Composition for Pufferfish Privacy

Jiamu Bai
Penn State University
jvb6867@psu.edu

Guanlin He
Penn State University
gbh5146@psu.edu

Xin Gu
Penn State University
xingu@psu.edu

Daniel Kifer
Penn State University
duk17@psu.edu

Kiwan Maeng
Penn State University
kvm6242@psu.edu

ABSTRACT

When creating public data products out of confidential datasets, inferential/posterior-based privacy definitions, such as Pufferfish, provide compelling privacy semantics for data with correlations. However, such privacy definitions are rarely used in practice because they do not always compose. For example, it is possible to design algorithms for these privacy definitions that have no leakage when run once but reveal the entire dataset when run more than once. We prove necessary and sufficient conditions that must be added to ensure linear composition for Pufferfish mechanisms, hence avoiding such privacy collapse. These extra conditions turn out to be differential privacy-style inequalities, indicating that achieving both the interpretable semantics of Pufferfish for correlated data and composition benefits requires adopting differentially private mechanisms to Pufferfish. We show that such translation is possible through a concept called the $a(b)$-influence curve, and many existing differentially private algorithms can be translated with our framework into a composable Pufferfish algorithm. We illustrate the benefit of our new framework by designing composable Pufferfish algorithms for Markov chains that significantly outperform prior work.

PVLDB Reference Format:
Jiamu Bai, Guanlin He, Xin Gu, Daniel Kifer, and Kiwan Maeng. Composition for Pufferfish Privacy. PVLDB, 19(10): 2790 - 2802, 2026.
doi:10.14778/3828612.3828632

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/cmla-psu/composition-for-pufferfish.

1 INTRODUCTION

Differential privacy (DP) [15] is a gold standard in privacy protection and is deployed in many important real-world applications by organizations such as the U.S. Census Bureau [1, 31], Uber [23, 24], Apple [42], Meta [35, 49], Microsoft [10], and Google [16, 19, 45]. This success is not only due to differential privacy's rigorous guarantees against data reconstruction attacks [13, 15] but also due to its composition properties [25, 33]. Composition measures the total privacy loss associated with a collection of mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$ in terms of their individual privacy losses. If each $\mathcal{M}_i$ satisfies differential privacy with parameter ϵ_i, the combined release of all of their outputs satisfies differential privacy with parameter $\epsilon = \sum_{i=1}^{k} \epsilon_i$ (i.e., the privacy parameter composes linearly). This property critically allows a larger mechanism to be created out of smaller, simpler mechanisms. Without composition, performing a privacy analysis of a complex disclosure avoidance system becomes intractable [21].

The semantics of differential privacy use the hypothetical/counterfactual world framework [43]: for every hypothetical database $\mathcal{D}$ and secret about a person (e.g., the contents of one person's record), there is an associated counterfactual database $\mathcal{D}'$ in which the secret has been completely scrubbed from the database. This counterfactual database is considered a risk-free baseline for that person, i.e., any data release from the counterfactual database $\mathcal{D}'$ poses no risk to that person. The differential privacy constraints on a mechanism $\mathcal{M}$ guarantee that for any pair $(\mathcal{D}, \mathcal{D}')$ of the hypothetical database $\mathcal{D}$ and any of its counterfactual databases $\mathcal{D}'$, inference about a person based on the output of $\mathcal{M}(\mathcal{D})$ would be nearly the same as the inference based on $\mathcal{M}(\mathcal{D}')$.

However, for datasets with correlations, such a guarantee may not be meaningful [26, 27]. For example, consider a medical dataset $\mathcal{D}$ in which one person's medical status (e.g., whether the person has an infectious disease) can have a causal effect on others (e.g., they may infect close relatives, who may then go on to infect others). Removing one person's record creates a counterfactual database $\mathcal{D}'$ that is not necessarily risk-free for that person, as other records in the counterfactual database provide strong clues about the missing record. For differential privacy to provide a meaningful privacy guarantee in such cases, the risk-free counterfactual database must be the one with all the records removed (one contagious individual can potentially affect all records). However, ensuring that a data release from a hypothetical database is similar to a release from this risk-free (empty) database would entirely ruin utility.

