Intermediate Models of Differential Privacy: Properties, Tradeoffs, and Connections

Jonathan Huml

jhuml@g.harvard.edu

Abstract

Decentralized or "local" notions of differential privacy [1] predate the rigorous, centralized definition of differential privacy [2] by over 50 years. More recent work has explored intermediate models sitting between the centrally-curated and local paradigms, each with varying purposes, trust assumptions, and statistical utility that preserve user privacy. This paper will prepare the reader to explain, compare, and implement the shuffle and pan-private models. Working from the local model, we motivate the need for varying levels of trust and explore when those trust assumptions prove appropriate through both the engineering and formal mathematical perspective. Lastly, we provide an example algorithm comparison to further illuminate the differences in the models.

1 Introduction

In the *central curator* paradigm of differential privacy (DP), a trusted party aggregates raw user data, stores or maintains this data, and releases noisy estimates to user queries using the original $(\varepsilon, \delta)-$DP definition of privacy as given in [2]. Of course, this paradigm implies that the honest curator is able to aggregate, store, and maintain raw user data in a cryptographically secure fashion. In contrast to the centrally-curated paradigm, the *local* paradigm assumes no trust at any such point in the data pipeline: users run a randomizing mechanism $\mathcal{R}$ on their individual data before sending it to an (untrusted) aggregator [3]. This lack of trust statistically decouples user-level data from its original sender and preserves $(\varepsilon, \delta)-$DP privacy regardless of subsequent use. However, the noisy message may still be linked to the user, and this statistical decoupling comes at a heavy price. Even for simple tasks like private distributed summation or histogram construction, the local model incurs additive errors that scale with the square-root of n or square-root of both n and $\log(d)$, respectively, whereas this error is constant (independent of n or dimension d) in the central paradigm [4] [5] [6].

This scaling greatly limits the statistical utility of the local model to all but the massive "data black holes" like Google [7] and Apple [9]. Even with a massive population of users, the local model maintains difficulties with statistical utility and security concerns as result of this scaling. As the $\varepsilon-$budget of any randomized algorithm decreases or the dimensionality d of the data increases, the aggregator must become more sensitive to the signal inherent in the data. Because local differential privacy (LDP) considers *all* differing inputs x, x' in the input domain, as opposed to the "change-one" definition the central model, this signal is especially weak. While the central curator may consider entire databases that merely differ by one row, the analyzers of an $\varepsilon-$LDP protocol must consider an aggregation of individual users whose data is completely divorced from its senders given that each user is directly linked to their own data. Finding the "needle in the haystack" is thus a difficult statistical question, but also presents usability concerns from an engineering perspective. An adversary that can change the distribution of secure user messages, even if only slightly, can completely destroy the utility of the data. Due to the local protocol noise scaling, such a shift may only require a handful of adversaries that lie about their inputs. For any noninteractive protocol with m dishonest users, the distribution of an $\varepsilon-$LDP algorithm can be skewed in a *manipulation* *attack* by a factor of $\Theta\left(\frac{m\sqrt{d}}{\varepsilon n}\right)$ [19]. While the local model is explainable to users and often computationally efficient to implement, further mechanisms for ensuring honest user-level interaction is necessary in any framework that assumes an adversary will exploit any possible vulnerability if given an avenue to do so.

As such, a "fully-trusted" and "trustless" dichotomy of differential privacy may only be possible in a narrow (or less charitably, imaginary) sense. Even for "trustless" local paradigms, the protocol implementer may also be the one receiving the data as in the case of Apple [9]. Whether at the level of the user or curator, we must place trust somewhere in varying degrees along the data pipeline. Overcoming manipulation attacks, for example, requires ensuring honest user behavior at the level of computation or, more specifically, efficient cryptographic techniques for each node in a distributed system. Intermediate models attempt to balance these tradeoffs by placing this point of trust somewhere between the user and curator. This placement may overcome one problem while exacerbating another, and this paper will attempt to highlight these tradeoffs where they exist. We will explore, in depth, two of these intermediate paradigms: the *shuffle* and *pan-private* models. The *shuffle model* [18] is a specific instance of the more general Encode, Shuffle, Analyze (ESA) architecture that replaces the trusted curator with a trusted shuffler. In the *pan-private* model [27], an algorithm receives a stream of raw data, incrementally updating its internal state with information from a new element in an online setting. The conclusion of the stream results in a differentially private function of the internal states, where at one point, any one of these internal states may be leaked to an adversary. Thus, in contrast to the shuffle model, whose definition is closer to local or central definitions with differing post-processing, the pan-private model attempts to make the joint distribution of any possible internal state and the output insensitive to individual elements of the stream. While the use cases of these models seem orthogonal, we will explore surprising connections between the two, as well as tie each respective frameworks back to the more familiar central and local models.

Lastly, in §8, we provide an example of uniformity testing algorithms in the various paradigms. We show that the sample complexities imply a notion of ordering amongst these algorithms in terms of statistical utility. In the paper, we introduce the paradigms in the sequence of that utility starting from the poorest scaling in terms of error or sample complexity: from the local model in this introduction, we move to the shuffle, robust shuffle, and then pan-private paradigms, exploring the necessary trust assumptions along the way.

2 PROCHLO: Encode, Shuffle, Analyze (ESA)

The Encode, Shuffle, Analyze (ESA) [11] is a flexible privacy-preserving framework built by Google to monitor client software behavior. While the shuffle model is indeed an instance of ESA, the architecture is broad enough to discuss the shuffle, local, and central models in its context, and also provides an optimal starting point to discuss the engineering challenges of differential privacy in a practical setting.

