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, Fairfax, USA
{gordon,mliang5}@gmu.com
[2] University of Maryland, College Park, USA
[3] Algorand, Boston, USA
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, in some cases, the shuffle model 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.

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,

D. Gordon—Work supported by NSF awards #1942575 and #1955264.
J. Katz—Work supported in part by a Facebook Privacy Enhancing Technologies Award and NSF award #1837517. Portions of this work were done while at GMU.
J. Xu—Portions of this work were done while at GMU.

G. Ateniese and D. Venturi (Eds.): ACNS 2022, LNCS 13269, pp. 501–520, 2022.
https://doi.org/10.1007/978-3-031-09234-3_25

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* [4,8,16,36], 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 mode. However, fully secure distributed-shuffling protocols are inefficient in practice (see Sect. 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 Sect. 1.1.) Roughly, for any honest pair of 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 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.

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 having one less user—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 shuffle locally [24,29]. Either approach requires $\Omega(n^2)$ communication. While it is possible to improve the asymptotic communication complexity to $O(n \log n)$ by using $\Theta(\log n)$-size committees (cf. [9,17,31]), the concrete efficiency of that approach is unclear.

Movahedi et al. [31] considered a relaxed version of shuffling in which security may fail completely with probability $O(1/n^3)$; this can be viewed as a form of differential obliviousness. The communication complexity of their protocol is $O(n \cdot \mathsf{polylog}\, n)$. Their protocol and that of Ando et al. [1] (discussed earlier) are the only practical protocols for shuffling we are aware of with sub-quadratic communication complexity.

Bell et al. [6] proposed a different approach for achieving a relaxed form of shuffling. Their construction requires $O(n^2)$ communication, which can be improved to $O(n \log n)$ for constant size input domains. To the best of our knowledge, it 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 does not provide a smooth tradeoff between privacy and performance as our approach does.

We provide a concrete comparison between our shuffling protocol and prior work in Sects. 4.4 and 5.1.

In roughly 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 (sender-message pair unlinkability) nearly identical to our own definition of differential obliviousness. Neither of these works show new protocols realizing their definitions. Several recent anonymous communication systems [27,34,35] also define security in terms of differential privacy, but the 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 [1,21,33] 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 some theoretical analyses of the anonymity provided by onion routing exist [1,2,11,18,20,26,28], 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 [5,12,22,23,29,30]. 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 $\mathbf{x}, \mathbf{x'} \in \mathbf{D^n}$ and subsets $R' \subseteq R$ we have

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

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 that $\mathcal{R}$ 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* [4,8,16,36] 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 $\text{VIEW}_{\Pi,A}(x_1, \ldots, x_n)$ 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 $\text{VIEW}_{\Pi,A}(\mathbf{x_A}, \mathbf{x_H})$ 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*

$$\Pr[\text{VIEW}_{\Pi,A}(\mathbf{x_A}, \mathbf{x_H}) \in \mathbf{V}] \leq \mathbf{e}^{\epsilon} \cdot \Pr[\text{VIEW}_{\mathbf{\Pi},\mathbf{A}}(\mathbf{x_A}, \mathbf{x'_H}) \in \mathbf{V}] + \delta.$$

The above can be relaxed to *computational* DP 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 $\mathbf{y}, \mathbf{y}'$ *neihgboring* 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,*

$$\Pr[\text{VIEW}_{\Sigma,A}(\mathbf{y_A}, \mathbf{y_H}) \in \mathbf{V}] \leq \mathbf{e}^{\epsilon} \cdot \Pr[\text{VIEW}_{\mathbf{\Sigma},\mathbf{A}}(\mathbf{y_A}, \mathbf{y'_H}) \in \mathbf{V}] + \delta.$$

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 $\epsilon_0 \leq \log((n-t)/\log(1/\delta'))/2$, 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 prove the above in the full version of our paper; here, we focus on the particular case of randomized response. 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):

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

(Note that $\Pr[v_1 \mid \mathbf{x}'] = \Pr[\mathbf{v_1} \mid \mathbf{x}]$ 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

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

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', Eq. (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 $\mathbf{y_H} \in \mathbf{Y}$ gives the same distribution as sampling uniform $\mathbf{y_H} \in [\mathbf{Y}]$, 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 Eq. (2) holds.

