sum_{m=1}^{M} \log\det(\widehat{V}_{t_m}) - \log\det(\widehat{V}_{t_{(m-1)}}) = \log\left(\frac{\det(G_M + \lambda_{\min} I)}{\det(\lambda_{\min} I)}\right)$$

Algorithm 2 Local Randomizer $\mathcal{R}_{\text{Amp}}$

Parameters: σ_1, σ_2, d
function $R_1(\phi(c,a)y)$
  Sample fresh noise $n \sim \mathcal{N}(0, \sigma_2^2 I_{d\times d})$
  $M_1 = \phi(c,a)y + n$
  return $M_{t,1}$
end function
function $R_2(\phi(c,a)\phi(c,a)^\top)$
  Sample fresh noise $N_{(i,j)} \sim \mathcal{N}(0, \sigma_1^2), \forall i \le j \le d$ and let $N_{(j,i)} = N_{(i,j)}$
  $M_2 = \phi(c,a)\phi(c,a)^\top + N$
  return M_2
end function

Finally, using the same analysis as in (1), yields

$$|\widehat{\Psi}| \le \frac{d}{2\log 2} \log\left(1 + \frac{T}{d\lambda_{\min}}\right),$$

which directly implies the result of Claim A.6. □

Proof of Lemma A.4. To prove the result, thanks to Lemma A.2, we only need to determine the three constants $\lambda_{\max}, \lambda_{\min}$ and ν under the sub-Gaussian private noise assumption in Assumption A.3. To this end, we resort to concentration bounds for sub-Gaussian random vector and random matrix.

To start with, under (i) in Assumption A.3, by the concentration bound for the norm of a vector containing sub-Gaussian entries (cf. Theorem 3.1.1 in (Vershynin, 2018)) and a union bound over m, we have for all $m \in [M]$ and any $\alpha \in (0,1]$, with probability at least $1 - \alpha/2$, for some absolute constant c_1,

$$\left\|\sum_{i=1}^{m} n_i\right\| = \|h_m\| \le \Sigma_n := c_1 \cdot \widetilde{\sigma}_1 \cdot (\sqrt{d} + \sqrt{\log(M/\alpha)}).$$

By (ii) in Assumption A.3, the concentration bound for the norm of a sub-Gaussian symmetric random matrix (cf. Corollary 4.4.8 (Vershynin, 2018)) and a union bound over m, we have for all $m \in [M]$ and any $\alpha \in (0,1]$, with probability at least $1 - \alpha/2$,

$$\left\|\sum_{i=1}^{m} N_i\right\| \le \Sigma_N := c_2 \cdot \widetilde{\sigma}_2 \cdot (\sqrt{d} + \sqrt{\log(M/\alpha)})$$

for some absolute constant c_2. Thus, if we choose $\lambda = 2\Sigma_N$, we have $\|H_m\| = \|\lambda I_d + \sum_{i=1}^m N_i\| \le 3\Sigma_N$, i.e., $\lambda_{\max} = 3\Sigma_N$, and $\lambda_{\min} = \Sigma_N$. Finally, to determine ν, we note that

$$\|h_m\|_{H_m^{-1}} \le \frac{1}{\sqrt{\lambda_{\min}}} \|h_m\| \le c \cdot \left(\sigma \cdot (\sqrt{d} + \sqrt{\log(M/\alpha)})\right)^{1/2} := \nu,$$

where $\sigma = \max\{\widetilde{\sigma}_1, \widetilde{\sigma}_2\}$. The final regret bound is obtained by plugging the three values into the result given by Lemma A.2. □

B. Analysis of LDP Amplification Protocol

B.1. Pseudocode of $\mathcal{P}_{\text{Amp}}$

The shuffle protocol is given by $\mathcal{P}_{\text{Amp}} = (\mathcal{R}_{\text{Amp}}, \mathcal{S}_{\text{Amp}}, \mathcal{A}_{\text{Amp}})$, in which $\mathcal{R}_{\text{Amp}}$ is presented in Algorithm 2, $\mathcal{S}_{\text{Amp}}$ is presented in Algorithm 3, and $\mathcal{A}_{\text{Amp}}$ is presented in Algorithm 4.

Algorithm 3 Shuffler $\mathcal{S}_{\text{Amp}}$

Input: $\{M_{\tau,1}\}_{\tau\in\mathcal{B}}$ and $\{M_{\tau,2}\}_{\tau\in\mathcal{B}}$, in which $\mathcal{B}$ is a batch and $M_{\tau,1}\in\mathbb{R}^d$, $M_{\tau,2}\in\mathbb{R}^{d\times d}$ come from user τ
function $S_1(\{M_{\tau,1}\}_{\tau\in\mathcal{B}})$
  Generate a uniform permutation π of indexes in $\mathcal{B}$
  Set $Y_1=(M_{\pi(1),1},\dots,M_{\pi(B),1})$
  return Y_1
end function
function $S_2(\{M_{\tau,2}\}_{\tau\in\mathcal{B}})$
  Generate a uniform permutation π of indexes in $\mathcal{B}$
  Set $Y_2=(M_{\pi(1),2},\dots,M_{\pi(B),2})$
  return Y_2
end function

Algorithm 4 Analyzer $\mathcal{A}_{\text{Amp}}$

Input: Shuffled outputs $Y_1=(M_{\pi(1),1},\dots,M_{\pi(B),1})$ and $Y_2=(M_{\pi(1),2},\dots,M_{\pi(B),2})$
function $A_1(Y_1)$
  return $\sum_{i=1}^B M_{\pi(i),1}$
end function
function $A_1(Y_2)$
  return $\sum_{i=1}^B M_{\pi(i),2}$
end function

B.2. Main Results

Theorem B.1 (Restatement of Theorem 3.2)**.** *Fix time horizon $T\in\mathbb{N}$, batch size $B\in[T]$, confidence level $\alpha\in(0,1]$, , and privacy budgets $\varepsilon\in(0,\sqrt{\frac{\log(2/\delta)}{B}}]$, $\delta\in(0,1]$. Then, Algorithm 1 instantiated with shuffle protocol $\mathcal{P}_{Amp}$ with noise levels $\sigma_1=\sigma_2=\frac{4\sqrt{2\log(2.5B/\delta)\log(2/\delta)}}{\varepsilon\sqrt{B}}$, and regularizer $\lambda=\Theta(\sqrt{T}\sigma_1(\sqrt{d}+\sqrt{\log(T/B\alpha)})$, enjoys the regret*

$$Reg(T)=O\left(dB\log T+\frac{\log^{1/4}(B/\delta)\log^{1/4}(2/\delta)}{\varepsilon^{1/2}B^{1/4}}d^{3/4}T^{3/4}\log T\log(T/\alpha)\right),$$

with probability at least $1-\alpha$. Moreover, it satisfies $O(\varepsilon,\delta)$-shuffle differential privacy (SDP).

