Local Pan-Privacy for Federated Analytics

Vitaly Feldman [*] [1] **Audra McMillan** [*] [1] **Guy N. Rothblum** [*] [1] **Kunal Talwar** [*] [1]

Abstract

Pan-privacy was proposed by (Dwork et al., 2010) as an approach to designing a private analytics system that retains its privacy properties in the face of intrusions that expose the system's internal state. Motivated by federated telemetry applications, we study *local pan-privacy*, where privacy should be retained under repeated unannounced intrusions *on the local state*. We consider the problem of monitoring the count of an event in a federated system, where event occurrences on a local device should be hidden even from an intruder on that device. We show that under reasonable constraints, the goal of providing information-theoretic differential privacy under intrusion is incompatible with collecting telemetry information. We then show that this problem can be solved in a scalable way using standard cryptographic primitives.

1. Introduction

Private federated telemetry systems allow for collection of aggregate statistics from a population, while ensuring a strong privacy guarantee for individuals. For example, private federated learning and statistics have been used to collect webpage and search engine popularities from Chrome browsers (Erlingsson et al., 2014), learn popular emojis (Apple's Differential Privacy Team, 2017), collect Covid epedimiological metrics (Apple and Google, 2021), collect browser performance metrics (Helmer et al., 2018), collect operating system telemetry (Ding et al., 2017), and train keyboard models (Xu et al., 2023; Zhang et al., 2023).

These systems typically rely on the client device storing information about usage on device, and periodically, at appropriate times, taking part in a protocol that computes an aggregate. These protocols use encryption to protect this data from an eavesdropper. In some cases, additional cryptographic protocols are used to protect the individual contributions from the server computing the aggregate (Bonawitz et al., 2017; Corrigan-Gibbs and Boneh, 2017). Finally, the aggregate itself protects individual contributions by noise addition to provide a differential privacy guarantee.

In some applications, one may want to additionally protect against an attacker that has access to the device. For example, a computer in a public library may be accessed by one user, and later by another. In such a case, we would want to ensure that the second user cannot learn about the activity of the first user by inspecting the internal state of the device. In this work, we initiate the study of estimating population statistics while ensuring privacy against an on-device intruder. We call this model *local pan-privacy*, as it aims to protect against an intrusion at the local device, much as pan-privacy protects against an intrusion at the server.

We study locally pan-private algorithms for some fundamental statistical tasks in a simple streaming setting. There are n devices in total. For each device, at each time step, an event of interest, such as an application crash while using a feature, may or may not occur. Thus, each device's input is a sequence of T bits. The device can communicate with the server at the end of the T time steps, and we require that the protocol retains its privacy properties in the face of multiple intrusions on device (c.f. Remark 4). The first statistic we study is CountNonZero, which counts the number of devices on which the event occurred in at least one of the T time steps. In the standard local privacy model, this task can be accomplished by each device sending a randomized response of a bit corresponding to whether on not the event occurred at any of the T steps. Such a protocol can also lead to near-optimal central privacy guarantees via shuffling or aggregation (Feldman et al., 2020), when the responses are aggregated by a trusted server, or by a secure aggregation system. In this work, we focus on the setting where we either have a trusted server, or where the secure aggregation system uses a two-server architecture as in PRIO (Corrigan-Gibbs and Boneh, 2017).

We also study two additional statistics of the event count distribution: the *mean* number of occurrences per device, and a *histogram* of the number of occurrences, appropriately bucketed. As with the CountNonZero, these tasks also admit simple and commonly-used algorithms in the local privacy model, based on the Laplace mechanism (Dwork

[*]Equal contribution [1]Apple, Cupertino, CA, USA. Correspondence to: Kunal Talwar <ktalwar@apple.com>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

et al., 2006) and on RAPPOR (Erlingsson et al., 2014) respectively. We remark that these simple tasks underpin a large number of private federated statistics, and can enable a much richer set of data science tasks. For example, Zhu et al. (2020); Chadha et al. (2024) use histograms over small known domains to discover heavy hitters over large domains.

We show that local pan-privacy is severely limited if we insist on providing (information-theoretic) differential privacy in the face of intrusions. Our lower bound shows that for the COUNTNONZERO task, the error of any algorithm must be $\sqrt{T}$ times larger than that needed for local differential privacy alone. This lower bound holds even when the event occurs at most once on each device.

Theorem 1 (Informal version of Theorem 9)**.** *Any locally pan-private algorithm (for $\varepsilon = 1$) for* COUNTNONZERO *on n devices, for large enough T, must incur additive error $\Omega(\sqrt{nT})$, even though a local DP algorithm can estimate* COUNTNONZERO *with additive error $\Omega(\sqrt{n})$.*

We then show that under standard cryptographic assumptions, local pan-privacy can be ensured without this overhead. We present algorithms for all of the aforementioned problems, for both the single- and the two-server models, showing that local pan-privacy comes at *no* additional cost in the privacy-utility trade-off. Our protocols need a public-key encryption scheme, with a few additional properties that are satisfied by commonly-used encryption schemes. We note that public-key encryption is already used to protect the communication from a network intrusion, so local pan-privacy does not increase the complexity of the required assumptions. While we can define local pan-privacy broadly as a computational differential privacy guarantee against a local intruder, our protocols actually provide a stronger semantic security guarantee.

Theorem 2 (Informal version of Theorem 10)**.** *Suppose that we have a public-key encryption scheme that is rerandomizable. Then there is a streaming algorithm for* COUNTNONZERO *in the single-server and in the two-server model with the following properties.*

- *The on-device algorithm satisfies computational local pan-privacy; the on-device sequence of states on any pair of input streams are computationally indistinguishable.*
- *The on-device state consists of $O(1)$ ciphertexts, and the device sends one message consisting of $O(1)$ ciphertexts.*
- *In the local and aggregator model of differential privacy, the algorithm achieves privacy-utility trade-offs that are within constant factors of algorithms without the local pan-privacy constraint.*

In a rerandomizable encryption scheme, a ciphertext can be "rerandomized" (using only the public key) to create a new ciphertext encrypting the same plaintext, which is indistinguishable from a fresh encryption of a different plaintext, see Definition 2.8. A similar result to Theorem 2 holds for building approximate histograms, as well as for estimating the mean number of events. The latter requires slightly stronger assumptions on the encryption scheme (namely, the scheme needs to be additively homomorphic).

At a high level, this is achieved by maintaining a state on device that contains information about the stream so far, but in encrypted form. Since the local device does not have the private key, this ensures that the on-device state contains no useful information for an adversary that does not collude with the server. The challenge then is to maintain the state under updates to the stream, and we show that for our tasks of interest, this is feasible. We remark that a public-key fully homomorphic encryption (FHE) scheme can be used to achieve computational local pan-privacy, as any algorithm for the on-device computation can be made locally pan-private by keeping the state encrypted at all times and operating on the encrypted state. However, this is overkill: FHE schemes incur a large space and time overhead, and in this work we strive to rely on more minimal and more efficient cryptographic primitives.

We provide privacy at the level of the whole device, and not just at the event level. Thus a user may use the shared device multiple times during our collection period, and our algorithms will protect all of their interactions. Our model allows for continuous intrusion: the adversary can see the state of the device before and after each update. In our example of a shared device, this means that the attacker can see the memory contents of the device before and after a user is using it (but not while they are using it). Thus from our point of view, the updates to the state are atomic. Finally, we show that the existence of a public-key encryption scheme is necessary for the existence of an accurate locally pan-private algorithm for COUNTNONZERO.

Theorem 3 (Informal version of Theorem 6.1)**.** *Suppose that we have a locally pan-private algorithm for* COUNTNONZERO *that has additive error less than $n/4$ with high probability. Then we can build a public key encryption scheme.*

1.1. Related Work

Pan-privacy (Dwork et al., 2010) was proposed as an approach to designing data analysis algorithms that do not maintain a disclosive internal state. In particular, they maintain privacy even under intrusions. This notion was studied in several subsequent works, e.g. (Mir et al., 2011). The goal of maintaining privacy in the presence of potential intrusions on a user's device is common to several settings.

Examples include web browsing, where several popular browsers offer a mode where browsing history is not stored on device and chat apps where messages are ephemeral. In many situations, it may be undesirable to keep any information from events in such settings. In others, statistical information about events such as application crashes may be useful to improve the user experience. The notion of history independence (Micciancio, 1997; Naor and Teague, 2001) shares similar goals of ensuring that the memory representation of a data structure does not leak information about the history of interactions with a data structure. Oblivious RAM algorithms (Goldreich and Ostrovsky, 1996) share a similar goal of ensuring that the memory access pattern of an algorithm does not leak information about the inputs it is being run on. Forward Secrecy (Günther, 1990) addresses a similar goal of protecting old communications from future intrusions.

Organization:

We present some preliminaries and definitions in Section 2. Section 3 presents our lower bound for the information-theoretic privacy case. We present algorithms in the single- and two-server settings in Section 4 and Section 5 respectively. Section 6 shows that public-key cryptography is needed, and we conclude with some open problems in Section 7. For lack of space, some of the results, and some proofs are deferred to supplementary material.