Inferential/posterior-based privacy definitions have emerged as an alternative privacy concept for such settings where data are correlated [9, 18, 27, 46]. With these privacy definitions, the data curator specifies a set of distributions for the dataset, and it is ensured that a Bayesian adversary using those distributions would learn little about any target individual. However, posterior-based definitions are difficult to deploy in practice because *they do not guarantee composition*. We show that it is possible to design a mechanism that has zero privacy leakage under posterior-based definitions when run once but has a good chance of revealing the entire database (i.e., no privacy) when run more than once (Section 4).

To counter the lack of composition guarantees, we study the following question, focusing on one of the popular posterior-based

Proceedings of the VLDB Endowment, Vol. 19, No. 10 ISSN 2150-8097.
doi:10.14778/3828612.3828632

privacy definitions, *Pufferfish* [27]: what additional constraints need to be added to Pufferfish in order to guarantee linear composition? We prove necessary and sufficient conditions that establish a formal link between Pufferfish and differential privacy. That is, in order to ensure that the Pufferfish privacy parameter ϵ_{puffer} composes linearly, one should design mechanisms that satisfy differential privacy with some ϵ_{DP} and translate it to Pufferfish privacy. We show that such translation is possible through a concept called the $a(b)$-influence curve, and existing differential privacy mechanisms can be translated into composable Pufferfish mechanisms with our framework. Our result generalizes the previous framework of Song et al. [41], which showed the Laplace mechanism for differential privacy can be used to achieve composable Pufferfish for Markov networks. Thus, our work addresses a major concern hindering the development of query-answering platforms based on Pufferfish—currently, designing a new mechanism requires tedious manual analyses of how it interacts with *all* other supported mechanisms. Linear composition removes this bottleneck, and our theoretical results imply that systems designers should build upon existing DP platforms (and leverage their large library of existing mechanisms) simply by rescaling the privacy parameter. Our contributions are:

- We show that posterior-based definitions generally do not defend against privacy leakage via composition, by constructing mechanisms that have zero leakage under those definitions when run once yet provide no meaningful privacy when run multiple (as few as two) times (Section 4).
- We prove *necessary conditions* for a Pufferfish mechanism to ensure linear composition. Our result shows that a Pufferfish mechanism must meet a class of inequalities that we call the NfC constraints (Section 5), which subsume the differential privacy inequalities. We also show that only a subset of the NfC constraints are post-processing invariant, and those subsets still contain differential privacy-style inequalities but rule out many others. The implication is that Pufferfish should be augmented with differential privacy-style inequalities in order for its privacy parameter ϵ_{puffer} to compose.
- We next study *sufficient conditions* for composition—how to construct composable Pufferfish privacy mechanisms—for general tabular datasets (Section 6). We show that if a mechanism satisfies per-entry ϵ_{DP}-differential privacy, it also satisfies ϵ_{puffer}-Pufferfish privacy with a certain ϵ_{puffer}, and ϵ_{puffer} will compose linearly. We introduce a concept called an $a(b)$-*influence curve*, which links the ϵ_{puffer} parameter to the appropriate ϵ_{DP} parameter. This part is a generalization of the work of Song et al. [41], whose technique was limited to the Laplace mechanism and Markov network priors. We also study how to make the $a(b)$-curve robust to mis-specification of the data generating priors.
- We empirically compare our approach to prior work on composable Pufferfish mechanisms [41] to show that we can answer popularity queries over Markov chains with significantly better accuracy, as our framework allows adapting other differential privacy mechanisms that are better-suited to queries of interest.

2 BACKGROUND AND NOTATION

We use $\mathcal{U}$ as the set of all possible datasets, and $\mathcal{D} \in \mathcal{U}$ to represent a specific dataset. The dataset will be a table where rows are records and columns are attributes, except for Sections 4 and 5, whose results do not need $\mathcal{D}$ to have any particular structure. We use θ to represent a prior distribution over datasets (e.g., a distribution that an attacker may use to make inferences), and Θ for a set of prior distributions. We let *Data* denote a random variable over datasets, so that $\Pr[Data = \mathcal{D} \mid \theta]$ is the probability of the data curator having the dataset $\mathcal{D}$ according to the prior distribution θ.