Corollary B.2 (Utility-targeted)**.** *Under the same assumption in Theorem B.1 and Algorithm 1 is instantiated with $\mathcal{P}_{Amp}$. Let $B=O(d^{-1/5}\varepsilon^{-2/5}T^{3/5}\log^{1/5}(T/\delta)\log^{1/5}(2/\delta)$, Algorithm 1 achieves $O(\varepsilon,\delta)$-SDP with regret*

$$Reg(T)=\widetilde{O}\left(d^{4/5}T^{3/5}\varepsilon^{-2/5}\log^{1/5}(T/\delta)\log^{1/5}(2/\delta)\right).$$

Simultaneously, Algorithm 1 also achieves $O(\varepsilon,\delta)$-JDP and $O(\varepsilon_0,\delta_0)$-LDP where

$$\varepsilon_0=O\left(\varepsilon^{4/5}T^{3/10}d^{-1/10}\log^{1/10}(T/\delta)\log^{-2/5}(2/\delta)\right),\quad \delta_0=O\left(\delta d^{1/5}T^{-3/5}\varepsilon^{2/5}\log^{-1/5}(T/\delta)\log^{-1/5}(2/\delta)\right).$$

Corollary B.3 (Privacy-targeted)**.** *Let Assumption 3.1 hold and Algorithm 1 is instantiated with $\mathcal{P}_{Amp}$. For any $\varepsilon_0\in[0,1]$ and $\delta_0\in(0,1]$, let $\sigma_1=\sigma_2=\frac{4\sqrt{2\log(2.5/\delta_0)}}{\varepsilon_0}$. Then, for all $B\in[T]$, Algorithm 1 is (ε_0,δ_0)-LDP. Further suppose $B=O(d^{-1/4}T^{3/4}\varepsilon_0^{-1/2}\log^{1/4}(1/\delta_0))$, then Algorithm 1 achieves regret*

$$Reg(T)=\tilde{O}\left(d^{3/4}T^{3/4}\varepsilon_0^{-1/2}\log^{1/4}(1/\delta_0)\right).$$

Simultaneously, Algorithm 1 achieves $O(\varepsilon,\delta)$-SDP and $O(\varepsilon,\delta)$-JDP where

$$\varepsilon=O\left(\varepsilon_0^{5/4}T^{-3/8}d^{1/8}\log^{3/8}(1/\delta_0)\right),\quad \delta=O(\delta_0 d^{-1/4}T^{3/4}\varepsilon_0^{-1/2}\log^{1/4}(1/\delta_0)).$$

B.3. Proofs

To prove Theorem B.1, we need the following important lemma, which can be seen as a special case of Theorem 3.8 in (Feldman et al., 2020). In particular, in our paper, we consider a fixed local randomizer rather than the more general adaptive one in (Feldman et al., 2020). Another difference is that we consider the case of *randomizer-then-shuffle* rather than the *shuffle-then-randomizer*. However, as pointed in (Feldman et al., 2020), the two cases are equivalent when the local randomizer is a fixed one.

Lemma B.4 (Amplification by shuffling)**.** *Consider a one-round protocol $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ over n users. Let $\mathcal{R}$ be an $(\varepsilon_0, \delta_0)$-LDP mechanism. Then, for any $\delta' \in [0, 1]$ such that $\varepsilon_0 \leq \log(\frac{n}{16\log(2/\delta')})$, $\mathcal{P}$ is $(\widetilde{\varepsilon}, \widetilde{\delta})$-SDP, i.e., the analyzer's view is $(\widetilde{\varepsilon}, \widetilde{\delta})$-DP, where*

$$\widetilde{\varepsilon} \leq \log\left(1 + \frac{e^{\varepsilon_0} - 1}{e^{\varepsilon_0} + 1}\left(\frac{8\sqrt{e^{\varepsilon_0}\log(4/\delta')}}{\sqrt{n}} + \frac{8e^{\varepsilon_0}}{n}\right)\right), \widetilde{\delta} = \delta' + (e^{\varepsilon} + 1)\left(1 + \frac{e^{-\varepsilon_0}}{2}\right)n\delta_0,$$

That is, when $\varepsilon_0 > 1$, $\widetilde{\varepsilon} = O\left(\frac{\sqrt{e^{\varepsilon_0}\log(1/\delta')}}{\sqrt{n}}\right)$ and when $\varepsilon_0 \leq 1$, $\widetilde{\varepsilon} = O\left(\varepsilon_0\frac{\sqrt{\log(1/\delta')}}{\sqrt{n}}\right)$.

Roughly speaking, we have a privacy amplification by a factor $\sqrt{n}$ due to shuffling, which is the key to our analysis.

Proof of Theorem B.1. To apply Lemma B.4, we choose $\delta' = \delta/2$ and $\varepsilon_0 = \frac{\varepsilon\sqrt{B}}{\sqrt{\log(1/\delta')}} = \frac{\varepsilon\sqrt{B}}{\sqrt{\log(2/\delta)}}$. For any $\varepsilon \in (0, \sqrt{\log(2/\delta)}/\sqrt{B}]$, we have $\varepsilon_0 \leq 1$, which implies $\widetilde{\varepsilon} = O(\varepsilon)$. Meanwhile, we let $\delta_0 = \delta/B$ for any $\delta \in [0, 1]$, which implies that $\widetilde{\delta} = O(\delta)$. Now, we are only left to choose σ_1 and σ_2 in $\mathcal{R}_{\text{Amp}}$ so that it is $(\varepsilon_0, \delta_0)$-LDP. To this end, via the standard Gaussian mechanism and boundedness assumption, we have when

$$\sigma_1 = \sigma_2 = \frac{4\sqrt{2\log(2.5/\delta_0)}}{\varepsilon_0},$$

$\mathcal{R}_{\text{Amp}}$ is $(\varepsilon_0, \delta_0)$-LDP. Finally, plugging in $\varepsilon_0 = \varepsilon\sqrt{B}/\sqrt{\log(2/\delta)}$ and $\delta_0 = \delta/B$, yields

$$\sigma_1 = \sigma_2 = \frac{4\sqrt{2\log(2.5B/\delta)\log(2/\delta)}}{\varepsilon\sqrt{B}}.$$

