



# Mitigating Leakage from Data Dependent Communications in Decentralized Computing using Differential Privacy

Riad Ladjel, Nicolas Anciaux, Aurélien Bellet, Guillaume Scerri

## ► To cite this version:

Riad Ladjel, Nicolas Anciaux, Aurélien Bellet, Guillaume Scerri. Mitigating Leakage from Data Dependent Communications in Decentralized Computing using Differential Privacy. 2021. hal-03502320

**HAL Id: hal-03502320**

**<https://inria.hal.science/hal-03502320>**

Preprint submitted on 24 Dec 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Mitigating Leakage from Data Dependent Communications in Decentralized Computing using Differential Privacy

Riad Ladjel\*, Nicolas Anciaux\*, Aurélien Bellet†, Guillaume Scerri\*

\*Petrus team, Inria, France †Magnet team, Inria, France

**Abstract**—Imagine a group of citizens willing to collectively contribute their personal data for the common good to produce socially useful information, resulting from data analytics or machine learning computations. Sharing raw personal data with a centralized server performing the computation could raise concerns about privacy and a perceived risk of mass surveillance. Instead, citizens may trust each other and their own devices to engage into a decentralized computation to collaboratively produce an aggregate data release to be shared. In the context of secure computing nodes exchanging messages over secure channels at runtime, a key security issue is to protect against external attackers observing the traffic, whose dependence on data may reveal personal information. Existing solutions are designed for the cloud setting, with the goal of hiding all properties of the underlying dataset, and do not address the specific privacy and efficiency challenges that arise in the above context. In this paper, we define a general execution model to control the data-dependence of communications in user-side decentralized computations, in which differential privacy guarantees for communication patterns in global execution plans can be analyzed by combining guarantees obtained on local clusters of nodes. We propose a set of algorithms which allow to trade-off between privacy, utility and efficiency. Our formal privacy guarantees leverage and extend recent results on privacy amplification by shuffling. We illustrate the usefulness of our proposal on two representative examples of decentralized execution plans with data-dependent communications.

## I. INTRODUCTION

In many areas such as health, social networks or smart cities, the need to obtain information from citizens for the social good is growing. Hospitals are asking patient groups to provide health statistics<sup>1</sup> related to diseases for which little data is available or about recently recognized pathologies [46]. Researchers also ask citizen volunteers to collectively use their social media accounts to identify potential abuse and political propaganda [52]. Similarly, municipalities are using participatory sensing applications to collect urban statistics (e.g., on street noise exposure [29], [43], [13]). While legal frameworks such as data portability [18], [41] or data altruism [17] allow citizens to retrieve and share their personal data, asking citizens to provide raw data raises privacy concerns, with a risk of being perceived as mass surveillance [58] and limited chances for widespread adoption [33].

Instead, in this work we consider a setting where groups of citizens wish to collectively compute and share an *aggregate data release*, resulting from a distributed database query or

a federated machine learning process [31] executed on their personal data. In terms of security, this implies considering a distributed set of trusted computing nodes processing data locally, and exchanging results at runtime through secure communication channels. While techniques aiming to ensure that a computation does not leak information (including through communication patterns) exist, from secure multiparty computations [11] to secure outsourcing [16], they either have a very high overhead when massive number of parties are involved or rely on distributing trust between a small number of servers which is contradictory with our massively distributed setting. In order to avoid such drawbacks, we thus avoid changing the way distributed computations are performed and wish to keep computations at the level of client nodes. Securing local processing from an attacker can be tackled by solutions ranging from software security [5], [10] to hardware-based protection [42], [30], [40]. These solutions suggest that avoiding secure servers and relying only on simple client nodes is a feasible and sound approach.

In this paper, we focus on the problem of protecting against traffic analysis between computing nodes, whose dependence on data may reveal sensitive (e.g., personal) information. This is a standard, independent security problem that has been less studied than leakage from other side channels (such as memory access patterns [63], [61], [1], [37]), especially in the context of massively decentralized processing. Data-dependent communications in execution plans of distributed queries arise for efficiency reasons (to enable nodes to do more work in parallel) and depend on the query to be evaluated. For example, in distributed SQL and MapReduce analytics, nodes process data based on given (group-by) value intervals or (join) hashed key values (see e.g., [8]). In distributed data mining and machine learning algorithms, iterative updates to the models are split across computing nodes based on the distance to given centroids, according to regions of the feature space, or based on the similarity of local updates [62], [55], [53], [50], [28].

Existing solutions to hide the dependency between communication patterns and underlying data values have been proposed for the so-called *confidential computing* model [48], [54], [19], [20], [44], where computing nodes are allocated in cloud settings within secure enclaves and generate data exchanges at query time over secure communication channels. A first option is to make all communications (computationally) data independent. As a more practical alternative to the naive

<sup>1</sup>See e.g., the ComPaRe initiative in Paris hospitals: <https://compare.aphp.fr/>

option of broadcasting dummy messages to all nodes, [39] proposes to use communication padding (data transmitted from node to node is padded to a maximum size) and clipping (when the maximum size is reached, messages are discarded) for MapReduce computations. However, to avoid high communication overheads and preserve the utility of the computation, appropriately tuning the padding and clipping parameters is crucial. This requires prior knowledge on the data distribution to obtain a good balance between computing nodes, which is unrealistic in our context with potentially unknown sets of volunteers holding small data sets (in extreme cases, citizens contribute with a single personal record).

A second option is to anonymize communication patterns by resorting to secure shuffling [9], mixnets [23] or oblivious data exchanges [63]. However, the privacy guarantees obtained in a massively decentralized setting are difficult to formally quantify, especially against attackers with auxiliary knowledge. Even without such knowledge, anonymized communication patterns may still leak sensitive information (e.g., observing communication patterns to computing nodes that collect personal data for a given range of values would reveal the number of involved citizens with values in this range).

Hence, hiding communication patterns in the case of massively decentralized computations on end-user nodes calls for new solutions. A key challenge is to design techniques that can appropriately trade-off between privacy, utility and efficiency.

In this work, we propose to mitigate the leakage from data-dependent communication patterns with the tools of differential privacy, a mathematical definition of privacy with many interesting properties (including robustness against auxiliary knowledge). Our first contribution is to define a general execution model, in which the differential privacy guarantees of communications patterns resulting from global execution plans can be analyzed by combining guarantees obtained for local clusters of nodes through composition. Our second contribution is a set of algorithms to hide cluster-level communication patterns. Our algorithms rely on a combination of local sampling (each node randomizing the targets of messages), flooding (adding dummy messages) and shuffling subsets of messages with scramblers. We prove analytical differential privacy guarantees for our solutions by relying and extending recent results on privacy amplification by shuffling [14], [27], [4]. Finally, our last contribution is to highlight the practical usefulness of our solutions on two representative examples of execution plans with data-dependent communication, showing possible privacy-utility-efficiency trade-offs.

The rest of the paper is organized as follows. Section II defines the problem. Section III introduces a first approach where each node locally randomizes its messages. In Section IV, we propose algorithms where the noise can be shared across nodes. Section V presents an evaluation of our approach on two concrete use-cases. We review some related work in Section VI, and conclude in Section VII.

## II. PROBLEM DEFINITION

In this section, we describe our execution and adversary models, formulate the problem of mitigating the dependency of communication to data when executing distributed queries, and present the performance metrics used to evaluate the quality of solutions. We begin with a motivating example used throughout the section to illustrate the various concepts.

### A. Motivating Example

We introduce here a representative example of distributed query producing aggregated values of certain attributes grouped by other sets of attributes for a dataset contributed by a large set of participants. This type of queries, called *Aggregate* in the paper, is used to understand frequency distributions and collect marginal statistics from the data, as routinely performed in a data exploration phase. Such queries are also used in many data mining/machine learning algorithms (e.g., to learn decision trees). The data, computation and distribution models are as follows.

a) *Data model*: Personal data is hosted in users' source nodes and forms a partition of the dataset under consideration, as in federated machine learning or federated database scenarios. For simplicity, the dataset is represented by a single table  $\mathcal{D}(\mathbb{A})$ , with  $\mathbb{A}$  a fixed set of attributes used in the computation, and any source node hosts a single tuple  $t$  in  $\mathcal{D}$ .

b) *Computation model*: An Aggregate query on the dataset  $\mathcal{D}$  is expressed as  $Q = \{q\}$  a set of sub-queries, where each sub-query  $q = (\mathcal{G}_q, \mathcal{V}_q, \mathcal{F}_q)$  is defined by a set  $\mathcal{F}_q$  of statistical functions (e.g.,  $\{\text{count}, \text{avg}, \text{max}\}$ ) evaluated over a set of aggregate attributes  $\mathcal{V}_q \subset \mathbb{A}$  and grouped by another set of attributes  $\mathcal{G}_q \subset \mathbb{A}$ . Note that  $Q$  is a typical case of a SQL aggregate database query with a *grouping sets* clause.

c) *Distribution model*: For evaluating query  $Q$ , the dataset  $\mathcal{D}$  is partitioned vertically by sub-query  $q$ , and horizontally on grouping attribute values of  $\mathcal{G}_q$  for each sub-query. Each partition  $\mathcal{D}_q^i$  of the dataset is then assigned to a distinct compute node  $c_q^i$  in charge of computing  $q(\mathcal{D}_q^i)$ . The partitions  $\mathcal{D}_q^i$  can be formed by locally (i.e., at source node) assigning any tuple  $t \in \mathcal{D}$  to the appropriate partitions. The result of  $Q = \cup_{q,i} \{q(\mathcal{D}_q^i)\}$  is obtained by the union of the results produced by all compute nodes  $\{c_q^i\}$  on their partition  $\mathcal{D}_q^i$ .

For illustration purposes, in the rest of this section we consider the following query instance as a running example.

**Example 1** (Running example, Figure 1). A dataset for a diabetes study consists of a table  $\mathcal{D}(A, B, I)$  with two grouping columns, Age range ( $A$ ) and Body mass index ( $B$ ), and one aggregation column, Insulin rate ( $I$ ). Each row of the table is a tuple  $t_k = (a, b, i)$  with the values taken for a contributing patient. A query  $Q$  is the union of  $q_1 = (A, I, \text{avg})$  and  $q_2 = (B, I, \text{avg})$  which compute respectively the average insulin rate grouped by age range and grouped by body mass index. Note that this query is equivalent to the SQL database query: `SELECT A,B,AVG(I) FROM D GROUP BY GROUPING SETS (A),(B)`. The dataset  $\mathcal{D} = \{t_k\}_{k \in [1,d]}$  is distributed on a set of source nodes each

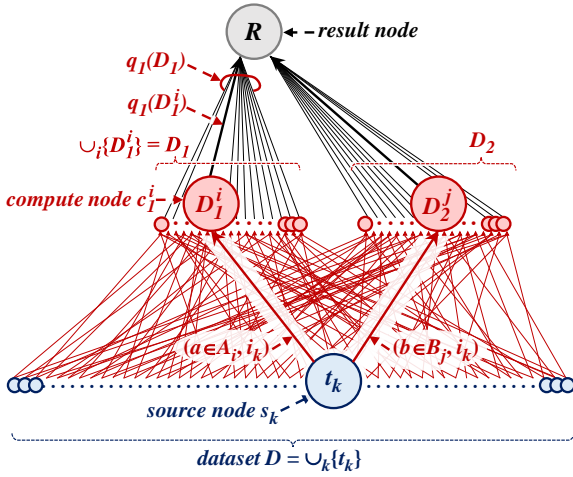


Fig. 1: Distributed query plan (Example 1).

holding one tuple  $t_k$ . To evaluate the query, a first vertical partition of the dataset  $\mathcal{D}_1 = \mathcal{D}(A, I)$  is used to evaluate  $q_1$  and a second  $\mathcal{D}_2 = \mathcal{D}(B, I)$  to evaluate  $q_2$ . The grouping domains  $A$  for  $q_1$  and  $B$  for  $q_2$  are respectively partitioned horizontally into  $C$  parts  $\{A_i\}_{i \in [1, C]}$  and  $\{B_j\}_{j \in [1, C]}$ , with  $\mathcal{D}_1^i = \{t \in \mathcal{D}_1, t.a \in A_i\}$  and  $\mathcal{D}_2^j = \{t \in \mathcal{D}_2, t.b \in B_j\}$ . The query execution is distributed on two groups  $\mathcal{C}_1$  and  $\mathcal{C}_2$  of  $C$  compute nodes each, computing  $q_1(\mathcal{D}_1^i)$  and  $q_2(\mathcal{D}_2^j)$  respectively. A source node holding  $t_k = (a_i, b_j, i_k)$  with  $a_i \in A_i$  and  $b_j \in B_j$  sends  $(a_i, i_k)$  to compute node  $c_1^i \in \mathcal{C}_1$  which evaluates  $q_1(\mathcal{D}_1^i)$  and  $(b_j, i_k)$  to  $c_2^j \in \mathcal{C}_2$  which evaluates  $q_2(\mathcal{D}_2^j)$ . The result  $R$  of the query is the union of the results produced by compute nodes.

**Data leakage from communications.** In general, an attacker observing the destination of (encrypted) messages sent by a user contributing to the computation deduces the values of the grouping attributes  $\mathcal{G}_1, \dots, \mathcal{G}_n$  for this user. For example, the user contributing tuple  $(A = 19, B = 18, I = 128)$  in Example 1 sends  $(A = 19, I = 128)$  to the compute node processing  $A = 19$ , and  $(B = 18, I = 128)$  to the one processing  $B = 18$ , and thus reveals their age and body mass index to the attacker. By extension, an attacker observing a compute node identifies users sharing the values processed by that compute node. If some compute nodes process sensitive values (e.g., nodes corresponding to low or high BMI values in Example 1), the communications reveal potentially discriminated users. More generally, the higher the number of grouping sets (desirable in terms of utility) and the finer the partitioning (beneficial from a performance viewpoint), the more extensive and accurate is the information inferred about contributors. Ultimately, an attacker observing the messages in the computation could infer all the grouping values of the participants, which amounts to revealing their profile (if not their identity, as a few demographic attributes are enough to uniquely identify an individual [47]).

## B. Execution Model

Our execution model aims to abstract away the specific computation to focus on capturing generic communication patterns. In order to achieve this, we break down the distributed computation between a set of elementary nodes that execute arbitrary computations but have simple communication behavior. We assume that nodes communicate on secure channels enforcing integrity and secrecy of communications (e.g. TLS). We then combine these nodes in order to obtain an *execution plan*.

a) *Nodes*: The simplest building block of our execution model is a node, which executes an elementary step of the computation. We assume that each node has a unique unforgeable public identifier (typically provided by a PKI). The input of this elementary computation consists of data received from other nodes and/or internal data of the node (represented as a set of tuples). As we focus on communication between nodes, we allow nodes to run any program on received data but explicitly require that a node sequentially follows three steps:

- 1) Receive a number of input messages from a set of source nodes (if any);
- 2) Process input data together with internal data of the node;<sup>2</sup>
- 3) Establish secure channels with target nodes and send a number of messages to a set of target nodes (if any).

We assume that the number of messages received and sent by a given node is fixed (if it is not, we add a dummy target node that represents discarding the message). We further assume that the size of messages does not depend on the value taken by the data (content of the message).

Given a node  $\nu$  and its input data  $\mathcal{D}_\nu$  (a set of tuples consisting of internal data and messages received from other nodes), the execution of  $\nu$  on  $\mathcal{D}_\nu$  generates some communication in the form of a set of triplets  $(\nu, m_i, t_i)$  indicating that node  $\nu$  sent a message with content  $m_i$  to target node  $t_i$ . We denote by  $\nu(\mathcal{D}_\nu)$  the set of triplets produced by the execution of  $\nu$  on  $\mathcal{D}_\nu$ . Since messages are indistinguishable to attackers (as we use secure channels between nodes and assume same size messages), in the rest of the paper we will often omit  $m_i$ .

**Example 2.** The computation described in Example 1 consists of four types of nodes:

- A set of source nodes, each holding one data tuple  $t_k = (a_i, b_j, i_k)$  where  $a_i \in A_i$  and  $b_j \in B_j$ . They do not receive messages and send two messages:  $(a_i, i_k)$  to the compute node  $c_1^i$  in  $\mathcal{C}_1$  processing  $\mathcal{D}_1^i$ , and  $(b_j, i_k)$  to the compute node  $c_2^j$  in  $\mathcal{C}_2$  processing  $\mathcal{D}_2^j$ .
- Two types of compute nodes  $\mathcal{C}_1 = \{c_1^i : 1 \leq i \leq C\}$  and  $\mathcal{C}_2 = \{c_2^j : 1 \leq j \leq C\}$  which receive a number of messages from the source nodes and send out one message (the aggregated result).
- A result node  $R$  which receives the (partially) aggregated data from the compute nodes.

<sup>2</sup>We do not consider the leakage that may occur during computation due to side channels (e.g. timing), see Section VI for a review of mitigation techniques based e.g. on constant time code, ORAMs, etc.

b) *Execution plans*: An execution plan is a set of nodes with disjoint node identifiers (if an “agent” needs to execute multiple computation steps, it executes several nodes). A distributed execution is simply the execution of all nodes where each message is taken as input to the specified target node.

**Definition 1** (Communication graph of an execution plan). *Given a set of nodes  $\mathcal{N}$ , a dataset  $\mathcal{D} = \bigcup_{\nu \in \mathcal{N}} \mathcal{D}_\nu$ , and  $r$  the randomness used in the computation (divided across all nodes), we define the communication graph  $\mathcal{N}^r(\mathcal{D})$  of the execution plan  $\mathcal{N}$  on data  $\mathcal{D}$  with randomness  $r$  as the union of the communication outputs  $\{(\nu, t_i)\} = \bigcup_{\nu \in \mathcal{N}} \nu(\mathcal{D}_\nu)$  produced by each node  $\nu \in \mathcal{N}$  with input data  $\mathcal{D}_\nu$  (message contents are excluded as they are hidden by secure channels and have same size) during the computation, with edges labeled by the order number of the message sent.*

Probabilities will be taken over  $r$ . When clear from the context,  $r$  may be omitted.