2.1 Three Generations of Systems: Why PROCHLO?

We consider distributed systems in a DP sense across three "generations" of approaches [12]. As motivated by the example of manipulation attacks, there exists an imperative to emulate the central paradigm beyond direct statistical utility, instead emphasizing cryptographic and security-related factors. Software attestation is of critical importance to defend against malicious code, especially as compute and distributed systems become simultaneously ubiquitous. The seminal DP definition is useful precisely *because* it is limited to an information-theoretic perspective. However, this implies that, in practice, differential privacy is always embedded into larger systems of which it is one subcomponent. Though we will consider the DP formalisms in subsequent sections, there exists a lineage of differential privacy systems built for the wild, each balancing statistical utility, computational feasibility, maintainability, and explainability. The first generation of these systems, secure multiparty connection [13], directly attempted to simulate a central paradigm through $n-$party protocols that, *in principle*, could implement any desired query or computation through distributed interactivity.

Noise generation is cooperatively shared and collected across participants, eliminating the need for a trusted administrator while retaining the required noise level to meet $(\varepsilon, \delta)-$DP given that at least two-thirds of participants are honest. By securing processing channels through encryption, adversaries become limited to (probabilistic) polynomial time computations, which allows values to be shared, verified, and reconstructed. By analogy, each party holds a broken shard of plate, and even if a fraction of those shards come from different plates, the whole plate can be glued back together and used at no loss in utility. While this approach is backed by a rigorous framework, this modification of *in principle* can be tenuous in practice: with many moving parts, this is not only computationally difficult but often unclear how to "glue" each node value back from algorithm to algorithm. This motivates the second-generation of distributed systems with untrusted servers, or local frameworks.

Despite the well-documented shortcomings of the local protocol, there are several benefits to this second-generation of distributed systems. While the addition of noise at a user level is cheap and efficient, the collection process largely avoids the pitfalls of secure multi-party computation, as aggregation requires no special structure. Furthermore, there exists a rigorous formalism behind the framework that also happens to be highly explainable to all involved stakeholders. Thus, despite the downsides, there exists several reasons for such a system in practice, which has a variety of implementations. Most notably, we highlight RAPPOR [7], a previous privacy-preserving software monitoring tool also built by Google. The RAPPOR framework ensured pure local differential privacy (LDP) data without any assumptions of client-side trust. This introduced two specific problems of note: first, even with hundreds of millions of users, the utility of this LDP data was limited to very specific use cases (namely, very common problems of high mass concentration, like a power distribution); second, developers and clients may have heterogeneous data pipelines with varying permissions, privacy guarantees, existing tools and processes, which makes unclear the ability to combine this data for statistical insight. In the task of measuring application programming interface (API) usage, for example, several thousand applications and hundreds of APIs may be of relevance: to obtain a clear signal in such a scenario requires over one hundred times the number of humans on Earth [8].

As such, a third-generation of distributed systems has arisen to combine aspects of differential privacy with fast, efficient cryptography. ESA builds upon the shortcomings of RAPPOR, sharing similar goals to earlier cryptography-based privacy-protection and hybrid systems like BLENDER [14], federated learning [15], or Prio [16]. BLENDER considers a combination of users that contribute to either of the central or local model (termed "opt-in" or "client" users, respectively). In the context of local search, this dichotomy has uses when beta versions of software are released, and privacy preferences may differ among early adopters. The data is then funneled through a *blending* stage to extract information from the union of these two user groups. Federated learning was conceived in the context of shared machine learning, where a central server coordinates a network of devices that locally store training data and locally run, for example, stochastic gradient descent on this training data. This allows the devices to send the updates instead of the data. The problem of *secure aggregation* is the process of combining these updates (which may themselves be sensitive) for server use. This is done, roughly speaking, by batching the data in a structured fashion. Prio computes aggregate statistics through a variant of verifiable computation, where the client must prove correct function execution to the server. As such, these frameworks incorporate aspects of the first and second-generation technologies. From ESA, however, a new formalism emerges.

2.2 ESA Framework

The key insight of ESA is that the noise scaling under a system like RAPPOR can be avoided by *partitions* of correlated data with nested encryption to ensure that only user-trusted parties are granted processing permission for analysis. The framework then also overcomes the second difficulty of RAPPOR (heterogeneous pipelines) by explicitly mapping permissions from clients to servers. Thus, ESA is designed in modular fashion. This makes the framework highly flexible: differential privacy can be ensured at any point of control, or may not be ensured at all if desired. These points of control can be decomposed as follows:

1. *Encoder*: The client-facing point of control is the encoding step. Users specify trust assumptions through a nested encryption step, granting access permissions and transmitting prepared data to a network of trusted shufflers. Data can be prepared through adding noise or fragmentation or any desired transformation, and is then marked with a crowd ID for use by the specific shuffler.

2. *Shuffler*: The trusted shuffler, assumed to be honest but curious, has access to this ID and a host of metadata associated with the user, which is useful for admission control. However, the shuffler removes the metadata for anonymization, and uses the crowd ID to batch the data via thresholding: the shuffler queues the stripped, shuffled data and forwards this data only when a (potentially randomized) threshold has been met, which prevents adversarial analyzers from timing and observing the ordering of the queue. Note that there are potentially many shufflers receiving a single instance of randomized user data.

3. *Analyzer*: After the shuffler forwards the anonymized batch, the analyzer decrypts, stores, aggregates, releases, or potentially attacks the received batch. In the actual PROCHLO implementation of ESA, there are almost always keys associated with a specific analysis. While the publicity of the analyzer module can vary from application to application, permissions are often restricted only to a small subset applications and APIs. Therefore, attacks may be considered in the post-analysis stage if ones wishes to conceptualize all three ESA modules as one closed system as opposed to a closed encoder-shuffler framework.