Finally, plugging the value $\sigma = \frac{4\sqrt{2T\log(2.5B/\delta)\log(2/\delta)}}{\varepsilon\sqrt{B}}$ (since there are total at most T noise) into the regret bound in Lemma A.4 yields the required results. □

Proof of Corollary B.2. To establish the regret bound, we simply choose a balanced B in the regret bound given by Theorem B.1. To prove the JDP guarantee, we will use the powerful *Billboard lemma* (cf. Lemma 9 in (Hsu et al., 2016)), which says that an algorithm is JDP if the action recommended to each user is a function of her private data and a common signal computed in a differential private way. In our case, the private data is user's context and the common signal is the updated policy (i.e., $\widehat{\theta}_m$ and design matrix V_m), which is a post-processing of shuffle outputs. Thus, the SDP guarantee directly implies the JDP guarantee in our case. Finally, the LDP guarantee simply follows from the standard Gaussian mechanism with parameter $\varepsilon_0 = \varepsilon\sqrt{B}/\sqrt{\log(2/\delta)}$ and $\delta_0 = \delta/B$. □

Proof of Corollary B.3. The LDP guarantee follows from standard Gaussian mechanism. To show the regret bound, we will use the result in Theorem B.1. In particular, comparing the values of σ_1, σ_2 in Corollary B.3 and the values in Theorem B.1, we can plug $\varepsilon = \frac{\varepsilon_0\sqrt{\log(2/\delta)}}{\sqrt{B}}$ and $\delta = \delta_0 B$ into the regret bound in Theorem B.1. Then, with a balanced choice of B, we obtain the required regret. The SDP guarantee also follows from Theorem B.1 with $\varepsilon = \frac{\varepsilon_0\sqrt{\log(2/\delta)}}{\sqrt{B}}$ and $\delta = \delta_0 B$. Finally, as in the proof of Corollary B.2, the JDP guarantee follows from SDP guarantee and Billboard lemma. □

Algorithm 5 Local Randomizer $\mathcal{R}_{\text{Vec}}$

Parameters: g, b, p, d
Local randomizer for a scalar within $[0, \Delta]$
function $\mathcal{R}^*(x, \Delta)$
  Set $\bar{x} = \lfloor xg/\Delta \rfloor$
  Sample rounding value $\gamma_1 \sim \textbf{Ber}(xg/\Delta - \bar{x})$
  Set $\widehat{x} = \bar{x} + \gamma_1$
  Sample binomial noise $\gamma_2 \sim \textbf{Bin}(b, p)$
  Set m be a multi-set containing $\widehat{x} + \gamma_2$ copies of 1 and $(g + b) - (\widehat{x} + \gamma_2)$ copies of 0.
  return m
end function
function $R_1(\phi(c, a)y)$
  Set $\Delta_1 = 1$
  for each coordinate $k \in [d]$ **do**
    Shift data $w_k = [\phi(c, a)y]_k + \Delta_1$
    Run the scalar randomizer $m_k = \mathcal{R}^*(w_k, \Delta_1)$
  end for
  # Labeled outputs (all bits in m_k are labeled by k)
  $M_1 = \{(k, m_k)\}_{k\in[d]}$
  return M_1
end function
function $R_2(\phi(c, a)\phi(c, a)^\top)$
  Set $\Delta_2 = 1$
  for all $i \le j \le d$ **do**
    Shift data $w_{(i,j)} = [\phi(c, a)\phi(c, a)^\top]_{(i,j)} + \Delta_2$
    Run the scalar randomizer to obtain $m_{(i,j)} = \mathcal{R}^*(w_{(i,j)}, \Delta_2)$ and $m_{(j,i)} = m_{(i,j)}$
  end for
  # Labeled outputs
  $M_2 = \{((i, j), m_{(i,j)})\}_{(i,j)\in[d]\times[d]}$
  return M_2
end function

C. Analysis of Vector Summation Protocol

C.1. Pseudocode of $\mathcal{P}_{\text{Vec}}$

The shuffle protocol is given by $\mathcal{P}_{\text{Vec}} = (\mathcal{R}_{\text{Vec}}, \mathcal{S}_{\text{Vec}}, \mathcal{A}_{\text{Vec}})$, in which $\mathcal{R}_{\text{Vec}}$ is presented in Algorithm 5, $\mathcal{S}_{\text{Vec}}$ is presented in Algorithm 6, and $\mathcal{A}_{\text{Vec}}$ is presented in Algorithm 7. Note that the original algorithm for the analyzer in (Cheu et al., 2021) has a small issue in the de-bias process (cf. Algorithm 2 in (Cheu et al., 2021)). In particular, instead of subtracting the norm Δ, one needs to subtract $B \cdot \Delta$, see Lines 11 and 19 in Algorithm 7. Here, B corresponds to n in Algorithm 2 of (Cheu et al., 2021).

C.2. Main Results

Theorem C.1 (Restatement of Theorem 3.5)**.** *Fix batch size $B \in [T]$, privacy budgets $\varepsilon \in (0, 15]$, $\delta \in (0, 1/2)$. Then, Algorithm 1 instantiated with $\mathcal{P}_{\text{Vec}}$ with parameters $p = 1/4$, $g = \max\{2\sqrt{B}, d, 4\}$ and $b = \frac{24\cdot 10^4 \cdot g^2 \cdot \log^2\left(4(d^2+1)/\delta\right)}{\varepsilon^2 B}$ is (ε, δ)-SDP. Furthermore, for any $\alpha \in (0, 1]$, setting $\lambda = \Theta\left(\frac{\log(d^2/\delta)\sqrt{T}}{\varepsilon\sqrt{B}}(\sqrt{d} + \sqrt{\log(T/(B\alpha))})\right)$, it enjoys the regret*

$$\mathit{Reg}(T) = O\left(dB\log T + \frac{\log^{1/2}(d^2/\delta)}{\varepsilon^{1/2}B^{1/4}} d^{3/4}T^{3/4}\log T \log(T/\alpha)\right),$$

with probability at least $1 - \alpha$.