**Example 3.** *In our example, the communication graph for the execution plan outlined in Example 1 is the graph presented in Figure 1 where a source node  $s$  holding tuple  $t_k = (a_i, b_j, i_k)$  with  $a_i \in A_i$  and  $b_j \in B_j$  has an edge labeled 1 to the compute node  $c_1^i \in \mathcal{C}_1$  processing  $(a_i, i_k)$ , and an edge labeled 2 to the compute node  $c_2^j \in \mathcal{C}_2$  processing  $(b_j, i_k)$ , and all compute nodes have an outgoing edge to the result node  $R$ .*

### C. Adversary and Security Models

We assume that node identifiers cannot be forged by the adversary in order to impersonate nodes. This could typically be achieved through the use of a public key infrastructure (PKI). In the case of SGX for example, Intel would play the role of such a PKI, ensuring that the computation environment is what we expect. We assume that all nodes communicate on secure channels that provide two-way authentication, secrecy of messages (i.e. an adversary cannot distinguish between a true message and a random message of the same size), and integrity of communications (i.e. an adversary may not convince the receiver that a forged message was sent by the sender), under some cryptographic assumption  $\mathcal{H}$  (e.g. decisional Diffie–Hellman). For example, TLS with client authentication would satisfy these conditions under the cryptographic hypotheses ensuring that the adversary may not break the underlying encryption and signature schemes.

We consider an adversary observing all communications, and only communications, in particular the adversary cannot observe the internal state of nodes. Additionally we assume that the adversary cannot break the cryptographic hypotheses  $\mathcal{H}$  (and is otherwise unbounded). Therefore, the adversary cannot break the security of secure channels and thus cannot observe the content of messages.

The natural way to model the amount of information leakage of an execution plan  $\mathcal{N}$  is differential privacy, applied here to communication patterns. Formally, given an execution plan  $\mathcal{N}$  and a dataset  $\mathcal{D}$ , the adversary is given access to a

computation of  $\mathcal{N}(\mathcal{D})$  and tries to infer information about individual tuples in  $\mathcal{D}$  (e.g., the values of attributes  $A$  and  $B$  of a user in Example 1) leading to a natural definition similar to the computational differential privacy of [38]. As usual with differential privacy, we do not make any assumption on the auxiliary knowledge of the adversaries (they may have arbitrary knowledge and even know some tuples of the dataset). We then require that the adversary should not be able to distinguish two runs of the protocol with neighboring datasets (i.e. datasets differing by exactly one tuple) with good probability. As the adversary cannot break secure channels, we can abstract away the content of messages and the construction of secure channels (for more details see Appendix A), giving only the communication graph as observables for the adversary. Formally, under hypotheses  $\mathcal{H}$ , we have the following privacy definition.

**Definition 2** (DP for execution plans). *An execution plan  $\mathcal{N}$  is  $(\epsilon, \delta)$ -differentially private if for any neighboring  $\mathcal{D}_0, \mathcal{D}_1$  (differing by at most one tuple), and for all possible sets  $\mathcal{O}$  of communication graphs, we have:*

$$P[\mathcal{N}(\mathcal{D}_0) \in \mathcal{O}] \leq e^\epsilon P[\mathcal{N}(\mathcal{D}_1) \in \mathcal{O}] + \delta.$$

Note that providing privacy guarantees for releasing the result of the computation is an orthogonal problem, but one advantage of using differential privacy for protecting communications is that it will compose well with techniques that provide differential privacy guarantees for protecting the result or other side channels (see Section VI).

### D. Reducing the Problem to Clusters of Nodes

In order to analyze the problem and control data dependency, we restrict the type of nodes we consider as follows.

**Definition 3** (Simple node). *A simple node is a node such that modifying one input tuple in  $\mathcal{D}$  may only change one output message of this node (content and target).*

Importantly, this does not restrict the type of distributed computations that we can do in practice. For instance, although in Example 2 a source node is not simple (two messages depend on the tuple  $t_k$ ), it can be seen as a pair of simple “logical” nodes running on the same client, one sending  $(a, i)$ , the other one sending  $(b, i)$ . Note that when doing this transformation the input data of nodes is no longer disjoint between nodes. These two “logical” nodes share the same base identity (the identity of the physical node) and are differentiated by the order number of the message they send.

As a preliminary step towards achieving differential privacy for communications in an arbitrary execution plan, we reduce the problem to considering a set of simple nodes with the same communication behavior, namely a *cluster* of nodes. This allows for a generic analysis as we will see in Section III.<sup>3</sup> Moreover, working at the level of a cluster will allow us to

<sup>3</sup>In particular, for a cluster, the only data dependency is which target receives a message, and differential privacy of all communications essentially becomes recipient anonymity as defined in [3].

share noise addition for these nodes (see Section IV), obtaining better guarantees with less traffic while reasoning locally on the potential communications. Crucially, we show that if we provide privacy guarantees for each cluster of an execution plan, they translate into privacy guarantees for the overall execution plan by composition (see Theorem 1).

**Definition 4** (Cluster of nodes). *In an execution plan  $\mathcal{N}$ , a set of nodes  $\mathcal{C} \subseteq \mathcal{N}$  is a cluster if*

- *the set of potential target nodes  $\mathcal{T} \subseteq \mathcal{N}$  for each message sent by any node in  $\mathcal{C}$  is the same,*
- *all nodes in  $\mathcal{C}$  are simple,*
- *nodes in  $\mathcal{C}$  operate on disjoint data.*

**Example 4.** *In our running example, source nodes (users contributing data) are not simple as they produce two messages (see Figure 1) and thus cannot directly be divided into clusters. However, breaking down each source node  $s$  into two logical nodes  $s_1$  and  $s_2$  sending out messages for the first (resp. second) set of target nodes  $\mathcal{C}_1$  (resp.  $\mathcal{C}_2$ ), all nodes are simple. Denoting by  $\mathcal{S}_1$  (resp.  $\mathcal{S}_2$ ) the nodes communicating with  $\mathcal{C}_1$  (resp.  $\mathcal{C}_2$ ), note that the nodes in  $\mathcal{S}_1$  operate on disjoint data (different tuples of  $\mathcal{D}$ ), as do nodes in  $\mathcal{S}_2$  (but nodes in  $\mathcal{S}_1$  share data with nodes in  $\mathcal{S}_2$ , as nodes  $s_1$  and  $s_2$  process the same tuple  $t \in \mathcal{D}$ ). The computation can therefore be divided into five clusters:  $\mathcal{S}_1$ ,  $\mathcal{S}_2$ ,  $\mathcal{C}_1$  and  $\mathcal{C}_2$  and  $R$ , as shown in Figure 2. Note that communications originating from  $\mathcal{C}_1$  and  $\mathcal{C}_2$  are not data dependent (they only send out one message to  $R$ ), hence they do not need any countermeasure to hide their communication patterns.  $R$  does not send messages at all, and will therefore be ignored in the remainder of the paper.*

For simplicity and without loss of generality, in the rest of the paper we assume that nodes in a cluster only send one message. Indeed, as each input tuple of a simple node may only change one output message, all output messages can be computed independently.

As our goal is to modify execution plans without altering the underlying computation, we are restricted to working on communication graphs rather than the internal workings of nodes. Therefore, all our algorithms will take communication graphs as input and return new, perturbed communication graphs. With a reduction similar to the one presented in [3], based on our previous assumptions, we can abstract away the dependence on data and provide the following differential privacy definition, which is equivalent to Definition 2 for clusters of nodes. Indeed, with our definition of simple nodes, two neighboring datasets may only produce communication patterns differing by the recipient of at most one message, defining *neighboring communication graphs*.<sup>4</sup>

**Definition 5** (DP for communication graphs). *An algorithm  $\mathcal{A}$  is  $(\epsilon, \delta)$ -differentially private if for any two neighboring communication graphs  $\mathcal{G}_1, \mathcal{G}_2$  (differing in at most one message) and any set of communication graphs  $\mathcal{O}$ , we have:*

$$P[\mathcal{A}(\mathcal{G}_1) \in \mathcal{O}] \leq e^\epsilon P[\mathcal{A}(\mathcal{G}_2) \in \mathcal{O}] + \delta.$$

<sup>4</sup>This is in line with the notion of adjacency for recipient anonymity in [3].

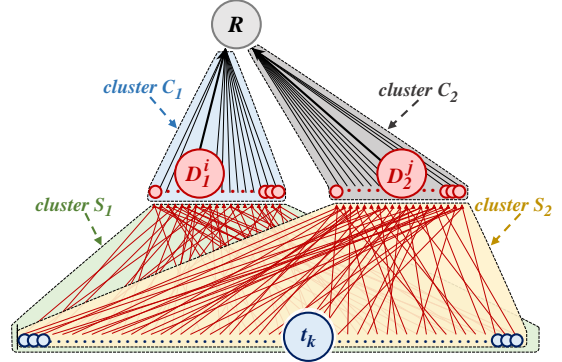


Fig. 2: Clusters of nodes (Example 4).

If  $\mathcal{A}$  can be written as a transformation of an execution plan  $\mathcal{N}$  (potentially by adding nodes in the middle of the original communication paths), with a slight abuse of notations we denote this transformation as  $\mathcal{A}(\mathcal{N})$ .

Finally, as execution plans may be composed of multiple clusters, we need a way of combining the privacy guarantees of each cluster into a guarantee for the overall execution plan. In order to do this, we take advantage of the composition properties of differential privacy and use the simple observation that a given data tuple may only influence communications along the communication path originating from its source.

**Theorem 1.** *Let  $\mathcal{N}$  be a valid execution plan where vertices form a set  $I$  of disjoint clusters  $(\mathcal{N}_i)_{i \in I}$ . If for each  $i \in I$ ,  $\mathcal{A}(\mathcal{N}_i)$  is  $(\epsilon_i, \delta_i)$ -differentially private, then  $\mathcal{A}(\mathcal{N})$  is  $(\epsilon, \delta)$ -differentially private where  $\epsilon$  and  $\delta$  are the worst possible sums of  $\epsilon_i$ 's and  $\delta_i$ 's encountered along the path of a data item the communication graph:*

$$\epsilon = \max_{\text{path } p \in \mathcal{N}} \sum_{i: \exists v \in \mathcal{N}_i \text{ s.t. } n \in p} \epsilon_i,$$

$$\delta = \max_{\text{path } p \in \mathcal{N}} \sum_{i: \exists v \in \mathcal{N}_i \text{ s.t. } n \in p} \delta_i.$$

*Proof.* The result follows from applying a combination of classical sequential and parallel composition results for differential privacy, see [26].  $\square$

**Remark 1.** *Execution plans typically require a small number of clusters. For instance, a single cluster is sufficient for simple Aggregate queries (with one grouping set). For computations that require an iterative process (e.g., K-means and machine learning algorithms in general), we make the graph acyclic by considering that each iteration is done by a different set of nodes (and each “agent” will execute several nodes). In terms of clusters, this means that each iteration adds more clusters and the privacy guarantee we obtain degrades with the number of iterations, as usual when considering differential privacy for this type of computation. Some optimizations can be made for specific computations, but we leave this for future work.*

**Example 5.** *In our running example, each data item encounters clusters  $\mathcal{S}_1, \mathcal{S}_2, \mathcal{C}_1, \mathcal{C}_2$  along its path. As mentioned before,  $\mathcal{C}_1$  and  $\mathcal{C}_2$  have data-independent communication outputs, and*



thus provide perfect privacy and do not need countermeasures. Therefore, if  $\mathcal{A}$  provides  $(\epsilon_1, \delta_1)$  and  $(\epsilon_2, \delta_2)$ -DP for  $\mathcal{C}_1$  and  $\mathcal{C}_2$  respectively, then  $\mathcal{A}(\mathcal{N})$  is  $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ -DP.

**Example 6.** If one is willing to trade off utility for a gain in privacy, another possible way of implementing (an analogue of) this computation would be for each participant to decide whether they want to participate to the aggregate on  $a$  or the aggregate on  $b$ . In this new execution plan  $\mathcal{N}'$ , each data tuple goes through exactly two clusters along its path (either  $\mathcal{S}_1, \mathcal{C}_1$  or  $\mathcal{S}_2, \mathcal{C}_2$ ) and  $\mathcal{A}(\mathcal{N}')$  is  $(\max\{\epsilon_1, \epsilon_2\}, \max\{\delta_1, \delta_2\})$ -DP.

As any execution plan can be broken down into clusters, and Theorem 1 provides DP guarantees on the whole execution plan from guarantees on clusters, we focus on providing guarantees at the cluster level. Specifically, Section III takes advantage of the fact that all nodes in a cluster have similar communication behavior in order to derive a DP bound using local differential privacy, while Section IV uses this common behavior to mutualize the cost of countermeasures across clusters while amplifying privacy guarantees. We then empirically show in Section V that for typical execution plans the guarantees provided at the cluster level transfer well to the execution plan level.

### E. Performance Metrics

In the general case it is theoretically impossible to have both perfect privacy and low overhead (in terms of latency and bandwidth) and solutions have to trade one for the other, as shown in [21]. While we study a more specific problem, solutions similarly lie on a spectrum from no privacy and no overhead to perfect privacy and huge overhead (e.g., broadcast). In this work, we explore trade-offs between privacy, utility and efficiency, defined as follows:

- *Privacy*, measured by the parameters  $\epsilon$  and  $\delta$  of differential privacy (which bound the amount of leakage about individual data point).
- *Utility*, measured as the number of tuples effectively used in the final result.
- *Efficiency*, divided into three distinct dimensions:
  - Network load: the amount of additional traffic generated by our solution w.r.t. non-private communications,
  - Individual load on users' nodes: the number of secure channels per node,
  - Additional users' consents: the number of additional user's nodes involved in the execution plan.

Note that depending on the specific implementation and computation considered, the relative importance of these metrics may vary. For instance, efficiency aspects may be crucial to make the approach practical when using secure hardware. Our solutions will provide simple ways to adjust the trade-off between these metrics.

## III. SOLUTION BY LOCAL SAMPLING AND FLOODING

Following the reduction described in Section II-D, we consider a single cluster of nodes composed of a set  $\mathcal{S}$  of

$S$  sources nodes, each source seeking to send one message to a set  $\mathcal{T}$  of  $T$  target nodes. In this context, a communication graph  $\mathcal{G}$  can be represented as a set of  $S$  messages  $\mathcal{G} = \{(s_1, t_1), \dots, (s_S, t_S)\}$  where each  $s_i \in \mathcal{S}$  and  $t_i \in \mathcal{T}$ . Our goal is to design a differentially private algorithm  $\mathcal{A}$  which takes as input a (non-private) communication graph  $\mathcal{G}$  and returns a perturbed graph  $\mathcal{A}(\mathcal{G})$  which preserves the utility and efficiency of the distributed computation as much as possible.

In this section, we propose and analyze a baseline solution in which each message  $(s, t)$  is randomized at the source node  $s$ , independently of others. In other words, the algorithm  $\mathcal{A}$  will be of the form

$$\mathcal{A}(\mathcal{G}) = \{\mathcal{R}(s_1, t_1), \dots, \mathcal{R}(s_S, t_S)\},$$

where  $\mathcal{R}$  is a *local randomizer* applied independently to each message. This setting corresponds to the so-called local model of differential privacy [24].

### A. Proposed Algorithm

Our algorithm is based on two principles: *sampling* and *flooding*. Sampling consists in randomizing the target of the message, following the idea of randomized response [59], [32]. Note however that in contrast to typical use-cases in which randomized response is used to perturb the content of the message, here we use it *perturb the destination of the message*. Sampling allows to trade utility for privacy without affecting efficiency. On the other hand, flooding consists in producing additional dummy messages that are indistinguishable from real messages from the attacker point of view, but can be discarded by target nodes at execution time so they do not affect the final result of the computation. Flooding alone cannot guarantee differential privacy (except in the extreme case when it becomes equivalent to a broadcast), but combined to sampling we will show that it allows to trade efficiency for improved privacy without affecting utility.

Formally, let  $\sigma \in [0, 1]$  be the sampling parameter and  $d \in \{1, \dots, T - 1\}$  the flooding parameter. Our local randomizer  $\mathcal{R}_\sigma^d$ , executed at each source node  $s$ , takes as input a message  $(s, t)$  and returns a collection of  $d + 1$  messages to be sent by the source node. These messages, which constitute the output observable by an adversary, are generated as follows: first, with probability  $1 - \sigma$ , the source node keeps the true message to the true target  $t$ , otherwise it sends a dummy message to a target  $t'$  chosen uniformly at random;<sup>5</sup> then, the source node creates  $d$  additional dummy messages aimed at a set of  $d$  targets chosen uniformly without replacement from  $\mathcal{T}$  (excluding the target of the first message). From this local randomizer  $\mathcal{R}_\sigma^d$ , we define the global algorithm as  $\mathcal{A}_\sigma^d(\mathcal{G}) = \{\mathcal{R}_\sigma^d(s_1, t_1), \dots, \mathcal{R}_\sigma^d(s_S, t_S)\}$ , which applies  $\mathcal{R}_\sigma^d$  to each message in  $\mathcal{G}$ . For convenience, we denote the sampling-only variants by  $\mathcal{R}_\sigma := \mathcal{R}_\sigma^0$  and  $\mathcal{A}_\sigma := \mathcal{A}_\sigma^0$ .