$\mathcal{D}$	Dataset
$Data$	A random variable representing the true dataset
$\mathcal{U}$	Database domain (set of all possible datasets)
S	The set of all potential secret pairs
s	A potential secret
θ	A prior probability distribution over datasets
Θ	The set of all possible prior distributions
$\mathcal{M}$	A privacy-preserving mechanism or algorithm
ϵ_{puffer}	Pufferfish Privacy budget
ϵ_{DP}	Differential Privacy budget
ω	An output from a mechanism $\mathcal{M}$
$\mathcal{D}_{\mathcal{L}}$	A low-influence part of $\mathcal{D}$
$Data_{\mathcal{L}}$	A random variable over $\mathcal{D}_{\mathcal{L}}$

Table 1: Table of Notation

Protecting privacy in the Pufferfish framework [27] requires specifying what needs to be protected. A potential secret s is a statement, such as "Bob has cancer", which should be protected from an attacker. That is, an attacker should have difficulty inferring whether s is true or not. We use the notation $s(\mathcal{D}) = true$ to indicate that in dataset $\mathcal{D}$, secret s is true. When potential secrets are specified in (mutually exclusive) pairs, like (s_1, s_2), it means the goal is to prevent the attacker from figuring out whether s_1 is true or whether s_2 is true [27]. Similarly, specifying $(s, \neg s)$ as a pair means the goal is to prevent an attacker from learning whether or not s is true. Let S denote a set of potential secret pairs that a data curator wishes to protect.

A mechanism $\mathcal{M}$ is a deterministic or randomized algorithm whose input is a confidential dataset $\mathcal{D}$ and whose output is supposed to be a privacy-preserving data product (e.g., data summaries, trained machine learning models, etc.). We let ω denote the output of a mechanism. The notation we use is summarized in Table 1.

2.1 Differential Privacy

A privacy definition is a set of restrictions that a mechanism $\mathcal{M}$ has to follow. One of the most foundational is ϵ_{DP}-differential privacy, also known as pure differential privacy.

Definition 1 (Pure Differential Privacy [15]). *Given a privacy parameter $\epsilon_{\text{DP}} \geq 0$, a mechanism $\mathcal{M}$ satisfies ϵ_{DP}-differential privacy if, for all pairs of datasets $\mathcal{D}, \mathcal{D}'$ that differ on the contents of one person's record, and all ω in range($\mathcal{M}$),*

$$\Pr[\mathcal{M}(\mathcal{D}) = \omega] \;\leq\; e^{\epsilon_{\text{DP}}} \Pr[\mathcal{M}(\mathcal{D}') = \omega].$$

(Note: the probability is with respect to the randomness in $\mathcal{M}$ only.)

One can think of $\mathcal{D}$ as a hypothetical dataset and $\mathcal{D}'$ as a counterfactual dataset in which the record of one person has been replaced. Differential privacy guarantees that inference about a person based

on the output $\mathcal{M}(\mathcal{D})$ would be similar to inference based on $\mathcal{M}(\mathcal{D}')$. Hence, if the person's influence is truly eliminated in counterfactual worlds, her privacy is protected.

Differential privacy has two important properties: *post-processing invariance* and *linear composition*. Post-processing invariance means that if $\mathcal{M}$ satisfies ϵ_{DP}-differential privacy, and f is any postprocessing function, the combined function $f \circ \mathcal{M}$ satisfies differential privacy with the same privacy parameter ϵ_{DP}. Linear composition means that if $\mathcal{M}_1, \ldots, \mathcal{M}_k$ are a sequence of mechanisms, and each $\mathcal{M}_i$ satisfies $\epsilon_{\text{DP}i}$-differential privacy, the mechanism $\mathcal{M}^*$, which runs all of those mechanisms and returns their outputs, satisfies $\sum_{i=1}^{k} \epsilon_{\text{DP}i}$-differential privacy. In this paper, when we say a mechanism ensures composition, we mean it ensures linear composition or better (sub-linear). A special variant of differential privacy that is particularly useful for this paper is per-entry differential privacy:

Definition 2 (Per-Entry DP). *Given a privacy parameter $\epsilon_{\text{DP}} \geq 0$, a mechanism $\mathcal{M}$ satisfies per-entry ϵ_{DP}-differential privacy if, for all pairs of tabular datasets $\mathcal{D}, \mathcal{D}'$ that differ on the contents of one attribute of one person's record, and all ω in range$(\mathcal{M})$,*

$$\Pr[\mathcal{M}(\mathcal{D}) = \omega] \;\leq\; e^{\epsilon_{\text{DP}}} \Pr[\mathcal{M}(\mathcal{D}') = \omega].$$

