Spreading the Privacy Blanket: Differentially Oblivious Shuffling for Differential Privacy

Dov Gordon[1]*, Jonathan Katz[2]**, Mingyu Liang[1], and Jiayu Xu[3]***

[1] George Mason University
{gordon,mliang5}@gmu.com
[2] University of Maryland
jkatz2@gmail.com
[3] Algorand
jiayux@uci.edu

Abstract. In the *shuffle model* for differential privacy, n users locally randomize their data and submit the results to a trusted "shuffler" who mixes the results before sending them to a server for analysis. This is a promising model for real-world applications of differential privacy, as several recent results have shown that the shuffle model sometimes offers a strictly better privacy/utility tradeoff than what is possible in a purely local model.

A downside of the shuffle model is its reliance on a trusted shuffler, and it is natural to try to replace this with a distributed shuffling protocol run by the users themselves. While it would of course be possible to use a fully secure shuffling protocol, one might hope to instead use a more-efficient protocol having weaker security guarantees.

In this work, we consider a relaxation of secure shuffling called *differential obliviousness* that we prove suffices for differential privacy in the shuffle model. We also propose a differentially oblivious shuffling protocol based on onion routing that requires only $O(n \log n)$ communication while tolerating any constant fraction of corrupted users. We show that for practical settings of the parameters, our protocol outperforms existing solutions to the problem in some settings.

Keywords: Differential privacy · Onion routing.

1 Introduction

Differential privacy [19] has become a leading approach for privacy-preserving data analysis. Traditional mechanisms for differential privacy operate in the *curator model*, where a trusted server holds all the sensitive data and releases noisy statistics about that data. To reduce the necessary trust assumptions,

* Work supported by NSF awards #1942575 and #1955264.

** Work supported in part by a Facebook Privacy Enhancing Technologies Award and NSF award #1837517. Portions of this work were done while at GMU.

*** Portions of this work were done while at GMU.

researchers subsequently proposed the *local model* of differential privacy. Here, each user applies a local randomizer $\mathcal{R}$ to its sensitive data x_i to obtain a noisy result y_i, and then forwards y_i to a server who analyzes all the noisy data it obtains. A drawback of local mechanisms is that, in some cases, they provably require more noise (and hence offer reduced utility) than mechanisms in the curator model for a fixed level of privacy. For example, computing a differentially private mean of n users' inputs can be done with $O(1)$ noise in the centralized curator model [19] but requires $\Omega(\sqrt{n})$ noise in the local model [5, 13].

A recent line of work has explored an intermediate model that provides a tradeoff between these extremes. In the *shuffle model* [8, 16, 36, 4], users locally add noise to their data as in the local model, but also have access to a trusted entity $\mathcal{S}$ (a "shuffler") that anonymizes their data before it is forwarded to the server. That is, whereas in the local model the server obtains the ordered vector of noisy inputs $(y_1, \ldots, y_n)$, in the shuffle model the server is given only the multiset $\{y_i\} := \mathcal{S}(y_1, \ldots, y_n)$ which hides information about which element was contributed by which user. (The $\{y_i\}$ can be encrypted with the server's public key before being sent to the shuffler so the shuffler does not learn the value submitted by any user.) Balle et al. [4] analyze the result of composing a local differentially private mechanism with a shuffler, and show a setting where the shuffle model offers a strictly better privacy/utility tradeoff than what is possible in the local model.

Although the shuffle model relies on a weaker trust assumption than the curator model, it may still be undesirable to rely on a trusted shuffler who is assumed not to collude with the curator. It is thus natural to consider replacing the shuffler by a distributed protocol executed by the users themselves. Clearly, using a fully secure shuffling protocol to instantiate the shuffler preserves the privacy guarantees of the shuffle model. However, fully secure distributed-shuffling protocols are inefficient in practice (see Section 1.1).

Our contributions. We consider a relaxation of oblivious shuffling that we call *differential obliviousness.* (Prior work has considered the same or similar notions in other settings; see Section 1.1.) Roughly, for any pair of honest users and any pair of values y, y', a differentially oblivious shuffling protocol hides (in the same sense as differential privacy) whether the first user contributed y and the second user contributed y', or vice versa. Generalizing the results of Balle et al. [4], we analyze the privacy obtained by composing a local differentially private mechanism with any *differentially oblivious* shuffling protocol, and show that such shuffling protocols suffice to replace the trusted shuffler.

With this result in place, we then seek an efficient differentially oblivious shuffling protocol. In the context of anonymous communication, Ando et al. [1] show a differentially oblivious shuffling protocol using $O(n \log n)$ communication.[1] Their protocol is based on *onion routing*, in which each user routes its

[1] Ando et al. consider a "many-to-many" variant of shuffling, where each of the n users wants to send a message to a distinct recipient, in contrast to our setting where all n inputs are sent to a designated receiver. Nevertheless, their results can be applied to our setting with minor modifications, so we ignore the distinction.

message to the server via a path of randomly chosen users, with nested encryption being used to hide from each intermediate user everything about the route except for the previous and next hops. Ando et al. analyze the privacy of onion routing against an adversary who corrupts some fraction of the users in the network in addition to the server, and who is also assumed to be able to eavesdrop on all communication in the network. While such an adversary may be appropriate in the context of using anonymous communication to evade state-sponsored censorship, we believe it is overkill for most deployments of differential privacy that could benefit from the shuffle model. Instead, we consider a weaker adversary who can only monitor the communications of corrupted users, and analyze the differential obliviousness of onion routing in this model. Our analysis uses very different techniques from those of Ando et al., and results in better concrete parameters as well as an asymptotic improvement in the average per-user communication complexity for a fixed level of privacy.

As in the work of Ando et al., we can adapt our protocol to handle a malicious adversary by routing dummy messages alongside real ones and checking partway along the route whether any dummy messages have been dropped. Focusing on the application to the shuffle model, we observe that the overall privacy degrades smoothly if only a few (real) messages are dropped—a dropped message is similar to reducing the number of honest users by one—and thus a secure protocol only needs to abort when many messages are dropped by the adversary. As a consequence, we are able to address malicious behavior with lower overhead (compared to the semi-honest setting) than Ando et al.

1.1 Related Work

Secure shuffling. There is a long line of work studying secure shuffling protocols. We survey some of what is known, restricting attention to protocols secure against $t = \Theta(n)$ corruptions.

Fully secure shuffling can be done via secure computation of a permutation network [24, 32], or by having $t+1$ parties sequentially perform a local shuffle [24, 29]. Either approach requires $\Omega(n^2)$ communication. While the asymptotic communication complexity can be improved to $O(n \log n)$ by using $\Theta(\log n)$-size committees (cf. [17, 9, 31]), the concrete efficiency of that approach is unclear.

Movahedi et al. [31] considered a relaxed notion of shuffling where security may fail with probability $O(1/n^3)$; this can be viewed as a form of differential obliviousness. The communication complexity of their protocol is $O(n \cdot \mathsf{polylog}\, n)$. Bell et al. [6] proposed a different approach for achieving a relaxed form of shuffling. Their construction requires $O(n^2)$ communication in general, but can be improved to $O(n \log n)$ for constant-size input domains. These protocols and that of Ando et al. [1] (discussed earlier) are the only practical protocols for shuffling we know of with sub-quadratic communication complexity.

To the best of our knowledge, the protocol of Bell et al. has the best concrete efficiency of any prior shuffling protocol. They are also motivated by applications to the shuffle model, but do not prove that their relaxation provides differential privacy when composed with a local differentially private mechanism. Their protocol also does not provide a smooth tradeoff between privacy and performance as our approach does. We provide a comparison between our shuffling protocol and prior work (for the malicious setting) in Section 6.

In concurrent work, Bünz et al. [10] propose a differentially oblivious shuffling protocol that relies on a very strong form of trusted setup.

Anonymous communication. Sender-anonymous communication can be used to implement oblivious shuffling. DC-nets [15] and mix networks [14], two classical approaches for anonymous communication, both require $\Omega(n^2)$ communication for security against a constant fraction of corrupted parties.

Backes et al. [3] proposed a security definition for anonymous routing inspired by differential privacy, and Kuhn et al. [25] gave a definition of security nearly identical to our definition of differential obliviousness. Neither of theoe works show protocols realizing their definitions. Several recent anonymous communication systems [34, 35, 27] also define security in terms of differential privacy, but the maximum per-user communication complexity of these systems is $\Omega(n)$. None of these works consider how anonymous-communication protocols compose with other differentially private mechanisms.

Bellet et al. [7] study "gossip" protocols that provide differential privacy. The model they consider is quite different from ours, and they focus on one-to-many communication rather than many-to-one communication as we do here.

The onion routing protocol [21, 33, 1] that we study in this paper is used as part of the Tor anonymous communication network (though Tor uses paths with only three intermediate nodes). Although Tor has received a lot of attention in the security community, most of that work focuses on active attacks and/or attacks that are specific to Tor. While theoretical analyses of the anonymity provided by onion routing exist [28, 20, 2, 1, 11, 18, 26], none (other than the work of Ando et al. [1]) prove differential obliviousness.

Differentially private computation. The idea of relaxing security for distributed protocols in the context of differential privacy has appeared in a number of prior works [5, 23, 29, 12, 22, 30]. Beimel et al. [5] first proposed the idea, and studied how the relaxation impacts efficiency for the problem of secure summation. He et al. [23] and Groce et al. [22] construct differentially private set-intersection protocols that are more efficient than fully secure protocols for the same task. Mazloom and Gordon [29], and Mazloom et al. [30] leverage differential privacy to make graph-parallel computations more efficient. Chan et al. [12] consider a version of differential obliviousness (defined differently from ours) in the client/server model, studying sorting, merging, and range-query data structures under that relaxation.

2 Definitions

Differential privacy. We use the standard notion of (approximate) differential privacy. Two vectors of inputs $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{x}' = (x'_1, \ldots, x'_n)$ are called *neighboring* if they differ at a single index; i.e., if there exists an index i such that $x_i \neq x'_i$ but $x_j = x'_j$ for $j \neq i$. Let f denote a randomized process mapping a vector of inputs $(x_1, \ldots, x_n) \in D^n$ to an output in some range R. We say that f satisfies (ϵ, δ)-*approximate differential privacy* if for all neighboring vectors $\mathbf{x}, \mathbf{x}' \in D^n$ and subsets $R' \subseteq R$ we have

$$\Pr[f(\mathbf{x}) \in R'] \leq e^{\epsilon} \cdot \Pr[f(\mathbf{x}') \in R'] + \delta.$$

If f satisfies $(\epsilon, 0)$-approximate differential privacy then we simply say that f is ϵ-*differentially private.* For compactness, we abbreviate these as (ϵ, δ)-DP/ϵ-DP.