<sup>5</sup>If it happens that  $t' = t$ , we obviously send the true message to maximize utility.

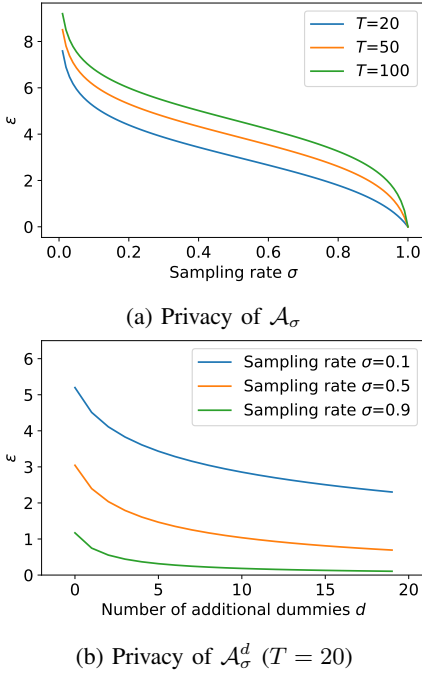


Fig. 3: Privacy of  $\mathcal{A}_\sigma$  and  $\mathcal{A}_\sigma^d$  varying  $\sigma$  and  $d$ .

### B. Privacy Analysis

We can show the following differential privacy guarantees for  $\mathcal{A}_\sigma^d$ . The proof can be found in Appendix B.

**Theorem 2.** Let  $\sigma \in [0, 1]$  and  $d \in \{1, \dots, T - 2\}$ . The algorithm  $\mathcal{A}_\sigma^d$  satisfies  $\epsilon$ -differential privacy with

$$\epsilon = \ln \left( \frac{(1 - \sigma)T}{\sigma(d + 1)} + 1 \right).$$

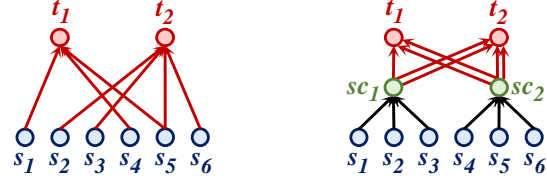
For the case where  $d = T - 1$ ,  $\mathcal{A}_\sigma^d$  is equivalent to a broadcast and thus  $\epsilon = 0$ .

To illustrate the influence of all parameters, we plot the values of  $\epsilon$  as given in Theorem 2 for different number of targets  $T$ , sampling parameter  $\sigma$  and flooding parameter  $d$ . Figure 3a shows the case of the sampling-only variant  $\mathcal{A}_\sigma$  (no flooding). We see that achieving reasonable values of  $\epsilon$  (typically,  $\epsilon$  is recommended to be smaller than  $\ln(3)$  [25]) requires a large  $\sigma$  (typically larger than  $\sigma = 0.9$ ), making  $\mathcal{A}_\sigma$  quite impractical for use-cases that require low sampling (e.g., when recruiting additional consenting users to act as source nodes is very costly). Flooding helps to reduce  $\epsilon$  while keeping  $\sigma$  fixed, as can be seen in Figure 3b. Interestingly, there are diminishing returns: the bulk of the gains in privacy come from the first dummies. We also see that the bigger  $\sigma$ , the faster the decrease of  $\epsilon$  with  $d$ .

### C. Performance Analysis

We analyze the performance in terms of *utility* and *efficiency* metrics defined in Section II-E.

*a) Utility:* In order to maintain the same utility (i.e., number of real contributions taken into account in the compute nodes) as the non-private algorithm with  $S$  sources, the total



(a) No scrambler (Section III) (b) 2 scramblers (Section IV)

Fig. 4: Cluster with 2 target, 6 source and 2 scrambler nodes.

number of source nodes must be  $S_t = \frac{S}{(1 - \sigma)}$ , with also an impact on efficiency with  $S_t - S$  additional users' consents.

*b) Efficiency:* In  $\mathcal{A}_\sigma^d$ ,  $d + 1$  messages are sent by each source to at most  $d + 1$  targets. The individual load on computation (target) nodes in total number of secure communication channels to be initiated is hence bounded by  $S_t \times (d + 1)$  and the total volume of exchanged messages is  $S_t \times (d + 1) \times \mu$ , with  $\mu$  the size of a single message. Overall,  $\mathcal{A}_\sigma^d$  introduces a factor  $(d + 1)$  overhead compared to a non-private execution.

While effective to enforce high privacy while maintaining high utility, this baseline solution based on sampling and flooding thus comes at a significant cost in efficiency.

## IV. AMPLIFYING PRIVACY VIA SCRAMBLERS

In the baseline algorithm proposed in the previous section, the privacy of each communication only comes from the sampling rate and flooding applied locally by the source node. In this section, we propose an approach where source nodes can benefit from the sampling applied at other source nodes as well as from shared dummy messages, thereby “hiding in the crowd”. To this end, we introduce an additional type of node called *scrambler* whose role is to collect a set of  $n$  locally randomized messages (from  $n$  source nodes), add  $d$  extra dummy messages and shuffle the output before transmitting the messages to the target nodes.

After introducing our approach in Section IV-A, we first state a pure  $\epsilon$ -DP guarantee in Section IV-B. Arguing that this result is quite conservative, in Section IV-C we prove  $(\epsilon, \delta)$ -DP guarantees which capture the desired “hiding in the crowd” effect, by extending recent results on amplification by shuffling [4]. Finally, we briefly discuss the efficiency of the proposed approach in Section IV-D.

### A. Proposed Algorithm

One of the tools at our disposal to balance privacy, utility and efficiency is to add computation nodes. We propose to add a set of *scrambler nodes*, which are assumed to have the same security properties as the other nodes and will be responsible for collecting messages from source nodes before transmitting them to target nodes. Recall that the content of messages is encrypted by source nodes and can only be decrypted by the target node. As relying on a single scrambler would introduce a single point of failure and require that the scrambler opens secure channels with each source node (which is impractical considering the load limitation constraints stated in Section II), we propose to add a set of  $S/n$  *scramblers* and assign each



---

**Algorithm 1:** Algorithm  $\mathcal{S}_d$  executed at each scrambler

---

**Input:** Set of messages  $\{(s_i, t'_i)\}_{i=1}^n$ , flooding parameter  $d$ , list of potential targets  $\mathcal{T}$ , boolean  $B$  (optional: to cap the number of messages per target, default `False`)

**Output:** set of messages  $\mathcal{O}$

```
1  $\mathcal{O} = \{t'_i\}_{i=1}^n$ 
2 for  $j \leftarrow 1$  to  $d$  do // add  $d$  dummy messages
3   if  $B = \text{True}$  then
4      $\mathcal{T} \leftarrow \mathcal{T} \setminus \{t : \mathcal{O}.\text{count}(t) = n\}$ 
5      $t_j \leftarrow$  uniformly random target from  $\mathcal{T}$ 
6      $\mathcal{O} \leftarrow \mathcal{O} \cup \{t_j\}$ 
7  $\mathcal{O}.\text{shuffle}()$  // random permutation
8 return  $\mathcal{O}$ 
```

---

source node to one scrambler in an input-independent manner. The parameter  $n$ , assumed to divide  $S$  for simplicity, thus corresponds to the number of source nodes assigned to each scrambler. Figure 4 shows the difference between this new architecture and the one considered in Section III.

We now describe the algorithm we propose in this setting. Let  $\sigma \in [0, 1]$  be the sampling parameter and  $d \in \mathbb{N}$  the flooding parameter. Given an input communication graph  $\mathcal{G} = \{(s_1, t_1), \dots, (s_S, t_S)\}$ , each message  $(s_i, t_i)$  is first processed by its source node  $s_i$  who applies the local randomizer  $\mathcal{R}_\sigma(s_i, t_i)$  (i.e., with sampling rate  $\sigma$  but no flooding) and sends the resulting message  $(s_i, t'_i)$  to the scrambler. Then, each scrambler collects the  $n$  inputs from its assigned sources, adds  $d$  dummy messages, shuffles the whole set of messages and sends them to the corresponding targets. The scrambler algorithm is shown in Algorithm 1. Note that the default behavior for creating dummy messages is to select targets uniformly at random with replacement, which we denote by  $\mathcal{S}_d$ . We will also consider a variant where the number of messages sent to each target is capped by  $n$ , which we denote by  $\bar{\mathcal{S}}_d$ . In any case, the scrambler can be implemented by a simple shuffling primitive, which is becoming standard in the design of private systems [9], [14], [27], [4], [64].

Crucially, the communication between sources and scramblers is input-independent, therefore from the point of view of the adversary the output can be restricted to the communication between scramblers and targets. Furthermore, since each scrambler operates over a distinct partition of  $n$  source nodes, from the differential privacy point of view it will be sufficient to analyze the algorithm at the level of a single partition. We denote the partition-level algorithm by  $\mathcal{A}_\sigma^{n,d}$ , which can be written as a sequential composition of the local randomizer  $\mathcal{R}_\sigma$  and the scrambler algorithm  $\mathcal{S}_d$ :

$$\mathcal{A}_\sigma^{n,d}(\mathcal{G}) = \mathcal{S}_d(\mathcal{R}_\sigma(s_1, t_1), \dots, \mathcal{R}_\sigma(s_n, t_n)). \quad (1)$$

Similarly, we denote by  $\bar{\mathcal{A}}_\sigma^{n,d}$  the variant based on  $\bar{\mathcal{S}}_d$ .

Intuitively,  $\mathcal{A}_\sigma^{n,d}$  and  $\bar{\mathcal{A}}_\sigma^{n,d}$  should achieve a better privacy-utility-efficiency trade-off than the baseline approach of Section III because an adversary can only infer information from

the communication pattern between the scrambler and the targets. In particular, (i) each dummy message added by the scrambler should improve privacy for all sources nodes in the partition, and (ii) local sampling at each source node should further help to hide each message destination. This is what we will examine in our privacy analysis.

### B. Privacy Analysis: Pure $\epsilon$ -DP

Our first result quantifies the privacy guarantees provided by the algorithm in terms of (pure)  $\epsilon$ -DP. For dummies to provide some benefit in terms of  $\epsilon$ -DP, we need to consider the variant  $\bar{\mathcal{A}}_\sigma^{n,d}$  where the number of messages to each target is capped by  $n$ . We have the following result.

**Theorem 3.** Let  $\sigma \in [0, 1]$  and  $d \in \{1, \dots, n-1\}$ . The algorithm  $\bar{\mathcal{A}}_\sigma^{n,d}$  satisfies  $\epsilon$ -differential privacy with

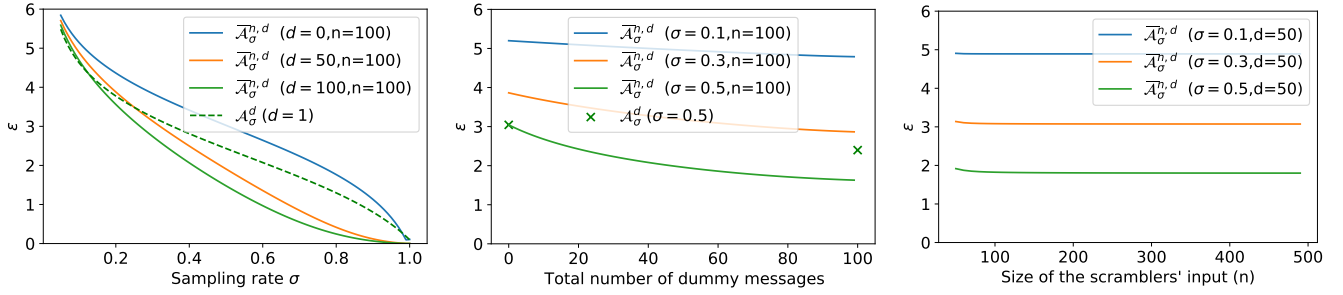
$$\epsilon^\epsilon = \frac{\sum_{k=0}^d \binom{d}{k} \binom{n-1}{k} (1-\sigma)^k R_{lie}^{n-k-1} \left(1 - \sigma + k \cdot \frac{R_{lie}^2}{1-\sigma}\right)}{\sum_{k=0}^d \binom{d}{k} \binom{n-1}{k} (1-\sigma)^k R_{lie}^{n-k-1} \left(R_{lie} + k \cdot \frac{R_{lie}^2}{1-\sigma}\right)}$$

where  $R_{lie} = \frac{\sigma}{T-1}$  and  $\binom{n}{k}$  is the binomial coefficient.

*Sketch of proof.* The main challenges are to deal with the combinatorial number of possible inputs and outputs, and the fact that each dummy message depends on the input messages as well as previously drawn dummies (due to the cap on the maximum number of messages per target). Our proof is based on factorizing the probability distribution of outputs in a way that allows us to identify the combination of neighboring communication graphs  $\mathcal{G}, \mathcal{G}'$  and output  $\mathcal{O}$  which produces the worst-case ratio  $P[\mathcal{A}_\sigma^{n,d}(\mathcal{G}) = \mathcal{O}] / P[\mathcal{A}_\sigma^{n,d}(\mathcal{G}') = \mathcal{O}]$ . We can then compute the exact value of this ratio, which gives  $\epsilon$ . The detailed proof can be found in Appendix C.  $\square$

As the formula in Theorem 3 is difficult to interpret, we plot the value of  $\epsilon$  when varying the parameters  $\sigma$ ,  $d$  and  $n$  in Figure 5. Figures 5a and 5b confirm that the privacy guarantees provided by  $\bar{\mathcal{A}}_\sigma^{n,d}$  increase with the local sampling rate  $\sigma$  and the number of dummies  $d$  added by the scrambler. More interestingly, we see that  $\bar{\mathcal{A}}_\sigma^{n,d}$  provides stronger privacy than the local algorithm  $\mathcal{A}_\sigma^d$  (without scrambler) when compared at equal sampling rate  $\sigma$  and total number of dummies. Equivalently,  $\bar{\mathcal{A}}_\sigma^{n,d}$  can match the privacy of  $\mathcal{A}_\sigma^d$  with fewer dummy messages, i.e., with better efficiency.

However, Figure 5c shows that the number of messages  $n$  given as input to the scrambler does not have a significant impact on the privacy guarantee. This is due to the fact that pure  $\epsilon$ -DP is governed by the ratio of probabilities of the worst-case output, even if the probability of that output actually occurring is extremely small. To illustrate the fact that  $\epsilon$ -DP is quite restrictive and gives a somewhat pessimistic view of the privacy guarantees provided by our algorithms, we performed a numerical simulation on a small problem instance for a large number of random runs (see Appendix D for details on this simulation). Figure 6 shows how many times each output occurred across runs of algorithm  $\mathcal{A}_\sigma^{n,d}$  on two neighboring communication graphs (the red and blue



(a) Varying sampling rate  $\sigma$  ( $n = 100$ ) (b) Varying number  $d$  of dummies ( $n = 100$ ) (c) Varying number  $n$  of messages ( $d = 50$ )

Fig. 5: Impact of the parameters  $\sigma$ ,  $d$  and  $n$  on the privacy of  $\bar{\mathcal{A}}_\sigma^{n,d}$ , measured by  $\epsilon$ , for  $T = 20$  targets. For comparison purposes, we also plot the privacy of  $\mathcal{A}_\sigma^d$  (the local algorithm without scrambler), which adds  $nd$  dummy messages in total.

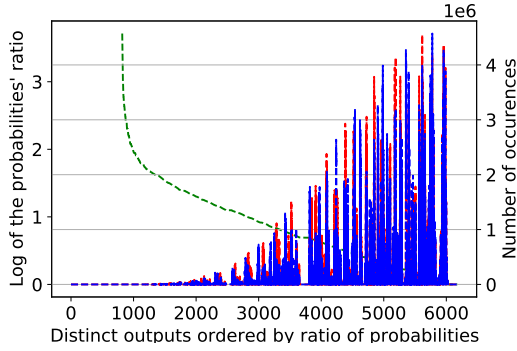


Fig. 6: Number of occurrences and corresponding log-ratio of probabilities (in decreasing order) for the distinct outputs obtained across  $7 \times 10^8$  runs of  $\mathcal{A}_\sigma^{n,d}$  ( $T = 4$ ,  $\sigma = 0.2$ ,  $d = 20$  and  $n = 20$ ). See main text and Appendix D for details.

histograms) and gives the estimated log-ratio of probabilities for each of these outputs (the green dots). The results clearly show the largest log-ratios (i.e., the ones that drive up the value of  $\epsilon$ ) correspond to very rare outputs. This was observed consistently across all inputs we tried.

Our simulation results suggest that Theorem 3 may not fully reflect the privacy benefits of our algorithms. In the next section, we prove  $(\epsilon, \delta)$ -differential privacy results which give a more faithful account of the privacy guarantees of  $\mathcal{A}_\sigma^{n,d}$ .

### C. Privacy Analysis: $(\epsilon, \delta)$ -DP via Amplification by Shuffling

The goal of this section is to prove  $(\epsilon, \delta)$ -DP guarantees for our algorithm  $\mathcal{A}_\sigma^{n,d}$ . Obtaining such guarantees is more challenging than proving  $\epsilon$ -DP, as the latter boils down to identifying and computing (or upper bounding) the worst-case ratio of probabilities over all possible outputs. In contrast, establishing  $(\epsilon, \delta)$ -DP requires to characterize the distribution of outputs so that its “long tails” (where the desired  $\epsilon$ -DP guarantees may not hold) can be accounted for in  $\delta$ .