2. Definitions and Preliminaries

We first recall the definition of differential privacy.

Definition 2.1 ((ε, δ)-Indistinguishability)**.** *We say that two random variables Y and Y' on a finite set R are (ε, δ) indistinguishable if for for every $S \subseteq R$,*

$$\Pr[Y \in S] \leq e^{\varepsilon} \Pr[Y' \in S] + \delta,$$
$$\text{and} \quad \Pr[Y' \in S] \leq e^{\varepsilon} \Pr[Y \in S] + \delta.$$

When two r.v.'s are $(\varepsilon, 0)$-indistinguishabile, we will often call them ε-indistinguishable.

Definition 2.2 (Differential Privacy)**.** *A mechanism $M : D^n \to R$ is (ε, δ)-differentially private if for any pair of neighboring datasets $\mathbf{x}, \mathbf{x}'$, the random variables $M(\mathbf{x})$ and $M(\mathbf{x}')$ are (ε, δ)-indistinguishable.*

Here neighboring datasets typically means datasets that differ in the input of one user.

We consider a data stream $\mathbf{x} = x_1, \ldots, x_T$, where each $x_i \in D$. In this work we will be concerned with the case that $D = \{0, 1\}$. A streaming algorithm is defined by a set of functions. The function **Initialize** (usually left implicit) sets up the state on device, including the state s_0, and potentially some state on the server to allow coordination between the two. We have a sequence of functions, where $\textbf{State}_t : D \times \mathcal{S} \to \mathcal{S}$ is a (possibly randomized) function that takes the input at time t and the current state s_{t-1}, and maps to the new state. Given an input stream $x_1, \ldots, x_T \in D$ and an initial state s_0 (we will sometimes omit s_0 as an argument for brevity), let $s_t = \textbf{State}_t(x_t; s_{t-1})$ and $\textbf{State}(x_1, \ldots, x_T) = (s_0, s_1, \ldots, s_T)$.

An estimation algorithm is defined by additional functions **Out** and $\mathcal{A}$, where **Out** maps s_T to an output space $\mathcal{O}$, and **Est** takes a vector of n elements from $\mathcal{O}$ and computes an estimate of the desired statistic.

Definition 2.3 (Local pan-privacy)**.** *We say a streaming algorithm defined by a set of functions $\textbf{State}_t$ is ε-locally pan-private if for any pair of streams $\mathbf{x} = x_1, \ldots, x_T$ and $\mathbf{x}' = x'_1, \ldots, x'_T$, the state vectors $\textbf{State}(\mathbf{x})$ and $\textbf{State}(\mathbf{x}')$ are ε-indistinguishable.*

Remark 4. *Several remarks are in order. (a) The definition considers neighboring datasets to differ in the full stream at one device. Thus it provides user-level privacy. (b) We require that privacy holds against an adversary that can monitor the internal state on the device after each event, and all communication out of the device. In other words, we guard against multiple intrusions, or continuous intrusion. The problem becomes easier if we only had to guard against a single intrusion. However, for the examples that motivate this work (shared device such a public computer), restricting the adversary to a single intrusion is unnatural. (c) A natural generalization of our model can allow the device to send multiple messages, while making sure to account for the potential information leakage from the event of sending or not-sending a message (that may be observable by an adversary). This generalization does not make the model stronger, as an algorithm that is locally pan-private in this model can store the messages it would have sent, and send them at step T.*[1] *(d) In some settings, one may only be interested in a smaller set of* admissible *streams $\mathbf{x}$, and one can naturally adapt the definition above by requiring the streams $\mathbf{x}$ and $\mathbf{x}'$ to be admissible. We have aimed to keep the definition above simple, at the expense of generality. One can replace the quantifier over $\mathbf{x}, \mathbf{x}'$ above to require them to lie in some set $\mathcal{X}$ of admissible streams.*

The following definitions capture the standard notions of privacy for such algorithms.

Definition 2.4. *We say an estimation algorithm is (ε, δ)-locally differentially private if for any pair of streams $\mathbf{x} = x_1, \ldots, x_T$ and $\mathbf{x}' = x'_1, \ldots, x'_T$, the distributions $\textbf{Out}(\textbf{State}(\mathbf{x}))$ and $\textbf{Out}(\textbf{State}(\mathbf{x}'))$ are (ε, δ)-indistinguishable.*

Definition 2.5. *We say an estimation algorithm is (ε, δ)-aggregator differentially private if for any pair of streams $\mathbf{x}$*

[1]If we were concerned about the size of $\mathcal{S}$, the many-messages version of the model may be more powerful.

*and $\mathbf{x}'$, and any set of $(n-1)$ streams $\{\mathbf{y}^{(j)}\}_{j=1}^{n-1}$ the distributions defined by $\sum_{i=1}^{n-1}$ **Out**(**State**$(\mathbf{y}^{(i)})) +$ **Out**(**State**$(\mathbf{x}))$ and $\sum_{i=1}^{n-1}$ **Out**(**State**$(\mathbf{y}^{(i)})) +$ **Out**(**State**$(\mathbf{x}'))$ are (ε, δ)-indistinguishable.*

We next define computational indistinguishability. A "security" parameter λ will control various quantities in these definitions. The adversary will be computationally bounded to be polynomial in λ, and we say a function in λ is *negligible* if it approaches zero faster than the inverse of any polynomial in λ.

Definition 2.6. *Let $\{D_\lambda\}_\lambda$ and $\{D'_\lambda\}_\lambda$ be two ensembles of distribution. We say that they are $f(\lambda)$-computationally indistinguishable if for any non-uniform probabilistic polynomial time algorithm A,*

$$|\Pr_{z\sim D_\lambda}[A(z)=1] - \Pr_{z\sim D'_\lambda}[A(z)=1]| \le f(\lambda).$$

When two ensembles are $f(\lambda)$-computationally indistinguishable for negligible function f, we say that they are computationally indistinguishable.

We use a public-key encryption scheme.

Definition 2.7. *A public key encryption scheme is defined by the following set of probabilistic polynomial time (p.p.t.) algorithms.*

Key Generation $\texttt{KeyGen}(\cdot)$ *takes a security parameter in unary 1^λ and outputs a pair of keys (k_{priv}, k_{pub}), each in $\{0,1\}^*$.*

Encryption $\texttt{Enc}(m, k_{pub})$ *takes a message m and a public key k_{pub} and outputs an encryption $c \in \{0,1\}^*$.*

Decryption $\texttt{Dec}(c, k_{priv})$ *takes a ciphertext c and a private key k_{priv} and outputs a message m.*

These functions have the following properties:

Correctness *Suppose that $(k_{priv}, k_{pub}) \leftarrow \texttt{KeyGen}(1^\lambda)$ for some λ. Then for any $m \in \{0,1\}^k$, it holds that $\texttt{Dec}(\texttt{Enc}(m, k_{pub}), k_{priv})$ equals m.*

Semantic Security *For any m, m', the encryptions $\texttt{Enc}(m, k_{pub})$ and $\texttt{Enc}(m', k_{pub})$ are computationally indistinguishable.*

We rely on encryption schemes that allow re-encryption of an encrypted message.[2] Informally, a rerandomizable encryption allows for an encrypted message to be "rerandomized", i.e. to be replaced by a new encryption of the same message, that is indistinguishable from a new encryption. Here and in the rest of the paper ${\Phi_r}^t(\cdot)$ denotes the t-fold composition $\Phi_r(\Phi_r(\ldots(\Phi_r(\cdot))))$.

[2]This is slightly weaker than the usual notion of rerandomizable encryption, as we allow c and $\Phi_r(c, k_{pub})$ to be distinguishable. We show that this notion is necessary and sufficient for our purposes.

Definition 2.8. *A public key encryption scheme is* rerandomizable *if there is a function $\Phi_r(c, k_{pub})$ with the following properties. For any m, m', and any $t \ge 0$, let $c = {\Phi_r}^t(\texttt{Enc}(m, k_{pub}))$ and $c' = {\Phi_r}^t(\texttt{Enc}(m', k_{pub}))$, and let $\bar{c} = \Phi_r(c, k_{pub})$ and $\bar{c'} = \Phi_r(c', k_{pub})$. Then $\texttt{Dec}(c, k_{priv}) = \texttt{Dec}(\bar{c}, k_{priv})$. Furthermore, the tuple $(c, \bar{c})$ is computationally indistinguishable from $(c, \bar{c'})$.*

Note that this definition implies that for any t and any m, m', the distributions ${\Phi_r}^t(\texttt{Enc}(m))$ and ${\Phi_r}^t(\texttt{Enc}(m'))$ are computationally indistinguishable. We remark that we do not require a rerandomized ciphertext to be indistinguishable from a freshly generated one: it need only be indistinguishable from a rerandomized encryption of any different plaintext (for the same number of rerandomizations t).

Mironov et al. (2009) defined and related different notions of computational differential privacy, and those notions can be extended to local pan-privacy. We state a version of this definition next.