Since

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

combining Eqs. (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 $\mathbf{y_A} = (\mathcal{R}_{\gamma,\mathbf{D}} \times \cdots \times \mathcal{R}_{\gamma,\mathbf{D}})(\mathbf{x_A})$, i.e., the adversary's inputs to Σ. Following Balle et al. [4], we also include in v_1 the vector $\mathbf{b} = (\mathbf{b_1}, \ldots, \mathbf{b_m})$ 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 $\mathbf{y_H} \in \mathbf{D^m}$ 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) $\mathcal{S}(\mathbf{y_A}, \mathbf{y_H}) = \mathbf{h}$ (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 $\mathcal{R}_{\gamma,D}$

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:*

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

The proof is given in the full version.

We also state a useful corollary. Define

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

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

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

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 $\mathbf{y_H} \in \mathbf{Y}$ then $y_{H,m} = 1$, and if $\mathbf{y'_H} \in \mathbf{Y'}$ 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 $\{\mathbf{x_{H,i}} \mid \mathbf{b_i} = \mathbf{0}, \mathbf{i} \neq \mathbf{m}\}$ (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 $\mathbf{y_H} \in \mathbf{Y}$, 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 $\mathbf{y'_H} \in \mathbf{Y'}$, 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 $\mathbf{y_H} \in \mathbf{Y}$. 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 $\mathbf{y_H} \in \mathbf{Y}$. Similarly, we let $[Y']$ be a multiset consisting of c_1 copies of each element $\mathbf{y'_H} \in \mathbf{Y'}$. 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 $\text{VIEW}_{\Sigma,A}(\mathbf{y_A}, \mathbf{y_H})$. We may now prove the main result of this section.

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

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

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

$$\Pr\left[\text{VIEW}_{\Sigma,A}(\mathbf{y_A}, \mathbf{y_H}) \in \mathbf{V_2}\right] \leq e^{\epsilon} \cdot \Pr[\text{VIEW}_{\Sigma,A}(\mathbf{y_A}, \phi(\mathbf{y_H})) \in \mathbf{V_2}] + \delta.$$

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

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

Combining Corollary 2 and Lemma 5 allows us to prove Theorem 4. Details are given in the full version.

4 A Differentially Oblivious Shuffle Protocol

In this section, we describe a construction of a differentially oblivious shuffler. We present the protocol in Sect. 4.1 and analyze its obliviousness (for a semi-honest adversary) in Sects. 4.2 and 4.3. We compare its concrete performance to relevant prior work in Sect. 4.4. We defer a discussion of how to deal with malicious behavior to the full version.

Inputs: Each user i has input y_i.

Round 1: Each user chooses $r-1$ users $i_1, \ldots, i_{r-1} \leftarrow [n]$ 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 $\ell = 2, \ldots, r-1$**:** 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 $y_1, \ldots, y_n$, 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 $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_n, \mathsf{sk}_n)$, 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 $i_1, \ldots, i_{r-1} \leftarrow [n]$ 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 along 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 have obtained to the server. (We assume a synchronous communication network.) See Fig. 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 received. 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 $1 \le \ell \le r-1$, 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 $0 \le j < r-1$) 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 "flipped" (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 Fig. 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 $j \ge 1$ 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* $x_{n/3,r} \ge 1 - 0.85^r$*. Thus, for* $r > 1$ *the protocol of Fig. 1 is* $(0,\ 0.85^r)$*-differentially oblivious for* $n/3$ *corrupted users.*

For $r > 1$*, it holds that* $x_{n/2,r} \ge 1 - 0.95^r$*. Thus, for* $r > 1$ *the protocol of Fig. 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).

4.4 Performance Analysis

To analyze the performance of our protocol and compare it with prior work, 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 plus 20 bits for the user ID. An r-layer onion ciphertext thus requires $384 + 276(r - 1)$ bits.

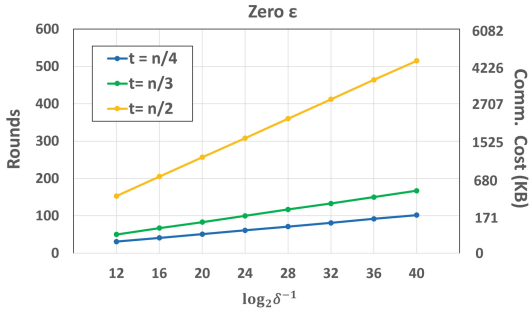

Fig. 2. Round complexity and per-user communication complexity for achieving $\epsilon = 0$ and different δ for various corruption thresholds, assuming 20-bit user IDs.

The $\epsilon = 0$ case. In Fig. 2, we give the number of rounds and per-user communication complexity needed to achieve $(0, \delta)$-differential obliviousness for several values of δ and various corruption thresholds. Note that these results are independent of the number of parties n. Our results compare favorably to prior work of Movahedi et al. [31], especially when the number of parties is large. In particular, for a corruption threshold of $t \approx n/3$ the protocol of Movahedi et al. [31] uses 500 rounds and communication of 128 MB per user when $n = 33,000$, and approximately 0.5–1 GB over 1,000 rounds when $n = 10^6$.