Recall that  $\mathcal{A}_\sigma^{n,d}$  can be seen as the composition of two algorithms: a local randomizer  $\mathcal{R}_\sigma$  and a scrambler  $\mathcal{S}_d$  (see Eq. 1). To prove  $(\epsilon, \delta)$ -DP guarantees, we leverage and extend techniques from the recent literature on privacy amplification by shuffling [14], [27], [4]. Privacy amplification by shuffling

allows to prove differential privacy guarantees for algorithms of the form  $\mathcal{A} = \mathcal{S}_0 \circ \mathcal{R}^n$ , where  $\mathcal{R}$  is any differentially private local randomizer and  $\mathcal{S}_0$  simply returns a shuffle of its inputs. Privacy amplification by shuffling was designed to provide privacy for the *content* of messages. Here, we use this idea in the novel context of providing privacy for the *communication patterns*, and also extend the proof of [4] to account for the additional privacy brought by dummy messages added by our scrambler  $\mathcal{S}_d$ .

A key quantity in our analysis is the so-called *privacy amplification random variable* introduced by [4]. In our context, given  $\epsilon > 0$ , two messages  $(s, t)$  and  $(s, t')$  with  $t \neq t'$  and a random variable  $t'' \sim \text{Uniform}(\{1, \dots, T\})$  uniformly distributed over the set of targets, the privacy amplification random variable can be written as:

$$L_\epsilon^{t,t'} = \sigma(1 - e^\epsilon) + (1 - \sigma)T(\mathbb{I}[t'' = t] - e^\epsilon \mathbb{I}[t'' = t']), \quad (2)$$

where  $\mathbb{I}$  is the indicator function.  $L_\epsilon^{t,t'}$  is related to the difference in the probabilities that  $\mathcal{R}_\sigma$  outputs  $(s, t'')$  when applied to two different inputs  $(s, t)$  and  $(s, t')$ . It will quantify the contribution of each input-independent random message present in the final output (be it obtained from sampling by source nodes or from dummy messages added by the scrambler) to the  $(\epsilon, \delta)$ -differential privacy guarantees of  $\mathcal{A}_\sigma^{n,d}$ .

a) *Analytical bound:* We are now ready to state the main result of this section: an analytical  $(\epsilon, \delta)$ -DP guarantee for  $\mathcal{A}_\sigma^{n,d}$  (the proof can be found in Appendix E).

**Theorem 4.** *Let  $d \in \mathbb{N}$ . If  $\sigma > 0$ , then the algorithm  $\mathcal{A}_\sigma^{n,d}$  satisfies  $(\epsilon, \delta)$ -differential privacy for any  $\epsilon > 0$  with*

$$\delta \geq \frac{1}{\sigma n} \sum_{m=1}^n \frac{m}{m+d} \binom{n}{m} \sigma^m (1 - \sigma)^{n-m} \frac{b^2}{4a} e^{-\frac{2(m+d)a^2}{b^2}},$$

where  $a = 1 - e^\epsilon$  and  $b = (1 - \sigma)T(1 + e^\epsilon) - 2\sigma(1 - e^\epsilon)$ .

If  $\sigma = 0$ , then  $\mathcal{A}_\sigma^{n,d}$  satisfies  $(\epsilon, \delta)$ -DP for any  $\epsilon > 0$  with

$$\delta \geq \frac{1}{d+1} \frac{b^2}{4a} e^{-\frac{2(d+1)a^2}{b^2}}.$$

**Remark 2** (Generality of our analysis). *Beyond the particular case of the local randomizer  $\mathcal{R}_\sigma$  we use in this work, our analysis readily applies to any other differentially private local randomizer  $\mathcal{R}$ . The only condition is that dummy messages*

drawn by  $\mathcal{S}_d$  must follow a specific distribution which depends on the choice of  $\mathcal{R}$ , see Appendix E for details.

Theorem 4 highlights the trade-off between  $\epsilon$  and  $\delta$ . Given a value for  $\epsilon$ , the formulas directly give the lowest admissible value of  $\delta$ . Conversely, we can fix  $\delta$  and numerically search for the lowest value of  $\epsilon$  for which the inequality is satisfied. As  $\delta$  can be interpreted as the probability that the privacy loss exceeds  $\epsilon$ , it should be kept to a small value. A general guideline for  $\delta$  is that it must be smaller than  $1/n$  to provide meaningful guarantees [26]. Crucially, Theorem 4 shows that the privacy guarantees improve with the number  $n$  of messages, as each original message will become input-independent with probability  $\sigma$ . Dummy messages further contribute by guaranteeing a minimal number  $d$  of input-independent messages in the output. As shown by the second inequality, we can obtain  $(\epsilon, \delta)$ -DP guarantees even if  $\sigma = 0$ , i.e., without affecting the utility but only the efficiency.

For the sake of simplicity, Theorem 4 relies on Hoeffding's inequality to upper bound the sum of privacy amplification random variables. Numerically tighter (albeit more complex) bounds can be obtained by applying Bennett's inequality (which leverages the variance of  $L_e^{(t, t')}$ ), see Appendix E for details.

We use this improved version of our analytical bound to plot the privacy guarantees of  $\mathcal{A}_{\sigma}^{n, d}$  as a function of the different parameters and comparing to our previous  $\epsilon$ -DP result (Theorem 3). Figure 7 shows that our amplified  $(\epsilon, \delta)$ -DP guarantees provide large improvements over the previous  $\epsilon$ -DP result in regimes where the expected number of input-independent messages in the output (roughly  $\sigma n + d$ ) is sufficiently large (in the order of 300). When  $\sigma n$  and  $d$  are both small, Theorem 3 does not provide any privacy improvement and we fall back on the privacy guarantees provided by the local randomizer  $\mathcal{R}_{\sigma}$  alone.

*b) Further improvements:* The result of Theorem 4 is in fact quite pessimistic when both  $\sigma n$  and  $d$  are small: this is because the concentration inequalities used to bound  $\mathbb{E}[\sum_{i=1}^{m+d+1} L_i]_+$  are known to be loose when the number  $m + d + 1$  of terms in the sum is small. To get tighter privacy guarantees in such regimes, we can instead compute a Monte Carlo estimate of  $\mathbb{E}[\sum_{i=1}^{m+d+1} L_i]_+$ , i.e., we can approximate the expectation empirically using a finite number  $R$  of random samples. The procedure is outlined in Algorithm 2. Note that drawing a sample of the privacy amplification random variable amounts to fixing two arbitrary targets  $t \neq t'$ , sampling a target  $t''$  uniformly at random from  $\mathcal{T}$ , and computing Eq. 2.

The larger  $R$ , the closer the empirical estimate  $\hat{L}$  is to the true value. We can bound the deviation  $|\hat{L} - \mathbb{E}[\sum_{i=1}^{m+d+1} L_i]_+|$  with high probability using concentration inequalities, which gives a high probability bound on the error in estimating  $\delta$ . In all our plots and experiments we use  $R = 5000$ , which is sufficient for Hoeffding's inequality to ensure that the probability of the relative estimation error of  $\delta$  being larger than  $1/1000$  is negligibly small (below  $10^{-30}$ ).

Figure 7 shows the strong gains in the privacy guarantees

---

**Algorithm 2:** Empirical estimation of  $\mathbb{E}[\sum_{i=1}^{m+d+1} L_i]_+$

---

**Input:** Number of random samples  $R$

---

```

1 for  $r \leftarrow 1$  to  $R$  do
2   | Draw  $L_1^r, \dots, L_{m+d}^r$  according to Eq. 2
3   |  $L^r \leftarrow [\sum_{i=1}^{m+d} L_i^r]_+$ 
4  $\hat{L} \leftarrow \frac{1}{R} \sum_{r=1}^R L^r$ 
5 return  $\hat{L}$ 

```

---

obtained using this empirical estimation: we are able to obtain significant privacy amplification compared to pure  $\epsilon$ -DP (Theorem 3) even in regimes where both  $\sigma n$  and  $d$  are small.

In summary, we have derived analytical  $(\epsilon, \delta)$ -DP guarantees for our algorithm  $\mathcal{A}_{\sigma}^{n, d}$  by leveraging and extending techniques from the literature of privacy amplification by shuffling. We have also shown how to obtain tighter empirical bounds. We will see in Section V how to use our results to tackle practical use-cases.

#### D. Performance Analysis

As before, recall that we consider a simple computation with  $S = |\mathcal{S}|$  source nodes delivering a single message to one of the  $T = |\mathcal{T}|$  potential target nodes.

*a) Utility:* In order to maintain the same utility (i.e., number of real contributions) as the non-private algorithm with  $S$  sources, the total number of source nodes must be  $S_t = \frac{S}{(1-\sigma)}$ . We thus consider  $SF = S_t/n$  scramblers.

*b) Efficiency:* Each scrambler must open a secure communication channel with  $n$  sources and  $T$  targets and exchanges  $2 \times n + d$  messages. Hence the total number of secure channels is  $S_t + SF \times T$  and the volume of exchanged messages is  $S_t + SF \times (n + d) \times \mu$ .

### V. EVALUATION

User-side collaborative computing is gaining interest with the emergence of (1) cross-device federated learning [31], [12] where large sets of personal devices collaboratively train machine learning models, and (2) personal database management systems [56], [2] where populations of trusted user devices are engaged in collective database aggregation queries [34], [36].

Data-dependent communication schemes increase performance by distributing data to compute nodes based on the data values of group-by/join keys (e.g., parallel Oracle SQL Analytics [8]), data points distance to given centroids or regions of feature space (e.g., parallel  $K$ -means [62],  $K$ -medoids [55], DBSCAN [53]) or similarity of users' profiles [50], [28].

We focus on two types of distributed queries representative of these contexts, and illustrate the trade-offs between privacy, utility and efficiency obtained with our proposal.

#### A. Queries and Datasets

We consider two queries representative of above cases, called *Aggregate* and *K-means*. *Aggregate* is used to understand frequency distributions and collect marginal statistics

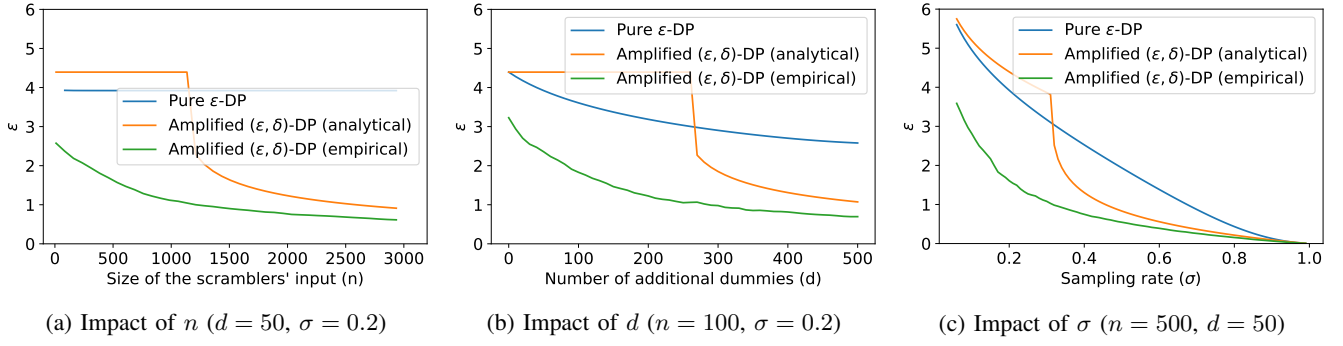


Fig. 7: Impact of  $n$ ,  $d$  and  $\sigma$  on the privacy of  $\mathcal{A}_q^{n,d}$ , measured by  $\epsilon$ , for  $T = 20$  targets. For  $(\epsilon, \delta)$ -DP, we set  $\delta = 10^{-4}$ .

from a set of consenting users (see example in Section II-A).  $K$ -means is representative of iterative data processing algorithms used for instance in data mining and machine learning. We describe the two queries and their execution plans below.

*Aggregate* (see Section II-A). A set of  $C \times G$  compute nodes, with  $G$  the number of grouping sets of attributes (e.g.,  $G = 2$  in Example 1), evaluates statistical functions (min, max, avg), with each compute node processing a partition of the input dataset and sending its output to the result node. A set of  $S$  source nodes, each holding a single tuple with numeric values (on which statistics are computed) and grouping values (according to which tuples are grouped), send values from their tuple to  $G$  compute nodes.

*K-means*: A set of  $S$  source nodes, each holding a single data tuple, compute the distance of their tuple to the  $K$  centroids and send their tuple to the compute node managing the closest centroid. A set of  $C \times I$  compute nodes (with  $I$  the number of iterations, and  $C = K$  the number of centroids), each managing a single centroid for a single iteration, update their centroid using the data tuples received from the source nodes, send their updates to the compute nodes for the next iteration, and propagate the updated centroids back to the source nodes they interact with. These steps are repeated for a fixed number  $I$  of iterations. The final result is transmitted to the result node. The initial state (first iteration) consists of  $K$  (random) points representing the initial centroids of  $K$  clusters.

In both cases, communication patterns between source and compute nodes reveal potentially sensitive information about the input data (grouping keys in Aggregate and close/similar users' tuples in  $K$ -means). Our proposal adds local sampling at source nodes and a set of SC scrambler nodes per grouping set/iteration between the source and compute nodes (see Figure 4b). For  $K$ -means, the new centroid obtained by each compute node at the end of the current iteration is sent back to all scrambler nodes which propagate them back to the source nodes they interact with to initiate the next iteration.

*Datasets*. For Aggregate, we use a synthetic dataset. We tested both uniform and biased distributions: for both cases we generated from 10k to 100k records distributed across 20 grouping intervals for 4 grouping attributes. For  $K$ -means, we use the classic MNIST dataset of handwritten digits, composed

of 70k records in dimension 784 and distributed among 10 classes. We execute  $K$ -means on the training set (60k), and measure the quality of the clusters we obtain on the test set (10k) using the rand index metric to compare to the ground-truth class labels and evaluate utility.

### B. Practical Trade-offs and Results

*Privacy evaluation*. In both execution plans any source node potentially sends messages (via scrambler nodes) to any compute node, and any partition of mutually disjoint sets of source nodes can define a set of clusters of nodes. Each of the SC scrambler nodes associated to a given grouping set (in Aggregate) or iteration (in  $K$ -means) takes as input a partition of the source nodes and hence belongs to different clusters that are never on the same data path. On the contrary, scrambler nodes assigned to successive iterations or different grouping sets use as inputs the same sets of input source nodes and are therefore on the same data path. In Aggregate, according to Theorem 1 the privacy of the overall execution plan is thus given by  $\epsilon \leq G \times \max_{1 \leq i \leq SC} (\epsilon_i)$  and  $\delta \leq G \times \max_{1 \leq i \leq SC} (\delta_i)$  where  $\epsilon_i$  and  $\delta_i$  denote the privacy guarantees for the cluster with the  $i$ -th scrambler. Similarly, the privacy of the execution plan for  $K$ -means is  $\epsilon = I \times \max_{1 \leq i \leq SC} (\epsilon_i)$  and  $\delta = I \times \max_{1 \leq i \leq SC} (\delta_i)$ .

*Parameters*. Different trade-offs between privacy, utility, and performance can be studied. The parameters of the experiments are shown in Table I.  $C$  is the number of compute nodes and determines the degree of parallelism of the algorithm. Its value depends on the computation and cannot be changed without impacting the performance. The value of the number  $S$  of source nodes (consenting users) influences both efficiency and utility. It is varying in our experiments between 10k and 20k. SC is the number of scrambler nodes involved per grouping set (for Aggregate, the number  $G$  of grouping sets varies from 1 to 4) and per iteration (for  $K$ -means, the number  $I$  of iterations is 10). Parameters  $\epsilon$  and  $\delta$  determine privacy. The numbers  $n$  of source nodes per scrambler, and  $d$  of dummy messages added per scrambler, affect privacy and efficiency, while the sampling rate  $\sigma$  affects privacy and utility. Fixing two of these three parameters and increasing the third would increase privacy.

*Results*. To study the impact of the different parameters in the trade-offs between privacy, utility and efficiency metrics

Name	Range
$C$	$20 (\times G \times I)$
$SC$	$20-100 (\times G \times I)$
$I$	10
$G$	1-4
$S$	$10k - 20k$
$\epsilon$	$0 - 5$
$\delta$	$10^{-4}$
$n$	10-600
$d$	0-1000
$\sigma$	0-1

TABLE I: Range of parameters for measures.

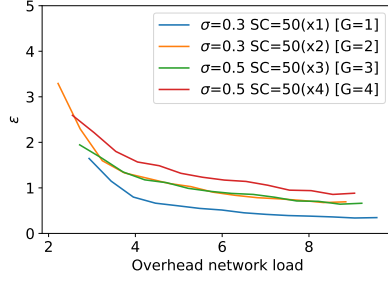


Fig. 9: Aggr.: privacy vs net. load.

(see Section II-E), for simplicity, we work at fixed utility, i.e., we fix the number of tuples effectively used by a compute node (except in Fig. 8d which varies utility). We then plot privacy as a function of the efficiency metrics along its three dimensions: (i) the *network load* overhead, evaluated as the number of messages added compared to a regular distributed execution, depends on the number of scrambler nodes  $SC$  added per grouping set or iteration, number of dummies  $d$  introduced by each scrambler node, and number of source nodes added for privacy reasons, (ii) the *individual load*, evaluated as the number of secure channels created per node, which depends mainly on the number  $n$  of source nodes per scrambler node<sup>6</sup> and is limited by the power/bandwidth of end-user devices, (iii) the number of additional *users' consents* required, influenced by the sampling rate  $\sigma$  and the number of contributors  $S$ . Note that we consider the three metrics for Aggregate, but only the first two for  $K$ -means. Indeed, the impact of sampling on utility is known to be negligible for  $K$ -means [51]. Preliminary measures using the rand index metric against the ground-truth clusters confirm that the proportion of correctly clustered tuples for high sampling rates ( $\sigma = 0.9$ ) is very close to that obtained when we do not sample.