The ESA framework is thus general enough to consider the collapse or collusion of combinations of the points of control as three paradigms of differential privacy–central, local and shuffle models–given that noise is indeed added at the appropriate point of control. The collusion of an adversarial analyzer and shuffler reduces to the local model when noise is added in the encoding step; full trust at the encoding, shuffling, and analyzing steps reduces to the central model when noise is added at the analyzing step; the trust of the shuffler, but not the analyzer, motivates the shuffle model and, essentially, its variants of the pure shuffle paradigm (trusted encoder) and robust variant (untrusted encoder).

2.3 Shuffling: Trust and Hardware

As a lightweight cryptographic implementation, PROCHLO introduces unique cryptographic primitives and an *oblivious shuffling mechanism* to further guarantee user privacy. While the encoder may enforce LDP, the shuffler is fully trusted and thus a central aspect of the ESA framework and its implementation. These contributions enable two possibilities of note: first, with trusted hardware, the shuffler and analyzer may be hosted by the same organization; second, by cryptographic blinding, the shuffler may be distributed across a network of parties. PROCHLO utilizes Intel's Software Guard Extensions (SGX) [17] to, in principle, eliminate the need for a distinct trusted third party. With SGX, a user could transmit data to a shuffler hosted by the analyzer even if the analyzer is untrustworthy, given the hardware to perform remote secure computation. This is just one possible approach to the software attestation problem motivated earlier in the first generation lineage, but presents several problems of note. First, this particular hardware is vulnerable to a subset of attacks (passive address translation attacks, firmware attacks on the Management Engine, etc.), and as such, may not be fully trustworthy. Second, the private memory constraints are often far too limited for scalable systems the size of Google's, and the shuffling mechanism must therefore be as efficient as possible. As such, PROCHLO presents a new oblivious shuffling algorithm for this purpose. In general, these algorithms compare and swap items in a data-independent fashion, whose high-level motivation is much like a simple sorting algorithm. However, given the scalability of such sorting primitives in the context of limited memory constraints, the PROCHLO shuffler manipulates encrypted items in buckets small enough to fit in private memory. The specific shuffle protocol will dictate the number of messages and bits per message sent, i.e. the size and number of these items. For example, there exist summation protocols that require multiple single-bit messages, and others that require single multiple-bit messages. Therefore, the number of rounds required for this memory bucketing can vary based on the specific use case at hand. The shuffling mechanism executed on this hardware is also of critical importance since the permutations on the user data are done with public operations. Even if the contents of the computation are hidden correctly by SGX, an adversarial shuffler could still track and reverse engineer the actual operations.

Lastly, the shuffler also performs (potentially randomized) cardinality thresholding on the batches of stripped, shuffled data or otherwise *count* and *filter* the crowd IDs. However, in the case that the analyzer hosts the shuffle, the distributions or values of the crowd IDs must remain hidden to the greater organization. These IDs are crucial since they specify user permissions and route messages to various shufflers, but they also allow for direct linkage to a specific user. As we explore in future sections, this shuffling step introduces privacy amplification that allows us to add less noise at the encoding step. Taking advantage of this result without securing crowd IDs, however, results in worse privacy guarantees for users. PROCHLO allows the organization to see the filter threshold since the batch size would ostensibly be visible to the analyzer anyway during the forwarding process, but SGX allows this thresholding computation to fit in private memory.

3 The Shuffle Model

The shuffle model is a particular instance of ESA with a rigorous formalism developed in the past few years. In the shuffle model, users add noise directly to their data, which is then sent to a trusted shuffler and subsequently released to the public. For randomizer $\mathcal{R}$ and analyzer $\mathcal{A}$, this is defined as follows:

Definition 1 (Shuffle Differential Privacy): A protocol $\mathcal{P} = (\mathcal{R}, \mathcal{A})$ is (ε, δ)-shuffle differentially private if, for all $n \in \mathbb{N}$ and $w \in \{0,1\}^r$, the algorithm $(\mathcal{S} \circ \mathcal{R}^n)(\vec{x}) := \mathcal{S}(\mathcal{R}(x_1, w), \dots, \mathcal{R}(x_n, w))$ is (ε, δ)-differentially private. [18]

Note here that the (semi)public random bit w can be fruitfully conceptualized as the crowd ID from the ESA encoder step; this may or may not be sensitive in the context of the shuffler, though obviously should never be revealed to the analyzer. Typically w and $\mathcal{S}$ are either dropped or abstracted in proofs of statistical utility: by post-processing, the composition will be at least as private as the randomizer $\mathcal{R}$ allows, and we assume that the shuffler is indeed honest. Note also that this definition has slight variations in practice, but is still labeled "shuffle-DP." Among the first privacy amplification results [20] applies $\mathcal{R}^n$ after $\mathcal{S}$. For a single fixed randomizer, the two definitions are indeed equivalent. However, the "shuffle-then-randomize" definition may be of relevance with *sets* of randomizers. Consider the case of k distinct randomizers. Each could individually be ε_k-LDP but have disjoint output spaces over the set, such that their combination could only be kept differentially private with initial anonymity in the shuffling step. In practice, because ESA is a true data pipeline with more moving parts, seemingly small details like these can introduce more potential failure modes. However, that flexibility manifests itself in design decisions that can overcome limitations of the local or central model.

The interaction between the shuffler and the randomizer provides *privacy amplification* by allowing the individual users to "hide" among the crowd or batch. By trusting the shuffler, we can expect to gain some additional statistical utility even with users applying direct noise via $\mathcal{R}$. Regardless of the composition order, however, this definition implicitly assumes that all users follow $\mathcal{P}$ honestly in addition to the trusted shuffler. Consider a rather extreme case where adversaries control $n-1$ of the n inputs x_i in batch b. By failing to execute $\mathcal{R}$ on the $n-1$ entries, the output $(\mathcal{S} \circ \mathcal{R}^n)(\vec{x})$ can simply be differenced with the set $(x_1, \dots x_{n-1})$ to obtain the honest user's data. This user's data will no longer satisfy the same level of differential privacy as before, especially if the local randomization noise level is known across all users. Thus, similarly to the local model of DP, the encoder step may not necessarily be trusted. By "dropping out," adversaries can attack the shuffle model even with a cryptographically secure shuffler. This motivates the notion of *robust shuffle privacy*, which is a stronger paradigm than the shuffle model.