Local differential privacy and the randomized response mechanism. In the setting of local differential privacy (LDP), each user U_i applies a randomized function $\mathcal{R}$ to their own input x_i and then sends the result y_i to an untrusted server. Translating the guarantees of differential privacy to this setting, we say that $\mathcal{R}$ is (ϵ, δ)-LDP if for all $x, x' \in D$ and $R' \subseteq R$ we have

$$\Pr[\mathcal{R}(x) \in R'] \leq e^{\epsilon} \cdot \Pr[\mathcal{R}(x') \in R'] + \delta.$$

If $\mathcal{R}$ is $(\epsilon, 0)$-LDP then we simply say it is ϵ-LDP.

Let $\gamma \in (0, 1)$ be a parameter, and let D denote a discrete domain in which users' inputs lie. The randomized response mechanism $\mathcal{R}_{\gamma,D}$ is defined as

$$\mathcal{R}_{\gamma,D}(x) = \begin{cases} x & \text{with probability } 1-\gamma \\ y \leftarrow D & \text{with probability } \gamma \end{cases};$$

i.e., with probability γ a user replaces its input with a uniform value in D, and with the remaining probability leaves its input unchanged. It is not hard to show that if $\gamma \geq |D|/(e^{\epsilon} + |D| - 1)$ then $\mathcal{R}_{\gamma,D}$ is ϵ-LDP.

The shuffle model. In the *shuffle model* [8, 16, 36, 4] each user U_i computes $y_i \leftarrow \mathcal{R}(x_i)$ as in the local model, but then sends y_i to a trusted "shuffler" $\mathcal{S}$. After receiving a message from all n users, $\mathcal{S}$ outputs the multiset (which can also be viewed as a *histogram*) $h = \{y_i\}$. If we overload notation and let $\mathcal{S}$ also denote the process of mapping a list of elements to the multiset containing those elements, then $\mathcal{R}$ defines the randomized process

$$\mathcal{S} \circ \mathcal{R}^{\otimes n} \stackrel{\text{def}}{=} \mathcal{S} \circ (\mathcal{R} \times \cdots \times \mathcal{R})(x_1, \ldots, x_n) = \mathcal{S}\left(\mathcal{R}(x_1), \ldots, \mathcal{R}(x_n)\right).$$

Balle et al. [4] showed that under certain conditions the shuffle model improves the privacy of an LDP mechanism.[2]

Theorem 1. *Let $\mathcal{R}$ be an ϵ-LDP mechanism. If $\epsilon \leq \log(n/\log(1/\delta))/2$, then $\mathcal{S} \circ \mathcal{R}^{\otimes n}$ is (ϵ', δ)-DP with $\epsilon' = O(\min\{1, \epsilon\} \cdot e^{\epsilon}\sqrt{\log(1/\delta)/n})$.*

For the particular case of randomized response they show

Theorem 2. *Fix values n, ϵ, δ, and D. If $\gamma \geq \max\left\{\frac{14 \cdot |D| \log(2/\delta)}{(n-1)\cdot\epsilon^2}, \frac{27 \cdot |D|}{(n-1)\cdot\epsilon}\right\}$, then $\mathcal{S} \circ \mathcal{R}_{\gamma,D}^{\otimes n}$ is (ϵ, δ)-DP.*

[2] For clarity, we state a slightly looser bound than what they prove.

Differentially private protocols. More generally, we may consider interactive protocols executed by a server and n users, each of whom initially holds an input x_i. The server has no input, and is the only party to generate an output. We say that a protocol Π *implements* a (randomized) function f if the honest execution of Π when the users hold inputs $x_1, \ldots, x_n$, respectively, results in the server generating output distributed according to $f(x_1, \ldots, x_n)$.

In this setting, the server's view may contain more than just its output. It is also natural to consider that some of the users executing the protocol may themselves be corrupted and colluding with the server. (For simplicity, in what follows we assume semi-honest corruptions; i.e., we assume corrupted parties—including the server—follow the protocol as directed, but may then try to learn additional information based on their collective view of the protocol execution. The definitions can be extended in the obvious way to handle malicious behavior.) Given a set of parties A (that we assume by default always includes the server), we let $\textsc{view}_{\Pi,A}(x_1, \ldots, x_n)$ be the random variable denoting the joint view of the parties in A in an execution of protocol Π when the users initially hold inputs $x_1, \ldots, x_n$. Let H denote the set of users not in A; let $\mathbf{x}_A$ denote the inputs of users in A; and let $\mathbf{x}_H$ denote the inputs of users outside of A. Then:

Definition 1. *Protocol Π is (ϵ, δ)-*DP for t corrupted users *if for any set A containing the server and up to t users and any $\mathbf{x}_A$, the function mapping $\mathbf{x}_H$ to $\textsc{view}_{\Pi,A}(\mathbf{x}_A, \mathbf{x}_H)$ is (ϵ, δ)-DP, i.e., for any neighboring $\mathbf{x}_H, \mathbf{x}'_H$ and any set V of possible (joint) views of the parties in A, we have*

$$\Pr[\textsc{view}_{\Pi,A}(\mathbf{x}_A, \mathbf{x}_H) \in V] \leq e^{\epsilon} \cdot \Pr[\textsc{view}_{\Pi,A}(\mathbf{x}_A, \mathbf{x}'_H) \in V] + \delta.$$

The above can be relaxed to *computational* differential privacy as well.

One can also consider protocols operating in a hybrid world. The shuffle model is a special case of this, where the parties have access to an ideal functionality $\mathcal{S}$ implementing the shuffler. Concretely, the protocol $(\mathcal{R}_{\gamma,D} \times \cdots \times \mathcal{R}_{\gamma,D})^{\mathcal{S}}$ corresponding to the randomized response mechanism is the one in which each user locally computes $y_i \leftarrow \mathcal{R}_{\gamma,D}(x_i)$ and then sends y_i to $\mathcal{S}$, which sends the result $\{y_i\} := \mathcal{S}(y_1, \ldots, y_n)$ to the server. The fact that some of the users themselves might be corrupted, however, now needs to be taken into account. For example, the following is a corollary of Theorem 2:

Corollary 1. *Fix n, t, ϵ, δ, and D. If $\gamma \geq \max\left\{\frac{14 \cdot |D| \log(2/\delta)}{(n-t-1) \cdot \epsilon^2}, \frac{27 \cdot |D|}{(n-t-1) \cdot \epsilon}\right\}$, then $(\mathcal{R}_{\gamma,D} \times \cdots \times \mathcal{R}_{\gamma,D})^{\mathcal{S}}$ is (ϵ, δ)-DP for t corrupted users in the $\mathcal{S}$-hybrid model.*

Shuffle protocols. A protocol Σ run by n users and a server is a *shuffle protocol* if it implements $\mathcal{S}$, i.e., if the output generated by the server when running Σ is the multiset consisting of the users' inputs. We are interested in shuffle protocols that ensure differential privacy when used to implement the shuffle model. Note, however, that we cannot use differential privacy to analyze a shuffle protocol; no shuffle protocol is differentially private, since two neighboring inputs $\mathbf{y}, \mathbf{y}'$ lead to different outputs. Instead, we use a related definition that we call *differential* *obliviousness*. Call vectors $\mathbf{y}, \mathbf{y}'$ *neighboring* if they differ by a transposition, i.e., there exist i, j such that $y'_i = y_j$, $y'_j = y_i$, and $y'_k = y_k$ for $k \notin \{i, j\}$ (so $\mathbf{y}'$ and $\mathbf{y}$ are identical except the elements at positions i, j are swapped). Then:

Definition 2. *Shuffle protocol Σ is (ϵ, δ)*-differentially oblivious for t corrupted users *if for any set A containing the server and up to t users, any $\mathbf{y}_A$, any neighboring $\mathbf{y}_H, \mathbf{y}'_H$, and any set V of possible (joint) views of the parties in A,*

$$\Pr[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V] \le e^{\epsilon} \cdot \Pr[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}'_H) \in V] + \delta.$$

3 Distributing the Privacy Blanket

Generalizing the result of Balle et al. [4], we show that a differentially oblivious shuffle protocol suffices for implementing the shuffle model. Specifically:

Theorem 3. *Let Σ be a shuffle protocol that is (ϵ, δ)-differentially oblivious for t corrupted users, and let $\mathcal{R}$ be an ϵ_0-LDP mechanism. For any δ' such that $\epsilon_0 \le \log((n-t)/\log(1/\delta'))/2$, protocol $(\mathcal{R}^{\otimes n})^{\Sigma}$ is $(\epsilon + \epsilon', \delta + \delta')$-differentially private for t corrupted users, where $\epsilon' = O(\max\{1, \epsilon_0\} \cdot e^{\epsilon_0} \sqrt{\log(1/\delta')/(n-t)})$.*

We defer the proof to Appendix A. Here, we focus on the important special case where $\mathcal{R}$ is the randomized response mechanism. Specifically, we show:

Theorem 4. *Let Σ be a shuffle protocol that is (ϵ, δ)-differentially oblivious for t corrupted users. If $(\mathcal{R}_{\gamma,D} \times \cdots \times \mathcal{R}_{\gamma,D})^{\mathcal{S}}$ is (ϵ', δ')-differentially private for t corrupted users, then $(\mathcal{R}_{\gamma,D} \times \cdots \times \mathcal{R}_{\gamma,D})^{\Sigma}$ is $(\epsilon+\epsilon', \delta+\delta')$-differentially private for t corrupted users.*

Overview of the proof of Theorem 4. Throughout this section, we let Π denote $\mathcal{R}_{\gamma,D} \times \cdots \times \mathcal{R}_{\gamma,D}$; our goal is to prove differential privacy of Π^{Σ}. We provide a formal proof starting in the next subsection; here, we provide an overview.

Fix some neighboring inputs $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_H)$ and $\mathbf{x}' = (\mathbf{x}_A, \mathbf{x}'_H)$, and some set of adversarial views V. (Each view in V includes the views of the server and t corrupted users in an execution of Π^{Σ}.) Conceptually, we separate each view $v \in V$ into three components: a component v_1 reflecting the adversary's view of the input to Σ (in particular, v_1 includes the randomized inputs $\mathbf{y}_A$ of the corrupted parties); the final multiset h output by the server (which has the same distribution as the multiset that would be output by the shuffler in $\Pi^{\mathcal{S}}$ conditioned on v_1); and the view v_2 that results from execution of Σ itself.