The curves shown in Figures 8 and 9 show our results. In each curve, we evaluate the trade-offs between efficiency/utility parameters (X-axis) and privacy (Y-axis).

*Network load vs privacy (Figures 8a, 8e and 9).* We vary the number  $d$  of dummies introduced by the scrambler nodes, fixing the number of tuples effectively used to  $10k$ , for different configurations. We consider low sampling rates for Aggregate (to maximize utility) and high sampling rates for  $K$ -Means (without utility loss). Network overhead exists even without dummy (left side of the curves) due to the introduction of scrambler nodes. When  $d$  is increased, the privacy gains are significant, especially for the first dummies. Configurations with good privacy ( $\epsilon \leq 1$ ) and acceptable network load can be achieved.

*Additional users' consents vs privacy (Figures 8c and 8d).* Fig. 8c shows privacy at constant utility ( $10k$  significant contributions) for fixed network overheads, increasing the sampling rate  $\sigma$  hence the number  $S$  of users' consents (up to  $20k$ ), with the number  $d$  of dummies kept compliant with the target network overhead. We observe that it is more privacy efficient to increase  $\sigma$  (users' consents) than to increase  $d$ , especially with few messages per scrambler (low  $n$ , then

higher  $SC$  and lower  $d$ ). Similarly, Fig. 8d shows that at fixed number of users' consents ( $S=10k$ ) and dummies, sampling has a drastic effect on privacy, especially with fewer dummies.

*Individual load vs privacy (Figures 8b and 8f).* Individual load is typically determined by the application context (i.e. limitations of individual participants). Figures 8b and 8f show that increasing individual load on scramblers yields better privacy, however this effect also diminishes when individual load increases. In particular, these figures suggest that a few tens to a few hundreds channels are enough in most cases.

Overall, depending on the constraints of the use case (acceptable utility loss, maximum individual load, ...), a number of good configurations can be reached by tuning the number of scramblers, the sampling rate and the number of dummies.

## VI. RELATED WORK

*Anonymous communications.* Providing ways of communicating anonymously is far from a new problem. While our approach is focused on tackling data dependency in communications, it is closely related to various lines of work seeking to hide endpoints of communications.

The first and probably simplest way of providing anonymous communications is to use mix networks (mixnets) [49]. Recent work has studied how shuffling messages with mixnets can amplify local differential privacy guarantees (see Section VI). While we take inspiration from such works in our use of scramblers (which we do not deliberately call mixnets or shufflers as they also have the function of adding dummy messages), simply using mixnets would not provide differential privacy guarantees in our case as they do not hide the number of messages sent to each target. Similar solutions seeking to achieve anonymous routing (e.g TOR [22]) have the same problem, and typically induce a fairly high overhead in terms of communications and cryptographic computations at client side.

Differentially private messaging is more closely related to our goal. Vuvuzela [57] and a number of following works [35] seek to provide differentially private communications. These would satisfy our goal of hiding data dependency in a differentially private manner. However, these aim at a larger goal, which is to fully hide who communicates with who (and even the fact that users are communicating at all) rather than restricting the problem to hiding data dependency. In our case, we are willing to disclose the fact that a source is talking to a target, we simply want to hide which specific target. As a consequence, these works have a very high overhead as they need to drown all actual traffic within fake traffic (the system should behave in roughly the same way whether people are communicating or not), leading to network load being orders of magnitude greater than the number of actual messages sent. In our work we leverage the fact that distributed computations typically do not exhibit arbitrary communication patterns to obtain a much more efficient solution.

Finally, the work of [3] aims at modeling and providing tools for analyzing protocols where cryptographic guarantees and differential guarantees coexist and providing differential

<sup>6</sup>And to a lesser extent on the number  $C$  of compute nodes, as  $C$  is small.



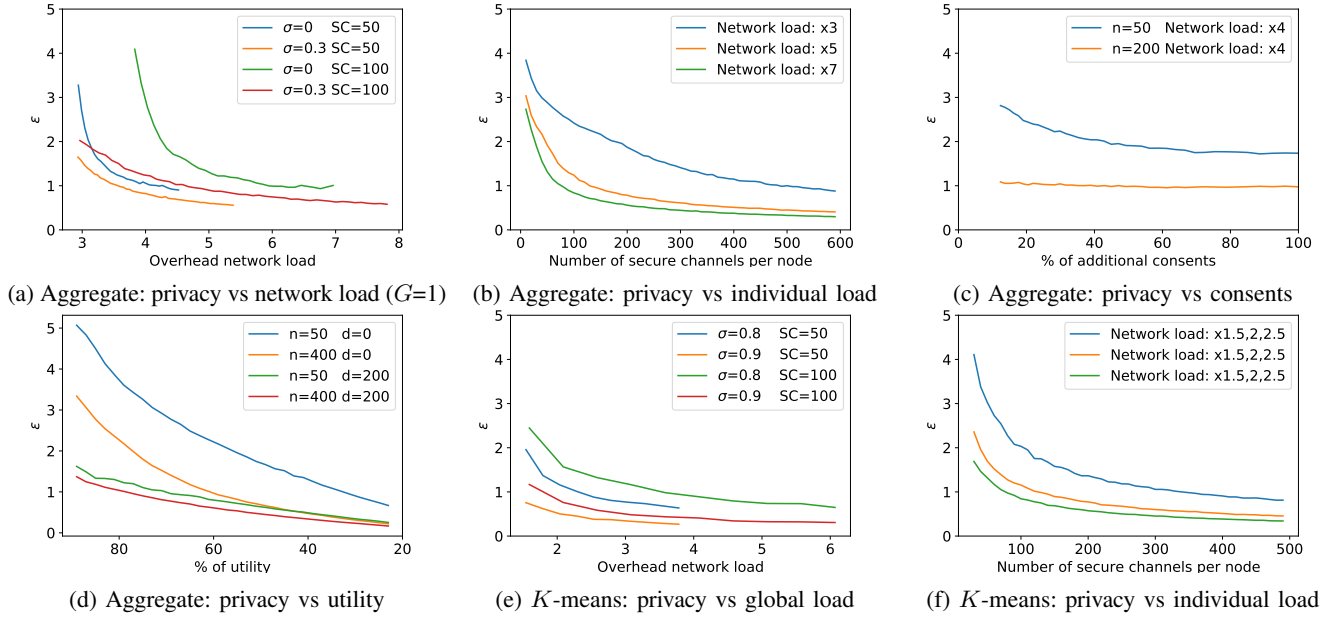


Fig. 8: Trade-offs between privacy, utility and efficiency in Aggregate and  $K$ -means.

privacy notions suitable for anonymous communications. Our adversary model is largely inspired by this work: in particular, our reduction from computational differential privacy and the adjacency notion for communication graphs in clusters is similar to the one in [3].

*Differentially private data analysis.* In terms of techniques used in this paper our work is closely related to differential privacy in the shuffle model [14], [27], [4] which provides an intermediate model between central and local DP. The shuffle model can reduce the utility cost of the local model by passing randomized data points through a secure shuffler (mixnet) before they are shared with an untrusted third party. However, many queries do not admit accurate solutions in the shuffle model [15]. In contrast we rely here on user-side trusted environments for distributed query evaluation, which provides different trade-offs. In particular, we avoid the loss in utility of local DP without requiring a trusted third party, and allow to accurately evaluate general queries (albeit at a potentially large cost in efficiency if trusted execution environments must be used to secure user-side computation). An original aspect of our work is to leverage DP and amplification by shuffling to guarantee the privacy of *data-dependent communications patterns* (and thereby mitigate traffic analysis attacks), while the above work on the shuffle and local models uses DP to guarantee the privacy of the *content of messages* with data-independent communication patterns. We also stress the fact that our approach nicely composes with central DP in use-cases where the result of the query evaluated in our framework is released in a differentially private way. In particular, if we provide an  $(\epsilon_1, \delta_1)$ -DP guarantee for communication patterns and an  $(\epsilon_2, \delta_2)$ -DP guarantee for releasing the query result, then by the composition property of DP we obtain an  $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ -DP guarantee against an adversary who observes both the communication patterns and the final result.

*Hiding memory access patterns and input/output size.* Many solutions were proposed to hide the query and/or the size of inputs and outputs of operators in execution plans [6], [7], [61], [45]. While the goal of these works is to hide data dependency in execution flows, these approaches tackle a different problem from ours. Indeed, only trees are considered (i.e. any operator has a single successor) and the goal is to hide the query executed within an operator and/or the result size, for example, hiding the number of tuples returned by a selection query. In contrast, our proposal supports any query plan, and prevents attackers from inferring information about individual tuples by observing the communications.

Another line of work consists in hiding memory access patterns in distributed cloud computations and local computations with multiple processors. In particular, [1], [37] provide differential privacy guarantees for memory access patterns. Specifically, [37] offers modifications to secure two-party computations between two non-colluding servers that hold individual data, and [1] offers a definition of oblivious differential privacy which is roughly the counterpart of ours when considering memory access pattern rather than communications but focuses on a single process rather than on interactions between multiple processes. Interestingly, these approaches can be used to protect individual nodes (which may perform complex computations), and the differential privacy guarantees they provide would compose nicely with ours.

*Hiding data dependency in communication patterns.* Several existing works use various anonymous communication techniques to hide data exchange between nodes in distributed query plans in a cloud setting [9], [23], [63]. While these are somewhat related to ours, they assume users have a (very) large amount of data and offer either unsatisfactory guarantees in our massively decentralized setting or massive overheads.

A more direct way to hide the dependency between com-



munication patterns and private data values is to make all communications data-independent, as done in [39] by padding and clipping messages in MapReduce computations for confidential computing in the cloud. However, this technique would produce massive overheads or very imprecise results in our case, where there is no prior knowledge on the data distribution to appropriately tune the padding and clipping parameters.

## VII. CONCLUSION

In this paper, we proposed a differentially private solution to mitigate the leakage from data-dependent communications in massively distributed computations. We leveraged recent work on privacy amplification by shuffling to formally prove privacy guarantees for our solution. We also showed how to balance privacy, utility and efficiency on two use-cases representative of distributed computations, highlighting the genericity of our solution.

We hope that our proposal will contribute to the development of new decentralized models for Data Altruism [17], in which citizens contribute the computation of socially useful information, with community control over the computation performed on the user side. Many research questions remain open, such as formulating and validating a collective computing “manifesto”. Another concrete challenge relates to the implementation of a platform to support these technologies. An interesting future work is to build on the emergence of Personal Data Management Systems (PDMS) [2], [56], which provide new tools for individuals to collect their personal data and control how they share results of local computations.

## ACKNOWLEDGMENTS

This work was supported by the French National Research Agency (ANR) through grants ANR-16-CE23-0016 (Project PAMELA) and ANR-20-CE23-0015 (Project PRIDE).

## REFERENCES

- [1] Joshua Allen, Bolin Ding, Janardhan Kulkarni, Harsha Nori, Olga Ohrimenko, and Sergey Yekhanin. An algorithmic framework for differentially private data analysis on trusted processors. In *NeurIPS*, pages 13635–13646, 2019.
- [2] Nicolas Anciaux, Philippe Bonnet, Luc Bouganin, Benjamin Nguyen, Philippe Pucheral, Iulian Sandu Popa, and Guillaume Scerri. Personal data management systems: The security and functionality standpoint. *Inf. Syst.*, 80:13–35, 2019.
- [3] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. Anoa: A framework for analyzing anonymous communication protocols. *J. Priv. Confidentiality*, 7(2), 2016.
- [4] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 638–667. Springer, 2019.
- [5] Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel N. Kho, and Jennie Rogers. SMCQL: secure query processing for private data networks. *Proc. VLDB Endow.*, 10(6):673–684, 2017.
- [6] Johes Bater, Xi He, William Ehrich, Ashwin Machanavajjhala, and Jennie Rogers. Shrinkwrap: Efficient SQL query processing in differentially private data federations. *Proc. VLDB Endow.*, 12(3):307–320, 2018.
- [7] Johes Bater, Yongjoo Park, Xi He, Xiao Wang, and Jennie Rogers. SAQE: practical privacy-preserving approximate query processing for data federations. *Proc. VLDB Endow.*, 13(11):2691–2705, 2020.
- [8] Srikanth Bellamkonda, Hua-Gang Li, Unmesh Jagtap, Yali Zhu, Vince Liang, and Thierry Cruanes. Adaptive and big data scale parallel execution in oracle. *Proc. VLDB Endow.*, 6(11):1102–1113, 2013.
- [9] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnés, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *SOSP*, pages 441–459. ACM, 2017.
- [10] Peter Boggetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.
- [11] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, pages 1175–1191. ACM, 2017.
- [12] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards Federated Learning at Scale: System Design. In *MLSys*, 2019.
- [13] Mariem Braham, Guillaume Scerri, Nicolas Anciaux, and Valérie Issarny. Consent-driven data use in crowdsensing platforms: When data reuse meets privacy-preservation. In *PerCom*, pages 1–10. IEEE, 2021.
- [14] Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed Differential Privacy via Shuffling. In *EUROCRYPT*, pages 375–403. Springer, 2019.
- [15] Albert Cheu and Jonathan R. Ullman. The Limits of Pan Privacy and Shuffle Privacy for Learning and Estimation. Technical report, arXiv:2009.08000, 2020.
- [16] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Crypt: Crypto-assisted differential privacy on untrusted servers. In *SIGMOD Conference*, pages 603–619. ACM, 2020.
- [17] EU Commission. Proposal for a regulation of the european parliament and of the council on european data governance (data governance act), com/2020/767., 25 October 2020.
- [18] EU Commission. Regulation (eu) 2016/679 of the european parliament and of the council on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)., 27 April 2016.
- [19] Confidential Computing Consortium. *Confidential Computing Consortium Defining and Enabling Confidential Computing*, 2020.
- [20] Confidential Computing Consortium. *Confidential Computing: Hardware-Based Trusted Execution for Applications and Data*, 2020.
- [21] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In *IEEE Symposium on Security and Privacy*, pages 108–126. IEEE Computer Society, 2018.
- [22] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320. USENIX, 2004.
- [23] T. T. Anh Dinh, P. Saxena, E. C. Chang, B. C. Ooi, and C. Zhang. M2R: enabling stronger privacy in mapreduce computation. In *USENIX Security Symposium*, 2015.
- [24] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438. IEEE Computer Society, 2013.
- [25] Cynthia Dwork. Differential privacy: A survey of results. In *TAMC*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2008.
- [26] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.
- [27] 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*, pages 2468–2479. SIAM, 2019.
- [28] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *NeurIPS*, 2020.
- [29] Sara Hachem, Vivien Mallet, Raphael Ventura, Animesh Pathak, Valérie Issarny, Pierre-Guillaume Raverdy, and Rajiv Bhatia. Monitoring noise pollution using the urban civics middleware. In *BigDataService*, pages 52–61. IEEE Computer Society, 2015.