Definition 2 (Robust Shuffle Privacy): For continuous, positive, and non-increasing functions $\tilde{\varepsilon}(\gamma) < \infty, \tilde{\delta}(\gamma) < 1$ for all $\gamma \in [\tau, 1]$, $\tau > 0$ a shuffle protocol $\mathcal{P} = (\mathcal{R}, \mathcal{A})$ is $(\tilde{\varepsilon}, \tilde{\delta})-$robust shuffle-DP for n users such that for $\lceil \gamma n \rceil \in \mathbb{N}$, the algorithm $\mathcal{S} \circ \mathcal{R}^{\lceil \gamma n \rceil}$ is $(\tilde{\varepsilon}(\gamma), \tilde{\delta}(\gamma))$-DP. Essentially, γ fraction of the users follow the protocol. We will denote this as $(\varepsilon, \delta, \gamma)$-robust shuffle DP [18]

While shuffle privacy does not imply robust shuffle privacy, many algorithms that are shuffle private are robustly shuffle private (or otherwise may degrade linearly with m). The shuffle-private algorithm presented in §8 that is derived from Algorithm 2 does not satisfy robust-shuffle privacy, however, and typically we observe different sample complexity or error scaling for each. More details on the robust shuffle private uniformity tester can be found in [33]. Though we will focus on the single-message, noninteractive shuffle protocol for the sake of simplicity and clarity, multiple rounds of communication can greatly enhance protocol ability and are the focus of many recent, more cutting-edge results. There are two flavors of local protocol interactivity, sequential and full, that inform the potential shuffle modifications. In the former, a user only sends one message to analyzer, but the encoding step can depend on the (private) totality of messages sent by other users. In the latter, a user can communicate with an analyzer multiple times, where the totality of communications must be kept differentially private. The fully interactive shuffle protocol has been shown to be able to simulate the central model: for any randomized algorithm $\mathcal{M}$ that is (ε, δ)-DP in the central model, there exists a two-round fully interactive shuffle protocol that simulates $\mathcal{M}$ [22].

3.1 Separations and The Privacy Blanket

A central notion for intermediate models is that of the "privacy blanket," which decomposes any local randomizer $\mathcal{R}$ into a linear combination of a "blanket" (or pure noise) $\mathcal{B}$ and user-dependent distribution $\mathcal{D}$ [10]. Specifically, there exists a parameter p such that $\mathcal{R}(x) = p\mathcal{B} + (1-p)\mathcal{D}(x)$. Approximately pn parties submit pure noise, while $(1-p)n$ parties submit their true value. This corresponds to a histogram on the union of these two sets of parties. We can thus view all shuffle-private algorithms as a variant of secure addition, where the privacy is derived from appropriately calibrating p to provide $(\varepsilon, \delta)-$DP, which will necessarily depend on the algorithm at hand. For example, an ε_0-LDP Laplace mechanism on the unit interval satisfies $p = e^{-\varepsilon_0/2}$, which we note is a decreasing function of ε_0; that is, for larger privacy budgets, we can add less noise to the blanket decomposition. This value is computed for general measures on $\mathcal{R} : \mathbb{X} \to \mathbb{Y}$, where we denote μ_x as the output distribution. In [10], the authors consider a characterization of the total variation distance over these measures, defined as:

$$D_{TV}(\mu || \mu') = 1 - \int \min\{\mu_y, \mu'(y)\} dy$$

The key insight of the paper is to generalize from pairs μ, μ' to arbitrary sets of measures. For a set $\Lambda = \{\mu_x\}_{x \in \mathbb{X}}$ over $\mathbb{Y}$, the total variation similarity is defined as:

$$p_\Lambda = \int \inf_x \mu_x(y) dy$$

This notion is thus general, which yields:

Theorem 1: Every ε_0-LDP randomizer $\mathcal{R}$ admits a unique maximal mixture decomposition where one of the components is independent of the input, for $\mathcal{R}(x) = p\mathcal{B} + (1-p)\mathcal{D}(x)$ and $e^{\varepsilon_0} \leq p \leq 1$ [10]

A key result of [10] is a shuffle-private optimal summation protocol for bounded-value sums, which shows that the shuffle model sits squarely between the local and central models for linear problems. As we recall from the introduction, the mean-squared error is $\Omega(\sqrt{n})$ for the local model and $\mathcal{O}(\frac{1}{\varepsilon^2})$ for the central model. For single-message, shuffle-private protocols, however, the optimal summation protocol achieves mean-squared error of $\Omega(n^{\frac{1}{3}})$. Because the given estimator is unbiased, this essentially gives the scaling of the variance of the protocol estimator.

4 The Pan-Private Model

As motivated in the ESA framework, we can view the central, local, and shuffle paradigms as variations of trust in the encoder, shuffler, and analyzer, where the shuffler does indeed exist at all. In essence, these paradigms are combinations of compositions of differentially private functions, where the "privacy wall" is enforced at the appropriate step. The pan-private model fits less clearly into this interpretation–though as we will explore, there are non-obvious and surprising connections between the paradigms. In a *streaming algorithm*, user data x_t populates a server or queue at time $t \leq T$. The server then computes on x_t with some algorithm $\mathcal{A}$, which produces an internal state S_t, or $\mathcal{A}(x_t) = S_t$. At the end of stream, where $t = T$, the protocol outputs some Z. Previous approaches to this problem included techniques like subsampling: much like a shuffler, a curator could batch inputs and randomly select sample from this batch. While the probability of selecting any one individual decreases inversely with respect to the batch size, timing attacks and breaches are significant problems of this approach.