For some first component v_1 and output multiset h, let $Y(v_1, h)$ denote the set of (possibly modified) honest inputs $\mathbf{y}_H$ to Σ that are consistent with v_1, h, and $\mathbf{x}$, and let $Y'(v_1, h)$ denote the set of $\mathbf{y}_H$ consistent with v_1, h, and $\mathbf{x}'$. Using Corollary 1 and letting $m = n - t$, we show (cf. Lemma 1):

$$\sum_{(v_1,h)\,:\,(v_1,h,v_2)\in V} \Pr[v_1 \mid \mathbf{x}] \cdot \Pr\left[\mathcal{R}_{\gamma,D}^{\otimes m}(\mathbf{x}) \in Y(v_1,h) \mid v_1\right]$$
$$\le e^{\epsilon'} \cdot \sum_{(v_1,h)\,:\,(v_1,h,v_2)\in V} \Pr[v_1 \mid \mathbf{x}'] \cdot \Pr\left[\mathcal{R}_{\gamma,D}^{\otimes m}(\mathbf{x}') \in Y'(v_1,h) \mid v_1\right] + \delta'. \quad (1)$$

(Note that $\Pr[v_1 \mid \mathbf{x}'] = \Pr[v_1 \mid \mathbf{x}]$ since v_1 only depends on the true inputs of the corrupted parties.) For v_1, h as above, let $V_2(v_1, h) = \{v_2 \mid (v_1, h, v_2) \in V\}$. In what is the most technical part of the proof, we then use differential obliviousness of Σ to show (cf. Lemma 5) that for any v_1, h we have

$$\Pr_{\mathbf{y}_H \leftarrow Y(v_1,h)}[v_2 \in V_2(v_1, h)] \leq e^{\epsilon} \cdot \Pr_{\mathbf{y}'_H \leftarrow Y'(v_1,h)}[v_2 \in V_2(v_1, h)] + \delta. \tag{2}$$

The proof of the above follows from a combinatorial analysis of the two sets Y and Y'. Recall that an element in Y and an element in Y' are neighboring if they differ by a single transposition. Differential obliviousness of Σ guarantees that neighboring vectors give rise to (roughly) the same view. If we can establish a bijection between Y and Y', mapping each element of Y to a neighboring element in Y', Equation (2) would follow immediately. Unfortunately, Y and Y' do not necessarily have the same size, and so such a bijection may not exist. Nevertheless, we show how to extend Y and Y' to multisets $[Y]$ and $[Y']$ (by duplicating certain elements) having the same size, and so that the resulting multisets preserve the probabilities of each vector (so sampling uniform $\mathbf{y}_H \in Y$ gives the same distribution as sampling uniform $\mathbf{y}_H \in [Y]$, and similarly for Y' and $[Y']$). We then show that there is a bijection $\phi : [Y] \to [Y']$ such that $\mathbf{y}_H$ and $\phi(\mathbf{y}_H)$ are neighboring. This allows us to prove that Equation (2) holds.

Since

$$\begin{aligned}\Pr[(v_1, h, v_2) \in V \mid \mathbf{x}] &= \sum_{(v_1,h,v_2)\in V} \Pr[(v_1, h, v_2) \mid \mathbf{x}] \\ &= \sum_{(v_1,h)\,:\,(v_1,h,v_2)\in V} \Pr[v_1 \mid \mathbf{x}] \cdot \Pr[\mathcal{R}_{\gamma,D}^{\otimes m}(\mathbf{x}) \in Y(v_1, h) \mid v_1] \\ &\qquad\qquad \cdot \Pr_{\mathbf{y}_H \leftarrow Y(v_1,h)}[v_2 \in V_2(v_1, h)],\end{aligned}$$

combining Equations (1) and (2) allows us to prove Theorem 4.

3.1 Notation and Preliminaries

We now formalize the preceding intuition. We assume t users are corrupted and let $m = n - t$ be the number of uncorrupted users. Fix some neighboring inputs $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_H)$ and $\mathbf{x}' = (\mathbf{x}_A, \mathbf{x}'_H)$, and for $i \in [m]$ let $x_{H,i}$ be the input of the ith honest user. Without loss of generality, we assume $\mathbf{x}_H$ and $\mathbf{x}'_H$ differ on the input of the mth user, and further assume that $x_{H,m} = 1$ and $x'_{H,m} = 2$.

The adversary's view. We now make explicit the components of the adversary's view in an execution of Π^{Σ} on input $\mathbf{x}$. The first component of the view, which we denote by v_1, includes $\mathbf{y}_A = (\mathcal{R}_{\gamma,D} \times \cdots \times \mathcal{R}_{\gamma,D})(\mathbf{x}_A)$, i.e., the adversary's inputs to Σ. Following Balle et al. [4], we also include in v_1 the vector $\mathbf{b} = (b_1, \ldots, b_m)$ indicating which of the honest users' inputs are replaced by a random value, i.e., if $b_i = 0$ then $y_{H,i} = x_{H,i}$ and if $b_i = 1$ then $y_{H,i} \leftarrow D$. The second component of the adversary's view is the multiset $h = \mathcal{S}(\mathbf{y}_A, \mathbf{y}_H)$ output by Σ, in which $\mathbf{y} = (\mathbf{y}_A, \mathbf{y}_H)$ denotes the vector of inputs the parties provide to Σ; note that parts of $\mathbf{y}_H$ (corresponding to inputs that have not been randomized) can be deduced from v_1. The third component v_2 of the adversary's view consists of the entire view of the adversary in the execution of Σ on inputs $\mathbf{y}$. (Although v_2 determines h, we find it useful to treat h separately.)

For the rest of the proof, fix some set of views $V = \{(v_1, h, v_2)\}$. Note that views for which $b_m = 1$ are equiprobable regardless of whether the honest inputs are $\mathbf{x}_H$ or $\mathbf{x}'_H$; therefore, we assume without loss of generality that all views in V have $b_m = 0$. We let $V' = \{(v_1, h) \mid \exists v_2 : (v_1, h, v_2) \in V\}$ and, for any $(v_1, h) \in V'$, we let $V_2(v_1, h) = \{v_2 \mid (v_1, h, v_2) \in V\}$.

For some fixed v_1, h, let $Y(v_1, h)$ denote the set of honest inputs $\mathbf{y}_H$ consistent with v_1, h, and $\mathbf{x}$. That is, $Y(v_1, h)$ contains all $\mathbf{y}_H \in D^m$ such that (1) for all i with $b_i = 0$, we have $y_{H,i} = x_{H,i}$ (so, in particular, $y_{H,m} = x_{H,m} = 1$), and (2) $\mathcal{S}(\mathbf{y}_A, \mathbf{y}_H) = h$ (where $\mathbf{y}_A$ is fixed by v_1). Similarly, we let $Y'(v_1, h)$ denote the set of $\mathbf{y}_H$ consistent with v_1, h, and $\mathbf{x}'$.

3.2 Step 1: Using Local Differential Privacy of $\mathcal{R}_{\gamma,D}$

A proof of the following is straightforward:

Lemma 1. *If $\Pi^{\mathcal{S}}$ is (ϵ', δ')-DP for t corrupted users, then for any set of views V and any pair of neighboring inputs $\mathbf{x}, \mathbf{x}'$, we have:*

$$\begin{aligned} &\sum_{(v_1,h)\in V'} \Pr[v_1 \mid \mathbf{x}] \cdot \Pr\left[\mathcal{R}^{\otimes m}_{\gamma,D}(\mathbf{x}) \in Y(v_1,h) \mid v_1\right] \\ &\le e^{\epsilon'} \cdot \sum_{(v_1,h)\in V'} \Pr[v_1 \mid \mathbf{x}'] \cdot \Pr\left[\mathcal{R}^{\otimes m}_{\gamma,D}(\mathbf{x}') \in Y'(v_1,h) \mid v_1\right] + \delta'. \end{aligned}$$

We also state a useful corollary. Define

$$\begin{aligned} &\Delta(v_1,h) \overset{\text{def}}{=} \\ &\max\left\{\Pr[\mathcal{R}^{\otimes m}_{\gamma,D}(\mathbf{x}) \in Y(v_1,h) \mid v_1] - e^{\epsilon'} \cdot \Pr[\mathcal{R}^{\otimes m}_{\gamma,D}(\mathbf{x}') \in Y'(v_1,h) \mid v_1],\ 0\right\}. \end{aligned}$$

Using the fact that $\Pr[v_1 \mid \mathbf{x}] = \Pr[v_1 \mid \mathbf{x}']$, we then have:

Corollary 2. *If $\Pi^{\mathcal{S}}$ is (ϵ', δ')-DP for t corrupted users, then for any set of views V and any pair of neighboring inputs $\mathbf{x}, \mathbf{x}'$, it holds that:*

$$\sum_{(v_1,h)\in V'} \Pr[v_1 \mid \mathbf{x}] \cdot \Delta(v_1,h) \le \delta'.$$

3.3 Step 2: Using Differential Obliviousness of Σ

In this section we fix some $(v_1, h) \in V'$, and write Y, Y', and V_2 for $Y(v_1, h)$, $Y'(v_1, h)$, and $V_2(v_1, h)$, respectively. For simplicity, we assume both Y and Y'

are non-empty; the case where one or both are empty can be addressed by Lemma 1. Recall that if $\mathbf{y}_H \in Y$ then $y_{H,m} = 1$, and if $\mathbf{y}'_H \in Y'$ then $y'_{H,m} = 2$.

Let $\bar{h}$ denote the multiset that remains after removing from h the multiset given by the elements of $\mathbf{y}_A$ and the multiset $\{\mathbf{x}_{H,i} \mid b_i = 0, i \neq m\}$ (both of which are determined by v_1). Let c_1 be the number of 1's in $\bar{h}$, and let c_2 be the number of 2's in $\bar{h}$; note that $c_1, c_2 \neq 0$ since Y and Y' are non-empty. The following characterizes the relative sizes of Y and Y' in terms of c_1 and c_2:

Lemma 2. $\frac{|Y|}{|Y'|} = \frac{c_1}{c_2}$.

Proof. Let C be the number of ways of distributing all the elements of $\bar{h}$ that are not equal to 1 or 2 among the honest users who have changed their inputs. A vector $\mathbf{y}_H$ is consistent with v_1, h, and $\mathbf{x}$ only if a 1 is associated with the last user, and the remaining $c_1 + c_2 - 1$ elements of $\bar{h}$ that are 1 or 2 are distributed among the $c_1 + c_2 - 1$ users who remain from those who have changed their inputs. Thus,

$$|Y| = C \cdot \binom{c_1 + c_2 - 1}{c_1 - 1}.$$

Siilarly,

$$|Y'| = C \cdot \binom{c_1 + c_2 - 1}{c_2 - 1}.$$

The lemma follows. □

Lemma 3. *For every $\mathbf{y}_H \in Y$, there are c_2 vectors in Y' that result from transposing the final entry of $\mathbf{y}_H$ with some other entry of $\mathbf{y}_H$. Similarly, for every $\mathbf{y}'_H \in Y'$, there are c_1 vectors in Y that result from transposing the final entry of $\mathbf{y}'_H$ with some other entry of $\mathbf{y}'_H$.*

Proof. We prove the first statement; the second follows symmetrically. Fix some $\mathbf{y}_H \in Y$. The final entry of $\mathbf{y}_H$ is 1, and there are c_2 other entries of $\mathbf{y}_H$ that are equal to 2 and that correspond to users who have changed their inputs. Transposing the final entry of $\mathbf{y}_H$ with the entries at any of those locations gives a vector in Y'. □

Mapping between Y and Y'. Ideally, we would like to construct a bijection between Y and Y' such that a vector in Y is mapped to a vector in Y' iff they are transpositions of each other. Then for each pair of such vectors $\mathbf{y}_H$ and $\mathbf{y}'_H$, we could argue that $\text{VIEW}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H)$ and $\text{VIEW}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}'_H)$ must be "close" by differential obliviousness of Σ. Unfortunately, as shown in Lemma 2, the cardinalities of Y and Y' might be different, so such a bijection might not exist.