- [30] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel. Ryoan: A distributed sandbox for untrusted computation on secret data. *ACM Trans. Comput. Syst.*, 35(4):13:1–13:32, 2018.
- [31] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrede Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and Open Problems in Federated Learning. Technical report, arXiv:1912.04977, 2019.
- [32] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. In *NIPS*, 2014.
- [33] Jan Krämer, Pierre Senellart, and Alexandre de Streel. *Making data portability more effective for the digital economy: Economic implications and regulatory challenges*. Centre on Regulation in Europe asbl (CERRE), 2020.
- [34] Riad Ladjel, Nicolas Anciaux, Philippe Pucheral, and Guillaume Scerri. Trustworthy distributed computations on personal data using trusted execution environments. In *TrustCom/BigDataSE*, pages 381–388. IEEE, 2019.
- [35] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *OSDI*, pages 711–725. USENIX Association, 2018.
- [36] Julien Loudet, Iulian Sandu Popa, and Luc Bouganim. SEP2P: secure and efficient P2P personal data processing. In *EDBT*, pages 145–156. OpenProceedings.org, 2019.
- [37] Sahar Mazloom and S. Dov Gordon. Secure computation with differentially private access patterns. In *CCS*, pages 490–507. ACM, 2018.
- [38] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil P. Vadhan. Computational differential privacy. In *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 126–142. Springer, 2009.
- [39] Olga Ohrimenko, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Markulf Kohlweiss, and Divya Sharma. Observing and preventing leakage in mapreduce. In *CCS*, pages 1570–1581. ACM, 2015.
- [40] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious multi-party machine learning on trusted processors. In *USENIX Security Symposium*, pages 619–636. USENIX Association, 2016.
- [41] Stuart L. Pardo. The California Consumer Privacy Act: Towards a European-Style Privacy Regime in the United States. *J. Tech. L. & Pol'y*, 23:68, 2018.
- [42] R. Pires, D. Gavrill, P. Felber, E. Onica, and M. Pasin. A lightweight mapreduce framework for secure processing with SGX. In *CCGrid*, 2017.
- [43] Iulian Sandu Popa, Dai Hai Ton That, Karine Zeitouni, and Cristian Borcea. Mobile participatory sensing with strong privacy guarantees using secure probes. *Geoinformatica*, 25(3):533–580, 2021.
- [44] Fahmida Y Rashid. The rise of confidential computing: Big tech companies are adopting a new security model to protect data while it's in use-[news]. *IEEE Spectrum*, 57(6):8–9, 2020.
- [45] Kui Ren, Yu Guo, Jiaqi Li, Xiaohua Jia, Cong Wang, Yajin Zhou, Sheng Wang, Ning Cao, and Feifei Li. Hybridx: New hybrid index for volume-hiding range queries in data outsourcing services. In *ICDCS*, pages 23–33. IEEE, 2020.
- [46] Carole Robert, Jean Imbert, Mohamed Lajnef, Camille Noûs, Gilbert Cabiran, Serge Robert, Françoise Cabiran, and Flavie Mathieu. Production of knowledge using data collected by associations of patients: The fibromyalgia example. *Med Sci (Paris)*, 37(1):81–88, 2021.
- [47] Luc Rocher, Meenatchi Sundaram Muthu, and Yves-Alexandre de Montjoye. The observatory of anonymity: An interactive tool to understand re-identification risks in 89 countries. In *WWW (Companion Volume)*, pages 687–689. ACM / IW3C2, 2021.
- [48] Mark Russinovich, Manuel Costa, Cédric Fournet, David Chisnall, Antoine Delignat-Lavaud, Sylvan Clebsch, Kapil Vaswani, and Vikas Bhatia. Toward confidential cloud computing: Extending hardware-enforced cryptographic protection to data while in use. *ACM Queue*, 19(1):49–76, 2021.
- [49] Krishna Sampigethaya and Radha Poovendran. A survey on mix networks and their secure applications. *Proc. IEEE*, 94(12):2142–2181, 2006.
- [50] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Trans. Neural Networks Learn. Syst.*, 32(8):3710–3722, 2021.
- [51] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.
- [52] Márcio Silva, Lucas Santos de Oliveira, Athanasios Andreou, Pedro Olmo Stanciolli Vaz de Melo, Oana Goga, and Fabrício Benevenuto. Facebook ads monitor: An independent auditing system for political ads on facebook. In *WWW*, pages 224–234. ACM / IW3C2, 2020.
- [53] Hwanjun Song and Jae-Gil Lee. RP-DBSCAN: A superfast parallel DBSCAN algorithm based on random partitioning. In *SIGMOD Conference*, pages 1173–1187. ACM, 2018.
- [54] David Sturzenegger, Aetienne Sardon, Stefan Deml, and Thomas Hardjono. Confidential computing for privacy-preserving contact tracing. *CoRR*, abs/2006.14235, 2020.
- [55] Ying ting Zhu, Fu zhang Wang, Xing hua Shan, and Xiao yan Lv. K-medoids clustering based on mapreduce and optimal search of medoids. In *9th International Conference on Computer Science Education*, pages 573–577, 2014.
- [56] Lachlan Urquhart, Neelima Sailaja, and Derek McAuley. Realising the right to data portability for the domestic internet of things. *Pers. Ubiquitous Comput.*, 22(2):317–332, 2018.
- [57] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: scalable private messaging resistant to traffic analysis. In *OSDP*, pages 137–152. ACM, 2015.
- [58] Anne Wagner, Aleksandra Matulewska, and Sarah Marusek. Pandemica panoptica: Biopolitical management of viral spread in the age of covid-19. *International Journal for the Semiotics of Law-Revue internationale de Sémiotique juridique*, pages 1–37, 2021.
- [59] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.
- [60] Gary C White. *Capture-recapture and removal methods for sampling closed populations*. Los Alamos National Laboratory, 1982.
- [61] Min Xu, Antonis Papadimitriou, Andreas Haeberlen, and Ariel Feldman. Hermetic: Privacy-preserving distributed analytics without (most) side channels. *Technical report*, 2019.
- [62] Weizhong Zhao, Huifang Ma, and Qing He. Parallel K-means clustering based on mapreduce. In *CloudCom*, volume 5931 of *Lecture Notes in Computer Science*, pages 674–679. Springer, 2009.
- [63] Wenting Zheng, Ankur Dave, Jethro G. Beekman, Raluca Ada Popa, Joseph E. Gonzalez, and Ion Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In *NSDI*, pages 283–298. USENIX Association, 2017.
- [64] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, Shuffle, Analyze Privacy Revisited: Formalizations and Empirical Evaluation. Technical report, arXiv:2001.03618, 2020.

APPENDIX A  
COMPUTATIONAL DIFFERENTIAL PRIVACY

We explain how we can reduce our attention to the privacy notion defined in Definition 2 under our hypotheses. Similarly to [38], we start from a notion which explicitly restricts the power of the adversary.

**Definition 6** (Computational DP for execution plans). *An execution plan  $\mathcal{N}$  is  $(\epsilon, \delta)$ -differentially private if for any neighboring  $\mathcal{D}_0, \mathcal{D}_1$  (differing by at most one tuple), and any probabilistic polynomial time adversary  $\mathcal{A}$  we have:*

$$P[\mathcal{A} \text{ interacting with } \mathcal{N}(\mathcal{D}_0) \text{ guesses } 0] \leq e^\epsilon P[\mathcal{A} \text{ interacting with } \mathcal{N}(\mathcal{D}_1) \text{ guesses } 0] + \delta.$$

We can make two immediate simplifications to this definition using our assumption that communication channels are secure. The first one is to exclude adversaries that influence the computation by injecting messages (as such an adversary would need to break secure channel integrity). We can therefore restrict ourselves to a passive adversary. The second one is that we may also omit the content of messages, as under secure channels all messages can be replaced by random messages without the adversary being able to differentiate the two situations. Finally, as two identical communication graphs are trivially indistinguishable when the content of messages is random, the question reduces to whether different neighboring datasets generate similar communication graphs, leading to our Definition 2 in the main text.

APPENDIX B  
PROOF OF THEOREM 3

*Proof.* Let  $\mathcal{G}$  and  $\mathcal{G}'$  be two neighboring communication graphs and let  $(s, t)$  and  $(s, t')$  with  $t \neq t'$  be the messages that differ in  $\mathcal{G}$  and  $\mathcal{G}'$  respectively. To prove that  $\mathcal{R}_\sigma^d$  satisfies  $\epsilon$ -differential privacy, we need to show that for any possible output  $\mathcal{O}$ :

$$\frac{P[\mathcal{A}_\sigma^d(\mathcal{G}) = \mathcal{O}]}{P[\mathcal{A}_\sigma^d(\mathcal{G}') = \mathcal{O}]} \leq e^\epsilon.$$

Since each message is processed independently by  $\mathcal{R}_\sigma^d$ , and  $\mathcal{G}$  and  $\mathcal{G}'$  are neighboring, we have:

$$\frac{P[\mathcal{A}_\sigma^d(\mathcal{G}) = \mathcal{O}]}{P[\mathcal{A}_\sigma^d(\mathcal{G}') = \mathcal{O}]} = \frac{P[\mathcal{R}_\sigma^d(s, t) = \mathcal{O}]}{P[\mathcal{R}_\sigma^d(s, t') = \mathcal{O}]} \quad (3)$$

We now seek to bound the above ratio for the worst-case output. By construction of  $\mathcal{R}_\sigma^d$ , the probability of producing a message  $(s, t'')$  with  $t'' \neq t \neq t'$  is the same for  $\mathcal{R}_\sigma^d(s, t)$  and  $\mathcal{R}_\sigma^d(s, t')$ . Therefore, we can focus on four cases for the output  $\mathcal{O}$  produced by  $\mathcal{R}_\sigma^d$ : (i)  $(s, t) \in \mathcal{O}$  and  $(s, t') \in \mathcal{O}$ , (ii)  $(s, t) \notin \mathcal{O}$  and  $(s, t') \notin \mathcal{O}$ , (iii)  $(s, t) \in \mathcal{O}$  and  $(s, t') \notin \mathcal{O}$  and (iv)  $(s, t) \notin \mathcal{O}$  and  $(s, t') \in \mathcal{O}$ .

The probability of the first two cases is the same under both inputs, hence their ratio is 1. The two last cases are symmetric, so without loss of generality we consider an output  $\mathcal{O}$  such that  $(s, t) \in \mathcal{O}$  and  $(s, t') \notin \mathcal{O}$ . Consider sampling  $d+1$  elements without replacement from  $\mathcal{T}$ : for two distinct  $t, t' \in \mathcal{T}$ , let

$E_{\neg t'}$  be the event where  $t'$  is not selected,  $E_{\neg t \wedge \neg t'}$  the event where neither  $t$  nor  $t'$  are selected and  $E_{t \wedge \neg t'}$  the event where  $t$  is selected but not  $t'$ . We have:

$$\begin{aligned} P[E_{t \wedge \neg t'}] &= P[E_{\neg t'}] - P[E_{\neg t \wedge \neg t'}] \\ &= \frac{T-d-1}{T} - \frac{(T-d-1)(T-d-2)}{T(T-1)} \\ &= (d+1) \frac{T-d-1}{T(T-1)}. \end{aligned}$$

Using the above, we can compute the ratio of probabilities in Eq. 3:

$$\begin{aligned} \frac{P[\mathcal{R}_\sigma^d(s, t) = \mathcal{O}]}{P[\mathcal{R}_\sigma^d(s, t') = \mathcal{O}]} &= \frac{(1-\sigma) \frac{T-(d+1)}{T-1} + \sigma P[E_{t \wedge \neg t'}]}{\sigma P[E_{t \wedge \neg t'}]} \\ &= \frac{(1-\sigma) \frac{T-(d+1)}{T-1} + \sigma(d+1) \frac{T-d-1}{T(T-1)}}{\sigma(d+1) \frac{T-d-1}{T(T-1)}} \\ &= \frac{(1-\sigma)T}{\sigma(d+1)} + 1 \leq e^\epsilon, \end{aligned} \quad (4)$$

which combined with Eq. 3 shows that  $\mathcal{A}_\sigma^d$  satisfies  $\epsilon$ -DP.  $\square$

Incidentally, Eq. 4 shows that the local randomizer  $\mathcal{R}_\sigma^d$  satisfies  $\epsilon$ -local differential privacy.

APPENDIX C  
PROOF OF THEOREM 3

To facilitate the reading, we introduce additional notations. Let  $\mathcal{G}$  and  $\mathcal{G}'$  be two neighboring communication graphs and let  $(s, t)$  and  $(s, t')$  with  $t \neq t'$  be the messages that differ in  $\mathcal{G}$  and  $\mathcal{G}'$  respectively. We abstract the graphs  $\mathcal{G}$  as  $\mathcal{G} = \{x, y, z\}$  where  $x$  is the number of messages targeting  $t$ ,  $z$  is the number of messages targeting  $t'$  and  $y$  is the number of messages targeting  $t'' \in \mathcal{T} \setminus \{t, t'\}$  with  $t \neq t'$ . Following the same principle we denote the output as  $\mathcal{O} = \{\alpha, \beta, \gamma\}$  where  $\alpha$  is the number of messages the scrambler sends to the target  $t$ ,  $\gamma$  is the number of messages are sent to the target  $t'$  and  $\beta$  is the number of messages are sent to  $t'' \in \mathcal{T} \setminus \{t, t'\}$ . The other used notations are summarized below:

- $\mathbb{P}_{n,d}(\frac{\alpha}{\beta} | \frac{x}{y})$  is the probability to get an output  $\mathcal{O} = \{\alpha, \beta, \gamma\}$  with the algorithm  $\bar{\mathcal{A}}_\sigma^{n,d}$  given a graph  $\mathcal{G} = \{x, y, z\}$
- $R_n^d(\frac{\alpha}{\beta} | \frac{x}{y})$  is the ratio  $\frac{\mathbb{P}_{n,d}(\frac{\alpha}{\beta} | \frac{x+1}{y})}{\mathbb{P}_{n,d}(\frac{\alpha}{\beta} | \frac{x}{y+1})}$  and it is equal to  $e^\epsilon$
- $\mathbb{P}_{n,d}^{\text{dum}}(\frac{k_1}{k_2} | \frac{\alpha - k_1}{\beta - k_2})$  is the probability to send  $k_1$  dummies (resp.  $k_2, k_3$ ) to the target  $t$  (resp.  $t'' \in \mathcal{T} \setminus \{t, t'\}$ ).
- $\Phi_{u,v}$  is the probability to draw  $u$  times  $\alpha$ ,  $v$  times  $\gamma$  and  $x+z-u-v$  times  $\beta$  (i.e.  $\mathbb{P}_{n,0}(\frac{x+z-u-v}{x} | \frac{u}{v})$ ).
- $\mathbb{I}(\text{conditions})$  is the indicator function (equal to 1 when the conditions are met and 0 otherwise).

a) *Useful formulas:* based on the notations above, we provide below some useful formulas used in the remaining of the chapter.

$$\mathbb{P}_{n,d}\left(\frac{\alpha}{\beta} \middle| \frac{x}{y}\right) = \sum_{k_1+k_2=d} \mathbb{P}_{n,d}^{\text{dum}}\left(d-k_1-\frac{k_1}{k_2} \middle| \frac{\alpha-k_1}{\beta-(d-k_1-k_2)} \right) \cdot \mathbb{P}_{n,0}\left(\beta-(d-k_1-k_2) \middle| \frac{x}{y}\right) \quad (5)$$

$$\mathbb{P}_{n,0}\left(\frac{\alpha}{\beta} \middle| \frac{x}{y}\right) = \frac{y! \bar{\sigma}^{\beta} \left(\frac{\sigma}{T-1}\right)^{y-\beta}}{\alpha! \beta! \gamma!} \mathbb{I}\left(\begin{smallmatrix} \alpha & \geq & 0 \\ \beta & \geq & 0 \\ \gamma & \geq & 0 \end{smallmatrix}\right) \quad (6)$$

$$\mathbb{P}_{n,0}\left(\frac{\alpha}{\beta} \middle| \frac{x}{y}\right) = \sum_{u+v \leq x+z} \Phi_{u,v} \mathbb{P}_{n,0}\left(\beta-(x+z-\frac{\alpha-u}{\gamma-v}) \middle| \frac{x}{y}\right) \quad (7)$$

By replacing formula (6) in formula (7) we obtain:

$$\mathbb{P}_{n,0}\left(\frac{\alpha}{\beta} \middle| \frac{x}{y}\right) = \sum_{u+v \leq x+z} \Phi_{u,v} \frac{y! \bar{\sigma}^{\beta-v} \left(\frac{\sigma}{T-1}\right)^{y-\beta+v}}{(\alpha-u)! (\beta-(x+z-u-v))! (\gamma-v)!} \times \mathbb{I}\left(\begin{smallmatrix} u & \leq & \alpha-k_1 \\ x+z-u-v & \leq & \beta-(d-k_1-k_2) \\ v & \leq & \gamma \end{smallmatrix}\right) \quad (8)$$

*Proof.* We first need to determine the input and the output producing the higher ratio<sup>7</sup> for two neighboring graphs. In other words we need to find  $\mathcal{G} = \{x+1, y, z\}$ ,  $\mathcal{G}' = \{x, y, z+1\}$  and  $\mathcal{O} = \{\alpha, \beta, \gamma\}$  such that  $R_n^d\left(\frac{\alpha}{\beta} \middle| \frac{x}{y}\right) = \frac{P[\mathcal{A}(\mathcal{G}) \in \mathcal{O}]}{P[\mathcal{A}(\mathcal{G}') \in \mathcal{O}]}$  is maximum.