Definition 2.9 (Computational (ε, δ)-indistinguishability)**.** *Let $\{D_\lambda\}_\lambda$ and $\{D'_\lambda\}_\lambda$ be two ensembles of distribution. We say that they are computationally (ε, δ)-indistinguishable if for any non-uniform probabilistic polynomial time algorithm A,*

$$\Pr_{z\sim D_\lambda}[A(z)=1] \le e^\varepsilon \cdot \Pr_{z\sim D'_\lambda}[A(z)=1]| + \delta + \texttt{negl}(\lambda).$$

When two ensembles of r.v.'s are computationally $(\varepsilon, 0)$-indistinguishabile, we will often call them computationally ε-indistinguishable.

Definition 2.10 (Computational Local Pan-Privacy,IND-CDP version)**.** *We say a streaming algorithm defined by a set of functions **State**$_t$ is computationally ε-locally pan-private if for any pair of streams $\mathbf{x} = x_1, \ldots, x_T$ and $\mathbf{x}' = x'_1, \ldots, x'_T$, the state vectors **State**$(\mathbf{x})$ and **State**$(\mathbf{x}')$ are computationally ε-indistinguishable.*

We recall some basic mechanisms that will be useful. We refer the reader to Dwork and Roth (2014) for their privacy proofs.

Definition 2.11 (Randomized Response)**.** *Let $\varepsilon > 0$. The randomized response mechanism $2RR_\varepsilon : \{0,1\} \to \{0,1\}$ is an ε-DP mechanism defined as:*

$$2RR_\varepsilon(b) = \begin{cases} b & \textit{with probabilty } \frac{e^\varepsilon}{1+e^\varepsilon} \\ 1-b & \textit{with probabilty } \frac{1}{1+e^\varepsilon} \end{cases}$$

The following observation about implementing randomized response will be useful.

Observation 5. *Let $b \in \{0,1\}$ and $\varepsilon > 0$. Then $2RR_{\varepsilon}(b)$ can be implemented by outputting b with probability $\frac{e^{\varepsilon}-1}{e^{\varepsilon}+1}$, and outputting a random bit otherwise.*

We will use the following lemma that says that any 2-input local randomizer is a post-processing of a randomized response.

Lemma 2.1. *(Kairouz et al., 2015) Let $\mathcal{R} : \{0,1\} \to \mathcal{S}$ be an ϵ-DP local randomizer. Then there exists a post-processing function $h : \{0,1\} \to \mathcal{S}$ such that $\mathcal{R}(0) = h(2RR_{\epsilon}(0))$ and $\mathcal{R}(1) = h(2RR_{\epsilon}(1))$.*

The following utility bounds for randomized response are standard (Feldman et al., 2020).

Theorem 6. *Let $\varepsilon_0 > 0$. Then for any $b_1, \ldots, b_n \in \{0,1\}$, one can post-process their randomized responses $\{y_i = 2RR_{\varepsilon_0}(b_i)\}$ to derive an estimate $\hat{S}$ of $S = \sum_i b_i$ such that*

$$\mathbb{E}[\hat{S}] = S; \quad \mathbb{E}[|\hat{S} - S|] \leq O\left(\sqrt{n\left(1 + \frac{e^{\varepsilon_0}}{(1+e^{\varepsilon_0})^2}\right)}\right).$$

Theorem 7. *Let $(\varepsilon, \delta) \in (0,1)$. Then there is an ε_0 such that for any $b_1, \ldots, b_n \in \{0,1\}$, the randomized responses $\{y_i = 2RR_{\varepsilon_0}(b_i)\}$ are (ε,δ)-DP in the aggregator model. Further their sum can be post-processed to derive an estimate $\hat{S}$ of $S = \sum_i b_i$ such that*

$$\mathbb{E}[\hat{S}] = S; \quad \mathbb{E}[|\hat{S} - S|] \leq O(\sqrt{\log \tfrac{1}{\delta}}/\varepsilon).$$

Let $\mathbf{x} = x_1, \ldots, x_T$ be an input stream with each $x_i \in \{0,1\}$. We denote by $\mathbb{1}(\mathbf{x})$ the predicate $(\max_i x_i = 1)$. For a set of streams $\{\mathbf{x}^{(i)}\}_{i=1}^{n}$, the COUNTNONZERO value is defined as $\sum_i \mathbb{1}(\mathbf{x}^{(i)})$.

3. Lower Bound

In this section, we prove a lower bound showing that information-theoretic local pan-privacy incurs a non-trivial cost on the achievable accuracy for the basic COUNTNONZERO task. Without local pan-privacy, this task can be solved easily. Each device maintains $\mathbb{1}(\mathbf{x}_{1:t})$, which can be done with a single bit of state. Randomized response on this state after T steps allows us to estimate COUNTNONZERO with additive error $\tilde{O}(1/\varepsilon)$ in the aggregator DP setting (Theorem 7), and $\tilde{O}(\sqrt{n(1 + \frac{e^{\varepsilon}}{(e^{\varepsilon}-1)^2}}))$ in the local DP setting (Theorem 6). In contrast, we show that information-theoretic local pan-privacy entails a polynomial dependence on T. This lower bound holds for the case when the input streams are restricted to have at most a single '1'.

Each client receives a stream of inputs $x_1, \ldots, x_T \in \{0,1\}$, can communicate once to the server, and the goal of the server is to compute the number of clients such that $\mathbb{1}(\mathbf{x}) = 1$. We will prove the following theorem. Note that this means that the correlation between the message a device sends, and $\mathbb{1}(\mathbf{x})$ is at most $O(1/\sqrt{T})$

Theorem 8. *Let $\mathcal{A}$ be an ε-locally pan-private client-side algorithm for the COUNTNONZERO task comprising of the pair **State** and **Out**. Then there exists inputs $\mathbf{x}$ and $\mathbf{x}'$ such that $\mathbb{1}(\mathbf{x}) = 0$, $\mathbb{1}(\mathbf{x}') = 1$, and*

$$TV(\textbf{\textit{Out}}(\textbf{\textit{State}}(\mathbf{x})), \textbf{\textit{Out}}(\textbf{\textit{State}}(\mathbf{x}'))) \leq O((e^{\varepsilon} - 1)/\sqrt{Te^{\varepsilon}}).$$

This bound, coupled with standard techniques (e.g. (Duchi et al., 2013)), yields the following lower bound. This shows that for large T, local pan-privacy must come at a significant cost.

Theorem 9. *Let $\mathcal{A}$ be an ε-locally pan-private algorithm run on n clients, and let $\hat{S}$ be any estimate of COUNTNONZERO$(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)})$ derived from the the outputs of $\mathcal{A}$. For $T \geq 16e^{\varepsilon}$, $\hat{S}$ has expected error $\Omega(\sqrt{nTe^{\varepsilon}/(e^{\varepsilon}-1)^2})$.*

Algorithm 1 f_D

```
1: Require: State, Out, initial state $s_0$
2: Input: $b$
3: if $b = 0$ then
4:     Return Out(State($(0, \ldots, 0); s_0$))
5: else
6:     Select $\tilde{t} \sim$ Uniform($[T]$)
7:     Set $x_{\tilde{t}} = 1$ and $x_t = 0$ for $t \in [T] \backslash \{\tilde{t}\}$
8:     Return Out(State($(x_1, \ldots, x_T); s_0$))
9: end if
```

We will now prove Theorem 8. We start by using the locally pan-private algorithm $\mathcal{A}$ to construct a randomized function f_D from $\{0,1\}$ to the output space of the algorithm. On input '0', this algorithm runs $\mathcal{A}$ on the all 0's stream. On input '1', it will put a 1 at a uniformly random place in the stream (with other x_t's being 0) and run $\mathcal{A}$ on the resulting stream. Since $\mathbb{1}(\mathbf{x})$ is different on these two streams, we expect this f_D to behave differently enough on 0 and 1. We will argue that the local pan-privacy constraint prevents these distributions from being too far.

Towards this goal, we first argue that the output of a locally pan-private algorithm can be obtained as a post-processing of randomized responses of individual x_t's.

Lemma 3.1. *Given an initial state s_0, there exists a post-processing function g such that*

$$\textbf{State}((x_1, \ldots, x_T); s_0) = g(2RR_{\epsilon}(x_1), \ldots, 2RR_{\epsilon}(x_T)).$$

Proof. We will first show that if **State** is ϵ-DP then for all $t \in [T]$ and any state s_{t-1}, $\textbf{State}_t(\cdot; s_{t-1})$ is ϵ-DP.