Corollary C.2 (Utility-targeted)**.** *Under the same assumption in Theorem C.1 and Algorithm 1 is instantiated with $\mathcal{P}_{\text{Vec}}$. Let*

Algorithm 6 Shuffler $\mathcal{S}_{\text{Vec}}$

Input: $\{M_{\tau,1}\}_{\tau\in\mathcal{B}}$ and $\{M_{\tau,2}\}_{\tau\in\mathcal{B}}$, in which $\mathcal{B}$ is a batch of users. $M_{\tau,1} = \{(k, m_k)\}_{k\in[d]}$ and $M_2 = \{((i,j), m_{(i,j)})\}_{(i,j)\in[d]\times[d]}$ are labeled data of user τ
function $S_1(\{M_{\tau,1}\}_{\tau\in\mathcal{B}})$
  Uniformly permutes all messages, i.e., a total of $(g+b)\cdot B\cdot d$ bits
  Set y_k be the collection of bits labeled by $k\in[d]$
  Set $Y_1 = \{y_1, \ldots, y_d\}$
  return Y_1
end function
function $S_2(\{M_{\tau,2}\}_{\tau\in\mathcal{B}})$
  Uniformly permutes all messages, i.e., a total of $(g+b)\cdot B\cdot d^2$ bits
  Set $y_{(i,j)}$ be the collection of bits labeled by $(i,j)\in[d]\times[d]$
  Set $Y_2 = \{y_{(i,j)}\}_{(i,j)\in[d]\times d}$
  return Y_2
end function

$B = O(d^{-1/5}\varepsilon^{-2/5}T^{3/5}\log^{2/5}(d^2/\delta))$, *Algorithm 1 achieves* (ε,δ)*-SDP with regret*

$$Reg(T) = \widetilde{O}\left(d^{4/5}T^{3/5}\varepsilon^{-2/5}\log^{2/5}(d^2/\delta)\right).$$

Simultaneously, Algorithm 1 also achieves $O(\varepsilon,\delta)$*-JDP and* $O(\varepsilon_0,\delta_0)$*-LDP where*

$$\varepsilon_0 = O\left(\varepsilon^{4/5}T^{3/10}d^{-1/10}\log^{1/5}(d^2/\delta)\right), \quad \delta_0 = O(\delta).$$

Corollary C.3 (Privacy-targeted)**.** *Let Assumption 3.1 hold and Algorithm 1 is instantiated with* $\mathcal{P}_{\text{Vec}}$*. For any* $\varepsilon_0\in(0,15]$ *and* $\delta_0\in(0,1/2)$*, let*

$$g = \max\{d,4\}, \quad b = \frac{24\cdot 10^4\cdot g^2\cdot\left(\log\left(\frac{4\cdot(d^2+1)}{\delta_0}\right)\right)^2}{\varepsilon_0^2}, \quad p = 1/4,$$

Then, for all $B\in[T]$*, Algorithm 1 is* (ε_0,δ_0)*-LDP. Further suppose* $B = O(d^{-1/4}T^{3/4}\varepsilon_0^{-1/2}(\log(d^2/\delta_0))^{1/2})$*, then Algorithm 1 achieves regret*

$$Reg(T) = \widetilde{O}\left(d^{3/4}T^{3/4}\frac{\log^{1/2}(d^2/\delta_0)}{\sqrt{\varepsilon_0}}\right).$$

Simultaneously, Algorithm 1 also achieves $O(\varepsilon,\delta)$*-SDP and* $O(\varepsilon,\delta)$*-JDP where*

$$\varepsilon = O\left(\varepsilon^{5/4}T^{-3/8}d^{1/8}(\log(d^2/\delta_0))^{-1/4}\right), \quad \delta = O(\delta_0).$$

C.3. Proofs

Proof of Theorem C.1. The privacy part follows from the one-round SDP guarantee of vector summation protocol in (Cheu et al., 2021). In particular, by Theorem 3.2 in (Cheu et al., 2021), we have to properly choose parameters g, b, p in $\mathcal{R}_{\text{Vec}}$. To this end, by adapting the results of Lemma 3.1 in (Cheu et al., 2021), we have in our case when one chooses

$$g = \max\{2\sqrt{B}, d, 4\}, \quad b = \frac{24\cdot 10^4\cdot g^2\cdot\left(\log\left(\frac{4\cdot(d^2+1)}{\delta}\right)\right)^2}{\varepsilon^2 B}, \quad p = 1/4,$$

$\mathcal{P}_{\text{Vec}}$ is (ε,δ)-SDP. It is worth pointing out that here we choose b such that $p = 1/4$, which is necessary for our following analysis on the tail of the private noise. This is the key difference compared to the original one in (Cheu et al., 2021) where the variance of the noise is sufficient.

Algorithm 7 Analyzer $\mathcal{A}_{\text{Vec}}$

Input: Shuffled outputs $Y_1 = \{y_k\}_{k\in[d]}$ and $Y_2 = \{y_{(i,j)}\}_{(i,j)\in[d]\times d}$
Initialize: g, b, p
Analyzer for a collection y of $(g+b)\cdot B$ bits using Δ
function $\mathcal{A}^*(y, \Delta)$
  return $\frac{\Delta}{g}\left((\sum_{i=1}^{(g+b)\cdot B} y_i) - p\cdot b\cdot B\right)$
end function
function $A_1(Y_1)$
  $\Delta_1 = 1$
  for each coordinate $k \in [d]$ **do**
    Run analyzer on k-th labeled data to obtain $z_k = \mathcal{A}^*(y_k, \Delta_1)$
    Re-center: $o_k = z_k - B\cdot\Delta_1$
  end for
  return $\{o_1, \ldots, o_k\}$
end function
function $A_2(Y_2)$
  $\Delta_2 = 1$
  for all $i \le j \le d$ **do**
    Run analyzer on (i,j)-th labeled data to obtain $z_{(i,j)} = \mathcal{A}^*(y_{(i,j)}, \Delta_2)$
    Re-center: $o_{(i,j)} = z_{(i,j)} - B\cdot\Delta_2$ and $o_{(j,i)} = o_{(i,j)}$
  end for
  return $\{o_{(i,j)}\}_{(i,j)\in[d]\times[d]}$
end function