In the pan-private model, we *mostly* trust this server, but with an essential caveat: while a potential adversary sees Z as in any of the typical paradigms, at any one point t, the internal state S_t is completely exposed. We assume that the data x can be discarded after server processing such that the adversary cannot access the inputs at t. Thus, we trust the curator to honestly collect and process our data, but not store it in perpetuity. This scenario may be reasonable or likely under changes in or the pressure of law (considering subpoenas or mandated transparency, for example), human factors (privacy policy reversal), or non-authorized access to the stream, whether internal or external to the curator. The intention of the pan-private model is to ensure that the joint distribution of the output and any internal state is private in a DP sense. Unless otherwise stated, we allow for the adversary to fully access only one of these internal states S_t in addition to Z, but there exist several results for multiple and continual intrusions.

Definition 3 (Pan-Privacy): A protocol is (ε, δ) pan-private if the protocol $\mathcal{P}_t(x) = (\mathcal{A}_t(x_{1:t}), Z(x))$ is $(\varepsilon, \delta)-$DP for all $t \in [T]$ [27]

Pan-privacy has various levels of interactivity much like the shuffle or local model. In this work, we highlight two: *user-level* and *record-level* pan-privacy. In the former, one user may contribute multiple elements to the stream, while in the latter, neighboring streams differ in at most one element. In the results and figures presented here, we focus on the latter definition.

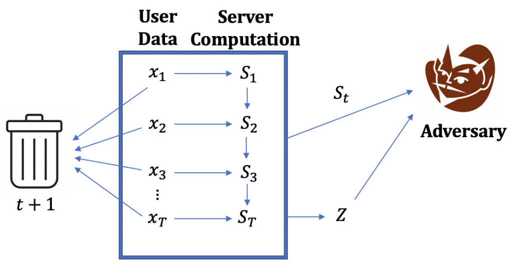

Figure 1: A pan-private data flow. There now exists a time-dependency to the stream, and the adversary also gets complete access to one state S_t for some $t \in [T]$

We also note that there are differing levels of adversarial access across variations of pan-private models. While we present a definition (and algorithms) for the case where only one internal state is fully exposed, more leaks can be incorporated into the framework at the cost of more noise injected into the stream.

5 Connections

This survey has largely focused on the trust assumptions of the various paradigms because there may exist variants of each respective model that allow us to simulate the statistical utility of the other models within that particular framework. While this is not always the case as, for example, we simply cannot perform certain tasks in the local model that we can in the central model, intermediate models exhibit adaptability even where the connections are not immediately obvious.

5.1 Shuffle and Pan-Privacy

In the shuffle definition we have considered, a database is distributed amongst n users, where each user holds exactly one element. In the case of record-level pan-privacy, where neighboring streams differ in at most one element, one user contributes data x_t at time t, which is then discarded after updating the internal state of the stream. By the definitions considered, it is unclear whether or not the respective use cases can even translate, much less if one model implies the other. Due to a clever construction, however, we can simulate robust shuffle-private protocols in a pan-private protocol, thus implying a notion of ordering between the shuffle and pan-private models, given that the robust shuffle definition does not necessarily imply pure shuffle privacy. However, we note that generic structural conversions between the robust shuffle and pan-private algorithms are lacking in the literature.

Theorem 2: Let $\mathcal{P} = (\mathcal{R}, \mathcal{A})$ be an $(\varepsilon, \delta, \frac{1}{3})$-robust shuffle DP $\alpha-$uniformity tester with sample complexity n. Given more than n elements, there exists an $(\varepsilon, \delta)-$DP pan-private tester with sample complexity $\frac{n}{3}$ built from this robust shuffle tester. [23]

In this construction, the pan-private algorithm maintains its internal state with shuffled messages, using a combination of actual user data and $\lceil 2n/3 \rceil$ draws or "dummy noise" $\mathcal{U} \sim \mathcal{R}(1)$. The stream is initialized with $\lceil n/3 \rceil$ of these draws, processes the stream x_t, and is finally capped with another set of $\lceil n/3 \rceil$ draws before outputting Z. This dummy noise can be conceptualized as a set of malicious users m from the robust shuffle privacy definition. As we will see below in §6, this idea surfaces in actual algorithm construction for pan-privacy. We can initialize and cap private computations with noise, which allows us to effectively treat the stream of data as a single batch instead of online, sequential timesteps. This is how we treat our pan-private algorithms in the code in §8.

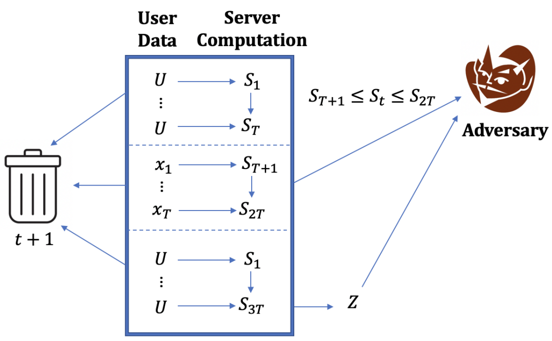

Figure 2: Simulating the robust shuffle model in the pan-private streaming problem. Note that the in this construction, the adversary sees an exposed internal state from the middle (i.e. non-uniform) third of the stream