Let $t \in [T]$, $s_{t-1} \in \mathcal{S}$, and $x_t, x_t' \in \{0,1\}$. Further, let $\{x_{t'}\}_{t' \in [T] \setminus \{t\}} \in \{0,1\}^{T-1}$ and $\{s_{t'}\}_{t' \in [T] \setminus \{t\}} \in \mathcal{S}^{T-1}$, then

$$\begin{aligned}
&\frac{\Pr(\mathbf{State}_t(x_t; s_{t-1}) = s_t)}{\Pr(\mathbf{State}_t(x_t'; s_{t-1}) = s_t)} \\
&\quad= \frac{\Pr(\mathbf{State}_t(x_t; s_{t-1}) = s_t)}{\Pr(\mathbf{State}_t(x_t'; s_{t-1}) = s_t)} \times \\
&\qquad \prod_{t' \in [T] \setminus \{t\}} \frac{\Pr(\mathbf{State}_{t'}(x_{t'}; s_{t'-1}) = s_{t'})}{\Pr(\mathbf{State}_{t'}(x_{t'}; s_{t'-1}) = s_{t'})} \\
&\quad= \frac{\Pr(\mathbf{State}((x_1, \ldots, x_t, \ldots, x_T); s_0) = (s_1, \ldots, s_T))}{\Pr(\mathbf{State}((x_1, \ldots, x_t', \ldots, x_T); s_0) = (s_1, \ldots, s_T))} \\
&\quad\le e^{\epsilon}.
\end{aligned}$$

Thus the map $\mathbf{State}_t(\cdot; s_{t-1})$ can be viewed as a local DP mechanism, and thus Lemma 2.1 implies that it can be written as a postprocessing of a randomized response. Given $t \in [T]$ and $s_{t-1} \in \mathcal{S}$, let $h_{t,s_{t-1}} : \{0,1\} \to \mathcal{S}$ be such that for $b \in \{0,1\}$, $\mathbf{State}_t(b; s_{t-1}) = h_{t,s_{t-1}}(\texttt{2RR}_{\epsilon}(b))$. Then we can define $g(b_1, \ldots, b_T) = (s_1, \ldots, s_T)$ where $s_t = h_{t,s_{t-1}}(b_t)$. □

We next argue that this, and the fact that our distributions on $\mathbf{x}$ are exchangeable implies that the output really is a post-processing of the *count* of the number of 1s in the randomized responses.

Lemma 3.2. *Given any initial state s_0, there exists a post-processing function h such that $f_D(0) = h(\text{Bin}(T, \frac{1}{e^{\epsilon}+1}))$ and $f_D(1) = h(\text{Bin}(T-1, \frac{1}{e^{\epsilon}+1}) + \text{Ber}(\frac{e^{\epsilon}}{e^{\epsilon}+1}))$. Further,*

$$TV(f_D(0), f_D(1)) \tag{1}$$

$$\le TV\left(\text{Bin}(T, \tfrac{1}{e^{\epsilon}+1}), \text{Bin}(T-1, \tfrac{1}{e^{\epsilon}+1}) + \text{Ber}(\tfrac{e^{\epsilon}}{e^{\epsilon}+1})\right) \tag{2}$$

$$= O\left((e^{\varepsilon} - 1)\sqrt{\frac{1}{Te^{\varepsilon}}}\right). \tag{3}$$

Proof. By Lemma 3.1, there exists $g : \{0,1\}^T \to \mathcal{S}^T$ such that $\mathbf{State}((x_1, \ldots, x_T); s_0)$ can be written as $g(\texttt{2RR}_{\epsilon}(x_1), \ldots, \texttt{2RR}_{\epsilon}(x_T))$. Under our distribution on $\mathbf{x}$ conditioned on b, the random variables x_t's are exchangeable, and thus conditioned on the count of 1s in $\texttt{2RR}_{\varepsilon}(x_i)$'s, all permutations are equally likely. The function h is then described in Algorithm 2. This implies the first part of the claim. The inequality Eq. (2) is a consequence of the data processing inequality. The second inequality is standard. For completeness, we give a proof in Appendix A.

□

Lemma 3.2 implies Theorem 8 and completes the proof of our lower bound.

Algorithm 2 h

```
1: Require: $g$, Out
2: Input: $m \in [T]$
3: Sample $\mathbf{b}$ uniformly at random from $\{\mathbf{b} \in \{0,1\}^T \mid |\mathbf{b}|_1 = m\}$.
4: Return Out($g(\mathbf{b})$).
```

4. The Single-Server Model

In this section, we will discuss the single-server model. We will require a rerandomizable public-key encryption scheme. Our computational local pan-privacy will be achieved by the device storing encryptions of any sensitive state. As the device does not hold the private key, the on-device state is computationally indistinguishable from a sequence of encryptions of 0.

4.1. Counting Devices that have at least one occurrence

Let us first consider the problem of differentially privately computing the number of clients for which the sensitive event occurs at least once. In Section 4.2 we will discuss how to extend our approach to building a histogram of the number of occurrences of the sensitive event.

We start by describing how the count will be stored and updated on device. At the beginning of the collection period, the client initializes the state to be an encryption of 0. At each time step, the device updates the state. If x_t is 1, i.e. if the sensitive event has occurred in this time step, the device replaces the state with a fresh encryption of 1, with an appropriate number of rerandomizations applied. If x_t is 0, the device rerandomizes the current state. This continues until the end of the time horizon. Pseudo-code is given in Algorithm 3. This algorithm does not reveal anything about whether or when updates have occurred to an intruder, even if the intruder can view the internal state of the device after each time step. The intrusion resistance comes from the fact that an intruder can not distinguish between a fresh (rerandomized) encryption of 1, and a rerandomization a current encrypted value. Formally, the state after t steps is computationally indistinguishable from ${\Phi_r}^t(\texttt{Enc}(0))$. Finally, the device uses randomized response to send their encrypted bit to the server. This can be done without needing to decrypt the state using Observation 5. The sent message is computationally indistinguishable from ${\Phi_r}^{T+1}(\texttt{Enc}(0))$. We remark that for most practical cryptosytems, ${\Phi_r}^t(\texttt{Enc}(b))$ has the same distribution as $\texttt{Enc}(b)$ so that we can optimize the algorithm by replacing the ${\Phi_r}^t(\texttt{Enc}(b))$ steps by $\texttt{Enc}(b)$.

At the end of the collection period, the goal of the server is

Algorithm 3 COUNTNONZERO, Client Algorithm

```
Require: $T, \epsilon_0, \texttt{Enc}, \Phi_r, \mathbf{x} = x_1, \ldots, x_T$
1: Initialization
2: $c = \texttt{Enc}(0)$
3: State Update
4: for $t = 1 : T$ do
5:     if $x_t = 1$ then
6:         $c = {\Phi_r}^t(\texttt{Enc}(1))$
7:     else
8:         $c = \Phi_r(c)$
9:     end if
10: end for
11: Send to Server
12: $r = \text{Ber}(\frac{e^{\epsilon_0}-1}{e^{\epsilon_0}+1})$
13: if $r = 0$ then
14:     $r' = \text{Ber}(1/2)$
15:     if $r' = 1$ then
16:         $c = {\Phi_r}^{T+1}(\texttt{Enc}(0))$
17:     else
18:         $c = {\Phi_r}^{T+1}(\texttt{Enc}(1))$
19:     end if
20: else
21:     $c = \Phi_r(c)$
22: end if
23: Return $c$
```

to compute a differentially private estimate of the number of devices for which the sensitive event occurred at least once. Since they have the private key, they can decrypt the local reports from each client, then aggregate and de-bias the resulting sum in the same way they would for randomized response. The following result follows:

Theorem 10. *Let $\varepsilon_0 > 0$. Suppose that each client i uses Algorithm 3 on its input $\mathbf{x}^{(i)}$. Then the server can estimate* COUNTNONZERO *on inputs $\{\mathbf{x}^{(i)}\}_{i=1}^n$ with expected error $O\left(\sqrt{n\left(1+\frac{e^{\varepsilon_0}}{(1+e^{\varepsilon_0})^2}\right)}\right)$ and ε_0-local differential privacy. For any $(\varepsilon, \delta) \in (0, 1)$, there is an ε_0 such that the mechanism is (ε, δ)-aggregator DP, and has expected error $O(\sqrt{\log\frac{1}{\delta}}/\varepsilon)$. The client algorithm satisfies computational 0-local pan-privacy.*

4.2. Computing Histograms of the Number of Occurrences

In this section we will consider how to compute a histogram of the number of clients that have a particular number of occurrences. We will solve a slightly more general problem, where we are given a k and the goal is to estimate for each $i \in \{0, 1, \ldots, k-1\}$ the number of devices with count $|\mathbf{x}|_1$ equal to i, as well as the number of devices with count at least k. While one could always set $k = T$, this generalization can allow more efficient solutions in the regime where T is large and the number of occurrences per device is much smaller than T. To handle this last bucket of count at least k, we will maintain an (encrypted) indicator d_i for each i, of the event $|\mathbf{x}|_1 \geq i$.