Now, we turn to regret analysis. Thanks to our general regret bound in Corollary A.4, we only need to verify the condition of sub-Gaussian private noise in the protocol $\mathcal{P}_{\text{Vec}}$ (in particular $\mathcal{R}_{\text{Vec}}$). To this end, we need a more careful analysis compared to (Cheu et al., 2021) as the issue pointed above. Fix any coordinate $k \in [d]$, we will determine the private noise in k, which motivates us to check the scalar randomizer $\mathcal{R}^*$ in $\mathcal{R}_{\text{Vec}}$. Consider a batch of users. Let z_i denote the sum of $g+b$ bits generated by user i using $\mathcal{R}^*$. That is, we have

$$z_i = \bar{x}_i + \gamma_{1,i} + \gamma_{2,i}.$$

This implies that

$$z_i - bp = \frac{g}{\Delta}x_i + \gamma_{1,i} + \bar{x}_i - \frac{g}{\Delta}x_i + \gamma_{2,i} - bp.$$

Define shifted random variables $\iota_{1,i} := \gamma_{1,i} + \bar{x}_i - \frac{g}{\Delta}x_i$ and $\iota_{2,i} := \gamma_{2,i} - bp$. Thus, taking the summation over all i within a given batch $\mathcal{B}$ of size B, yields

$$\sum_{i\in\mathcal{B}} z_i - B\cdot b\cdot p = \frac{g}{\Delta}\sum_{i\in\mathcal{B}} x_i + \sum_{i\in\mathcal{B}}\iota_{1,i} + \sum_{i\in\mathcal{B}}\iota_{2,i},$$

which implies that

$$\frac{\Delta}{g}\left(\sum_{i\in\mathcal{B}} z_i - B\cdot b\cdot p\right) = \sum_{i\in\mathcal{B}} x_i + \frac{\Delta}{g}\sum_{i\in\mathcal{B}}\iota_{1,i} + \frac{\Delta}{g}\sum_{i\in\mathcal{B}}\iota_{2,i}.$$

Note that the above is exactly the output of the analyzer $\mathcal{A}^*$ in $\mathcal{P}_{\text{Vec}}$. Thus, to verify the sub-Gaussian condition in Assumption A.3, we only need to show that the last two terms above are zero-mean and sub-Gaussian random variables. To this end, we note that $\gamma_{1,i}$ is draw from $\textbf{Ber}(\frac{g}{\Delta}x_i - \bar{x}_i)$. Hence, $\mathbb{E}[\iota_{1,i}] = 0$ and $\iota_{1,i}$ is sub-Gaussian with variance 1 since $\iota_{1,i} \in [-1, 1]$. By independence of private noise across i, we have $\sum_{i\in\mathcal{B}}\iota_{1,i}$ is sub-Gaussian with variance of B. Similarly, since $\gamma_{2,i}$ is independently sampled from binomial $\textbf{Bin}(b, p)$, we have $\mathbb{E}[\iota_{2,i}] = 0$ and $\sum_{i\in\mathcal{B}}\iota_{2,i}$ can be viewed as a sum of $B\cdot b$ bounded random variable within $[0, 1]$, hence it is sub-Gaussian with variance of $B\cdot b/4$. Therefore, the total noise

$\frac{\Delta}{g}\sum_{i\in\mathcal{B}}\iota_{1,i}+\frac{\Delta}{g}\sum_{i\in\mathcal{B}}\iota_{2,i}$ is sub-Gaussian with variance given by

$$\frac{\Delta^2}{g^2}\cdot B+\frac{\Delta^2}{g^2}\cdot B\cdot b/4\overset{(a)}{=}\frac{1}{g^2}\cdot B+\frac{1}{g^2}\cdot B\cdot b/4=O\left(\frac{(\log(d^2/\delta))^2}{\varepsilon^2}\right).$$

where (a) holds by the fact that in $\mathcal{P}_{\text{Vec}}$, $\Delta=1$. Thus, this implies that $\widetilde{\sigma}_1^2,\widetilde{\sigma}_2^2$ in Assumption A.3 are satisfied with $O\left(M\frac{(\log(d^2/\delta))^2}{\varepsilon^2}\right)$, hence σ in Lemma A.4 is given by $\sigma=O\left(\sqrt{T/B}\frac{\log(d^2/\delta)}{\varepsilon}\right)$, which leads to the following regret bound

$$R(T)=\tilde{O}\left(dB+\frac{(\log(d^2/\delta))^{1/2}}{\sqrt{\varepsilon}}d^{3/4}B^{-1/4}T^{3/4}\right),$$

Hence, we finish the proof. □

Proof of Corollary C.2. The regret bound simply follows from a balanced choice of B in Theorem C.1. As before, JDP follows from SDP and Billboard lemma. To show the LDP guarantee, one way is to use DP property of Binomial mechanism and the refined advanced composition in (Cheu et al., 2021) across dimensions (cf. Lemma 3.3 in (Cheu et al., 2021)). However, there is a simple way to achieve this by noting that when $B=1$, the SDP guarantee of $\mathcal{P}_{\text{Vec}}$ also implies LDP guarantee since now the shuffle output is the same as the output at each local randomizer[9]. Thus, by comparing the values of b for a general B and the case when $B=1$, we can see that $\varepsilon_0=\varepsilon\sqrt{B}$ and $\delta_0=\delta$, i.e., an implicit privacy amplification by $\sqrt{B}$. Note that, this simple way might lead to a larger term in δ. A careful analysis via Binomial mechanism and the (refined) advanced composition could yield something like $\varepsilon_0=\varepsilon\sqrt{B}/\sqrt{\log(d^2/\delta)}$ and $\delta_0=\delta/d^2$, where d^2 comes from the $d\times d$ matrix in the computation. Here we choose the simple way to avoid additional complexity for clarity. □

Proof of Corollary C.3. The LDP guarantee follows from the same trick as in the proof of Corollary C.2 which helps to avoid Binomial mechanism and advance composition over dimensions. To establish the regret bound, we can compare the values of b in Corollary C.3 and the one in Theorem C.1. In particular, we can plug $\varepsilon=\frac{\varepsilon_0}{\sqrt{B}}$ and $\delta=\delta_0$ into the regret bound in Theorem C.1. Then, with a balanced choice of B, we obtain the required regret. The SDP guarantee also follows from Theorem C.1 with $\varepsilon=\frac{\varepsilon_0}{\sqrt{B}}$ and $\delta=\delta_0$. Finally, as in the proof of Corollary B.2, the JDP guarantee follows from SDP guarantee and Billboard lemma. □

D. Joint Differenital Privacy

In this section, we will give formal DP definitions in the central model for linear contextual bandits. In particular, we first present the standard (event-level) definition which assumes all users are unique and then generalize it to (user-level) definition that allows for returning users. To this end, we first give the following general DP definition.