To resolve this issue, we "duplicate" vectors in Y and Y' so that the resulting multisets $[Y]$ and $[Y']$ have the same cardinality. Concretely, we let $[Y]$ be a multiset consisting of c_2 copies of each element $\mathbf{y}_H \in Y$. Similarly, we let $[Y']$ be a multiset consisting of c_1 copies of each element $\mathbf{y}'_H \in Y'$. Note that sampling uniformly from $[Y]$ (resp., $[Y']$) is equivalent to sampling uniformly from Y (resp., Y'). Moreover, by Lemma 2, $[Y]$ and $[Y']$ have the same size. We show:

Lemma 4. *There is a bijection $\phi : [Y] \to [Y']$ such that for every $\mathbf{y}_H \in [Y]$, the vector $\phi(\mathbf{y}_H) \in [Y']$ is a transposition of $\mathbf{y}_H$.*

Proof. Consider the bipartite graph G with vertex sets $[Y]$ and $[Y']$, where there is an edge between $\mathbf{y}_H \in [Y]$ and $\mathbf{y}'_H \in [Y]'$ iff $\mathbf{y}'_H$ results from transposing the final entry of $\mathbf{y}_H$ with some other entry of $\mathbf{y}_H$. Using Lemma 3 and the fact that every vector in Y' is included c_1 times in $[Y']$, we see that each $\mathbf{y}_H \in [Y]$ has exactly $c_1 \cdot c_2$ edges. Reasoning analogously, each $\mathbf{y}'_H \in [Y']$ has $c_1 \cdot c_2$ edges. Hall's marriage theorem implies that G has a complete matching, which is also a perfect matching since $[Y]$ and $[Y']$ have the same size. Any such matching constitutes a bijection ϕ as claimed by the lemma. □

Recall that the third component of the adversary's view, v_2, is equal to $\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H)$. We may now prove the main result of this section.

Lemma 5. *If Σ is (ϵ, δ)-differentially oblivious for t corrupted users:*

$$\Pr_{\mathbf{y}_H \leftarrow Y}[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2] \leq e^\epsilon \cdot \Pr_{\mathbf{y}'_H \leftarrow Y'}[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}'_H) \in V_2] + \delta.$$

Proof. Let $\phi : [Y] \to [Y']$ be a bijection as guaranteed by Lemma 4. Differential obliviousness of Σ implies that for any $\mathbf{y}_H \in [Y]$:

$$\Pr\left[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2\right] \leq e^\epsilon \cdot \Pr[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \phi(\mathbf{y}_H)) \in V_2] + \delta.$$

Recalling that $[Y]$ and $[Y']$ have the same size, we thus have

$$\begin{aligned}
\Pr_{\mathbf{y}_H \leftarrow Y}[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2] &= \Pr_{\mathbf{y}_H \leftarrow [Y]}\left[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2\right] \\
&= \sum_{\mathbf{y}_H \in [Y]} \frac{\Pr\left[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2\right]}{|[Y]|} \\
&\leq \sum_{\mathbf{y}_H \in [Y]} \frac{e^\epsilon \cdot \Pr[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \phi(\mathbf{y}_H)) \in V_2] + \delta}{|[Y]|} \\
&= \sum_{\mathbf{y}'_H \in [Y']} \frac{e^\epsilon \cdot \Pr[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}'_H) \in V_2] + \delta}{|[Y']|} \\
&= e^\epsilon \cdot \Pr_{\mathbf{y}'_H \leftarrow Y'}[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}'_H) \in V_2] + \delta,
\end{aligned}$$

as required. □

Combining Corollary 2 and Lemma 5 proves Theorem 4.

4 A Differentially Oblivious Shuffle Protocol

In this section, we describe a construction of a differentially oblivious shuffler for semi-honest adversaries. We present the protocol in Section 4.1 and analyze its obliviousness in Sections 4.2 and 4.3. In Section 5 we discuss how to adapt the protocol for malicious adversaries.

Inputs: Each user i has input y_i.

Round 1: Each user chooses $r-1$ users $i_1, \dots, i_{r-1} \leftarrow [n]$ uniformly and independently, and then forms the onion encryption C_r as described in the text. It sends C_r to user i_1.

Rounds $\ell = 2, \dots, r-1$**:** For each ciphertext $C_{r-\ell+2}$ received in the previous round, compute $(i_\ell, C_{r-\ell+1}) := \mathsf{Dec}_{\mathsf{sk}_{i_{\ell-1}}}(C_{r-\ell+2})$ and forward $C_{r-\ell+1}$ to user i_ℓ.

Round r**:** For each ciphertext C_2 received in the previous round, compute $(S, C_1) := \mathsf{Dec}_{\mathsf{sk}_{i_{r-1}}}(C_2)$ and forward C_1 to the server S.

Output: S initializes $h := \emptyset$. Then, for each ciphertext C received in the previous round, compute $y := \mathsf{Dec}_{\mathsf{sk}_S}(C)$ and add y to h.

Fig. 1. A differentially oblivious shuffling protocol, parameterized by r.

4.1 A Shuffling Protocol

Recall that in our setting we have n users holding inputs $y_1, \dots, y_n$, respectively, who would like a server (that we treat as distinct from the n users) to learn the multiset $h = \{y_i\}$. We assume the parties have public/private keys $(\mathsf{pk}_1, \mathsf{sk}_1), \dots, (\mathsf{pk}_n, \mathsf{sk}_n)$, respectively, and that the server has keys $(\mathsf{pk}_S, \mathsf{sk}_S)$. Our protocol, which is based on onion routing [21, 33], works as follows. Let r be a parameter that we fix later. Each user U chooses $r-1$ users $i_1, \dots, i_{r-1} \leftarrow [n]$ uniformly and independently (it may be that U chooses itself), and then forms a nested ("onion") encryption of the form

$$C_r = \mathsf{Enc}_{\mathsf{pk}_{i_1}}(i_2, \mathsf{Enc}_{\mathsf{pk}_{i_2}}(i_3, \cdots (i_{r-1}, \mathsf{Enc}_{\mathsf{pk}_{i_{r-1}}}(S, \mathsf{Enc}_{\mathsf{pk}_S}(y))) \cdots)),$$

such that at each "layer" the identity of the next receiver is encrypted with an onion encryption whose outer layer can be removed by that receiver. In the first round, U sends C_r to the first receiver i_1, who decrypts to remove the outer layer and thus obtains i_2 and an onion encryption C_{r-1} that it forwards to i_2 in the next round. This process continues for $r-1$ rounds, until in the rth round all parties send the ciphertext $\mathsf{Enc}_{\mathsf{pk}_S}(y)$ they obtain to the server. See Figure 1.

The protocol requires r rounds of communication, and the total number of ciphertexts transmitted is exactly rn. Since ciphertexts have length $O(r \log n)$, the total communication complexity is $O(r^2 n \log n)$.

4.2 Analysis of Obliviousness ($\epsilon = 0$)

We assume a semi-honest adversary who corrupts up to t users as well as the server S. The attacker has access to the state of any corrupted user, and can also determine which user sent any message that it receives. However, we assume the attacker *cannot* eavesdrop on the communication between honest users, so in particular it cannot tell whether some honest user i sent a message to some other honest user j in some round. We treat encryption as ideal in our analysis of obliviousness in order to simplify our treatment.

Assume without loss of generality that users U_1, U_2 are honest and hold different inputs, and fix input vectors $\mathbf{y}$ and $\mathbf{y}'$ that are identical except the inputs of U_1 and U_2 are swapped. Let i^1_ℓ denote the ℓth intermediate user chosen by U_1 for $1 \leq \ell \leq r-1$, and set $i^1_0 = 1$; define $i^2_0, \ldots, i^2_{r-1}$ similarly. (We let round 0 refer to the beginning of the algorithm when U_1 and U_2 each hold their own input.) Say that *U_1 and U_2 can swap at round j* (with $0 \leq j < r-1$) if the routing paths of U_1 and U_2 both have an honest user in rounds j and $j+1$ (i.e., for which users i^1_j, i^1_{j+1}, i^2_j, and i^2_{j+1} are all honest). A key observation is that if there exists some j such that U_1 and U_2 can swap at round j then the distributions on the attacker's views are *identical* regardless of whether the input vector is $\mathbf{y}$ or $\mathbf{y}'$. The reason for this is that it is equally likely that the onion encryption of U_1 was routed from i^1_j to i^1_{j+1} and that of U_2 went from i^2_j to i^2_{j+1}, or that the communication was "swapped" (in which case we say *the swap happened*) so that the onion encryption of U_1 was routed from i^1_j to i^2_{j+1} and that of U_2 went from i^2_j to i^1_{j+1}. In other words, if there exists some j such that U_1 and U_2 can swap at round j, then perfect obliviousness is achieved. If we let $x_{t,r}$ denote the probability of this event in an execution of the protocol with parameter r when up to t users are corrupted, we have:

Theorem 5. *The protocol in Figure 1 is $(0, 1-x_{t,r})$-differentially oblivious for t corrupted users.*

Our problem is now reduced to lower bounding $x_{t,r}$. Let $p_t = (1-t/n)^2$ be the probability that U_1 and U_2 both choose an honest user in some fixed round $j \geq 1$ when t users are corrupted. By definition, we have $x_{t,1} = 0$, and $x_{t,2} = p_t$ since both U_1 and U_2 are honest in round 0. By conditioning on the outcomes of the final two rounds, we can derive the following recurrence relation for $r > 2$:

$$x_{t,r} = p_t^2 + (1-p_t) \cdot x_{t,r-1} + p_t \cdot (1-p_t) \cdot x_{t,r-2}.$$

Although it is possible to solve this recurrence, it is cleaner to simply bound $x_{t,r}$ for any desired t, r. The following can be proved by induction on r:

Theorem 6. *For $r > 1$, it holds that $x_{n/3,r} \geq 1 - 0.85^r$. Thus, for $r > 1$ the protocol of Figure 1 is $(0,\ 0.85^r)$-differentially oblivious for $n/3$ corrupted users.*

For $r > 1$, it holds that $x_{n/2,r} \geq 1 - 0.95^r$. Thus, for $r > 1$ the protocol of Figure 1 is $(0,\ 0.95^r)$-differentially oblivious for $n/2$ corrupted users.