At the beginning of the collection period, the device initializes $k+1$ counters to be an encryption of the initial histogram, i.e. $c_0 = \texttt{Enc}(1), c_1 = \texttt{Enc}(0), \ldots, c_k = \texttt{Enc}(0)$. Additionally, it initializes $d_0 = \texttt{Enc}(1)$ and $d_i = \texttt{Enc}(0)$ for each $i = 1, \ldots, k$. At each time step, if an event has not occurred, all the counters are rerandomized. If an event has occurred then all the counters are rerandomized *and* shifted by 1. The counter c_0 is set to be an $\texttt{Enc}(0)$, and d_0 is set $\texttt{Enc}(1)$. At the end of T steps, the device has the desired values $(v_0, v_1, \ldots, v_k)$ in $(c_0, c_1, \ldots, c_{k-1}, d_k)$, since c_i encrypts $\mathbb{1}(|\mathbf{x}|_1 = i)$ and d_k encrypts $\mathbb{1}(|\mathbf{x}|_1 \geq k)$. It uses randomized response on each of these to send the result to the server. We defer the full pseudocode to Algorithm 5.

Upon receiving the reports from each client, the server can decrypt the randomized responses. We sum these reports and use the standard de-biasing for randomized response to obtain an unbiased estimate of each histogram bucket count. Note that the sensitivity of the vector $(v_1, \ldots, v_k)$ is 1 as at most one of the values can be 1. Thus even though k randomized responses are sent, the noise added does not grow with k. Since we run randomized response on each coordinate of the histogram, Theorem 6 and Theorem 7 implies that

Theorem 11. *Algorithm 5 is computationally 0-locally pan-private, and ε_0-local DP with respect to the server. It estimates the histogram with expected error $O\left(\sqrt{n\left(1+\frac{e^{\varepsilon_0}}{(1+e^{\varepsilon_0})^2}\right)}\right)$ for each bucket count. Moreover, for any $(\varepsilon, \delta) \in (0, 1)$, there is an ε_0 such that the mechanism is (ε, δ)-aggregator DP, and has expected error $O(\sqrt{\log\frac{1}{\delta}}/\varepsilon)$ for each bucket count.*

4.3. Computing the Average Number of Occurrences

Computing the average number of occurrences per-device can be done using histograms, under the assumption that no count is larger than k. Alternatively, the resulting average can be viewed as an average of a truncation (at k) of the original counts. However using the estimate $\sum_j c_j = \sum_{i=1}^k i \cdot |\{j : c_i = j\}|$ will result in error that scales as $O(k^{\frac{3}{2}})$. In Appendix B, we show that using an encryption scheme that has one additional property, we can reduce the error to $O(k)$, matching the bound one would get in the central model.

5. The Two-Server Model

In this section we will address the two-server model of (Corrigan-Gibbs and Boneh, 2017), where the client sends secret-shares of their data to two servers. Any sensitive information stored locally will be secret-shared with the shares encrypted by the public keys of two separate servers at all times. This scheme is protected against continuous intrusion by an adversary that does not collude with either of the servers.

In the two-server model, we often want to additionally protect the server against malicious clients who seek to poison the result by sending secret-shares of invalid reports. For example, instead of sending a secret-sharing of 0 or 1, a malicious device may send a secret-sharing of a million to skew the result. We will build on zero-knowledge proofs of validity for the predicates of interest. Our construction will maintain encrypted secret-shares of the state. The additional complexity arising from these proofs is that to construct the proof of validity, one needs to know the secret-shared data. To address this, we make use of an important fact. The proofs in Prio depend on the input being secret-shared (as they must), but conditioned on the input, do not depend on the encryptions of the secret-shares. This allows the proofs to remain valid when we rerandomize the encryptions. Thus our algorithms will generate proofs of validity when the relevant secret-shares are created. One can rerandomize the encryptions of the secret-shares and the proofs, without impacting the validity of the proofs.

5.1. Counting Devices that have at least one occurrence

We first consider the problem of counting the number of devices that have at least one occurrence in the two-server model. The on-device and server-side algorithms will both be very similar to the single-server setting but with the addition of secret-sharing of any sensitive information on device, and creating validity proofs. For lack of space, we defer the detailed pseudocode to Algorithm 6 in Appendix D.

We remark that this algorithm can resist an intrusion by an adversary that does not collude with either of the two servers. This can be relaxed to allow the adversary to collude with one of the two servers. The place where the collusion with a server can create a challenge is that a colluding server can help the adversary detect whether its share is changed in step 9 or if only the encryption is rerandomized and the share itself is unchanged (step 14). If we use an additively homomorphic encryption scheme (see Definition B.1), we can create new secret-shares under the encryption instead, so that each $c^{(i)}$ is an encryption of a fresh random field element after each update. This allows for non-interactive proofs, including the SNIPs (Corrigan-Gibbs and Boneh, 2017) that use Beaver triples. Each client sends the secret-shares of their contribution to the two servers. Each server can then decrypt the shares with their private key and aggregate the result. The sum of the shares at the two servers can then be de-biased to give the final result. The servers can also validate the proofs to ensure that the secret-shared values are all in $\{0, 1\}$.

5.2. Computing Histograms of the Number of Occurrences

Using the same approach as we did for COUNTNONZERO, we can extend our algorithm for histograms in the single-server setting to work in the two-server model as well. As with the COUNTNONZERO algorithm, the validity property that is being validated is that each of the $k + 1$ reported values are in $\{0, 1\}$. Thus the same approach to constructing validity proofs suffices. For brevity, we omit a detailed description of the algorithm.

6. On the need for rerandomizable Public-key cryptography

We show that a rerandomizable public-key encryption scheme is necessary for computational 0-local pan-privacy. Recall that any locally pan-private algorithm also has an **Initialize** operation, that creates some shared state between the client and the server(s). In our implementations, this operation would be one that provides the public keys to the clients, and creates the initial state s_0.

Theorem 6.1. *Suppose that we have a set of algorithms* **Initialize**, $\{\mathbf{State}_t\}_{t=1}^{T}$, ***Out****, and* **Est** *that define a computationally* 0*-locally pan-private streaming algorithm that estimates* COUNTNONZERO *on any set of inputs with error at most* $n/4$*, with probability* $1 - \texttt{negl}(n)$ *for large enough* n*. Then for any security parameter* λ *and given a* $T = poly(\lambda)$*, we can define functions* `KeyGen`, `Enc`, `Dec` *that define a public-key encryption scheme, where each of these operations runs in poly* (λ) *time. The scheme is rerandomizable in the sense of Definition 2.8, supporting up to* $T - 1$ *rerandomizations.*

Proof. We will build an encryption scheme that can encrypt a single bit b. The basic idea of the construction is to simulate running the algorithm on n clients, where for each client i, the first input in the stream $x_1^{(i)}$ is set to b. The internal state of all the clients will comprise the encryption. The decryption will simulate the state transformations given $x_t^{(i)} = 0$ all the way to T, generating the outputs, and running **Est** on the outputs. The utility guarantee implies that the decryption recovers the original intended bit. The local pan-privacy implies the security of the scheme. Finally rerandomization is achieved by simulating a single state transformation. We give details next.

For a suitably large $n = \text{poly}(\lambda)$, `KeyGen` will run the

$\mathbf{Initialize}$ operation n times, and return the set of n client states, including the state $s_0^{(i)}$ for each client, as a public key k_{pub}, and the n server states as private key k_{priv}. Additionally, the parameter T will be included in both the keys. Given a bit b, $\texttt{Enc}(b, k_{pub})$ will simulate for each i, one step of the locally pan-private algorithm on input b starting at state $s_0^{(i)}$ to derive a set of states $s_1^{(i)}$. The encryption will be defined as $(1, \{s_1^{(i)}\}_{i=1}^n)$. In general, valid ciphertexts will look like $(t, \{s_t^{(i)}\}_{i=1}^n)$, for some $t \leq T$, and some set of states $\{s_t^{(i)}\}_{i=1}^n$. Given such an encryption, the $\texttt{Dec}$ algorithm will simulate, for each i, running the state transformations $\mathbf{State}_{t+1}, \ldots, \mathbf{State}_T$ on $s_t^{(i)}$ with input $x_{t+1}^{(i)} = 0$, followed by running **Out** to generate a set of n messages. It then runs $\mathbf{Est}$ on the collection of n messages, and returns 0 if the answer is smaller than $n/2$, and 1 otherwise. Finally, Φ_r on a ciphertext $c = (t, \{s_t^{(i)}\}_{i=1}^n)$ (for $t < T$) will simulate, for each i, running the state transformations $\mathbf{State}_{t+1}$ on $s_t^{(i)}$ with input $x_{t+1}^{(i)} = 0$ to get an updated state $s_{t+1}^{(i)}$. The updated ciphertext c' is then set to $(t+1, \{s_{t+1}^{(i)}\}_{i=1}^n)$. Any input ciphertexts with $t \geq T$ are rejected by Φ_r.

It is easy to see that $\texttt{Dec}(\texttt{Enc}(b, k_{pub}), k_{priv})$ comprises an exact simulation of running the COUNTNONZERO algorithm on n clients on inputs $(b, 0, 0, \ldots, 0)$, followed by rounding the final estimate. Since the correct answer to COUNTNONZERO on these inputs is bn and the estimation algorithm has error $n/4$ with high probability, we conclude that the encryption scheme satisfies $\texttt{Dec}(\texttt{Enc}(b, k_{pub}), k_{priv}) = b$ except with negligible probability.