Definition D.1 (General DP)**.** A randomized mechanism $\mathcal{M}:\mathcal{D}\to\mathcal{R}$ satisfies (ε,δ)-differential privacy if for any two adjacent datasets $X,X'\in\mathcal{D}$ and for any measurable subsets of outputs $\mathcal{Y}\subseteq\mathcal{R}$ it holds that

$$\mathbb{P}\left[\mathcal{M}(X)\in\mathcal{Y}\right]\leq\exp(\varepsilon)\mathbb{P}\left[\mathcal{M}(X')\in\mathcal{Y}\right]+\delta.$$

Remark D.2. All the DP definitions in our main paper can be viewed as a particular instantiation of Definition D.1 in terms of adjacent relation between two datasets and the corresponding output sequences.

A straightforward adaptation of Definition D.1 to linear contextual bandits in the central model is to consider the sequence of T unique users as the dataset, denoted by $U_T:=\{u_1,\ldots,u_T\}\in\mathcal{U}^T$, and the corresponding prescribed actions as the output sequence, denoted by $\mathcal{M}(U_T):=\{a_1,\ldots,a_t\}\in\mathcal{A}^T$. This is the central trust model because the learning agent in the protocol can have direct access to users' sensitive information, but all the prescribed actions via the deployed algorithm are indistinguishable on two neighboring user sequences. Unfortunately, it is not hard to see that this is in conflict with the goal of personalization of linear contextual bandits, which essentially requires the algorithm to prescribe different actions to different users according to their contexts. Indeed, as shown in (Shariff & Sheffet, 2018), any learning protocol that satisfy the above notion of privacy protection has to incur a linear regret. Hence, to obtain a non-trivial utility-privacy trade-off, we need to relax DP to the notion called *joint differential privacy* (JDP) (Kearns et al., 2014) in the central model, which

[9]Here, we can assume that each local randomizer already randomly orders the $g+b$ bits before they are sent out.

requires that simultaneously for any user $u_t \in U_T$, the joint distribution of the actions recommended to all users other than u_t be differentially private in the type of the user u_t. It weakens the classic DP notion only in that the action suggested specifically to u_t may be sensitive in her type (i.e., context and reward responses[10]), as required by personalization. However, JDP is still a very strong definition since it protects u_t from any arbitrary collusion of other users against her, so long as she does not herself reveal the action suggested to her. Formally, we let $\mathcal{M}_{-t}(U_T) := \mathcal{M}(U_T) \setminus \{a_t\}$ to denote all the actions prescribed by the deployed algorithm excluding the one recommended to u_t and based on it we have the definition of JDP as follows.

Definition D.3 (Joint Differential Privacy (JDP))**.** A learning process of linear contextual bandits is (ε, δ)-joint differentially private if its deployed algorithm $\mathcal{M} : \mathcal{U}^T \to \mathcal{A}^T$ satisfies that for all $t \in [T]$, for all neighboring user sequences $U_T, U_T' \in \mathcal{U}^T$ differing only on the t-th user and for all set of actions $\mathcal{A}_{-t} \subseteq \mathcal{A}^{T-1}$ given to all but the t-th user,

$$\mathbb{P}\left[\mathcal{M}_{-t}(U_T) \in \mathcal{A}_{-t}\right] \leq \exp(\varepsilon)\mathbb{P}\left[\mathcal{M}_{-t}(U_T') \in \mathcal{A}_{-t}\right] + \delta.$$

The above JDP definition assumes that all the T users are unique, which is the standard event-level DP considered in existing similar works (Shariff & Sheffet, 2018; Vietri et al., 2020; Chowdhury & Zhou, 2021). That is, since each user only contributes one event in the total T rounds, two user sequences U_T and U_T' are said to be adjacent if they only differ at one round $t \in [T]$.

However, a more practical situation is that one user could contribute her data at multiple rounds, i.e., returning users. This motivates us to consider a user-level JDP, in which two user sequences U_T and U_T' are adjacent if one replaces all the data associated with user u to u' in U_T results in U_T'. In this case, changing one user in the sequence could affect the data at multiple rounds. Accordingly, the output sequences need to remove all the actions at these rounds to avoid the conflict with personalization. Following the notations in (Dwork et al., 2010), we say U_T and U_T' are neighboring sequences if there exist u, u' such that if one replace some of u in U_T, the resultant sequence is U_T'. Formally, U_T, U_T' are neighboring with neighboring indices $\mathcal{I}$, if there exist $u, u' \in \mathcal{U}$ and index set $\mathcal{I} \subseteq [T]$ such that $U_T|_{\mathcal{I}:u\to u'} = U_T'$, in which $U_T|_{\mathcal{I}:u\to u'}$ means replacing u by u' in U_T at all indices in $\mathcal{I}$. Meanwhile, we let $\mathcal{M}_{-\mathcal{I}}(U_T) := \mathcal{M}(U_T) \setminus a_{\mathcal{I}}$, where $a_{\mathcal{I}}$ is the set of actions at indices in $\mathcal{I}$. With these notations, we have the following formal definition.

Definition D.4 (User-level JDP)**.** A learning process of linear contextual bandits is (ε, δ)-joint differentially private if its deployed algorithm $\mathcal{M} : \mathcal{U}^T \to \mathcal{A}^T$ satisfies that for all neighboring user sequences $U_T, U_T' \in \mathcal{U}^T$ with neighboring indices given by $\mathcal{I}$, and for all set of actions $\mathcal{A}_{-\mathcal{I}} \subseteq \mathcal{A}^{T-|\mathcal{I}|}$,

$$\mathbb{P}\left[\mathcal{M}_{-\mathcal{I}}(U_T) \in \mathcal{A}_{-\mathcal{I}}\right] \leq \exp(\varepsilon)\mathbb{P}\left[\mathcal{M}_{-\mathcal{I}}(U_T') \in \mathcal{A}_{-\mathcal{I}}\right] + \delta.$$

Remark D.5. A straightforward way to achieve user-level JDP via event-level JDP is to use group privacy property of DP (Dwork et al., 2014; Vietri et al., 2020). In particular, suppose a mechanism is (ε, δ)-JDP (event-level), then it is $(k\varepsilon, ke^{(k-1)\varepsilon}\delta)$-JDP (user-level) if each user contributes at most k rounds. This black-box approach leads to a large increase in δ. We will show that a careful and direct analysis can improve this part while the linear increase in ε is unchanged. This makes sense since the sensitivity now increases by a factor of k.