6 Uniformity Testing: An Example

Hypothesis testing is a standard statistical procedure. In uniformity testing, we reject or fail to reject the null hypothesis that some data $\mathcal{D}$ with m elements is uniformly distributed. Considering a continuous distribution with $d \in \mathcal{D} \subset [a, b]$, a simple (non-differentially private) test for uniformity bins $\mathcal{D}$ into k bins, taking the difference between the expected number of elements in bin k (i.e. $\frac{m}{k}$) and the actual number of elements z_j in bin k, squaring this difference, and dividing by the expected number of elements again [30]. We then sum over these elements to obtain a statistic:

$$Z^2 = \frac{\sum_{i=1}^{k}(z_j - \frac{m}{k})^2}{\frac{m}{k}}$$

that has distribution $Z^2 \sim \chi^2(k-1)$, which can then used to obtain a $p-$value for the desired confidence level. We will present and compare a handful of algorithms to compute a variant of this statistic in a differentially private manner. The problem statement in the privacy literature here is a bit different than this initial formulation, and illuminates some of the statistics versus computer science tradeoffs we often make under various constraints. In $\alpha-$uniformity testing, we attempt to report "uniform" with probability $\frac{2}{3}$ with $\mathcal{D}$ comes from uniform $\mathcal{U}$, and "not uniform" with probability $\frac{2}{3}$ when $||\mathcal{D} - \mathcal{U}|| > \alpha$ [26]. The main focus of this problem formulation (in order to meet these two requirements) is that of sample complexity: how many m are required to achieve this under differential privacy? In contrast to the original χ^2 formulation, we are only interested in obtaining a clear separation bound given we have "enough" samples. This does not imply that the resulting (pseudo) χ^2 statistic is "good" in a statistical sense, however (along the dimensions of variance, consistency, etc.). Most of the papers in which these algorithms are presented do not even report (or at least highlight) these standard estimator properties at all!

In a real-world hypothesis test, of course, we likely do not have access to an unbounded amount of data. Data collection is expensive and error-prone. A more practical use case for these sets of algorithms is checking whether or not a random number generator is truly random: here, data generation is cheap and sample complexity is (mostly) no big deal for low-polynomial sample complexity. Even still, several problems may arise. In [31], the authors consider a Laplaced χ^2 where $Y_i \sim \text{Lap}(\frac{1}{\varepsilon})$ is added directly to z_j before the uniformity test is applied. However, the variance of this statistic is bounded below by $\frac{20k^3}{\varepsilon^4 m^2}$ even when $\mathcal{D}$ is truly uniform. While the cubed term in the numerator looks scary, the sensitivity of these statistics (and variants thereof) are also quite high, so the scaling on ε can be gruesome as well. We provide a summary of the complexities below, and delve into more details of the algorithms:

	Local*	Robust-Shuffle	Pan-Private*	Central*
Complexity in m	$\Theta\left(\frac{k}{\alpha^2\varepsilon^2}\right)$	$O\left(\frac{k^{3/4}}{\alpha\varepsilon}\sqrt{\log\frac{1}{\delta}}\right)$	$\Theta\left(\frac{k^{2/3}}{\alpha^{4/3}\varepsilon^{2/3}}\right)$	$\Omega\left(\frac{\sqrt{k}}{\varepsilon^2}\right)$

Table 1: Algorithms marked with an asterisk denote optimal protocols. Note that k denotes the number of bins, α is the separation coefficient, and ε and δ are the usual privacy parameters in DP. The scaling with k shows clear separations between the various uniformity testers.

In this paper, we examine and implement [1] two specific uniformity testing algorithms that show the differences in sample complexities across the local, shuffle, and pan-private models. In the code, we also go more in-depth on the statistical properties of the estimators that inform their practical utility. The main differentiating factor across these algorithms is the threshold computation, which is derived to achieve the probability separation. Threshold aside, the algorithms are otherwise simple and conceptually identical: they operate by creating a noisy histogram and computing a bias-corrected version of this χ^2 statistic. The first algorithm is a pan-private uniformity tester that operates via a "Poissonification" of the stream that effectively randomizes its starting or stopping point, adding Laplace noise at its beginning and end to protect its elements and output, respectively. This algorithm achieves sample complexity $m = \Omega\left(\frac{k^{3/4}}{\alpha\varepsilon} + \frac{\sqrt{k}}{\alpha^2}\right)$, which is not *quite* optimal, but [28] considers a fine partition of the domain $[k]$ that achieves sample complexity $\Omega\left(\frac{k^{2/3}}{\alpha^{4/3}\varepsilon^{2/3}} + \frac{\sqrt{k}}{\alpha^2} + \frac{\sqrt{k}}{\alpha\varepsilon}\right)$. We consider the suboptimal protocol here for the sake of simplicity, but this improved version only requires a modified threshold in practice.

[1] Code for our implementations can be found at `https://github.com/jonathanhuml/cs208_final_project`

The Poisson sampling step of Algorithm 1, in particular, may be a point of confusion. Our interpretation here is that we access to *at most* m' samples. Thus our data is technically of size m' instead of m. However, we randomly start or stop the stream depending on the relative ordering of m and m', which affords extra privacy protections. Since the center of this randomization is m itself (also consider that the number of cases where $m > m'$ and $m < m'$ is roughly symmetric as $m \to \infty$), we consider the sample complexity in terms of m instead of m'.