The security of the encryption follows from the computational local pan-privacy. Indeed if we had an efficient algorithm A that can distinguish $\texttt{Enc}(0, k_{pub})$ and $\texttt{Enc}(1, k_{pub})$, then by a standard hybrid argument, it can be used to distinguish $\mathbf{State}(0, s_0)$ and $\mathbf{State}(1, s_0)$, which violates the computational 0-local pan-privacy.

The rerandomization is essentially simulating one step in the decoding. It is therefore immediate that for any valid ciphertext, $\texttt{Dec}(\Phi_r(c, k_{pub}), k_{priv})$ and $\texttt{Dec}(c, k_{priv})$ are running exactly the same simulation and thus are identically distributed. Finally, the security of the rerandomization follows once again from the hybrid argument: if we could distinguish iterated rerandomizations of 0 from those of 1, we would get a distinguisher that can learn the first input x_1 from the local state at some later time step t. □

We note that the proof uses instances which have either zero or one '1' in each input stream. Thus any accurate algorithm for histogram estimation implies one for COUNTNONZERO with the same accuracy, and hence would also imply a public-key encryption scheme.

We next extend this argument to handle a large class of ε-locally pan-private algorithms for $\varepsilon > 0$. This result will apply to locally pan-private algorithms that use at most logarithmic space, and which use the same state transformation function $\mathbf{State}$ at all time steps.

Theorem 6.2. *Let $\varepsilon \in (0, 1)$. Suppose that we have a set of algorithms $\mathbf{Initialize}, \mathbf{State}$, **Out**, and $\mathbf{Est}$ that define a computationally ε-locally pan-private streaming algorithm that estimates* COUNTNONZERO *on any set of inputs with error at most $n/4$, with probability $1 - \texttt{negl}(n)$ for large enough n, and for up to T steps, using S bits of space. Then for any security parameter λ, we can define functions $\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec}$ that define a $(\frac{\varepsilon}{\sqrt{T}} + T \cdot \texttt{negl}(\lambda))$-secure public-key encryption scheme, where each of these operations runs in poly $(\lambda, 2^S)$ time. The scheme is rerandomizable in the sense of Definition 2.8, supporting up to $T - 1$ rerandomizations.*

We defer the proof to Appendix C. Note that if T^{-1} is $\texttt{negl}(\lambda)$ and S is $O(\log \lambda)$, then the resulting public-key encryption scheme is secure.

7. Conclusions

In this work, motivated by privacy concerns on shared devices, we introduce the notion of local pan-privacy. We show that while information-theoretic local pan-privacy may be too strong a requirement for basic telemetry tasks, computational versions of this definition can be achieved without sacrificing on utility. We present algorithms for the fundamental tasks of counting the number of devices where a sensitive event occurs, as well as histograms of event counts, both in the trusted server and the two-server models. Our algorithms use public-key encryption schemes, and we show that such schemes are necessary to achieve computational local pan-privacy.

Our work raises many natural question. Our lower bound in Theorem 8 relies on instances with at most a single 1, and shows a $\sqrt{T}$ gap in the error. We conjecture that this can be strengthened to an $\Omega(T)$ gap when one allows the instances to have arbitrarily many 1s. While we have given algorithms for the most common telemetry tasks, other telemetry tasks may raise additional challenges. The validity proofs in the more general setting (when an adversary can collude with one of the servers) in our approach need to be non-interactive. We leave open the question of designing efficient zero-knowledge proofs for other predicates, that are compatible with local pan-privacy.

Impact Statement

This paper presents work whose goal is to advance the field of Differentially Private Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Apple and Google. Exposure notification privacy-preserving analytics (ENPA) white paper. https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ENPA_White_Paper.pdf, 2021.

Apple's Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(9), 2017.

Daniel Berend and Aryeh Kontorovich. A sharp estimate of the binomial mean absolute deviation with applications. *Statistics & Probability Letters*, 83(4):1254–1259, 2013. ISSN 0167-7152. doi: https://doi.org/10.1016/j.spl.2013.01.023. URL https://www.sciencedirect.com/science/article/pii/S0167715213000242.

Keith 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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3133982.

Clement Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *Journal of Privacy and Confidentiality*, 12(1), Jul. 2022. doi: 10.29012/jpc.784. URL https://journalprivacyconfidentiality.org/index.php/jpc/article/view/784.

Karan Chadha, Junye Chen, John Duchi, Vitaly Feldman, Hanieh Hashemi, Omid Javidbakht, Audra McMillan, and Kunal Talwar. Differentially private heavy hitter detection using federated analytics. In *2nd IEEE Conference on Secure and Trustworthy Machine Learning*, 2024. URL https://openreview.net/forum?id=1HF8tLztGX.

Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In Aditya Akella and Jon Howell, editors, *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 259–282. USENIX Association, 2017.

Abraham de Moivre. *Miscellanea analytica de seriebus et quadraturis*. J. Tonson & J. Watts, 1730.

Persi Diaconis and Sandy Zabell. Closed Form Summation for Classical Distributions: Variations on a Theme of De Moivre. *Statistical Science*, 6(3):284 – 302, 1991. doi: 10.1214/ss/1177011699. URL https://doi.org/10.1214/ss/1177011699.

Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3574–3583, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 429–438, 2013. doi: 10.1109/FOCS.2013.53.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In *Proceedings of The First Symposium on Innovations in Computer Science (ICS 2010)*. Tsinghua University Press, 2010. URL https://www.microsoft.com/en-us/research/publication/pan-private-streaming-algorithms/.

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 1054–1067, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329576.

Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. *CoRR*, abs/2012.12803, 2020. URL https://arxiv.org/abs/2012.12803. Preliminary version appears in FOCS 2021.

Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, May 1996. ISSN 0004-5411. doi: 10.1145/233551.233553. URL https://doi.org/10.1145/233551.233553.

Christoph G. Günther. An identity-based key-exchange protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, pages 29–37, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

Robert Helmer, Anthony Miyaguchi, and Eric Rescorla. Testing privacy-preserving telemetry with prio. https://hacks.mozilla.org/2018/10/testing-privacy-preserving-telemetry-with-prio/, 2018.

Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. volume 37 of *Proceedings of Machine Learning Research*, pages 1376–1385, Lille, France, 2015. PMLR.

Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5201–5212. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/kairouz21a.html.

Daniele Micciancio. Oblivious data structures: applications to cryptography. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, page 456–464, New York, NY, USA, 1997. Association for Computing Machinery. ISBN 0897918886. doi: 10.1145/258533.258638. URL https://doi.org/10.1145/258533.258638.

Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 37–48. ACM, 2011. doi: 10.1145/1989284.1989290. URL https://doi.org/10.1145/1989284.1989290.

Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 126–142, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03356-8.

Moni Naor and Vanessa Teague. Anti-persistence: history independent data structures. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 492–501, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133499. doi: 10.1145/380752.380844. URL https://doi.org/10.1145/380752.380844.

Zheng Xu, Yanxiang Zhang, Galen Andrew, Christopher Choquette, Peter Kairouz, Brendan Mcmahan, Jesse Rosenstock, and Yuanbo Zhang. Federated learning of gboard language models with differential privacy. In Sunayana Sitaram, Beata Beigman Klebanov, and Jason D Williams, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 629–639, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-industry.60. URL https://aclanthology.org/2023.acl-industry.60/.

Yuanbo Zhang, Daniel Ramage, Zheng Xu, Yanxiang Zhang, Shumin Zhai, and Peter Kairouz. Private federated learning in gboard, 2023. URL https://arxiv.org/abs/2306.14793.

Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. Federated heavy hitters discovery with differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 3837–3847. PMLR, 2020.

A. On the TV distance between Binomials

In this section, we will prove the following inequality.