E. Regret and Privacy Analysis Under Returning Users

We consider the following returning users case.

Assumption E.1 (Returning Users)**.** Fix a batch size B, any particular user can potentially participates in *all* $M = T/B$ batches, but within each batch $m \in [M]$, she only contributes once.

Under the above assumption, our previous SDP guarantee from one-round SDP protocol is no longer true. Instead, we now need to guarantee that outputs of all the batches together have a total privacy loss of (ε, δ), since all of them may reveal the sensitive information of a given user if she participates in all the batches, i.e., worst-case scenario. To this end, we resort to advanced composition theorem (Dwork et al., 2014), which is restated as follows for an easy reference.

Theorem E.2 (Advanced composition)**.** *Given target privacy parameters $\varepsilon' \in (0, 1)$ and $\delta' > 0$, to ensure $(\varepsilon', k\delta + \delta')$-DP for the composition of k (adaptive) mechanisms, it suffices that each mechanism is (ε, δ)-DP with $\varepsilon = \frac{\varepsilon'}{2\sqrt{2k\log(1/\delta')}}$.*

[10]Technically speaking, the type of the user is identified by the reward response she would give to all possible actions recommended based on her context information.

E.1. LDP Amplification Protocol

Theorem E.3 (Formal statement of (i) in Theorem 4.2)**.** *Let Assumption 3.1 and Assumption E.1 hold. For any* $\varepsilon \in [0, \frac{2}{B}\log(2/\delta)\sqrt{2T}]$, $\delta \in (0,1]$ *and* $B \in [T]$, *let* $\sigma_1 = \sigma_2 = \frac{16\log(2/\delta)\sqrt{T(\log(5T/\delta))}}{\varepsilon B}$. *Then, Algorithm 1 instantiated using* $\mathcal{P}_{Amp}$ *is* $O(\varepsilon,\delta)$*-SDP. Furthermore, for any* $\alpha \in (0,1]$, *setting* $\lambda = \Theta(\sqrt{T}\sigma_1(\sqrt{d} + \sqrt{\log(T/B\alpha)})$, *it has the following regret*

$$Reg(T) = O\left(\frac{dT}{M}\log T + \sqrt{\frac{MT}{\varepsilon}}d^{3/4}\log^{1/4}(T/\delta)\log^{1/2}(2/\delta)\log T\log(T/\alpha)\right).$$

The following corollary says that if the batch schedule also depends on privacy parameters, one can improve the dependence on ε, i.e., from $\varepsilon^{-1/2}$ to $\varepsilon^{-1/3}$.

Corollary E.4 (Utility-targeted)**.** *Under the same assumption in Theorem E.3 and* $B = O(d^{-1/6}\varepsilon^{-1/3}T^{2/3}(\log(T/\delta))^{1/2})$, *Algorithm 1 instantiated using* $\mathcal{P}_{Amp}$ *achieves* $O(\varepsilon,\delta)$*-SDP with regret*

$$R(T) = \tilde{O}\left(d^{5/6}T^{2/3}\varepsilon^{-1/3}\left(\log(T/\delta)\right)^{1/2}\right).$$

Proof of Theorem E.3. First, by advanced composition in Theorem E.2, if we let each batch's privacy parameters be $\varepsilon_m = \frac{\varepsilon}{2\sqrt{2M\log(2/\delta)}}$ and $\delta_m = \delta/(2M)$, then final privacy guarantee is (ε,δ)-DP. Thus, we only need to replace ε by ε_m and δ by δ_m in Theorem B.1 □

E.2. Vector Summation Protocol

Theorem E.5 (Formal statement of (ii) in Theorem 4.2)**.** *Let Assumption 3.1 and Assumption E.1 hold. Then, for any* $\varepsilon \le 15$, $\delta \in (0, 1/2)$ *and* $B \in [T]$, *let*

$$g = \max\{2\sqrt{B}, d, 4\}, \quad b = \frac{10^7\cdot\log(2/\delta)\cdot g^2\cdot T\cdot\left(\log\left(\frac{8\cdot T(d^2+1)}{B\delta}\right)\right)^2}{\varepsilon^2B^2}, \quad p = 1/4.$$

Algorithm 1 instantiated using $\mathcal{P}_{Vec}$ *is* (ε,δ)*-SDP. Furthermore, for any* $\alpha \in (0,1]$, *setting*

$$\lambda = \Theta\left(\frac{T\sqrt{\log(2/\delta)}\log(d^2T/(B\delta))}{B}\left(\sqrt{d} + \sqrt{\log(T/(B\alpha))}\right)\right),$$

then it has the regret bound

$$Reg(T) = O\left(\frac{dT}{M}\log T + \sqrt{\frac{MT}{\varepsilon}}d^{3/4}\log^{3/4}(d^2M/\delta)\log T\log(T/\alpha)\right).$$

Corollary E.6 (Utility-targeted)**.** *Under the same assumption in Theorem E.5,* $B = O(d^{-1/6}\varepsilon^{-1/3}T^{2/3}(\log(Td^2/\delta))^{1/2})$, *Algorithm 1 instantiated using* $\mathcal{P}_{Vec}$ *achieves* (ε,δ)*-SDP with regret*

$$R(T) = \widetilde{O}\left(d^{5/6}T^{2/3}\varepsilon^{-1/3}\left(\log(d^2T/\delta)\right)^{1/2}\right).$$

Proof of Theorem E.5. First, by advanced composition in Theorem E.2, if we let each batch's privacy parameters be $\varepsilon_m = \frac{\varepsilon}{2\sqrt{2M\log(2/\delta)}}$ and $\delta_m = \delta/(2M)$, then final privacy guarantee is (ε,δ)-DP. Thus, we only need to replace ε by ε_m and δ by δ_m in Theorem C.1 □