Algorithm 1 Pan-Private Uniformity Tester [28]

Require: $\varepsilon > 0$, domain $[k]$, closeness parameter α, data $\vec{x} \in \mathbb{R}^{m'}$
  Sample $m' \sim \text{Poisson}(m)$
  Set threshold $T_U = \frac{\alpha^2 m}{100} + \frac{4k^2}{\varepsilon^2 m} + 24\sqrt{2}\frac{k^{3/2}}{\varepsilon^2 m} + 16\sqrt{2}\frac{k}{\varepsilon\sqrt{m}} + 8\sqrt{2}\frac{k^{3/2}}{\varepsilon m}$
  Initialize private histogram $H \leftarrow \text{Lap}(\frac{1}{\varepsilon})^k \in \mathbb{R}^k$
  for $t \in [m']$ **do**
    if $x_t \in H_k$ **then**
      $H_k \leftarrow H_k + 1$ ▷ Increment the appropriate histogram bucket
    end if
  end for
  $H \leftarrow H + \text{Lap}(\frac{1}{\varepsilon})^k \in \mathbb{R}^k$
  $Z' \leftarrow \sum_{i=1}^{k} \frac{(H_i - \frac{m}{k})^2 - H_i}{\frac{m}{k}}$ ▷ Compute pseudo-χ^2 statistic
  if $Z' > T_U$ **then**
    Output "non-uniform"
  else
    Output "uniform"
  end if

The local and shuffle models are derived from Algorithm 2 [29], which are separate from Algorithm 1. However, we note that with uniform noise placed in appropriate proportions at the beginning and end of the stream in Algorithm 1, we could, in theory, use Algorithm 2 within Algorithm 1 as we found in §5. This would provide the same level of $(\varepsilon, \delta)-$DP, but the statistical guarantees on the estimators could be (likely are) far worse. The original local algorithm is based on the "private-coin" RAPPOR model, and achieves sample complexity $m = O\left(\frac{k^{3/2}}{\alpha^2\varepsilon^2}\right)$. However, using a "public-coin" mechanism, it is possible to achieve a bound that is instead linear in k, which has also been shown to be optimal [28]. Our shuffle implementation is based on a theorem of [32] that finds an ε_0 for the local randomizer of the shuffle mechanism given a fixed ε for any local mechanism, which is of course increasing function of m: given ε, we can add less noise (have a larger privacy budget) for the shuffle randomizer by the privacy amplification of the shuffler. While this theorem is quite useful, tight bounds for a pure shuffle-DP uniformity tester (to the best of our knowledge) are lacking. In [33], the authors consider a private-coin, robust shuffle mechanism that achieves sample complexity $m = O\left(\frac{k^{3/4}}{\alpha\varepsilon}\sqrt{\log\frac{1}{\delta}} + \frac{\sqrt{k}}{\alpha^2}\right)$.

The main takeaway from this section is the relative ordering of the sample complexities. We see improving sample complexity in m (improving with respect to k) as we move from the local to central models, respectively. Of course, in tying the paper together, we note that many of these paradigms overlap: we can often simulate one framework within another under very structured assumptions. Where the paradigms differ most are the trust assumptions. Here, even with "better" theoretical properties, we may elect to to choose an algorithm with higher sample complexity because the user-facing system is best equipped to handle those trust assumptions. If not for this important point, we could simply stop at the central model (as seen in Table 1). Differential privacy is always embedded into larger systems, and part of the difficulty of engineering these systems is the balance between this sample complexity or noise scaling and the actual use case or potential breaches of privacy that lie outside of the general differential privacy framework.

Algorithm 2 Locally-Private Uniformity Tester (RAPPOR-based) [29] [7]

Require: $\varepsilon > 0$, domain $[k]$, closeness parameter α, data $\vec{x} \in \mathbb{R}^m$

$a_R \leftarrow \frac{e^{\varepsilon/2}-1}{e^{\varepsilon/2}+1}$

$b_R \leftarrow \frac{1}{e^{\varepsilon/2}+1}$

for $i \in [m]$ **do**

$y_i \leftarrow$`ENCODE`(x_i) ▷ Turn x_i into a $k-$ary vector

$z_i \leftarrow$`PERTURB`(y_i) ▷ Flip bits of the $k-$ary vector w.p. b_R

end for

`AGGREGATE`(z_i) ▷ Add the counts and debias, setting $N_x = \frac{\sum_{j\in[k]} z_j - mb_R}{a_R}$

$Z' \leftarrow \sum_{i=1}^{k} \left(\left(N_x - (m-1)\left(\frac{a_R}{k} + b_R\right)\right)^2 - N - x\right) + k(m-1)\left(\frac{a_R}{k} + b_R\right)^2$

if $Z' > \frac{m(m-1)a_R^2\alpha^2}{k}$ **then**

Output "non-uniform"

else

Output "uniform"

end if

7 Conclusions

In this paper, we explored recent results on the shuffle and pan-private models. The shuffle model grew from the Encode, Shuffle, Analyze (ESA) architecture, which sought to build upon the shortcomings of the local model and its RAPPOR implementation. In this framework, a trusted shuffler receives locally randomized user input, where the shuffler provides a privacy amplification that allows less noise to be added by $\mathcal{R}$. Before the shuffle model, the pan-private model was conceptualized soon after the seminal, central definition. This online streaming algorithm maintains its privacy even if one of the internal states is completely exposed at any time. We found that the robust shuffle model, a variant of the shuffle model that protects against drop-out attacks, is able to be simulated in the pan-private framework, thus connecting these apparently separate models. Lastly, by considering the use case of uniformity testing, we were able to observe sample complexity separations that show us how each framework scales with respect to one another. Most importantly, beyond the statistical utility of each model, we enumerated the trust assumptions behind each framework. As we saw with ESA, differential privacy is always embedded into larger systems. Thus, while this probabilistic definition gives us a precise language to converse, debate, and budget privacy, the definition may not always directly determine the specific implementation in practice. When is it reasonable to have a trusted shuffler instead of a trusted central curator? Can we trust "trusted hardware"? When are streaming algorithms appropriate? By answering such questions, we may be able to balance statistical utility and trust assumptions as we build differential privacy systems in practice.