The difference between Definitions 1 and 2 is that in the latter, the counterfactual database is formed by only replacing part of a record (the value of one attribute) instead of the entire record. Definition 2 can naturally extend to a group of k entries, in which case the privacy linearly degrades with k:

Definition 3 (Group DP). *A per-entry ϵ_{DP}-differential privacy mechanism $\mathcal{M}$ achieves $k\epsilon_{\text{DP}}$-differential privacy for a group of k entries. That is, if $\mathcal{D}, \mathcal{D}'$ differ by k entries, for all ω in range$(\mathcal{M})$,*

$$\Pr[\mathcal{M}(\mathcal{D}) = \omega] \;\leq\; e^{k\epsilon_{\text{DP}}} \Pr[\mathcal{M}(\mathcal{D}') = \omega].$$

Group DP can handle correlations in the database by considering all the potentially correlated entries as a protected group, but doing so often leaves too little utility; for example, when all the entries are possibly correlated (even loosely)—as in our previous example of a person with an infectious disease—group DP degrades the privacy linearly with the number of total entries in the database.

2.2 Posterior-based Privacy Definitions

Unlike differential privacy, which treats datasets as deterministic, posterior-based privacy definitions (e.g., [9, 18, 27, 46]) model attackers as Bayesian reasoners who view the dataset as a random variable. Some of our results concern posterior-based privacy definitions in general, while others concern a specific definition called Pufferfish privacy [27]. We first distill common properties of posterior-based privacy, and then describe a special case called Pufferfish privacy.

Definition 4 (Posterior-based Privacy Properties). *A posterior-based privacy definition has a privacy parameter $\epsilon \geq 0$, a set Θ of prior distributions an attacker may use, and a collection of potential secrets $\{s_1, s_2, \ldots\}$. The definition must use these items to define a set of restrictions that a mechanism $\mathcal{M}$ must satisfy. Furthermore, if $\Pr(s_i \mid \theta) = \Pr(s_i \mid \mathcal{M}(\mathcal{D}), \theta)$ for all s_i and all $\theta \in \Theta$, then $\mathcal{M}$ satisfies the definition with privacy parameter $\epsilon = 0$. That is, if the posterior distribution of secrets is the same as the prior distribution, then the assessed privacy leakage is 0. Note that the probability is taken with respect to randomness in both $\mathcal{M}$ and θ, and larger values of ϵ represent weaker privacy protections than smaller values of ϵ.*

Definition 5 (Pufferfish Privacy [27]). *Given a set Θ of priors, a set S of potential secret pairs, and a privacy parameter $\epsilon_{\text{puffer}} \geq 0$, a randomized mechanism $\mathcal{M}$ satisfies ϵ_{puffer}-Pufferfish privacy if, for all pairs of potential secrets $(s_i, s_j) \in S$ and all priors $\theta \in \Theta$ such that $\Pr[s_i|\theta] > 0$ and $\Pr[s_j|\theta] > 0$, and for all possible outputs ω,*

$$\Pr[\mathcal{M}(Data) = \omega \mid s_i, \theta] \;\leq\; e^{\epsilon_{\text{puffer}}} \Pr[\mathcal{M}(Data) = \omega \mid s_j, \theta].$$

Let $\frac{\Pr(s_i \mid \theta)}{\Pr(s_j \mid \theta)}$ be the prior odds, specifying how likely the attacker thinks s_i is true compared to s_j being true before seeing the data. After seeing the data, the attacker forms the posterior odds, $\frac{\Pr(s_i \mid \mathcal{M}(\mathcal{D})=\omega, \theta)}{\Pr(s_j \mid \mathcal{M}(\mathcal{D})=\omega, \theta)}$. Pufferfish privacy guarantees that the posterior odds are at most $e^{\epsilon_{\text{puffer}}}$ times the prior odds, i.e., the attacker's belief changed by a factor of at most $e^{\epsilon_{\text{puffer}}}$. If the ratio between the prior and the posterior odds is the same, it means there is no privacy leakage ($\epsilon_{\text{puffer}} = 0$). Pufferfish satisfies post-processing invariance [27], but does not guarantee composition [27]. We show that this problem is very severe and can lead to privacy collapse (Section 4).