Theorem 12. *Let $T \geq 2$ and $p \in (0, \frac{1}{2})$. Then*

$$\begin{aligned} TV(Bin(T,p), Bin(T-1,p) + Bern(1-p)) &= 2(1-2p)\frac{\lceil Tp \rceil}{Tp} \Pr[Bin(T,p) = \lceil Tp \rceil] \\ &\leq (1-2p)/\sqrt{4p(1-p)T}) \end{aligned}$$

Proof. Note that $Bin(T,p) = Bin(T-1,p) + Bern(p)$. Since we can couple $Bern(p)$ and $Bern(1-p)$ so that they agree with probability $2p$, and differ by 1 otherwise, it follows that

$$TV(Bin(T,p), Bin(T-1,p) + Bern(1-p)) = (1-2p) \cdot TV(Bin(T-1,p), Bin(T-1,p)+1).$$

We now write

$$\begin{aligned} 2TV(Bin(T-1,p), Bin(T-1,p)+1) &= \sum_{m=0}^{T} |\Pr[Bin(T-1,p) = m] - \Pr[Bin(T-1,p) = m-1]| \\ &= \sum_{m=0}^{T} |\binom{T-1}{m} p^m (1-p)^{T-1-m} - \binom{T-1}{m-1} p^{m-1}(1-p)^{T-m}| \\ &= \sum_{m=0}^{T} \frac{1}{Tp(1-p)} \cdot \binom{T}{m} p^m (1-p)^{T-m} |p(T-m) - (1-p)m| \\ &= \sum_{m=0}^{T} \frac{1}{Tp(1-p)} \cdot \binom{T}{m} p^m (1-p)^{T-m} |pT - m| \end{aligned}$$

Letting $f(m)$ denote $pT - m$, we can thus write

$$2TV(Bin(T-1,p), Bin(T-1,p)+1) = \frac{1}{Tp(1-p)} \mathop{\mathbb{E}}_{X \sim Bin(T,p)} [|f(X)|].$$

The value $|f(X)|$ is the absolute deviation of a binomial random variable $Bin(T,p)$ from its expectation. An exact formula for the mean absolute deviation of a binomial was given by de Moivre (1730) (see Diaconis and Zabell (1991) for a historical perspective on this formula). De Moivre showed that

$$\begin{aligned} \mathop{\mathbb{E}}_{X \sim Bin(T,p)} [|X - Tp|] &= 2\lceil Tp \rceil (1-p) \Pr[Bin(T,p) = \lceil Tp \rceil] \\ &= 2\binom{T}{\lceil Tp \rceil} \lceil Tp \rceil p^{\lceil Tp \rceil} (1-p)^{T - \lceil Tp \rceil + 1}. \end{aligned}$$

Plugging this bound gives the claimed equality. To prove the upper bound, we use Jensen's inequality:

$$\sqrt{\mathop{\mathbb{E}}_{X \sim Bin(T,p)} [|f(X)|]} \leq \sqrt{\mathop{\mathbb{E}}_{X \sim Bin(T,p)} [f(X)^2]}.$$

Now observe that

$$\begin{aligned} \mathop{\mathbb{E}}_{X \sim Bin(T,p)} [f(X)^2] &= \mathop{\mathbb{E}}_{X \sim Bin(T,p)} [(X - pT))^2] \\ &= p(1-p)T. \end{aligned}$$

It follows that

$$2TV(Bin(T-1,p), Bin(T-1,p)+1) \leq \sqrt{\frac{1}{p(1-p)T}},$$

so that

$$TV(Bin(T,p), Bin(T-1,p)+Bern(1-p)) \leq (1-2p)\sqrt{\frac{1}{4p(1-p)T}}.$$

Berend and Kontorovich (2013) show that the bound from Jensen's inequality above (which is the only inequality in this proof) is tight up to a $\sqrt{2}$ factor for all $p \in (\frac{1}{T}, 1-\frac{1}{T})$. When $p < \frac{1}{T}$, the mean absolute deviation is bounded by $2Tp$ and this is tight up to a factor of e. A symmetric bound holds for $p > 1-\frac{1}{T}$. $\square$

B. Averages via Additively Homomorphic Encryption

We recall the definition of additively homomorphic encryption scheme. Several popular public-key encryption schemes or standard variants of those, including Paillier, RSA and El Gamal, and lattice-based schemes, satisfy these properties.

Definition B.1. *A public key encryption scheme as defined in Definition 2.7 is partially homomorphic over a group $(G,*)$ if there is a p.p.t. algorithm $\Phi_+(\cdot,\cdot,k_{pub})$ with the property that for any pair of valid ciphertexts c_1, c_2, $\texttt{Dec}(\Phi_+(c_1,c_2,k_{pub})) = \texttt{Dec}(c_1,k_{priv}) + \texttt{Dec}(c_2,k_{priv})$; and a p.p.t. algorithm Φ_* that takes a group element and a ciphertext, and outputs another ciphertext with the property that for any c and any $\alpha \in G$, $\texttt{Dec}(\Phi_*(\alpha,c,k_{pub}),k_{priv}) = \alpha * \texttt{Dec}(c,k_{priv})$.*

We will omit k_{pub} as an argument from Φ_+ and Φ_* in the rest of the section and write $\Phi_+(c_1,c_2,\ldots,c_k)$ to mean $\Phi_+(c_1,\Phi_+(c_2,\ldots,\Phi_+(c_{k-1},c_k)\ldots))$.

Given an additively homomorphic encryption scheme, we will build on Algorithm 5 to design a locally pan-private algorithm for means. Recall that the histogram algorithm already gives us encryptions of indicators of $|x|_1 = j$ for each $j \in \{0,1,\ldots,k\}$. We will only need to redefine the **Send to Server** subroutine. Building on these, the client computes an encryption of the number of occurrences by multiplying each encrypted bit by the number of occurrences and summing together. Since only one of the coordinates is an encryption of 1, the sum is the number of occurrences. The client can then privatize by adding discrete Gaussian noise (Canonne et al., 2022; Kairouz et al., 2021) to the encrypted sum before sending to the server.

Algorithm 4 Averaging, Client Algorithm

```
Require: $k, T, \texttt{Enc}, \Phi_+, \Phi_*, \mathbf{x}, \sigma^2$
1: Send to Server
2: for i = 0,1,...,k do
3:     $s_i = \Phi_*(i, c_i)$
4: end for
5: $r \sim \mathcal{N}_{\mathbb{Z}}(0, k\sigma^2)$
6: $s_{k+1} = \texttt{Enc}(r)$
7: Return $\Phi_+(s_0, \cdots, s_{k+1})$
```

Upon receiving the reports from each client, the server simply decrypts and takes the average of the noisy reports. For an appropriate choice of σ is easy to show the following.

Theorem 13. *There is a computationally 0-locally pan-private for estimating the sum, which is (ε_0,δ_0)-local DP with respect to the server. It estimates the sum with expected error $O\left(k\sqrt{n\log\frac{1}{\delta}_0}/\varepsilon_0\right)$. Moreover, for any $(\varepsilon,\delta) \in (0,1)$, there is an σ such that the mechanism is (ε,δ)-aggregator DP, and has expected error $O(k\sqrt{\log\frac{1}{\delta}}/\varepsilon)$.*

C. Proof of Theorem 6.2

We recall Theorem 6.2

Theorem 6.2. *Let $\varepsilon \in (0,1)$. Suppose that we have a set of algorithms* $\mathbf{Initialize}, \mathbf{State}$, ***Out****, and* $\mathbf{Est}$ *that define a computationally ε-locally pan-private streaming algorithm that estimates* CountNonZero *on any set of inputs with error at most $n/4$, with probability $1 - \texttt{negl}(n)$ for large enough n, and for up to T steps, using S bits of space. Then for any security parameter λ, we can define functions $\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec}$ that define a $(\frac{\varepsilon}{\sqrt{T}} + T \cdot \texttt{negl}(\lambda))$-secure public-key encryption scheme, where each of these operations runs in poly $(\lambda, 2^S)$ time. The scheme is rerandomizable in the sense of Definition 2.8, supporting up to $T - 1$ rerandomizations.*

Proof. We will use an encryption scheme very similar to the one in the proof above. For encryption, we will set x_t to one for a random $t \in \{1, 2, \dots, T/2\}$, instead of setting x_1 to one in the proof of Theorem 6.1. The encryption algorithm simulates the state transformation up to $T/2$ steps. Lemma 3.2 then allows us to argue the security of the encryption scheme. The decryption simulates the state transforms up to step T and the correctness proof is as before. The additional challenge is to argue that $\texttt{Enc}, \texttt{Dec}$ run in polynomial time. This uses the fact that the result of applying the same randomized transform τ times, on a state of size S can be computed in time $\exp(O(S)) \cdot \log(\tau)$. We give details next.

As before, $\texttt{KeyGen}$ will run the $\mathbf{Initialize}$ operation n times, and return the set of n client states, including the state $s_0^{(i)}$ for each client, as a public key k_{pub}, and the n server states as private key k_{priv}. Additionally, the parameter T will be included in both the keys. To define $\texttt{Enc}$, we consider the following randomized process. Given a bit b, we first create n independent random streams which are zero everywhere, except for a randomly picked $t_i \in \{1, \dots, T/2\}$, where $\mathbf{x}^{(i)}$ is b. We then simulate, for each i, $T/2$ steps of the locally pan-private algorithm on $\mathbf{x}^{(i)}$ starting at state $s_0^{(i)}$ to get $s_{T/2}^{(i)}$. $\texttt{Enc}(b, k_{pub})$ is then defined as $(T/2, \{s_{T/2}^{(i)}\}_{i=1}^n)$. In general, valid ciphertexts will look like $(t, \{s_t^{(i)}\}_{i=1}^n)$, for some $t \in [T/2, T]$, and some set of states $\{s_t^{(i)}\}_{i=1}^n$. Given such an encryption, the $\texttt{Dec}$ algorithm will simulate, for each i, running the state transformations $\mathbf{State}$ $(T - t)$ times on $s_t^{(i)}$ with input $x_{t+1}^{(i)} = 0$, followed by running **Out** to generate a set of n messages. It then runs **Est** on the collection of n messages, and returns 0 if the answer is smaller than $n/2$, and 1 otherwise. As before, Φ_r on a ciphertext $c = (t, \{s_t^{(i)}\}_{i=1}^n)$ (for $t < T$) will simulate, for each i, running the state transformations $\mathbf{State}$ on $s_t^{(i)}$ with input $x_{t+1}^{(i)} = 0$ to get an updated state $s_{t+1}^{(i)}$. The updated ciphertext c' is then set to $(t + 1, \{s_{t+1}^{(i)}\}_{i=1}^n)$.