8 Code

While tucked away in a footnote on the previous pages, we note that the uniformity testing code can be found at `https://github.com/jonathanhuml/cs208_final_project`

9 Acknowledgements

Thanks a ton for the help from the fantastic CS208 team all semester, especially the comments on the first draft of this paper from Dr. Vadhan, Alibi, and (future Dr.) Sarathy. This course was a privilege to take and I look forward to making more connections to differential privacy in the wild.

References

[1] Warner. "Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias." 1965.

[2] Dwork, McSherry, Nissim, Smith. "Calibrating Noise to Sensitivity in Private Data Analysis." 2006. https://people.csail.mit.edu/asmith/PS/sensitivity-tcc-final.pdf

[3] Kasiviswanathan, Lee, Nissim, Raskhodnikova, Smith. "What Can We Learn Privately?" 2009. https://arxiv.org/abs/0803.0924

[4] Chan, Shi, Song. "Optimal Lower Bound for Differentially Private Multi-Party Aggregation." 2012. https://eprint.iacr.org/2012/373.pdf

[5] Beimel, Nissim, Omri. "Distributed private data analysis: Simultaneously Solving How and What." 2008.

[6] Bassily, Smith. "Local, Private, Efficient Protocols for Succinct Histograms." 2015. https://arxiv.org/pdf/1504.04686.pdf

[7] Erlingsson, Pihur, Korolova, "RAPPOR: randomized aggregatable privacy-preserving ordinal response." 2014. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42852.pdf

[8] Fanti, Pihur, and Erlingsson. "Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries." 2016.

[9] Differential Privacy Team Apple. "Learning with privacy at scale." 2017. https://machinelearning.apple.com/research/learning-with-privacy-at-scale

[10] Balle, Bell, Gascon, Nissim. "The Privacy Blanket of the Shuffle Model." 2019. https://arxiv.org/pdf/1903.02837.pdf

[11] Bittau, Erlingsson, Maniatis, Mironov, et. al. "Prochlo: Strong Privacy for Analytics in the Crowd." 2017. https://arxiv.org/pdf/1710.00901.pdf

[12] Ullman. (2021, October 13). The Limits of Pan Privacy and Shuffle Privacy for Learning and Estimation [Conference presentation]. CMU Theory Lunch, Pittsburgh, PA, United States. https://www.youtube.com/watch?v=HvTZTDT-QkA

[13] Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M. (2006). Our Data, Ourselves: Privacy Via Distributed Noise Generation. In: Vaudenay, S. (eds) Advances in Cryptology - EUROCRYPT 2006. EUROCRYPT 2006.

[14] Avent, Korolova, Zeber, Hovden, Livshits. "BLENDER: Enabling Local Search with a Hybrid Differential Privacy Model." In Proc. of the 26th USENIX Security Symposium (2017), pp. 747–764. https://www.doc.ic.ac.uk/~livshits/papers/pdf/usenixsec17.pdf

[15] Bonawitz, Ivanov, Kreuter, Marcedone, McMahan, Patel, Ramage, Segal, Seth. "Practical Secure Aggregation for Privacy-Preserving Machine Learning." 2017.

[16] Corrigan-Gibbs, Boneh. "Prio: Private, Robust, and Scalable Computation of Aggregate Statistics." 2017. https://www.usenix.org/system/files/conference/nsdi17/nsdi17-corrigan-gibbs.pdf

[17] Coston, Devadas. "Intel SGX Explained." https://eprint.iacr.org/2016/086.pdf

[18] Cheu, Smith, Ullman, Zeber, Zhilyaev. "Distributed Differential Privacy via Shuffling." 2019. https://arxiv.org/pdf/1808.01394.pdf

[19] Cheu, Smith, Ullman. "Manipulation Attacks in Local Differential Privacy." 2019. https://arxiv.org/pdf/1909.09630.pdf

[20] Erlingsson, Feldman, Mironov, Raghunathan, Talwar, Thakurta. "Amplification by shuffling: From local to central differential privacy." 2019.

[21] Cheu. "Differential Privacy in the Shuffle Model." PhD Thesis. 2021.

[22] Beimel, Haitner, Nissim, Stemmer. "On the round complexity of the shuffle model." 2020.

[23] Balcer, Cheu, Joseph, Mao. "Connecting Robust Shuffle Privacy and Pan-Privacy." 2020. https://arxiv.org/pdf/2004.09481.pdf

[24] Cheu, Ullman. "The limits of pan privacy and shuffle privacy for learning and estimation." 2021. https://arxiv.org/pdf/2009.08000.pdf

[25] Joseph. "Differential Privacy beyond the Central Model." PhD Thesis. 2020. https://www.proquest.com/docview/2511196794?pq-origsite=gscholar&fromopenview=true

[26] Cheu. "Differential Privacy in the Shuffle Model: A Survey of Separations." 2021. https://arxiv.org/pdf/2107.11839.pdf

[27] Dwork, Naor, Toniann, Pitassi, Rothblum, Yekhanin. "Pan-Private Streaming Algorithms." 2010. https://www.cs.toronto.edu/~toni/Papers/panprivacy.pdf

[28] Amin, Joseph, Mao. "Pan-Private Uniformity Testing." 2020. http://proceedings.mlr.press/v125/amin20a.html

[29] Acharya, Canonne, Freitag, Tyagi. "Test without trust: Optimal locally private distribution testing." 2019.

[30] Cook, Gelman, Rubin. "Validation of Software for Bayesian Models Using Posterior Quantiles." 2006.

[31] Cai, Daskalakis, Kamath. "Priv'IT: Private and Sample Efficient Identity Testing." 2017. https://arxiv.org/pdf/1703.10127.pdf

[32] Feldman, McMillan, Talwar. "Hiding Among the Clones: A Simple and Nearly Optimal Analysis of Privacy Amplification by Shuffling." 2021.

[33] Cannone, Lyu. "Uniformity Testing in the Shuffle Model: Simpler, Better, Faster." 2021