4.3 Analysis of Obliviousness ($\epsilon > 0$)

We show here an alternate analysis that allows us to prove (ϵ, δ)-differential obliviousness for $\epsilon > 0$. (This analysis is incomparable to the analysis of the previous section since, for fixed r, we may obtain larger ϵ but smaller δ.)

We focus again on the case where we have input vectors $\mathbf{y}$ and $\mathbf{y}'$ that are identical except that the inputs of honest users U_1 and U_2 are swapped. The observation we rely on here is that even if there is no round j where U_1 and U_2 can swap at round j, it is still possible to achieve some privacy if their inputs can be swapped via some other honest users. For example, say there is an honest user U_3 and $0 \leq j < j' < j'' < r-1$ such that (1) U_1 and U_3 can swap at round j, (2) U_2 and U_3 can swap at round j', and (3) U_1 and U_3 can swap at round j''. Then the following events lead to the same view for the adversary: the input vector was $\mathbf{y}$ and none of the swaps happens; the input vector was $\mathbf{y}$ and (only) swaps #1 and #3 happen; or the input vector was $\mathbf{y}'$ and all three swaps happen. This gives some privacy (given a view consistent with these events, the adversary cannot determine with certainty whether the input was $\mathbf{y}$ or $\mathbf{y}'$), but the privacy is not perfect: since each swap is equally likely to happen or not, conditioned on the adversary's view being consistent with the above input $\mathbf{y}$ is twice as likely as input $\mathbf{y}'$. In this particular example the level of privacy obtained is relatively low, but privacy improves as more honest users can potentially be involved in the swaps.

In the full version we give a more detailed analysis of the ϵ, δ parameters obtained by considering swaps between multiple honest users; here we simply describe the qualitative conclusions of the analysis. Say U_1 and U_2 are *swap-compatible* if there are $0 \leq j < j' < j'' < r-1$ such that (1) the routing path of U_1 has an honest user in rounds j and $j+1$ as well as rounds j'' and $j''+1$, and (2) the routing path of U_2 has an honest user in rounds j' and $j'+1$ (or the similar event with the roles of U_1 and U_2 interchanged). If U_1, U_2 are swap-compatible then U_1 can potentially swap with some other honest users at round j, other honest users can potentially swap with U_2 at round j', and then U_1 can again potentially swap with other honest users at round j''. For that to occur requires other honest users who can potentially swap with U_1, U_2 at the appropriate rounds; roughly speaking, the more honest users can swap with U_1, U_2, the higher privacy will be achieved for U_1, U_2 .

Let δ_1 denote the probability that U_1, U_2 are not swap-compatible. Next, fix some desired value for $\epsilon > 0$. When U_1, U_2 are swap-compatible, we can derive a lower bound m on the number of other honest users that need to be able to swap with U_1, U_2 (we do not define this event more formally here) to ensure privacy bound ϵ. Letting δ_2 be the probability that there are fewer than m other honest users who can swap with U_1, U_2, we can then conclude that our protocol achieves $(\epsilon, \delta_1 + \delta_2)$-differential obliviousness. Note that δ_1 depends only on the corruption threshold and the number of rounds r, and decreases exponentially with r as in the $\epsilon = 0$ case. On the other hand, δ_2 also depends on the total number of parties n as well as the privacy parameter ϵ (since decreasing ϵ requires increasing m, which in turn increases the probability δ_2 of failing to have m other honest users who can swap with U_1, U_2).

5 Malicious Security

Here we describe how to adapt our protocol to address malicious attacks affecting privacy; denial-of-service attacks and other attacks that only affect correctness are out of scope. If the encryption scheme used in the protocol is non-malleable, and timestamps are included in each layer of the onion to prevent replay attacks (cf. [11]), then the only attack an adversary can carry out on the protocol is to drop messages to reduce the effective number of honest users contributing to the output histogram and thereby degrade privacy (cf. Corollary 1).

As in the work of Ando et al. [1], we can address such an attack by having honest users (1) route dummy messages alongside their real messages, (2) check partway through the shuffling that their dummy messages have not been dropped, and (3) abort the protocol if malicious behavior is detected. Compared to the work of Ando et al., however, we can achieve security against malicious behavior with much lower overhead, both because we assume the adversary cannot eavesdrop on communication between honest users and also because we focus on the eventual application of our protocol to differential privacy in the shuffle model. With regard to the latter point, note that although dropping even a single user's message can be catastrophic in the context of anonymous communication (e.g., if the adversary knows that either user 1 or user 2 is sending some message m, and drops the message sent by user 1, then the set of output messages reveals which of those two users was sending that message), dropping a few users' inputs has only a small effect on end-to-end differential privacy when the shuffle protocol is used to instantiate the shuffle model, as we explain further below.

Concretely, let $\hat{\mathcal{S}}_d$ represent an ideal shuffler that is identical to $\mathcal{S}$ except that the adversary can select d honest users whose messages are dropped. The following is a natural extension of Corollary 1:

Lemma 6. *Fix $n, t, d, \epsilon, \delta$, and D. If $\gamma \geq \max\left\{\frac{14\cdot|D|\log(2/\delta)}{(n-d-t-1)\cdot\epsilon^2}, \frac{27\cdot|D|}{(n-d-t-1)\cdot\epsilon}\right\}$, then $(\mathcal{R}_{\gamma,D} \times \cdots \times \mathcal{R}_{\gamma,D})^{\hat{\mathcal{S}}_d}$ is (ϵ, δ)-DP for t corrupted users in the $\hat{\mathcal{S}}_d$-hybrid model.*

Our goal is thus to design a protocol approximating the effect of $\hat{\mathcal{S}}_d$ (for some small d). Let r, s be two parameters. Our modified protocol has four stages:

1. Each user U_i runs the onion-routing protocol from Section 4 twice, in parallel, using parameter $r + s + 1$. In one execution it uses its real input y_i and in the other it uses a default dummy value $\perp$. We let R_i—called the "checker" for U_i—denote the rth user chosen by U_i for its dummy onion encryption.
2. At round r, each user U_i asks its checker R_i to respond with (sufficiently many bits of) the ciphertext corresponding to U_i's dummy onion encryption. U_i sets $\mathsf{cheat}_i := 0$ if R_i responds correctly, and sets $\mathsf{cheat}_i := 1$ otherwise.
3. The users run a protocol to determine whether any user set $\mathsf{cheat} = 1$. (We discuss below how this can be implemented efficiently.) If so, they all abort and do not run the next stage.
4. The users run the onion-routing protocol on the remaining ciphertexts for the remaining $s + 1$ rounds.

Informally, the argument for why this suffices to instantiate the shuffle model in the context of differential privacy is as follows. For concreteness, assume the attacker knows all users' inputs except that of user U_1. Note first that dropping U_1's onion encryptions does not help the attacker at all. As for the onion encryptions of other honest users, prior to round r the adversary cannot distinguish the real onion encryption of some user from the dummy onion encryption of that same user as long as the paths of both onion encryptions each have at least one honest user in their final s hops. By setting parameters appropriately, we can ensure that if the adversary drops too many of the honest users' onion encryptions before round r, then with high probability at least one of those will correspond to a dummy message; in that case, cheating will be detected and all honest users will abort. When that occurs, the adversary does not learn anything about the (real) input of U_1 unless the final s intermediate users chosen by U_1 for the onion routing of its real message are all corrupted; the probability that this occurs is at most $(t/n)^s$. We can thus claim privacy at round r, with the number of honest messages being at least $n-t-d$, just as we did in Section 4. The adversary can drop as many messages as it likes after round r, but doing so cannot degrade the privacy already achieved by round r.

We now make the above more precise. Fix d_1, d_2 with $d_1 \geq (n-t) \cdot (t/n)^s$. From now on we only consider onion encryptions generated by honest users. We consider different types of onion encryptions that the attacker could potentially drop by round r:

- The adversary might drop an onion encryption at some round $k \leq r$, when the intermediate hops $k, \ldots, r$ used by that onion encryption are all corrupted. (Note that in this case, even if the onion encryption turns out to be a dummy onion encryption, the [corrupted] checker will still be able to respond correctly and so such cheating will not be caught.) Dropping such an onion encryption, however, does not degrade privacy beyond what is achieved by the r-round semi-honest protocol.
- The adversary might drop an onion encryption of a user U when the final s intermediate hops of U's dummy onion encryption are all corrupted. (In such a case it is possible that the attacker also does not risk being caught, assuming it knows which two onion encryptions are associated with U, and can definitively determine which is the dummy onion encryption.) Per Lemma 6, this may reduce privacy for U_1. However, the expected number of dummy onion encryptions whose final s hops are all corrupted is $\mu = (n-t) \cdot (t/n)^s$, and (using Hoeffding's inequality) one can show that except with probability $e^{-2 \cdot (d_1 - \mu)^2/(n-t)}$ the number of such onion encryptions is at most d_1.
- The remaining case is that in some round $k < r$ the attacker drops an onion encryption associated with a user U, for which not all of the intermediate hops $k, \ldots, r$ are corrupted, and the final s intermediate hops of U's dummy encryption are not all corrupted. In that case the attacker (1) cannot distinguish U's dummy and real onion encryptions, and (2) will be caught if the dropped onion encryption turns out to be a dummy onion encryption. So the probability that the attacker can drop more than d_2 (real) onion encryptions of this type without being caught is $2^{-(d_2+1)}$.

In summary, except with probability at most $\delta_d = e^{-2 \cdot (d_1 - \mu)^2/(n-t)} + 2^{-(d_2+1)}$, the attacker drops at most d message or is caught. If the attacker drops at most d messages, then privacy is given by the analysis from Section 4.3 with the number

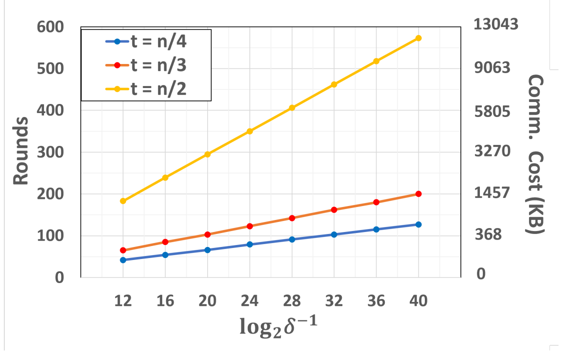

Fig. 2. Round complexity and per-user communication complexity for achieving $\epsilon = 0$ and different δ for various corruption thresholds.

of honest users reduced to $n - t - d$. If the attacker is caught, then privacy fails only if the final s hops of U_1's real onion encryption are all corrupted, which occurs with probability at most $\delta_s = (t/n)^s$.