Additionally, note that δ is often set to be $10^{-4} \geq \delta \geq 10^{-6}$ in the differential privacy literature. Using that range of values, we require $r \approx 55$–83 with $n/3$ corrupted users, and our per-user communication cost is reduced to 53–119 KB.

The $\epsilon > 0$ case. In Fig. 3, we show how $\delta = \delta_1 + \delta_2$ relates to n, r, and t, and ϵ. Specifically, in Fig. 3(a) we show how the round/communication complexity depends on δ_1, and in Fig. 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 Fig. 3(b) we see that $\delta_2 \approx 2^{-23}$. Using Fig. 3(a), we see that 43 rounds suffice for $\delta_1 \approx 2^{-23}$. Thus, the protocol is $(1, 2^{-22})$-differentially oblivious with 43 rounds. Assuming 20 bits for the user IDs, this corresponds to per-user communication of 32 KB.

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

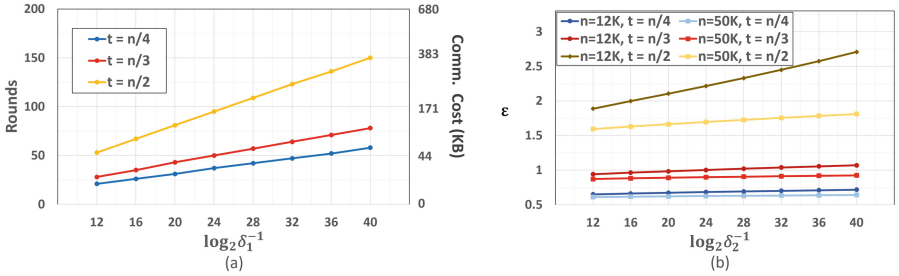

Fig. 3. (a) Round complexity and per-user communication complexity for achieving different δ_1 for various corruption thresholds, assuming 20-bit user IDs. (b) ϵ vs. δ_2 for various corruption thresholds and different n.

5 Malicious Security

We briefly discuss how to address malicious attacks affecting privacy; denial-of-service attacks and other attacks that affect correctness are out of scope. If the encryption scheme used by the protocol is non-malleable, and timestamps and identifiers are included in each layer of the onion to prevent replay attacks [11], then the only attack an adversary can carry out on the protocol of Sect. 4 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 the shuffle model. With regard to the latter point, note that although dropping even a single user's input can be catastrophic for differential obliviousness of a shuffling protocol (e.g., if $\mathbf{y}$ and $\mathbf{y}'$ are input vectors that differ by a transposition of the inputs of users 1 and 2, and the input of user 1 is dropped), 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. 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.*

It thus suffices to realize $\hat{\mathcal{S}}_d$ for small d. We describe our approach for doing so somewhat informally, and leave a detailed analysis for the full version. Let r, s be two parameters. At a high level, our modified protocol has four stages:

1. Each user U_i runs the onion-routing protocol from Sect. 4 twice, in parallel. It sends its real input y_i to the server using $r + s - 1$ intermediate hops, and sends a random dummy value to a randomly selected user R_i—called a "checker"–using $r - 1$ intermediate hops. Appropriate padding is used to make sure the onion encryptions are indistinguishable.
2. After round r, each user U_i asks R_i to respond with the random dummy value chosen by U_i. If R_i responds with the correct value, then U_i sets $\mathsf{cheat}_i := 0$; otherwise, it sets $\mathsf{cheat}_i := 1$.
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 phase.
4. Parties run the onion-routing protocol on the remaining real messages.

The overall argument for why this preserves privacy is as follows. Prior to round r, the adversary cannot distinguish real onion encryptions from dummy onion encryptions. Setting parameters appropriately, we can ensure that if a malicious adversary drops d or more of the honest users' onion encryptions before round r, then with high probability at least one of those will correspond to a dummy message associated with an honest checker; in that case, cheating will be detected and all honest users will abort. This, in turn, means that the real input of an honest user will be completely hidden from the adversary by the onion encryption unless the final s intermediate users chosen by that honest user for the onion-routing of its real message are all corrupted. The probability that this occurs for some honest user is at most $n \cdot (t/n)^s$.

The above shows that if the honest users do not abort by round r, then at most d of the honest users' real messages were dropped before round r. 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 Sect. 4. Nothing prevents the adversary from dropping as many messages as it likes after round r, but doing so cannot degrade the privacy already achieved by round r.