As before, $\texttt{Dec}(\texttt{Enc}(b, k_{pub}), k_{priv})$ comprises an exact simulation of running the CountNonZero algorithm on n clients, where the input streams satisfy $\mathbb{1}(\mathbf{x}^{(i)}) = b$, followed by rounding the final estimate. Since the correct answer to CountNonZero on these inputs is bn and the estimation algorithm has error $n/4$ with high probability, we conclude that the encryption scheme satisfies $\texttt{Dec}(\texttt{Enc}(b, k_{pub}), k_{priv}) = b$ except with negligible probability.

Now Lemma 3.2 implies that if the streaming algorithm was information-theoretically ε-locally pan-private, then the TV distance between the distributions $\texttt{Enc}(0, k_{pub})$ and $\texttt{Enc}(1, k_{pub})$ would be $O(\varepsilon/\sqrt{T})$. Mironov et al. (2009) show that computational (ε, δ)-indistinguishability (Definition 2.9) is equivalent the existence of distributions D_0 and D_1 such that D_0 and D_1 are $O(\varepsilon/\sqrt{T})$-indistinguishable, and D_b and $\texttt{Enc}(b, k_{pub})$ are computationally indistinguishable. It follows that $\texttt{Enc}(0, k_{pub})$ are $\texttt{Enc}(1, k_{pub})$ are computationally $(0, O(\varepsilon/\sqrt{T} + T \cdot \texttt{negl}(\lambda)))$-indistinguishable.

The rerandomization, as before, is essentially simulating one step in the decoding. It is therefore immediate that for any valid ciphertext, $\texttt{Dec}(\Phi_r(c, k_{pub}), k_{priv})$ and $\texttt{Dec}(c, k_{priv})$ are running exactly the same simulation and thus are identically distributed. Finally, the security of the rerandomization follows once again from the hybrid argument: if we could distinguish iterated rerandomizations of 0 from those of 1, we would get a distinguisher that can learn the b from the local state at some later time step t.

Finally, we argue computational efficiency. The $\texttt{Enc}$ and $\texttt{Dec}$ algorithms need to simulate applying $\mathbf{State}^\tau$ on input 0^τ, for a suitable τ, at most $2n$ times each. For a state space of size S, the transformation is defined by a stochastic matrix M of size $2^S \times 2^S$. Applying this Markov kernel defined by M τ times is equivalent to applying M^τ. Since computing M^τ can be done in time $poly(2^S) \cdot \log \tau$ by repeated squaring, and thus we get the claimed run time for the scheme. □

D. Deferred Pseudocode for algorithms

Algorithm 5 Counter, Client Algorithm

```
Require: $k, T, \texttt{Enc}, \Phi_r, \mathbf{x} = x_1, x_2, \ldots, x_T$.
 1: Initialization
 2: $c_0 = \texttt{Enc}(1)$; $d_0 = \texttt{Enc}(1)$
 3: for $i = 1 : k$ do
 4:     $c_i = \texttt{Enc}(0)$; $d_i = \texttt{Enc}(0)$
 5: end for
 6: State Update
 7: for $t = 1 : T$ do
 8:     if $x_t = 0$ then
 9:         for $i = 0 : k$ do
10:             $c_i = \Phi_r(c_i)$, $d_i = \Phi_r(d_i)$.
11:         end for
12:     else
13:         $c_0 = {\Phi_r}^t(\texttt{Enc}(0))$; $d_0 = {\Phi_r}^t(\texttt{Enc}(1))$
14:         for $i = 1 : k$ do
15:             $c_i = \Phi_r(c_{i-1})$; $d_i = \Phi_r(d_{i-1})$
16:         end for
17:     end if
18: end for
19: Send to Server
20: for $i = 0 : k - 1$ do
21:     $v_i = \Phi_r(c_i)$
22: end for
23: $v_k = \Phi_r(d_k)$
24: for $i = 0 : k$ do
25:     $r = \mathrm{Ber}(2/(e^{\epsilon_0} + 1)$
26:     if $r = 0$ then
27:         $r' = \mathrm{Ber}(1/2)$
28:         if $r' = 1$ then
29:             $v_i = {\Phi_r}^{T+1}(\texttt{Enc}(0))$
30:         else
31:             $v_i = {\Phi_r}^{T+1}(\texttt{Enc}(1))$
32:         end if
33:     end if
34: end for
35: Return $(v_0, \ldots, v_k)$
```

Algorithm 6 COUNTNONZERO, Client Algorithm, Two-server model

```
Require: $T, \texttt{Enc}_1, \Phi_{r1}, \texttt{Enc}_2, \Phi_{r2}, \mathbf{x} = x_1, \ldots, x_T$
 1: Initialization
 2: $(s^{(1)}, s^{(2)}) = \texttt{SecretShare}(0)$
 3: $(\pi^{(1)}, \pi^{(2)}) = \texttt{ValidityProof}(s^{(1)}, s^{(2)})$
 4: $(c^{(1)}, c^{(2)}) = (\texttt{Enc}_1(s^{(1)}), \texttt{Enc}_2(s^{(2)}))$
 5: $(p^{(1)}, p^{(2)}) = (\texttt{Enc}_1(\pi^{(1)}), \texttt{Enc}_2(\pi^{(2)}))$
 6: State Update
 7: for $t = 1 : T$ do
 8:   if $x_t = 1$ then
 9:     $(s^{(1)}, s^{(2)}) = \texttt{SecretShare}(1)$
10:     $(\pi^{(1)}, \pi^{(2)}) = \texttt{ValidityProof}(s^{(1)}, s^{(2)})$
11:     $(c^{(1)}, c^{(2)}) = (\Phi_{r1}^{t}(\texttt{Enc}_1(s^{(1)})), \Phi_{r2}^{t}(\texttt{Enc}_2(s^{(2)})))$
12:     $(p^{(1)}, p^{(2)}) = (\Phi_{r1}^{t}(\texttt{Enc}_1(\pi^{(1)})), \Phi_{r2}^{t}(\texttt{Enc}_2(\pi^{(2)})))$
13:   else
14:     $(c^{(1)}, c^{(2)}) = (\Phi_{r1}(c^{(1)}), \Phi_{r2}(c^{(2)}))$
15:     $(p^{(1)}, p^{(2)}) = (\Phi_{r1}(p^{(1)}), \Phi_{r2}(p^{(2)}))$
16:   end if
17: end for
18: Send to Server
19: $r = \text{Ber}(2/(e^{\epsilon_0} + 1)$
20: if $r = 0$ then
21:   $r' = \text{Ber}(1/2)$
22:   if $r' = 1$ then
23:     $(s^{(1)}, s^{(2)}) = \texttt{SecretShare}(0)$
24:     $(\pi^{(1)}, \pi^{(2)}) = \texttt{ValidityProof}(s^{(1)}, s^{(2)})$
25:     $(c^{(1)}, c^{(2)}) = (\Phi_{r1}^{T+1}(\texttt{Enc}_1(s^{(1)})), \Phi_{r2}^{T+1}(\texttt{Enc}_2(s^{(2)})))$
26:     $(p^{(1)}, p^{(2)}) = (\Phi_{r1}^{T+1}(\texttt{Enc}_1(\pi^{(1)})), \Phi_{r2}^{T+1}(\texttt{Enc}_2(\pi^{(2)})))$
27:   else
28:     $(s^{(1)}, s^{(2)}) = \texttt{SecretShare}(1)$
29:     $(\pi^{(1)}, \pi^{(2)}) = \texttt{ValidityProof}(s^{(1)}, s^{(2)})$
30:     $(c^{(1)}, c^{(2)}) = (\Phi_{r1}^{T+1}(\texttt{Enc}_1(s^{(1)})), \Phi_{r2}^{T+1}(\texttt{Enc}_2(s^{(2)})))$
31:     $(p^{(1)}, p^{(2)}) = (\Phi_{r1}^{T+1}(\texttt{Enc}_1(\pi^{(1)})), \Phi_{r2}^{T+1}(\texttt{Enc}_2(\pi^{(2)})))$
32:   end if
33: end if
34: $(c^{(1)}, c^{(2)}) = (\Phi_{r1}(c^{(1)}), \Phi_{r2}(c^{(2)}))$
35: $(p^{(1)}, p^{(2)}) = (\Phi_{r1}(p^{(1)}), \Phi_{r2}(p^{(2)}))$
36: Return $(c^{(1)}, p^{(1)}, c^{(2)}, p^{(2)})$
```