Efficient implementation of stage 3. In stage 3 we need a distributed protocol with the property that if any honest user holds $\mathsf{cheat} = 1$ then all honest users output 1. While this can be achieved using n executions of secure broadcast, doing so would be inefficient, and is overkill; in particular, it is acceptable for us if the adversary causes some honest users to output 1 even when no honest users hold $\mathsf{cheat} = 1$. Instead, we can use the following lightweight protocol based on any multisignature scheme. Every user who holds $\mathsf{cheat} = 0$ sends a signature on a designated message ok to the server. The server then combines these signatures into a single, constant-size signature, and sends it to every user. Each user locally verifies the signature it receives from the server with respect to every users' public key, and outputs 1 if verification fails (or if it does not receive any signature from the server). Note that even if all-but-one of the users are corrupted, an adversary cannot forge a valid multisignature on ok unless every honest user holds $\mathsf{cheat} = 0$.

6 Performance Analysis

To analyze the performance of our malicious protocol and compare it with prior work in the malicious setting, we assume encryption is done using the KEM-DEM paradigm, with the KEM portion having a length of 256 bits. We allocate 20 bits for user identities, which suffices for up to $n = 2^{20}$ users,[3] and we assume users' inputs are 128 bits long. The innermost ciphertext thus requires $256 + 128 = 384$ bits, and in each of the other layers we add 256 bits for the next key encapsulation, 20 bits for the user ID, and 20 bits counter to prevent replay attacks. An ℓ-layer onion encryption thus requires $384 + 296 \cdot (\ell - 1)$ bits.

[3] In fact, these identifiers are the only part of our construction that contribute to the $O(\log n)$ multiplicative factor in the overhead.

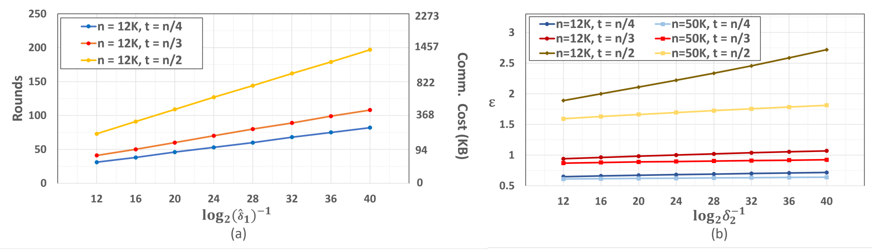

Fig. 3. (a) Round complexity and per-user communication complexity for achieving different $\hat{\delta}_1$ for various corruption thresholds. (b) Plot of ϵ vs. δ_2 for various corruption thresholds and different n.

The $\epsilon = 0$ case. In Figure 2, we fix the number of users $n = 12,000$ and plot the number of rounds and per-user communication complexity for achieving $(0, \delta)$-differential obliviousness for several values of δ and various corruption thresholds. For all our results, we set $d = 50$ so that (except with some small probability that we add to the δ term) the attacker can drop at most 50 messages without being caught. For simplicity, the communication complexity reflects only the two onion encryptions sent by each user, and we do not count the communication in stages 2 and 3 which is anyway dominated by the onion routing.

The $\epsilon > 0$ case. In Figure 3, we show how $\delta = \hat{\delta}_1 + \delta_2$ relates to n, r, s, t, d, and ϵ, where $\hat{\delta}_1 = \delta_1 + \delta_d + \delta_s$. (Terms δ_1 and δ_2 come from the analysis of the semi-honest protocol in Section 4.3, and δ_d, δ_s come from the analysis of malicious behavior in the previous section.) Specifically, in Figure 3(a) we show how the round/communication complexity depends on $\hat{\delta}_1$, and in Figure 3(b) we show how ϵ varies with δ_2.

We can use these figures to determine how to set parameters. For example, say we have $n = 12,000$ users and up to $t = n/3$ corruptions, and want to determine the δ achievable for $\epsilon = 1$. From Figure 3(b) we see that $\delta_2 \approx 2^{-23}$. Using Figure 3(a), we see that 68 rounds suffice for $\hat{\delta}_1 \approx 2^{-23}$. This corresponds to per-user communication of 171 KB.

Comparison to prior work. We compare to the solutions of Movahedi et al. [31] and Bell et al. [6], both of which achieve $\epsilon = 0$. Our results compare favorably to the work of Movahedi et al., especially when the number of parties is large. In particular, for a corruption threshold of $t \approx n/3$ their protocol uses 500 rounds and per-user communication of 128 MB when $n = 33,000$, and per-user communication of approximately 0.5–1 GB over 1,000 rounds when $n = 10^6$. For the same corruption threshold, $\epsilon = 0$, and[4] $\delta \leq 2^{-13}$, our protocol requires 70–103 rounds with per-user communication of 182–390 KB for any $n \leq 2^{20}$. Additionally, if we set $\epsilon = 1$, our protocol requires 44–60 rounds with per-user communication of 73–134 KB for $n \leq 2^{20}$.

[4] We choose $\delta \leq 2^{-13} \approx 10^{-4}$ to match what is often done in the differential privacy literature. Mohavedi et al. claim $\delta = O(1/n^3)$, but the constant term is unclear. Note also that they cannot set δ independently of n.

Our solution outperforms Bell et al. in the large-domain setting (i.e., when $|D| = \Omega(n)$). This is due to the fact that our per-user communication grows logarithmically in both the size of the domain and n, while theirs grows either linearly in the domain size and logarithmically in n, or super-linearly in n.

References

1. Megumi Ando, Anna Lysyanskaya, and Eli Upfal. Practical and provably secure onion routing. In *ICALP 2018*, volume 107 of *LIPIcs*, pages 144:1–144:14. Schloss Dagstuhl, July 2018.
2. Michael Backes, Ian Goldberg, Aniket Kate, and Esfandiar Mohammadi. Provably secure and practical onion routing. In *25th IEEE Computer Security Foundations Symposium (CSF)*, pages 369–385, 2012.
3. Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A framework for analyzing anonymous communication protocols. In *26th IEEE Computer Security Foundations Symposium (CSF)*, pages 163–178, 2013.
4. Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *Crypto 2019*, pages 638–667. Springer, 2019.
5. Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Crypto 2008*, pages 451–468. Springer, 2008.
6. James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly)logarithmic overhead. In *ACM CCS*, page 1253–1269. ACM Press, 2020.
7. Aurélien Bellet, Rachid Guerraoui, and Hadrien Hendrikx. Who started this rumor? Quantifying the natural differential privacy guarantees of gossip protocols. In *34th International Symposium on Distributed Computing (DISC)*, volume 179 of *LIPIcs*, pages 8:1–8:18, 2020.
8. Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Usharsee Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proc. 26th Symposium on Operating Systems Principles (SOSP)*, pages 441–459, 2017.
9. Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication locality in secure multi-party computation - how to run sublinear algorithms in a distributed setting. In *TCC 2013*, volume 7785 of *LNCS*, pages 356–376. Springer, 2013.
10. Benedikt Bünz, Yuncong Hu, Shinichiro Matsuo, and Elaine Shi. Non-interactive differentially anonymous router, 2021. Available at `https://eprint.iacr.org/2021/1242`.
11. Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In *Crypto 2005*, volume 3621 of *LNCS*, pages 169–187. Springer, 2005.
12. T.-H. Hubert Chan, Kai-Min Chung, Bruce M. Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. In *30th SODA*, pages 2448–2467. ACM-SIAM, 2019.
13. T-H. Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *European Symposium on Algorithms (ESA)*, pages 277–288. Springer, 2012.
14. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Comm. of the ACM*, 24(2):84–88, 1981.

15. D. Chaum. The dining cryptographers problem: unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.
16. Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Eurocrypt 2019, Part I*, volume 11476 of *LNCS*, pages 375–403. Springer, 2019.
17. Varsha Dani, Valerie King, Mahnush Movahedi, and Jared Saia. Brief announcement: breaking the $O(nm)$ bit barrier, secure multiparty computation with a static adversary. In *31st ACM PODC*, pages 227–228. ACM, 2012.
18. Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In *2018 IEEE Symposium on Security and Privacy*, pages 108–126. IEEE Computer Society Press, 2018.
19. Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC 2006*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.
20. Joan Feigenbaum, Aaron Johnson, and Paul F. Syverson. Probabilistic analysis of onion routing in a black-box model. *ACM Trans. Information and Systems Security*, 15(3):14:1–14:28, 2012.
21. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In *Proc. 1st International Workshop on Information Hiding*, volume 1174 of *LNCS*, pages 137–150. Springer, 1996.
22. Adam Groce, Peter Rindal, and Mike Rosulek. Cheaper private set intersection via differentially private leakage. *Proc. Privacy Enhancing Technologies (PETS)*, 2019(3):6–25, 2019.
23. Xi He, Ashwin Machanavajjhala, Cheryl J. Flynn, and Divesh Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *ACM CCS 2017*, pages 1389–1406. ACM Press, 2017.
24. Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, February 2012.
25. Christiane Kuhn, Martin Beck, Stefan Schiffner, Eduard A. Jorswieck, and Thorsten Strufe. On privacy notions in anonymous communication, 2018. Available at `https://arxiv.org/abs/1812.05638`.
26. Christiane Kuhn, Martin Beck, and Thorsten Strufe. Breaking and (partially) fixing provably secure onion routing. In *2020 IEEE Symposium on Security and Privacy*, pages 168–185. IEEE Computer Society Press, 2020.
27. David Lazar, Yossi Gilad, and Nickolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *Proc. 13th USENIX Conference on Operating Systems Design and Implementation*, page 711–725. USENIX Association, 2018.
28. S. Mauw, J. H. S. Verschuren, and E. P. de Vink. A formalization of anonymity and onion routing. In *ESORICS 2004*, pages 109–124. Springer, 2004.
29. Sahar Mazloom and S. Dov Gordon. Secure computation with differentially private access patterns. In *ACM CCS 2018*, pages 490–507. ACM Press, 2018.
30. Sahar Mazloom, Phi Hung Le, Samuel Ranellucci, and S. Dov Gordon. Secure parallel computation on national scale volumes of data. In *USENIX Security 2020*, pages 2487–2504. USENIX Association, 2020.
31. Mahnush Movahedi, Jared Saia, and Mahdi Zamani. Secure multi-party shuffling. In *Structural Information and Communication Complexity*, pages 459–473. Springer International Publishing, 2015.

32. Nigel P. Smart and Younes Talibi Alaoui. Distributing any elliptic curve based protocol. In *Proc. 17th IMA International Conference on Cryptography and Coding (IMACC)*, volume 11929 of *LNCS*, pages 342–366. Springer, 2019.
33. Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *1997 IEEE Symposium on Security and Privacy*, pages 44–54. IEEE Computer Society Press, 1997.
34. Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proc. 26th Symposium on Operating Systems Principles (SOSP)*, page 423–440. ACM Press, 2017.
35. Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proc. 25th Symposium on Operating Systems Principles (SOSP)*, page 137–152. ACM Press, 2015.
36. Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *SODA 2019*, pages 2468–2479, 2019.