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 for our purposes; in particular, it is acceptable for us if the adversary causes disagreement among the honest users. We propose the following lightweight protocol that can be based on any multisignature scheme. Every user who holds $\mathsf{cheat} = 0$ sends a signature on some designated message M 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 M unless every honest party held $\mathsf{cheat} = 0$.

5.1 Performance Analysis

We analyze the communication overhead of the malicious protocol relative to the semi-honest protocol for the same privacy guarantees. Using dummy messages incurs roughly 2× overhead compared with the semi-honest protocol using the same number of rounds. (For simplicity, we do not count the communication in stages 2 and 3 which is anyway dominated by the onion routing. In fact, since dummy messages are not routed in stage 4, the communication overhead is less than 2× of the semi-honest protocol with the same number of rounds.) However, since the total number of rounds must be increased in the malicious setting, the overall communication overhead is higher. (Note that the total communication complexity is quadratic in the number of rounds since the length of each onion encryption is linear in the number of rounds.)

Zero ϵ. For the values of t, δ in Fig. 2, we need to set s equal to anywhere from 5% to 52% of r. This results in a total communication overhead of 2.2–4.6× compared to the semi-honest protocol.

Non-zero ϵ. For the parameters in Fig. 3, we need to set s equal to anywhere from 30–80% of r. This results in a total communication overhead of 3.4–6.4× compared to the semi-honest protocol.

Comparison to Prior Work. For $n = 1,000,000$ users, $t = n/3$, and to achieve $(0, 2^{-40})$-differential privacy, our malicious protocol requires $r + s = 212$ rounds and 1.5 MB communication per party. In comparison, for 1,000,000 parties and $t = n/5$, we estimate[4] Bell et al. [6] costs 12 rounds and communication of 199 KB per party. While the performance of our protocol is inferior, we note that in practice, often worse privacy parameters are chosen, and our protocol would then out-perform that of Bell at al. For example, if $(1.25, 2^{-20})$-differential privacy suffices and $t = n/3$, our per-party communication cost reduces to 169 KB using only 70 rounds. If $(0.454, 2^{-20})$-differential privacy suffices and $t = n/5$, our per-party communication cost reduces to 70.9 KB using only 45 rounds.

Finally, if a DO shuffle is used in applications beyond the privacy blanket, we compare even more favorably when the input domain size is larger than $O(n^{1/3})$. Specifically, our communication cost per party grows logarithmically in the domain size, while theirs either grows linearly in the domain size, or super linearly in n.

References

1. Ando, M., Lysyanskaya, A., Upfal, E.: Practical and provably secure onion routing. In: ICALP 2018, volume 107 of LIPIcs, pp. 144:1–144:14. Schloss Dagstuhl, July 2018

[4] We assume the availability of PKI so we drop the cost for Merkle tree verification in their protocol. We also use their baseline solution (rather than the invertible Bloom lookup table solution) to take advantage of the small domain setting.

2. Backes, M., Goldberg, I., Kate, A., Mohammadi, E.: Provably secure and practical onion routing. In: 25th IEEE Computer Security Foundations Symposium (CSF), pp. 369–385 (2012)
3. Backes, M., Kate, A., Manoharan, P., Meiser, S., Mohammadi, E.: AnoA: a framework for analyzing anonymous communication protocols. In: 26th IEEE Computer Security Foundations Symposium (CSF), pp. 163–178 (2013)
4. Balle, B., Bell, J., Gascón, A., Nissim, K.: The privacy blanket of the shuffle model. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 638–667. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_22
5. Beimel, A., Nissim, K., Omri, E.: Distributed private data analysis: simultaneously solving how and what. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 451–468. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_25
6. Bell, J.H., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly)logarithmic overhead. In: ACM CCS, pp. 1253–1269. ACM Press (2020)
7. Bellet, A., Guerraoui, R., Hendrikx, H.: 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, pp. 8:1–8:18 (2020)
8. Bittau, A., et al.: Prochlo: strong privacy for analytics in the crowd. In: Proceedings 26th Symposium on Operating Systems Principles (SOSP), pp. 441–459 (2017)
9. Boyle, E., Goldwasser, S., Tessaro, S.: Communication locality in secure multi-party computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 356–376. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_21
10. Bünz, B., Hu, Y., Matsuo, S., Shi, E.: Non-interactive differentially anonymous router (2021). Available at https://eprint.iacr.org/2021/1242
11. Camenisch, J., Lysyanskaya, A.: A formal treatment of onion routing. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 169–187. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_11
12. Hubert Chan, T.-H., Chung, K.-M., Maggs, B.M., Shi, E.: Foundations of differentially oblivious algorithms. In: 30th SODA, pp. 2448–2467. ACM-SIAM (2019)
13. Chan, T.-H.H., Shi, E., Song, D.: Optimal lower bound for differentially private multi-party aggregation. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 277–288. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33090-2_25
14. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Comm. ACM **24**(2), 84–88 (1981)
15. Chaum, D.: The dining cryptographers problem: unconditional sender and recipient untraceability. J. Cryptol. **1**(1), 65–75 (1988)
16. Cheu, A., Smith, A., Ullman, J., Zeber, D., Zhilyaev, M.: Distributed differential privacy via shuffling. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 375–403. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_13
17. Dani, V., King, V., Movahedi, M., Saia, J.: Brief announcement: breaking the $O(nm)$ bit barrier, secure multiparty computation with a static adversary. In: 31st ACM PODC, pp. 227–228. ACM (2012)
18. Das, D., Meiser, S., Mohammadi, E., Kate, A.: Anonymity trilemma: strong anonymity, low bandwidth overhead, low latency - choose two. In 2018 IEEE Symposium on Security and Privacy, pp. 108–126. IEEE Computer Society Press (2018)

19. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_14
20. Feigenbaum, J., Johnson, A., Syverson, P.F.: Probabilistic analysis of onion routing in a black-box model. ACM Trans. Inf. Syst. Secur. **15**(3), 14:1–14:28 (2012)
21. Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding routing information. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 137–150. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61996-8_37
22. Groce, A., Rindal, P., Rosulek, M.: Cheaper private set intersection via differentially private leakage. Proc. Priv. Enhancing Technol. (PETS) **2019**(3), 6–25 (2019)
23. He, X., Machanavajjhala, A., Flynn, C.J., Srivastava, D.: Composing differential privacy and secure computation: a case study on scaling private record linkage. In: ACM CCS 2017, pp. 1389–1406. ACM Press (2017)
24. Huang, Y., Evans, D., Katz, J.: Private set intersection: are garbled circuits better than custom protocols? In: NDSS 2012. The Internet Society, February 2012
25. Kuhn, C., Beck, M., Schiffner, S., Jorswieck, E.A., Strufe, T.: On privacy notions in anonymous communication (2018). Available at https://arxiv.org/abs/1812.05638
26. Kuhn, C., Beck, M., Strufe, T.: Breaking and (partially) fixing provably secure onion routing. In: 2020 IEEE Symposium on Security and Privacy, pp. 168–185. IEEE Computer Society Press (2020)
27. Lazar, D., Gilad, Y., Zeldovich, N.: Karaoke: distributed private messaging immune to passive traffic analysis. In: Proceedings 13th USENIX Conference on Operating Systems Design and Implementation, pp. 711–725. USENIX Association (2018)
28. Mauw, S., Verschuren, J.H.S., de Vink, E.P.: A formalization of anonymity and onion routing. In: Samarati, P., Ryan, P., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 109–124. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30108-0_7
29. Mazloom, S., Dov Gordon, S.: Secure computation with differentially private access patterns. In: ACM CCS 2018, pp. 490–507. ACM Press (2018)
30. Mazloom, S., Le, P.H., Ranellucci, S., Dov Gordon, S.: Secure parallel computation on national scale volumes of data. In: USENIX Security 2020, pp. 2487–2504. USENIX Association (2020)
31. Movahedi, M., Saia, J., Zamani, M.: Secure multi-party shuffling. In: Scheideler, C. (ed.) SIROCCO 2014. LNCS, vol. 9439, pp. 459–473. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25258-2_32
32. Smart, N.P., Talibi Alaoui, Y.: Distributing any elliptic curve based protocol. In: Albrecht, M. (ed.) IMACC 2019. LNCS, vol. 11929, pp. 342–366. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35199-1_17
33. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: 1997 IEEE Symposium on Security and Privacy, pp. 44–54. IEEE Computer Society Press (1997)
34. Tyagi, N., Gilad, Y., Leung, D., Zaharia, M., Zeldovich, N.: Stadium: a distributed metadata-private messaging system. In: Proceedings 26th Symposium on Operating Systems Principles (SOSP), pp. 423–440. ACM Press (2017)
35. van den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: scalable private messaging resistant to traffic analysis. In: Proceedings 25th Symposium on Operating Systems Principles (SOSP), pp. 137–152. ACM Press (2015)
36. Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., Thakurta, A.: Amplification by shuffling: from local to central differential privacy via anonymity. In: SODA 2019, pp. 2468–2479 (2019)