2.3 Trust and Threat Model

We assume a trusted data curator who releases query results by running multiple privacy mechanisms on confidential data, and an adversary who attempts to infer protected secrets from the released outputs. We assume that the adversary can leverage prior information about the data distribution and the protected secret pairs. The role of the privacy mechanisms is to prevent the released outputs from significantly increasing the adversary's ability to distinguish between the secrets that the curator wishes to protect. This is the setting of Pufferfish [27], but we additionally consider the combined privacy leakage of multiple mechanisms (composition).

3 RELATED WORK

Inevitability of DP for Composition. Blanc et al. [6] recently studied composition in a general axiomatic setting. If a privacy definition satisfies their axioms (some of which are related to composition), then any mechanism for that definition can be replaced by an (ϵ, δ)-DP mechanism whose accuracy, on statistical tasks involving *independent* records, is nearly the same. However, their work cannot be applied to *correlated data*, and their axioms are restrictive: pure ϵ-differential privacy violates their composition axioms, and posterior-based privacy definitions almost always violate their symmetry axioms. Thus, the work targets different privacy definitions and data assumptions from ours.

Generally, composition results outside of differential privacy are limited in scope. Bhaskar et al. [5] studied composition for mechanisms where the only randomness comes from the prior. Their composition result required refreshing large parts of the data (i.e., resampling from the prior) in between queries. An approach by Farokhi [17] studies a prior-free noise-free privacy definition and proves composition without restrictions, but the definition cannot always prevent a mechanism from leaking secret information.

Although arbitrary Pufferfish mechanisms will not compose [27], Song et al. [40, 41] showed that some specific mechanisms compose with themselves. They considered the set Θ of priors to be all Markov networks with the same structure and showed that a Laplace mechanism satisfying differential privacy with a privacy parameter ϵ_{DP} can satisfy Pufferfish privacy with some other parameter ϵ_{puffer} and will compose. Our result is a generalization of this prior result: we propose a framework which allows translating *any* set of (per-entry) ϵ_{DP}-differential privacy mechanisms into composable ϵ_{puffer}-Pufferfish privacy mechanisms for *any* data prior through a concept called an $a(b)$-influence curve.

Inferential/Posterior-based Definitions. This work focuses on Pufferfish privacy [27], a popular posterior-based privacy definition that is a generalization of many other posterior-based definitions [18, 30, 46, 52]. Other notable approaches related to posterior-based privacy include: adapting DP to attackers with deterministic knowledge about the data [20], changing how the dissimilarities between posteriors are quantified [34, 38], and adapting the counterfactual framework of DP to compare attacker inference about a hypothetical database to a scrubbed database [3, 9].

Pufferfish Privacy and Composition. Lack of composition in Pufferfish has been sidestepped in several ways. Pierquin et al. [38], Zhang et al. [51], and Liang et al. [29] performed a joint privacy analysis across multiple runs of mechanisms. Tao et al. [50] proposed a framework for performing such a joint analysis. In many cases [38, 50, 51], the measure of dissimilarity in posterior distributions was changed from Definition 5 to mirror Renyi differential privacy [36] or (ϵ, δ)-differential privacy [14]. An alternative to a joint analysis is to consider restricted types of mechanisms that compose with themselves. For example, Song et al. [40, 41] proposed the Markov Quilt Mechanism (MQM) that adapts the Laplace mechanism to Pufferfish with Markov network priors. This mechanism was also used by Shafieinejad et al. [39] and Cao et al. [7], who adjusted the privacy budget of differentially private algorithms to account for temporal correlations in Markov chains. In other work, the compositional properties were not studied [11, 12, 41].

4 PRIVACY COLLAPSE

We next use simple examples to illustrate that posterior-based privacy can catastrophically fail upon composition. While pathological, the main message is that posterior-based privacy definitions do not have built-in safeguards against privacy collapse arising from composition. Necessary conditions for such safeguards are proved in Section 5 and sufficient conditions are presented in Section 6.