A Proof of Theorem 3

We start by introducing the generalized privacy blanket method for analyzing arbitrary ϵ_0-local differentially private mechanism in the original paper [4].

Privacy blanket decomposition. Let $\mathcal{R} : D \to R$ be a local randomizer, where R denotes a continuous space. For input $x \in D$, let μ_x denote the distribution of $\mathcal{R}(x)$; we also abuse the notation a bit and use $\mu_x(\cdot)$ to denote its probability density function. For the collection of all output distributions $\{\mu_x\}_{x\in D}$, we define their total variation similarity as:

$$\gamma_{\mathcal{R}} = \int_{-\infty}^{\infty} \inf_x \{\mu_x(y)\} dy;$$

let $\omega_{\mathcal{R}}$ be the blanket distribution with its probability density function $\omega_{\mathcal{R}}(y) = \inf_x \mu_x(y)/\gamma_{\mathcal{R}}$ for $y \in R$. In the rest of this section, we simply write them as γ and ω. Finally, define $\nu_x = (\mu_x - \gamma\omega)/(1-\gamma)$ as an input-dependent distribution. For every randomizer $\mathcal{R}$ and its collection of output distributions $\{\mu_x\}_{x\in D}$, we can decompose each output distribution μ_x into an input-dependent part and an input-independent part:

$$\mu_x = (1-\gamma)\nu_x + \gamma\omega$$

One can understand it as any party with some input x samples from an input-independent distribution ω with probability γ, and from an input-dependent distribution ν_x with probability $1-\gamma$. We first claim the following lemma:

Lemma 7. *If $\mathcal{R}$ is ϵ_0-local differentially private for some $\epsilon_0 \geq 0$, then the blanket distribution ω and all output distributions in $\{\mu_x\}_{x\in D}$ share full support of domain R.*

Proof. We first show that for any $x, x' \in D$, μ_x and $\mu_{x'}$ must have the same support. Otherwise without loss of generality, assume $\mu_x(y) \neq 0$ and $\mu_{x'}(y) = 0$;

then we have $\mu_x(y) > e^{\epsilon_0}\mu_{x'}(y)$ for any $\epsilon_0 \geq 0$, thus $\mathcal{R}$ cannot be ϵ_0-LDP. From the definition of ω, it follows that ω must share this same support with all distributions in $\{\mu_x\}_{x \in D}$.

Similar to the proof of Theorem 1, we assume a stronger adversary that can identify the contribution of any user among the first $m-1$ users that does not sample from the blanket distribution. This is also the assumption made in Section 3; however, notice that for the generalized LDP mechanism, as opposed to the randomized response mechanism, a party not sampling from blanket distribution ω could still enjoy some randomness by sampling from its input-dependent distribution ν_x. Thus, we need to explicitly provide these "partially" randomized values to the adversary and reflect this in our notation. Concretely, we modify the adversary's view v_1 defined earlier by including an additional vector $\hat{\mathbf{y}}_H = (\hat{y}_1, \ldots, \hat{y}_{m-1})$, where

$$\hat{y}_i = \begin{cases} y_{H,i} & \text{if } b_i = 0 \\ \perp & \text{otherwise} \end{cases}$$

The high-level intuition and structure of the proof of Theorem 3 are similar to those of Theorem 4. However, we need to re-introduce a generalized way to form a bijection.

We start by fixing v_1, h for which $Y(v_1, h)$ and $Y'(v_1, h)$ are both non-empty. For simplicity, we write Y for $Y(v_1, h)$ and Y' for $Y'(v_1, h)$. Also recall that $\bar{h}$ denotes the resulting multiset after removing from h the multiset given by the elements of $\mathbf{y}_A$ and the multiset $\{\hat{y}_i \mid b_i = 0\}$ (both of which are determined by v_1). To align our assumption with the assumption made in [4], we also loosen the earlier restriction that the mth honest party always submits its true input. Instead, this party, which either holds input $x_{H,m}$ or $x'_{H,m}$ in the two neighboring cases, samples from $\mu_{x_{H,m}}$ or $\mu_{x'_{H,m}}$, respectively. Thus, both $y_{H,m}$ and $y'_{H,m}$ can correspond to any element in $\bar{h}$ based on Lemma 7. And it immediately follows that $Y = Y'$ and both sets include all possible permutations of elements in $\bar{h}$. We keep the redundant notations Y and Y' throughout our proof (and later do the same for $[Y]$ and $[Y']$), as it allows us to draw analogy to our previous analysis given in Section 3.

Similar to the proof in Section 3.3, we apply a vector duplicating approach to generate $[Y]$ and $[Y']$ while claiming a specific bijection ϕ between $[Y]$ and $[Y']$. Roughly speaking, for ever pair of mapped vectors $\mathbf{y}_H$ and $\phi(\mathbf{y}_H)$, their probability densities have the same fraction with their respective sets $[Y]$ and $[Y']$'s probability densities. (This fraction may vary for different pairs of mapped vectors in this bijection.)

For simplicity, we start by assuming there are no duplicate values in $\bar{h}$, and later show how to address the case with duplicate values. Concretely, let $\bar{h} = \{a_i\}_{i=1}^{l}$ where all a_i are distinct. We abuse the notation ω and let $\omega(\bar{h})$ denote the probability density $\prod_{j=1}^{l} \omega(a_j)$.

We partition Y into subsets $Y_1, \ldots, Y_l$ with $Y_i = \{\mathbf{y}_H \in Y \mid y_{H,m} = a_i\}$. Notice that for all i, $|Y_i| = (l-1)!$. Furthermore, all vectors within each set Y_i

have the same probability density. In particular, for every vector $\mathbf{y}_H \in Y_i$, its probability density (conditioned on v_1) is given as:

$$f(\mathbf{y}_H \mid v_1) = \frac{\mu_{x_{H,m}}(a_i)}{\omega(a_i)} \cdot \omega(\bar{h})$$

Hence,

$$f(Y_i \mid v_1) = \sum_{\mathbf{y}_H \in Y_i} f(\mathbf{y}_H \mid v_1) = (l-1)! \cdot \frac{\mu_{x_{H,m}}(a_i)}{\omega(a_i)} \cdot \omega(\bar{h})$$

Likewise, we can partition Y' into subsets $Y'_1, \ldots, Y'_l$ in the same way (recall that they are identical sets). Similarly, for every vector $\mathbf{y}'_H \in Y'$, its probability density conditioned on v_1 is:

$$f'(\mathbf{y}'_H \mid v_1) = \frac{\mu_{x'_{H,m}}(a_i)}{\omega(a_i)} \cdot \omega(\bar{h})$$

Hence,

$$f'(Y'_i \mid v_1) = \sum_{\mathbf{y}'_H \in Y'_i} f'(\mathbf{y}_H \mid v_1) = (l-1)! \cdot \frac{\mu_{x'_{H,m}}(a_i)}{\omega(a_i)} \cdot \omega(\bar{h})$$

Relationship between vectors in Y and Y'. We start by examining the transposition relationship between vectors in $Y_1, \ldots, Y_l$ and vectors in $Y'_1, \ldots, Y'_l$. For all Y_i, every vector $\mathbf{y}_H \in Y_i$ has an identical vector in Y'_i, and for all $j \neq i$, y_H has exactly one vector with transposition distance 1 in each of the Y'_j. Collectively, we refer to these l vectors as $\mathbf{y}_H$'s "connected" vector and denote them as a set $C(\mathbf{y}_H)$. Likewise, we denote $\mathbf{y}'_H$'s "connected" vector as $C(\mathbf{y}'_H)$.

Similar to what we did in Section 3, we "duplicate" vectors in Y and Y' to form multisets $[Y]$ and $[Y']$. Concretely, we let $[Y]$ be a multiset consisting of l copies of each element $\mathbf{y}_H \in Y$ and $[Y']$ be a multiset consisting of l copies of each element $\mathbf{y}'_H \in Y'$. Given some $y_H \in Y_i$ and its connected vector $y'_H \in Y'_j$, we map y_H's jth duplicate $\mathbf{y}_H^{(j)}$ to y'_H's ith duplicate $\mathbf{y}_H'^{(i)}$ and we use $\phi(\mathbf{y}_H^{(j)}) = \mathbf{y}_H'^{(i)}$ to denote such mappings.

Lemma 8. *The mapping $\phi : [Y] \to [Y']$ is a bijection such that for every $\mathbf{y}_H \in [Y]$, the vector $\phi(\mathbf{y}_H) \in [Y']$ is either a transposition of $\mathbf{y}_H$, or identical to $\mathbf{y}_H$.*

Proof. The second part of statement is trivial as we only map a vector $\mathbf{y}_H$'s duplicate to the duplicates of vectors in $C(\mathbf{y}_H)$ and vice versa. For the first part, notice that $|[Y]| = |[Y']|$, as $|Y| = |Y'|$ and both $[Y]$ and $[Y']$ contain l duplicates for each vector. According to our description of ϕ, each $\mathbf{y}_H \in [Y]$ is mapped to exactly one vector $\mathbf{y}'_H \in [Y']$. Due to symmetry, each $\mathbf{y}'_H \in [Y']$ is mapped exactly once. Hence, ϕ is a bijection between $[Y]$ and $[Y']$.