E.3. JDP under Returning Users

As mentioned before, existing algorithm with JDP guarantee assumes *unique* users, i.e., event-level JDP given by Definition D.3. To handle returning users, we need to consider user-level JDP given by Definition D.4. One straightforward way is

to resort to group privacy (Dwork et al., 2014). That is, if any user appears at most M_0 rounds in the process, the original (ε, δ)-JDP algorithm proposed in (Shariff & Sheffet, 2018) now achieves $(M_0\varepsilon, M_0 \exp((M_0-1)\varepsilon)\delta)$-JDP (user-level). However, this black-box will incur a large loss in the δ term. To overcome this, we note that a simple modification of the added noise in the original algorithm in (Shariff & Sheffet, 2018) will work. In particular, we scale up the noise variance by a multiplicative factor of M_0^2, if any user participates in at most M_0 rounds. This follows from the fact that flipping one user now would change the ℓ_2 sensitivity of the expanded binary-tree nodes from $O(\sqrt{\log T})$ to $O(M_0\sqrt{\log T})$. Then, utilizing our derived generic regret bound in Lemma A.4, yields the following result.

Proposition E.7. *If any user participates in at most M_0 rounds, the algorithm in Shariff & Sheffet (2018), with the above modification to handle user-level privacy, achieves the high-probability regret bound*

$$\textit{Reg}(T) = \widetilde{O}\left(d\sqrt{T} + \sqrt{\frac{M_0 T}{\varepsilon}} d^{3/4} \log^{1/4}(1/\delta)\right).$$

Proof. The key idea behind the regret analysis in the central model for linear contextual bandits in (Shariff & Sheffet, 2018) is to utilize the following two properties of the so-called tree-based mechanism (or binary counting mechanism) (Chan et al., 2010): (i) change of each leaf-node (corresponding to a user's data) only incurs the change of l_2-sensitivity of the expanded binary-tree by $O(\sqrt{\log T})$; (ii) for any $t \in [T]$, the summation of data from time 1 to t only involves at most $O(\log T)$ tree nodes. Property (i) is used to compute the added noise at each node to guarantee privacy while property (ii) is used to compute the total noise in the private sum when bounding the regret. Now, in the case of returning users, if we flip one user's data, it will change the l_2-sensitivity of the expanded binary-tree by $O(M_0\sqrt{\log T})$, i.e., an additional M_0 factor in the sensitivity, which leads to the additional M_0^2 factor in the added noise. Property (ii) is the same as before, i.e., total number of noise is at most $O(\log T)$. Finally, by Lemma A.4, we have the result. □

F. Batched Algorithms for Local and Central Models

To start with, for the batched algorithm in the local model, one can simply replace the shuffler in Algorithm 1 by an identity mapping while using the same local randomizer as in (Zheng et al., 2020) (i.e., Gaussian mechanism). We call this algorithm *Batched-Local-LinUCB*. Thanks to Lemma A.4, we have the following privacy and regret guarantees.

Proposition F.1. *Let Assumption 3.1 hold. Fix any $\varepsilon_0 \in [0,1]$, $\delta_0 \in (0,1]$ and $\alpha \in [0,1]$, let $\sigma_1 = \sigma_2 = \frac{4\sqrt{2\log(2.5/\delta_0)}}{\varepsilon_0}$. Then, for all $B \in [T]$, Bathed-Local-LinUCB is $(\varepsilon_0, \delta_0)$-LDP and with probability at least $1-\alpha$*

$$R(T) = \tilde{O}\left(dB + T^{3/4} d^{3/4} \frac{(\log(1/\delta))^{1/4}}{\sqrt{\varepsilon_0}} \log(T/\alpha)\right).$$

Remark F.2. The above theorem indicates that it suffices to update every $B = \tilde{O}(T^{3/4})$ to ensure the same privacy and regret guarantees as in the sequential case.

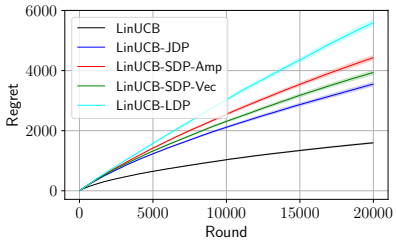
For the batched algorithm in the central model, we can make the following simple modification over the sequential one in (Shariff & Sheffet, 2018), which relies on the seminal tree-based algorithm (Chan et al., 2010) at the central server (analyzer) to balance between privacy and regret. In the batched case, instead of updating the binary-tree nodes after every round, the server updates them only after each batch by treating the the sum of the statistics (i.e., vectors or matrices) within the batch as a single new observation. We call this algorithm *Batched-Central-LinUCB*. With this modification and Lemma A.4, we have the following privacy and regret guarantees.

Proposition F.3. *Let Assumption 3.1 hold. Fix any $\varepsilon \in [0,1]$, $\delta \in (0,1]$ and $\alpha \in [0,1]$. Then, for all $B \in [T]$, Bathed-Central-LinUCB is (ε, δ)-JDP and with probability at least $1-\alpha$*

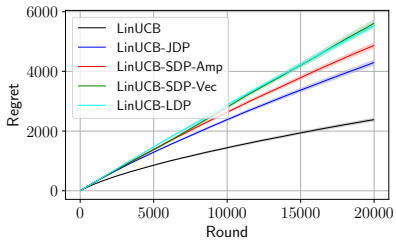
$$R(T) = \tilde{O}\left(dB + \sqrt{T} d^{3/4} \frac{(\log(1/\delta))^{1/4}}{\sqrt{\varepsilon}} \log(T/\alpha)\right).$$

Remark F.4. The above theorem indicates that it suffices to update every $B = \tilde{O}(\sqrt{T})$ to attain the same privacy-regret trade-off as in the sequential case.

G. Additional Experimental Results



(a) $d = 10$



(b) $d = 15$

Figure 2: Comparison of cumulative regret for LinUCB (non-private), LinUCB-JDP (central model), LinUCB-SDP (shuffle model) and LinUCB-LDP (local model) with privacy level $\varepsilon = 1$ for varying feature dimension $d = 10$ (a) and $d = 15$ (b). In all cases, regret of LinUCB-SDP lies perfectly in between LinUCB-JDP and LinUCB-LDP, achieving finer regret-privacy trade-off.