We have:

$$R_n^d\left(\frac{\alpha}{\beta} \middle| \frac{x}{y}\right) = \frac{\mathbb{P}_{n,d}\left(\frac{\alpha}{\beta} \middle| \frac{x+1}{y}\right)}{\mathbb{P}_{n,d}\left(\frac{\alpha}{\beta} \middle| \frac{x}{y+1}\right)}$$

We start by developing the numerator:

$$\begin{aligned} \mathbb{P}_{n,d}\left(\frac{\alpha}{\beta} \middle| \frac{x+1}{y}\right) &= \sum_{k_1+k_2=d} \mathbb{P}_{n,d}^{\text{dum}}\left(d-k_1-\frac{k_1}{k_2} \middle| \frac{\alpha-k_1}{\beta-(d-k_1-k_2)} \right) \cdot \mathbb{P}_{n,0}\left(\beta-(d-k_1-k_2) \middle| \frac{x+1}{y}\right) \\ \mathbb{P}_{n,d}\left(\frac{\alpha}{\beta} \middle| \frac{x}{y}\right) &= \sum_{k_1+k_2=d} \mathbb{P}_{n,d}^{\text{dum}}\left(d-k_1-\frac{k_1}{k_2} \middle| \frac{\alpha-k_1}{\beta-(d-k_1-k_2)} \right) \\ &\quad \cdot \left( \bar{\sigma} \mathbb{P}_{n,0}\left(\beta-(d-k_1-k_2) \middle| \frac{x}{y}\right) \right. \\ &\quad \left. + \left(\frac{\sigma}{T-1}\right) \mathbb{P}_{n,0}\left(\beta-(d-k_1-k_2)-\frac{1}{\gamma-k_2} \middle| \frac{x}{y}\right) \right. \\ &\quad \left. + \left(\frac{\sigma}{T-1}\right) \mathbb{P}_{n,0}\left(\beta-(d-k_1-k_2)-\frac{1}{\gamma-k_2-1} \middle| \frac{x}{y}\right) \right) \end{aligned}$$

We then apply the formula 8 to each  $\mathbb{P}_{n,0}$ :

$$\begin{aligned} \mathbb{P}_{n,0}\left(\beta-(d-k_1-k_2) \middle| \frac{x}{y}\right) &= \sum_{u+v \leq x+z} \frac{y! \bar{\sigma}^{\beta-(d-k_1-k_2)-v} \left(\frac{\sigma}{T-1}\right)^{y-\beta-(d-k_1-k_2)+v}}{(\alpha-k_1-1-u)! (\beta-(d-k_1-k_2)-(x+z-u-v))! (\gamma-k_2-v)!} \\ &\quad \cdot \Phi_{u,v} \cdot \mathbb{I}\left(\begin{smallmatrix} u & \leq & \alpha-k_1-1 \\ x+z-u-v & \leq & \beta-(d-k_1-k_2) \\ v & \leq & \gamma-k_2 \end{smallmatrix}\right) \end{aligned}$$

<sup>7</sup>Ignoring the symmetric case where the ratio is minimum.

$$\mathbb{P}_{n,0}\left(\beta-(d-k_1-k_2) \middle| \frac{x}{y}\right) = \sum_{u+v \leq x+z} f\left(\frac{\alpha}{\beta}, \frac{k_1}{k_2}\right) (\alpha-k_1-u) \mathbb{I}\left(\begin{smallmatrix} \alpha-k_1-u & \geq & 1 \\ 1 \end{smallmatrix}\right)$$

where:

$$f\left(\frac{\alpha}{\beta}, \frac{k_1}{k_2}\right) = \frac{y! \bar{\sigma}^{\beta-(d-k_1-k_2)-v} \left(\frac{\sigma}{T-1}\right)^{y-\beta-(d-k_1-k_2)+v}}{(\alpha-k_1-u)! (\beta-(d-k_1-k_2)-(x+z-u-v))! (\gamma-k_2-v)!} \Phi_{u,v} \cdot \mathbb{I}\left(\begin{smallmatrix} u & \leq & \alpha-k_1 \\ x+z-u-v & \leq & \beta-(d-k_1-k_2) \\ v & \leq & \gamma-k_2 \end{smallmatrix}\right)$$

In the same way we obtain for the two other  $\mathbb{P}_{n,0}$ :

$$\begin{aligned} \mathbb{P}_{n,0}\left(\beta-(d-k_1-k_2) \middle| \frac{x}{y}\right) &= \sum_{u+v \leq x+z} f\left(\frac{\alpha}{\beta}, \frac{k_1}{k_2}\right) \frac{\left(\frac{\sigma}{T-1}\right)^{\beta-(d-k_1-k_2)-(x+z-u-v)}}{\bar{\sigma}} \\ &\quad \cdot \mathbb{I}\left(\begin{smallmatrix} \beta-(d-k_1-k_2)-(x+z-u-v) & \geq & 1 \\ 1 \end{smallmatrix}\right) \end{aligned}$$

$$\mathbb{P}_{n,0}\left(\beta-(d-k_1-k_2) \middle| \frac{x}{y}\right) = \sum_{u+v \leq x+z} f\left(\frac{\alpha}{\beta}, \frac{k_1}{k_2}\right) (\gamma-k_2-v) \mathbb{I}\left(\begin{smallmatrix} \gamma-k_2-v & \geq & 1 \\ 1 \end{smallmatrix}\right)$$

By replacing in the numerator we obtain:

$$\begin{aligned} \mathbb{P}_{n,d}\left(\frac{\alpha}{\beta} \middle| \frac{x+1}{y}\right) &= \sum_{\substack{k_1+k_2 \leq d \\ u+v \leq x+z}} \Omega\left(\frac{\alpha}{\beta}, \frac{k_1}{k_2}\right) \cdot \left( \bar{\sigma} (\alpha-k_1-u) \mathbb{I}\left(\begin{smallmatrix} \alpha-k_1-u & \geq & 1 \\ 1 \end{smallmatrix}\right) \right. \\ &\quad \left. + \frac{\left(\frac{\sigma}{T-1}\right)^2 (\beta-(d-k_1-k_2)-(x+z-u-v))}{\bar{\sigma}} \right. \\ &\quad \cdot \mathbb{I}\left(\begin{smallmatrix} \beta-(d-k_1-k_2)-\frac{1}{\gamma-k_2-1}(x+z-u-v) & \geq & 1 \end{smallmatrix}\right) \\ &\quad \left. + \left(\frac{\sigma}{T-1}\right) (\gamma-k_2-v) \mathbb{I}\left(\begin{smallmatrix} \gamma-k_2-v & \geq & 1 \\ 1 \end{smallmatrix}\right) \right) \end{aligned}$$

$$\text{where } \Omega\left(\frac{\alpha}{\beta}, \frac{k_1}{k_2}\right) = \mathbb{P}_{n,d}^{\text{dum}}\left(d-k_1-\frac{k_1}{k_2} \middle| \frac{\alpha-k_1}{\beta-(d-k_1-k_2)} \right) f\left(\frac{\alpha}{\beta}, \frac{k_1}{k_2}\right).$$

With the same reasoning for the denominator, we obtain:

$$\begin{aligned} \mathbb{P}_{n,d}\left(\frac{\alpha}{\beta} \middle| \frac{x}{y+1}\right) &= \sum_{\substack{k_1+k_2 \leq d \\ u+v \leq x+z}} \Omega\left(\frac{\alpha}{\beta}, \frac{k_1}{k_2}\right) \left( \left(\frac{\sigma}{T-1}\right) (\alpha-k_1-u) \mathbb{I}\left(\begin{smallmatrix} \alpha-k_1-u & \geq & 1 \\ 1 \end{smallmatrix}\right) \right. \\ &\quad \left. + \frac{\left(\frac{\sigma}{T-1}\right)^2 (\beta-(d-k_1-k_2)-(x+z-u-v))}{\bar{\sigma}} \right. \\ &\quad \cdot \mathbb{I}\left(\begin{smallmatrix} \beta-(d-k_1-k_2)-\frac{1}{\gamma-k_2-1}(x+z-u-v) & \geq & 1 \end{smallmatrix}\right) \\ &\quad \left. + \bar{\sigma} (\gamma-k_2-v) \mathbb{I}\left(\begin{smallmatrix} \gamma-k_2-v & \geq & 1 \\ 1 \end{smallmatrix}\right) \right) \end{aligned}$$

The ratio can then be written as:

$$R_n^d\left(\frac{\alpha}{\gamma} \middle| \frac{x}{z}\right) = \frac{\sum_{\substack{k_1+k_2 \leq d \\ u+v \leq x+z}} \Omega\left(\frac{\alpha, k_1}{\beta, k_2; \gamma, k_3}\right) \cdot \left(\bar{\sigma}\chi_1 + \chi_2 + \left(\frac{\sigma}{T-1}\right)\chi_3\right)}{\sum_{\substack{k_1+k_2 \leq d \\ u+v \leq x+z}} \Omega\left(\frac{\alpha, k_1}{\beta, k_2; \gamma, k_3}\right) \cdot \left(\left(\frac{\sigma}{T-1}\right)\chi_1 + \chi_2 + \bar{\sigma}\chi_3\right)}$$

where:

$$\begin{aligned}\chi_1 &= (\alpha - k_1 - u) \mathbb{I}\left(\alpha - k_1 - u \geq 1\right) \\ \chi_2 &= \frac{\left(\frac{\sigma}{T-1}\right)^2 (\beta - (d - k_1 - k_2) - (x + z - u - v))}{\bar{\sigma}} \\ \chi_3 &= (\gamma - k_2 - v) \mathbb{I}\left(\gamma - k_2 - v \geq 1\right)\end{aligned}$$

As  $\bar{\sigma} \gg \left(\frac{\sigma}{T-1}\right)$ , to maximize the ratio, one need to maximize  $\chi_1$  and minimize  $\chi_3$ . On the one hand  $\chi_1$  is maximal when  $\alpha$  is maximal (i.e.  $\alpha = n$ , as the maximum one can send to the same target with algorithm  $\bar{\mathcal{A}}_{\sigma}^{n,d}$  is  $n$ ). On the other hand,  $\chi_3$  is minimal when  $\gamma = 0$ . Thus, the output producing the higher ratio is  $\mathcal{O} = \{n, d, 0\}$ .

To further increase the ratio, we need to minimize  $k_1$  and  $u$ . The first one depends on  $d$ , a fixed parameter of the algorithm we cannot change. The later one,  $u$ , is varying from 0 to  $x + z$  and takes its minimal value when  $x + z = 0$ . We deduce from this that the two inputs producing the higher ratio are  $\mathcal{G} = \{1, n - 1, 0\}$  and  $\mathcal{G}' = \{0, n - 1, 1\}$ .

By replacing the new indices in the ratio we obtain:

$$R_n^d\left(\frac{n}{0} \middle| \frac{0}{n-1}\right) = \frac{\sum_{k=0}^d \binom{d}{k} \binom{n-1}{k} \bar{\sigma}^k \left(\frac{\sigma}{T-1}\right)^{n-k-1} \left(\bar{\sigma} + k \cdot \frac{\left(\frac{\sigma}{T-1}\right)^2}{\bar{\sigma}}\right)}{\sum_{k=0}^d \binom{d}{k} \binom{n-1}{k} \bar{\sigma}^k \left(\frac{\sigma}{T-1}\right)^{n-k-1} \left(\frac{\sigma}{T-1} + k \cdot \frac{\left(\frac{\sigma}{T-1}\right)^2}{\bar{\sigma}}\right)} = e$$

which leads to the result.  $\square$

#### APPENDIX D DETAILS OF NUMERICAL SIMULATION

The results of our numerical simulation (Figure 6) were obtained as follows.

First, given a number of target  $T$ , we fix an input computation graph with  $n$  messages drawn from uniform and skewed target distributions. Then, two neighboring communication graphs  $\mathcal{G}_1$  and  $\mathcal{G}_2$  are generated by randomly changing the target of one message.

Second, for fixed parameters  $(\sigma, d$  and  $n)$ , we run our algorithm many times on the neighboring communication graphs with different random seeds. For each run and each output  $\mathcal{O}$  encountered, we count how many times  $\mathcal{O}$  appeared for each communication graph:

$$\begin{aligned}c_1(\mathcal{O}) &= \{\# \text{ times } \mathcal{O} \text{ occurred when the input was } \mathcal{G}_1\}, \\ c_2(\mathcal{O}) &= \{\# \text{ times } \mathcal{O} \text{ occurred when the input was } \mathcal{G}_2\},\end{aligned}$$

which correspond to the red and blue bars shown in Figure 6. We then compute an estimate of the log-ratio of the probabilities for each output  $\mathcal{O}$  as:

$$r(\mathcal{O}) = \ln \left( \max \left\{ \frac{c_1(\mathcal{O})}{c_2(\mathcal{O})}, \frac{c_2(\mathcal{O})}{c_1(\mathcal{O})} \right\} \right),$$

which correspond to the dotted green line in Figure 6.

The total number of runs are set such that nearly all possible outputs are drawn at least once. To make sure that we did enough runs, we applied the Capture-recapture [60] counting technique used in biology to estimate the size of populations of animals. More precisely, we follow a two-step approach to estimate the percentage of unseen outputs. First, we divide the number of runs into different batches. We execute a batch and record all the different output drawn. Second, we draw another batch and count the number of new outputs (i.e., not seen in the first batch). The percentage of new outputs in the second batch represents the percentage of all possible outputs that have not been seen in any batch.

Interestingly, we observed that the choice of input has a negligible impact on the results: although the probability of individual outputs obviously depend on the input, the overall shape of the output probability distribution (and thus the results in Figure 6) remains essentially the same.

#### APPENDIX E TECHNICAL DETAILS AND PROOFS FOR THE RESULTS OF SECTION IV-C

In this section, we provide a detailed exposition of our analysis leading to Theorem 4 in the main text. We first introduce some key technical concepts in Section E-A. In Section E-B, to keep our analysis as general as possible (see Remark 2), we first prove  $(\epsilon, \delta)$ -DP results for a generic local randomizer  $\mathcal{R}$ . Finally, in Section E-C, we apply our general result to our specific context.

For notational convenience, for any integer  $n \geq 1$ , we will denote the set  $\{1, \dots, n\}$  by  $[n]$ .  $\square$

##### A. Key Concepts

In this section, we review some key concepts from Balle et al. [4] that we need to prove our results.

a) *Decomposition of local randomizers.*: Let  $\mathcal{X}$  and  $\mathcal{Y}$  be some input and output domains respectively. Let  $\mathcal{R} : \mathcal{X} \rightarrow \mathcal{Y}$  be a *local randomizer* taking an input  $x \in \mathcal{X}$  and returning a randomized output  $y \in \mathcal{Y}$ . The *total variation similarity*  $\gamma_{\mathcal{R}}$  of a local randomizer  $\mathcal{R}$  measures the probability that  $\mathcal{R}$  produces an output which is independent from its input. When this happens, the output is sampled from some distribution  $\omega_{\mathcal{R}}$ , which is called the *blanket distribution* of  $\mathcal{R}$ . When it is clear from the context, we drop the subscript and simply write  $\gamma$  and  $\omega$ .

We will leverage a decomposition of  $\mathcal{R}$  as a mixture between an input-dependent and input-independent mechanism. Specifically, denoting by  $\mu_x$  the output distribution of  $\mathcal{R}(x)$ , we write  $\mu_x = (1 - \gamma)v_x + \gamma\omega$ . For a particular  $\mathcal{R} : \mathcal{X} \rightarrow \mathcal{Y}$ , the largest possible  $\gamma$  is given by  $\gamma = \int \inf_x \mu_x(y) dy$  and

the corresponding blanket distribution  $\omega$  is given by  $\omega(y) = \inf_x \mu_x(y)/\gamma$ . Balle et al. [4] show that  $\gamma \geq e^{-\epsilon_0}$  for any  $\epsilon_0$ -DP local randomizer, but it is possible to compute the exact value of  $\gamma$  for common local randomizers (see Lemma 5.1 in [4]).

We can illustrate these concepts on  $\mathcal{R}_\sigma$ , the local randomizer used in our algorithm. Since  $\mathcal{R}_\sigma$  only randomizes the target, we can abstract away the source node and we have  $\mathcal{X} = \mathcal{Y} = \mathcal{T}$ ,  $\gamma_{\mathcal{R}_\sigma} = \sigma$ ,  $v_{\mathcal{R}_\sigma, t}(t') = \mathbb{I}[t = t']$  and  $\omega_{\mathcal{R}_\sigma}(t') = 1/k$  for all  $t' \in \mathcal{T}$ .

b) *Hockey-stick divergence.*: Differential privacy can be conveniently expressed as a divergence between distributions. Divergences come with known results and properties that provide useful technical tools to derive differential privacy guarantees. Below, we will use the characterization of  $(\epsilon, \delta)$ -DP based on the so-called *hockey-stick divergence*. Precisely, the hockey-stick divergence of order  $e^\epsilon$  between distributions  $\mu$  and  $\mu'$  is defined as:

$$\mathbb{D}_{e^\epsilon}(\mu || \mu') = \int [\mu(y) - e^\epsilon \mu'(y)]_+ dy,$$

where  $[\cdot]_+ = \max(0, \cdot)$ . The following lemma from [4] shows the direct connection to  $(\epsilon, \delta)$ -DP.

**Lemma 1.** *An algorithm  $\mathcal{A} : \mathcal{X}^n \rightarrow \mathcal{Y}^m$  is  $(\epsilon, \delta)$ -DP if and only if  $\mathbb{D}_{e^\epsilon}(\mathcal{A}(\mathcal{D}) || \mathcal{A}(\mathcal{D}')) \leq \delta$  for any neighboring datasets  $\mathcal{D} = \{x_1, \dots, x_{n-1}, x_n\}$  and  $\mathcal{D}' = \{x_1, \dots, x_{n-1}, x'_n\}$ .*

#### B. Privacy Guarantee for a Generic Local Randomizer

Let  $\epsilon, \epsilon_0 \geq 0$ ,  $\delta \in (0, 1)$  and  $d \in \mathbb{N}$ . In this section, we prove an  $(\epsilon, \delta)$ -DP result for algorithms  $\mathcal{A} : \mathcal{X}^n \rightarrow \mathcal{Y}^{n+d}$  of the form  $\mathcal{A} = \mathcal{S}_{\mathcal{R}, d} \circ \mathcal{R}^n$ , where:

- $\mathcal{R}^n : \mathcal{X}^n \rightarrow \mathcal{Y}^n$  is such that

$$\mathcal{R}^n(x_1, \dots, x_n) = (\mathcal{R}(x_1), \dots, \mathcal{R}(x_n))$$

where  $\mathcal{R} : \mathcal{X} \rightarrow \mathcal{Y}$  is an arbitrary local randomizer satisfying  $\epsilon_0$ -DP.

- $\mathcal{S}_{\mathcal{R}, d} : \mathcal{Y}^n \rightarrow \mathcal{Y}^{n+d}$  randomly samples  $d$  “dummy” messages from the blanket distribution  $\omega_{\mathcal{R}}$  and shuffles (i.e., applies a random permutation to it) the multiset composed of the  $n$  input messages and the  $d$  dummy messages.

Note that  $\mathcal{S}_{\mathcal{R}, 0}$  (no dummy) corresponds to a standard shuffler: this is the setting covered by recent results on privacy amplification by shuffling, in particular those of [4]. The purpose of this section is to extend these results to account for the use of dummy messages, i.e., when  $d > 0$ .

a) *Step 1: Bounding the divergence in terms of i.i.d. random variables.*: Let  $\mathcal{D} = \{x_1, \dots, x_{n-1}, x_n\}$  and  $\mathcal{D}' = \{x_1, \dots, x_{n-1}, x'_n\}$  be two neighboring datasets that differ only in their last points  $x_n$  and  $x'_n$ . The key technical step of the proof is to bound the divergence  $\mathbb{D}_{e^\epsilon}(\mathcal{A}(\mathcal{D}) || \mathcal{A}(\mathcal{D}'))$  in terms of a sum of i.i.d. realizations of a “privacy amplification” random variable  $L_{\epsilon}^{x_n, x'_n}$  defined as:

$$L_{\epsilon}^{x_n, x'_n} = \frac{\mu_{x_n}(W) - e^\epsilon \mu_{x'_n}(W)}{\omega(W)}, \quad (9)$$

where  $W \sim \omega$ . We have the following result, which is the analog to Lemma 5.3 of [4] for the case with dummies.