Assigning probability density for duplicates. For every $\mathbf{y}_H \in Y$, rather than evenly distributing the probability density to each of its duplicates $\mathbf{y}_H^{(1)}, \ldots, \mathbf{y}_H^{(l)}$

(as done in the proof of Theorem 3.1), we assign the probability density proportionally to the probability density of its connected vectors $C(\mathbf{y}_H)$. Concretely, we have:

$$\begin{aligned}
&f(\mathbf{y}_H^{(i)} \mid v_1)\\
=&\frac{\mu_{\mathbf{x}'_{H,m}}(a_i)/\omega(a_i)}{\sum_{j=1}^{l}\mu_{\mathbf{x}'_{H,m}}(a_j)/\omega(a_j)} \cdot f(\mathbf{y}_H \mid v_1)\\
=&\frac{\mu_{\mathbf{x}'_{H,m}}(a_i)/\omega(a_i)}{\sum_{j=1}^{l}\mu_{\mathbf{x}'_{H,m}}(a_j)/\omega(a_j)} \cdot \frac{\mu_{\mathbf{x}_{H,m}}(a_i)/\omega(a_i)}{(l-1!)\cdot\sum_{j=1}^{l}\mu_{\mathbf{x}_{H,m}}(a_j)/\omega(a_j)} \cdot f(Y \mid v_1)
\end{aligned}$$

Similarly, for every $\mathbf{y}'_H \in Y'$, we assign the probability density to its duplicates $\mathbf{y}'^{(1)}_H, \ldots, \mathbf{y}'^{(l)}_H$:

$$\begin{aligned}
&f'(\mathbf{y}'^{(i)}_H \mid v_1)\\
=&\frac{\mu_{\mathbf{x}_{H,m}}(a_i)/\omega(a_i)}{\sum_{j=1}^{l}\mu_{\mathbf{x}_{H,m}}(a_j)/\omega(a_j)} \cdot f'(\mathbf{y}'_H \mid v_1)\\
=&\frac{\mu_{\mathbf{x}_{H,m}}(a_i)/\omega(a_i)}{\sum_{j=1}^{l}\mu_{\mathbf{x}_{H,m}}(a_j)/\omega(a_j)} \cdot \frac{\mu_{\mathbf{x}'_{H,m}}(a_i)/\omega(a_i)}{(l-1!)\cdot\sum_{j=1}^{l}\mu_{\mathbf{x}'_{H,m}}(a_j)/\omega(a_j)} \cdot f'(Y \mid v_1)
\end{aligned}$$

Lemma 9. *For every pair of $\mathbf{y}_H \in [Y]$ and $\phi(\mathbf{y}_H) \in [Y']$,*

$$\frac{f(\mathbf{y}_H \mid v_1)}{f([Y] \mid v_1)} = \frac{f'(\phi(\mathbf{y}_H) \mid v_1)}{f([Y'] \mid v_1)}$$

We omit the proof as it is straightforward from the probability density defined above and Lemma 8.

We are now ready to prove the following lemma, which is a generalized version of Lemma 5:

Lemma 10. *If Σ is (ϵ, δ)-differentially oblivious for t corrupted users, then for any set of views V_2 from an execution of Σ, we have:*

$$\Pr_{\mathbf{y}_H \leftarrow Y}[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2] \le e^{\epsilon} \cdot \Pr_{\mathbf{y}'_H \leftarrow Y'}[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}'_H) \in V_2] + \delta,$$

where the notation $\mathbf{y}_H \leftarrow Y$ denotes sampling a vector $\mathbf{y}_H$ from set Y according to the distribution described above (similar for $\mathbf{y}'_H \leftarrow Y'$).

Proof. Let $\phi : [Y] \to [Y']$ be the bijection defined in Lemma 8. Recall that Y is shorthand for $Y(v_1, h)$. Differential obliviousness of Σ implies that for any $\mathbf{y}_H \in [Y]$:

$$\Pr\left[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2\right] \le e^{\epsilon} \cdot \Pr[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \phi(\mathbf{y}_H)) \in V_2] + \delta.$$

We have:

$$\Pr_{\mathbf{y}_H \leftarrow Y}\left[\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2\right]$$

$$\begin{aligned}
&= \Pr_{\mathbf{y}_H \leftarrow [Y]}\left[\text{VIEW}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2\right] \\
&= \sum_{\mathbf{y}_H \in [Y]} \Pr\left[\text{VIEW}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H) \in V_2\right] \cdot \frac{f(\mathbf{y}_H \mid v_1)}{f([Y] \mid v_1)} \\
&\leq \sum_{\mathbf{y}_H \in [Y]} (e^{\epsilon} \cdot \Pr[\text{VIEW}_{\Sigma,A}(\mathbf{y}_A, \phi(\mathbf{y}_H)) \in V_2] + \delta) \cdot \frac{f(\mathbf{y}_H \mid v_1)}{f([Y] \mid v_1)} \\
&= \sum_{\mathbf{y}'_H \in [Y']} (e^{\epsilon} \cdot \Pr[\text{VIEW}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}'_H) \in V_2] + \delta) \cdot \frac{f(\mathbf{y}'_H \mid v_1)}{f([Y'] \mid v_1)} \\
&= e^{\epsilon} \cdot \Pr_{\mathbf{y}'_H \leftarrow [Y']}[\text{VIEW}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}'_H) \in V_2] + \delta \\
&= e^{\epsilon} \cdot \Pr_{\mathbf{y}'_H \leftarrow Y'}[\text{VIEW}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}'_H) \in V_2] + \delta.
\end{aligned}$$

Handling duplicates. In the case that $\bar{h} = \{a_i\}_{i=1}^{l}$ contains only $d < l$ distinct values, we essentially treat each element of $\bar{h}$ as distinct and adjust the probability density properly. Concretely, let the respective number of these d values be $c_1, \ldots, c_d$. For all $\mathbf{y}_H \in Y$, we create $\prod_{i=1}^{d} c_i!$ duplicates and assign each duplicate with an evenly divided probability density $f(\mathbf{y}_H \mid v_1)/\prod_{i=1}^{d} c_i!$. We do the same for all $\mathbf{y}'_H \in Y'$. As each duplicate is treated as a distinct vector, we can just proceed as what we did earlier with no duplicate values.

Finally, we handle the remaining changes of notations and relevant lemma due to our use of probability density. We first define the continuous counterpart of $\Pr[v_1 \mid \mathbf{x}]$ and $\Pr[v_1 \mid \mathbf{x}]$: let $g(v_1)$ and $g'(v_1)$ denote the corresponding probability density at point v_1, notice that $g = g'$. We also adjust the notation $\Delta(v_1, h)$. In particular, for any v_1, h, let

$$\Delta(v_1, h) \overset{\text{def}}{=} \max\left\{f(Y(v_1, h) \mid v_1) - e^{\epsilon'} \cdot f'(Y'(v_1, h) \mid v_1),\ 0\right\}.$$

Using the above notations, we give the following continuous counterpart of Lemma 2. We skip the proof as it is analogous:

Lemma 11. *If $\Pi^{\mathcal{S}}$ is (ϵ', δ')-DP for t corrupted users, then for any set $V' = \{(v_1, h)\}$ and any pair of neighboring inputs $\mathbf{x}, \mathbf{x}'$, we have:*

$$\int_{(v_1,h) \in V'} g(v_1) \cdot \Delta(v_1, h) \leq \delta'.$$

Proof of Theorem 3. It suffices to prove that for arbitrary v_1, h and set of Σ's view V_2 consistent with v_1, h, the following inequality holds:

$$\begin{aligned}
&g(v_1) \cdot f(Y \mid v_1) \cdot \Pr_{\mathbf{y}_H \leftarrow Y}[\text{VIEW}(\mathbf{y}_H) \in V_2] \\
\leq &e^{\epsilon+\epsilon'} \cdot g'(v_1) \cdot f'(Y' \mid v_1) \Pr_{\mathbf{y}'_H \leftarrow Y'}[\text{VIEW}(\mathbf{y}'_H) \in V_2] + g(v_1) \cdot \Delta(v_1, h) + g(v_1) \cdot f(Y \mid v_1) \cdot \delta
\end{aligned}$$

where $\textsc{view}(\mathbf{y}_H)$ is shorthand for $\textsc{view}_{\Sigma,A}(\mathbf{y}_A, \mathbf{y}_H)$. Notice that for the last two terms on the RHS of the inequality, for any set $V' = \{(v_1, h)\}$:

$$\int_{(v_1,h)\in V'} g(v_1)\cdot \Delta(v_1,h) + \int_{(v_1,h)\in V'} g(v_1)\cdot f(Y \mid v_1)\cdot \delta \le \delta' + \delta.$$

This is due to Lemma 11 and $g(v_1)\cdot f(Y(v_1,h) \mid v_1)$ integrated over all possible views is 1. Moreover, the ratio bound between the integral of the first terms on both sides preserves:

$$\begin{aligned}&\int_{(v_1,h)\in V'} g(v_1)\cdot f(Y \mid v_1)\cdot \Pr_{\mathbf{y}_H\leftarrow Y}[\textsc{view}(\mathbf{y}_H)\in V_2]\\ &\quad / \int_{(v_1,h)\in V'} g'(v_1)\cdot f'(Y' \mid v_1) \Pr_{\mathbf{y}'_H\leftarrow Y'}[\textsc{view}(\mathbf{y}'_H)\in V_2]\\ &\le e^{\epsilon+\epsilon'}.\end{aligned}$$

Hence, it suffices to focusing on a single pair of v_1, h here.

By Lemma 10, we have that for all v_1, h:

$$\Pr_{\mathbf{y}_H\leftarrow Y}[\textsc{view}(\mathbf{y}_H)\in V_2] \le \min\left\{e^{\epsilon}\cdot \Pr_{\mathbf{y}'_H\leftarrow Y'}[\textsc{view}(\mathbf{y}'_H)\in V_2],\ 1\right\} + \delta,$$

It follows that

$$\begin{aligned}&g(v_1)\cdot f(Y \mid v_1)\cdot \Pr_{\mathbf{y}_H\leftarrow Y}[\textsc{view}(\mathbf{y}_H)\in V_2]\\ &\le g(v_1)\cdot f(Y \mid v_1)\cdot \left(\min\left\{e^{\epsilon}\cdot \Pr_{\mathbf{y}'_H\leftarrow Y'}[\textsc{view}(\mathbf{y}'_H)\in V_2],\ 1\right\} + \delta\right)\\ &\le g(v_1)\cdot f(Y \mid v_1)\cdot \min\left\{e^{\epsilon}\cdot \Pr_{\mathbf{y}'_H\leftarrow Y'}[\textsc{view}(\mathbf{y}'_H)\in V_2],\ 1\right\} + g(v_1)\cdot f(Y \mid v_1)\cdot\delta\\ &\le g(v_1)\cdot \left(e^{\epsilon'}\cdot f'(Y' \mid v_1) + \Delta(v_1,h)\right)\cdot \min\left\{e^{\epsilon}\cdot \Pr_{\mathbf{y}'_H\leftarrow Y'}[\textsc{view}(\mathbf{y}'_H)\in V_2],\ 1\right\}\\ &\qquad\qquad +g(v_1)\cdot f(Y \mid v_1)\cdot\delta\\ &\le g(v_1)\cdot\left(e^{\epsilon+\epsilon'}\cdot f'(Y' \mid v_1)\cdot \Pr_{\mathbf{y}'_H\leftarrow Y'}[\textsc{view}(\mathbf{y}'_H)\in V_2] + \Delta(v_1,h)\right)\\ &\qquad\qquad +g(v_1)\cdot f(Y \mid v_1)\cdot\delta\\ &= e^{\epsilon+\epsilon'} g'(v_1)\cdot f'(Y' \mid v_1)\cdot \Pr_{\mathbf{y}'_H\leftarrow Y'}[\textsc{view}(\mathbf{y}'_H)\in V_2] + g(v_1)\Delta(v_1,h)\\ &\qquad\qquad +g(v_1)\cdot f(Y \mid v_1)\cdot\delta.\end{aligned}$$

This concludes our proof.