**Lemma 2.** *Let  $\epsilon > 0$  and let  $\mathcal{D} = \{x_1, \dots, x_{n-1}, x_n\}$  and  $\mathcal{D}' = \{x_1, \dots, x_{n-1}, x'_n\}$  two neighboring datasets with  $x_n \neq x'_n$ . We have:*

$$\begin{aligned} \mathbb{D}_{e^\epsilon}(\mathcal{A}(\mathcal{D}) || \mathcal{A}(\mathcal{D}')) &\leq \sum_{m=0}^{n-1} \frac{1}{m+d+1} C_m^{n-1} \gamma^m (1-\gamma)^{n-1-m} \mathbb{E} \left[ \sum_{i=1}^{m+d+1} L_i \right]_+ \\ &= \frac{1}{\gamma n} \sum_{m=1}^n \frac{m}{m+d} C_m^n \gamma^m (1-\gamma)^{n-m} \mathbb{E} \left[ \sum_{i=1}^{m+d} L_i \right]_+, \end{aligned}$$

where  $L_1, \dots, L_{m+d}$  are i.i.d. copies of  $L_{\epsilon}^{x_n, x'_n}$ .

*Proof.* Recall that  $\mathcal{A} = \mathcal{S}_{\mathcal{R}, d} \circ \mathcal{R}^n$ . For a fixed input dataset  $\mathcal{D} = \{x_1, \dots, x_{n-1}, x_n\}$ , we define the random variable  $Y_i \sim \mu_{x_i}$  for  $i \in [n]$ , where  $\mu_x = (1-\gamma)v_x + \gamma\omega$  is the output distribution of  $\mathcal{R}(x)$  as defined in Section E-A. For  $i \in [d]$ , we also define the random variable  $Z_i \sim \omega$ . Using these notations, the output of  $\mathcal{A}(\mathcal{D})$  can be seen as a realization of the random multiset  $\mathcal{O} = \{Y_1, \dots, Y_n, Z_1, \dots, Z_d\} \in \mathbb{N}_{n+d}^{\mathcal{Y}}$ , where  $\mathbb{N}_{n+d}^{\mathcal{Y}}$  denotes the collection of all multisets of size  $n+d$  of elements of  $\mathcal{Y}$ . Similarly, for a neighboring dataset  $\mathcal{D}' = \{x_1, \dots, x_{n-1}, x'_n\}$ , the output of  $\mathcal{A}(\mathcal{D}')$  is a realization of the random multiset  $\mathcal{O}' = \{Y_1, \dots, Y'_n, Z_1, \dots, Z_d\}$ . Our goal is thus to bound  $\mathbb{D}_{e^\epsilon}(\mathcal{O} || \mathcal{O}')$ , where we use a slight abuse of notation by applying the divergence to random variables rather than distributions.

To exploit the mixture decomposition  $\mu_x$ , we define additional random variables. Let  $V_i \sim v_{x_i}$  and  $W_i \sim \omega$  for  $i \in [n-1]$ . Hence we have:

$$Y_i = \begin{cases} V_i & \text{with probability } 1-\gamma, \\ W_i & \text{with probability } \gamma. \end{cases}$$

Finally, we define  $\mathcal{B} \subseteq [n-1]$  to be the random subset of inputs among the first  $n-1$  who sampled from the blanket, and let  $\bar{\mathcal{B}} = [n-1] \setminus \mathcal{B}$ . Note that for any  $B \subseteq [n-1]$  we have  $P[\mathcal{B} = B] = \gamma^{|B|} (1-\gamma)^{n-1-|B|}$ . With these notations, conditioned on a particular  $B$ , we have

$$\mathcal{O} | \{\mathcal{B} = B\} = \mathcal{W}_B \cup \mathcal{V}_{\bar{B}} \cup \mathcal{Z}_d \cup \{Y_n\},$$

where  $\mathcal{W}_B = \{W_i\}_{i \in B}$ ,  $\mathcal{V}_{\bar{B}} = \{V_i\}_{i \in [n-1] \setminus B}$  and  $\mathcal{Z}_d = \{Z_i\}_{i=1}^d$ .

By standard properties of the hockey-stick divergence (see Lemma A.1 in [4]), we have:

$$\mathbb{D}_{e^\epsilon}(\mathcal{O} || \mathcal{O}') \leq \sum_{B \subseteq [n-1]} \gamma^{|B|} (1-\gamma)^{n-1-|B|} \mathbb{D}_{B,d}^{(1)} \quad (10)$$

where  $\mathbb{D}_{B,d}^{(1)} = \mathbb{D}_{e^\epsilon}(\mathcal{W}_B \cup \mathcal{V}_{\bar{B}} \cup \mathcal{Z}_d \cup \{Y_n\} || \mathcal{W}_B \cup \mathcal{V}_{\bar{B}} \cup \mathcal{Z}_d \cup \{Y'_n\})$ .

By applying Lemma A.2 from [4], we further show that we can ignore the contributions of the first  $n-1$  inputs who did not sample from the blanket. Precisely:

$$\mathbb{D}_{B,d}^{(1)} \leq \mathbb{D}_{e^\epsilon}(\mathcal{W}_B \cup \mathcal{Z}_d \cup \{Y_n\} || \mathcal{W}_B \cup \mathcal{Z}_d \cup \{Y'_n\}). \quad (11)$$



Since the  $W_i$ 's are i.i.d., the distribution of  $\mathcal{W}_B$  depends on  $B$  only through its cardinality  $m = |B|$ . We thus define  $\mathcal{W}_m = \{W_1, \dots, W_m\}$  for any  $m \in [n-1]$ , with  $\mathcal{W}_0 = \emptyset$ . Rewriting (10) and (11), we have shown that:

$$\mathbb{D}_{e^\epsilon}(\mathcal{O}||\mathcal{O}') \leq \sum_{m=0}^{n-1} C_m^{n-1} \gamma^m (1-\gamma)^{n-1-m} \mathbb{D}_{B,d}^{(2)} \quad (12)$$

where  $\mathbb{D}_{B,d}^{(2)} = \mathbb{D}_{e^\epsilon}(\mathcal{W}_m \cup \mathcal{Z}_d \cup \{Y_n\} || \mathcal{W}_m \cup \mathcal{Z}_d \cup \{Y'_n\})$ .

We now upper bound the right-hand side of (12) in terms of the privacy amplification variables  $L_1, \dots, L_{m+d}$  arising from the  $m$  inputs who sampled from the blanket and the  $d$  dummy messages.

Let  $y \in \mathcal{Y}^{m+d}$  be a tuple of elements from  $\mathcal{Y}$  and  $Y \in \mathbb{N}_{m+d}^{\mathcal{Y}}$  be the corresponding multiset. We have:

$$\begin{aligned} & P[\mathcal{W}_{m-1} \cup \mathcal{Z}_d \cup \{Y_n\} = Y] \\ &= \frac{1}{(m+d)!} \sum_{\tau} P[(W_1, \dots, W_{m-1}, Y_n, Z_1, \dots, Z_d) = y_{\tau}], \end{aligned}$$

where  $\tau$  ranges over all permutations of  $\{1, \dots, m+d\}$  and we write  $y_{\tau} = (y_{\tau(1)}, \dots, y_{\tau(m+d)})$ . Since  $W_i \sim \omega$ ,  $Y_n \sim \mu_{x_n}$  and  $Z \sim \omega$ , we have:

$$\begin{aligned} & P[(W_1, \dots, W_{m-1}, Y_n, Z_1, \dots, Z_d) = y_{\tau}] \\ &= \omega(y_{\tau(1)}) \dots \omega(y_{\tau(m-1)}) \mu_{x_n}(y_{\tau(m)}) \omega(y_{\tau(m+1)}) \dots \omega(y_{\tau(m+d)}) \end{aligned}$$

Summing this expression over all permutations  $\tau$  and factoring out  $P[\mathcal{W}_m \cup \mathcal{Z}_d = Y]$  gives:

$$\begin{aligned} & \frac{1}{(m+d)!} \sum_{\tau} (\omega(y_{\tau(1)}) \dots \omega(y_{\tau(m-1)}) \mu_{x_n}(y_{\tau(m)}) \\ & \quad \times \omega(y_{\tau(m+1)}) \dots \omega(y_{\tau(m+d)})) \\ &= \left( \prod_{i=1}^{m+d} \omega(y_i) \right) \frac{1}{m+d} \sum_{i=1}^{m+d} \frac{\mu_{x_n}(y_i)}{\omega(y_i)} \\ &= P[\mathcal{W}_m \cup \mathcal{Z}_d = Y] \frac{1}{m+d} \sum_{i=1}^{m+d} \frac{\mu_{x_n}(y_i)}{\omega(y_i)}. \end{aligned}$$

Plugging this in the definition of  $\mathbb{D}_{e^\epsilon}$ , we get:

$$\begin{aligned} & \mathbb{D}_{e^\epsilon}(\mathcal{W}_{m-1} \cup \mathcal{Z}_d \cup \{Y_n\} || \mathcal{W}_{m-1} \cup \mathcal{Z}_d \cup \{Y'_n\}) \\ &= \int_{\mathbb{N}_m^{\mathcal{Y}}} [P[\mathcal{W}_{m-1} \cup \mathcal{Z}_d \cup \{Y_n\} = Y] \\ & \quad - e^\epsilon P[\mathcal{W}_{m-1} \cup \mathcal{Z}_d \cup \{Y'_n\} = Y]]_+ dY \\ &= \int_{\mathbb{N}_m^{\mathcal{Y}}} P[\mathcal{W}_m \cup \mathcal{Z}_d = Y] \left[ \frac{1}{m+d} \sum_{i=1}^{m+d} \frac{\mu_{x_n}(y_i) - e^\epsilon \mu_{x'_n}(y_i)}{\omega(y_i)} \right]_+ dY \\ &= \mathbb{E} \left[ \frac{1}{m+d} \sum_{i=1}^{m+d} \frac{\mu_{x_n}(y_i) - e^\epsilon \mu_{x'_n}(y_i)}{\omega(y_i)} \right]_+ = \mathbb{E} \left[ \frac{1}{m+d} \sum_{i=1}^{m+d} L_i \right]_+ \end{aligned}$$

Plugging this into (12) completes the proof.  $\square$

*b) Step 2: Bounding the sum of privacy amplification variables.* To control the term  $\mathbb{E}[\sum_{i=1}^{m+d} L_i]_+$  in Lemma 2, we need to resort to concentration inequalities for sums of i.i.d. random variables. We can trivially adapt Lemma 5.5 (based on Hoeffding's inequality) and Lemma 5.6 (based on Bennett's inequality) from [4] by replacing  $m$  by  $m+d$ .

**Lemma 3.** *Let  $L_1, \dots, L_{m+d}$  be i.i.d. bounded random variables with  $\mathbb{E}[L_i] = -a \leq 0$ . Suppose that  $b_- \leq L_i \leq b_+$  and let  $b = b_+ - b_-$ . Then, by Hoeffding's inequality, the following holds:*

$$\mathbb{E} \left[ \sum_{i=1}^{m+d} L_i \right]_+ \leq \frac{b^2}{4a} e^{-\frac{2(m+d)a^2}{b^2}}.$$

*If furthermore we have  $\mathbb{E}[L_i^2] \leq c$ , then by Bennett's inequality:*

$$\mathbb{E} \left[ \sum_{i=1}^{m+d} L_i \right]_+ \leq \frac{b_+}{a(m+d) \log(1 + \frac{ab_+}{c})} e^{-\frac{(m+d)c}{b_+^2} \phi(\frac{ab_+}{c})},$$

where  $\phi(u) = (1+u) \log(1+u) - u$ .

Bounding the sum of privacy amplification random variables with the above lemma requires the knowledge of their expected value as well as bounds on the values they can take. Bennett's inequality further requires a bound on the second moment: this can lead to a tighter (albeit more complex) bound compared to Hoeffding's. Balle et al. [4] provide such bounds that hold for any local randomizer  $\mathcal{R}$  satisfying  $\epsilon_0$ -DP, as we recall below.

**Lemma 4.** *Let  $\mathcal{R} : \mathcal{X} \rightarrow \mathcal{Y}$  be an  $\epsilon_0$ -DP local randomizer with total variation similarity  $\gamma$ . For any  $\epsilon \geq 0$  and any  $x, x' \in \mathcal{X}$ , the privacy amplification variable  $L = L_{\epsilon}^{x, x'}$  satisfies the following:*

- 1)  $\mathbb{E}[L] = 1 - e^\epsilon$ ,
- 2)  $\gamma e^{-\epsilon_0} (1 - e^{\epsilon+2\epsilon_0}) \leq L \leq \gamma e^{\epsilon_0} (1 - e^{\epsilon-2\epsilon_0})$ ,
- 3)  $\mathbb{E}[L^2] \leq \gamma e^{\epsilon_0} (e^{2\epsilon} + 1) - 2\gamma^2 e^{\epsilon-2\epsilon_0}$ .

*c) Step 3: Putting everything together.* Based on the intermediate results above, we can obtain an  $(\epsilon, \delta)$ -DP guarantee that holds for any  $\epsilon_0$ -DP local randomizer. We illustrate this using the simpler Hoeffding's inequality in Lemma 3.

**Theorem 5.** *Let  $\mathcal{R} : \mathcal{X} \rightarrow \mathcal{Y}$  be an  $\epsilon_0$ -DP local randomizer with total variation similarity  $\gamma$ , and  $\delta \geq 0$ . The algorithm  $\mathcal{A} = \mathcal{S}_{\mathcal{R}, \delta} \circ \mathcal{R}^n$  is  $(\epsilon, \delta)$ -DP for any  $\epsilon$  and  $\delta$  satisfying*

$$\frac{1}{\gamma n} \sum_{m=1}^n \frac{m}{m+d} C_m^m \gamma^m (1-\gamma)^{n-m} \frac{b^2}{4a} e^{-\frac{2(m+d)a^2}{b^2}} \leq \delta,$$

where  $a = 1 - e^\epsilon$  and  $b = \gamma(1 + e^\epsilon)(e^{\epsilon_0} - e^{-\epsilon_0})$ .

Although the above theorem does not give a simple expression for  $\epsilon$  and  $\delta$ , it can be easily evaluated numerically, for instance to identify the lowest achievable  $\epsilon$  given the  $\epsilon_0$  and  $\delta$  of  $\mathcal{R}$  and the desired  $\delta$ .

*C. Application to our Setting*

We can now apply our previous privacy guarantees, which hold for an arbitrary randomizer, to our specific context. Since

our local randomizer  $\mathcal{R}_\sigma$  only randomizes the target, for simplicity of notations we abstract away the source node and consider that  $\mathcal{R}_\sigma$  operates on  $\mathcal{X} = \mathcal{T}$  and returns an element of  $\mathcal{Y} = \mathcal{T}$ . We have  $\gamma_{\mathcal{R}_\sigma} = \sigma$ ,  $v_{\mathcal{R}_\sigma, t}(t') = \mathbb{I}[t = t']$  and  $\omega_{\mathcal{R}_\sigma}(t') = 1/k$  for all  $t' \in \mathcal{T}$ .  $\mathcal{R}_\sigma$  is in fact equivalent to  $T$ -ary randomized response [32], and we can refine the results of Lemma 4. This is shown in the following lemma, adapted from [4].

**Lemma 5.** *For any  $\epsilon > 0$  and  $t, t' \in \mathcal{T}$ , the privacy amplification variable  $L = L_\epsilon^{t, t'}$  satisfies the following:*

- $-(1 - \sigma)Te^\epsilon + \sigma(1 - e^\epsilon) \leq L \leq (1 - \sigma)T - \sigma(1 - e^\epsilon)$ ,
- $\mathbb{E}[L^2] = \sigma(2 - \sigma)(1 - e^\epsilon)^2 + (1 - \sigma)^2T(1 + e^{2\epsilon})$ .

We can use the results of Lemma 5 instead of the generic ones from Lemma 4 to obtain tighter privacy guarantees that are specific to randomized response. Doing this with Hoeffding's inequality and combining with Lemma 2 gives the statement for  $\sigma > 0$  in Theorem 4.

*a) Special case where source nodes do not sample ( $\sigma = 0$ ):* Interestingly, when  $\sigma = 0$  (i.e., source nodes never sample and always send their true message to the scrambler), the analysis is still valid and we can get  $(\epsilon, \delta)$ -DP guarantees that rely entirely on the dummies added by the scrambler. In this case, the set  $B$  in the proof of Lemma 2 (the set of source nodes who sample from the blanket) is always empty and therefore the result of Lemma 2 simplifies to:

$$\mathbb{D}_{e^\epsilon}(\mathcal{A}(\mathcal{D}) || \mathcal{A}(\mathcal{D}')) \leq \frac{1}{d+1} \mathbb{E} \left[ \sum_{i=1}^{d+1} L_i \right]_+.$$

The bounds in Lemma 5 also simplify when  $\sigma = 0$ , namely:  $-Te^\epsilon \leq L \leq T$  and  $\mathbb{E}[L^2] = T(1 + e^{2\epsilon})$ . We can thus apply Hoeffding's or Bennett's inequalities with the above values to bound the sum of i.i.d. variables as in Lemma 3 using  $d+1$  instead of  $m+d$ , and get privacy guarantees for this special case as well. Using Hoeffding's gives the statement for  $\sigma = 0$  in Theorem 4.