Single-secret/prior Example. We start with one secret and one prior and study to what extent remedies within the posterior-based framework (e.g., adding more potential secrets or adding more priors) are effective. The proofs can be found in the full version [2].

Example 4.1. Consider mutually exclusive potential secrets s_1 and $s_2 = \neg s_1$. Let $\Theta = \{\theta^*\}$ consists of a single prior. Let $\mathcal{U}_1$ (resp., $\mathcal{U}_2$) be the set of datasets for which s_1 (resp., s_2) is true and have nonzero probability under θ^*. Suppose $\mathcal{U}_1$ and $\mathcal{U}_2$ each contain at least 2 datasets. Define mechanism $\mathcal{M}^*$ as follows. On input $\mathcal{D}$,

- if $s_1(\mathcal{D})$ is true, sample $\mathcal{D}'$ from $\Pr(\cdot \mid s_2, \theta^*)$ and outputs $(\mathcal{D}, \mathcal{D}')$,
- if $s_2(\mathcal{D})$ is true, sample $\mathcal{D}'$ from $\Pr(\cdot \mid s_1, \theta^*)$ and outputs $(\mathcal{D}', \mathcal{D})$.

The next theorem shows this mechanism has zero leakage under posterior-based privacy definitions (see Definition 4) but can reveal the true database with finitely many (and as few as two) runs. The intuition is that $\mathcal{M}^*(\mathcal{D})$ returns a tuple where the left part is always the same if s_1 is true and the right part is always the same if s_2 is true. If $\mathcal{M}^*$ is run twice, and the output of the first run is different from the output of the second run, the tuple component which didn't change corresponds to the true input dataset.

Theorem 1. *Let $\mathcal{M}^*$ be the mechanism from Example 4.1. Given an output ω of $\mathcal{M}^*$, $\Pr(s_1 \mid \mathcal{M}^*(Data) = \omega, \theta^*) = \Pr(s_1 \mid \theta^*)$ and $\Pr(s_2 \mid \mathcal{M}^*(Data) = \omega, \theta^*) = \Pr(s_2 \mid \theta^*)$. Thus, $\mathcal{M}^*$ has zero leakage under posterior-based privacy definitions. However, if the mechanism is run multiple times, the expected number of runs before the input dataset is revealed is $\leq 1 + \frac{1}{1 - \max_{\mathcal{D}} \max_{s_i} \Pr(\mathcal{D} \mid s_i)}$.*

Multiple Secrets Example. *Is the problem that there were too few secrets to defend?* A plausible attempt to counter such privacy collapse is to add more secrets so that every bit in the dataset is covered by a potential secret. However, this is not always effective:

Example 4.2. Consider a database of $n \geq 3$ bits, $\mathcal{D} = \{x_1, \ldots, x_n\}$. Let the secrets $s_0^{(1)}, \ldots, s_0^{(n)}$ and $s_1^{(1)}, \ldots, s_1^{(n)}$ be defined as $s_j^{(i)}$ ="i-th bit of the data is j". Let $\Theta = \{\theta^*\}$, where the prior θ^* generates an n-bit dataset by flipping a fair coin n times. Define the mechanism $\mathcal{M}^*(x_1, \ldots, x_n)$ as the following ($\oplus$ is the xor operation):

- With probability 1/2, output the tuple containing these $n-1$ values: $(x_2 \oplus x_1,\ x_3 \oplus x_1,\ \ldots,\ x_n \oplus x_1)$. We call this output type A.
- Otherwise, if n is odd, output $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ (the parity of the database), and if n is even, output $x_2 \oplus \cdots \oplus x_n$ (the parity of the database after omitting the first bit). We call this output type B.

If an attacker sees output type A, they can xor the $n-1$ values together. If n is odd, this equals $x_2 \oplus \cdots \oplus x_n$. If n is even, this equals $x_1 \oplus x_2 \oplus \cdots \oplus x_n$. If the attacker later sees output type B, they can xor it with the output type A and learn the value of x_1. Once x_1 is revealed, the attacker can use it with output type A to determine all the other secrets. The dataset could be reconstructed by running $\mathcal{M}^*$ as few as 2 times, but when run only once, $\mathcal{M}^*$ still satisfies zero leakage under posterior-based privacy, summarized in Theorem 2:

Theorem 2. *Let $\mathcal{M}^*$ be the mechanism from Example 4.2. Given an output ω of $\mathcal{M}^*$, $\Pr(s_j^{(i)} \mid \mathcal{M}^*(Data) = \omega, \theta^*) = \Pr(s_j^{(i)} \mid \theta^*)$ for all $i = 1, \ldots, n$ and $j \in \{0, 1\}$. Thus, $\mathcal{M}^*$ has zero leakage under posterior-based privacy definitions. However, the expected number of runs before the input dataset is revealed is 3.*

Multiple Priors Example. Previous examples leveraged the prior distribution to reveal secrets. *Would the composition properties improve when using more priors?* Specifically analyzing Pufferfish privacy, we again show that this is not the case.

When Θ is the set of all product distributions, it is known that the resulting instantiation of Pufferfish will turn into ϵ-differential privacy [27]. However, this is typically not how posterior-based privacy definitions would be used. The ideal situation would be to specify a reference prior θ^* that is a good initial guess of the true distribution, and then to add to Θ many "nearby" priors to hedge against inaccuracies in θ^*. In this way, the collection Θ of priors would encode approximate knowledge, such as "the conditional

probability that a cancer patient is female is between 0.35 and 0.65", "the conditional probability that a cancer-free patient is female is between 0.4 and 0.6", etc.

Unfortunately, this kind of Θ causes problems with composition. We show a mechanism that has a finite Pufferfish privacy parameter when run once, but can reveal the secret when run twice.

Example 4.3. We simplify the discussion by considering one secret pair $(s_1, \neg s_1)$. The set Θ contains many priors, but in line with the above discussion, has the following structure: there exist two sets of databases $\mathcal{U}_T$ and $\mathcal{U}_F$ with the following properties:

(1) Secret s_1 is true for all $\mathcal{D} \in \mathcal{U}_T$ and s_1 is false for all $\mathcal{D} \in \mathcal{U}_F$.
(2) $\Pr(\mathcal{D} \in \mathcal{U}_T \mid s_1, \theta) \in [L_T, U_T]$ for all $\theta \in \Theta$ ($L_T > 0$, $U_T < 1$).
(3) $\Pr(\mathcal{D} \in \mathcal{U}_F \mid \neg s_1, \theta) \in [L_F, U_F]$ for all $\theta \in \Theta$ ($L_F > 0$, $U_F < 1$).

Define $\mathcal{M}^*$ that, on input $\mathcal{D}$, does the following:

- **Case 1.** If $s_1(\mathcal{D})$ is true: (**1a**) if $\mathcal{D} \in \mathcal{U}_T$, return ("1$a$", "2$a$") with probability 0.5 and ("1a", "2b") otherwise; (**1b**) if $\mathcal{D} \notin \mathcal{U}_T$, return ("1$b$", "2$a$") with probability 0.5 and ("1b", "2b") otherwise.
- **Case 2.** If $s_1(\mathcal{D})$ is false: (**2a**) if $\mathcal{D} \in \mathcal{U}_F$, return ("1$a$", "2$a$") with probability 0.5 and ("1b", "2a") otherwise; (**2b**) if $\mathcal{D} \notin \mathcal{U}_F$, return ("1$a$", "2$b$") with probability 0.5 and ("1b", "2b") otherwise.

$\mathcal{M}^*$ from Example 4.3 has 4 total cases (1a, 1b, 2a, 2b), and each case returns a tuple of case numbers, one of which is the actual case number. When $\mathcal{M}^*$ is run multiple times, as soon as two different outputs are observed, the true case, hence the secret, is revealed (the tuple element that didn't change is the real case number). The following theorem shows that $\mathcal{M}^*$ has a finite privacy parameter under Pufferfish, yet the expected number of runs before the secret is revealed is 3, meaning that composition breaks down quickly.

Theorem 3. *Let* $\epsilon = \max(\log|\frac{U_T}{L_F}|, \log|\frac{L_T}{U_F}|, \log|\frac{1-U_T}{U_F}|, \log|\frac{1-L_T}{L_F}|, \log|\frac{L_T}{1-L_F}|, \log|\frac{U_T}{1-U_F}|, \log|\frac{1-U_T}{1-L_F}|, \log|\frac{1-L_T}{1-U_F}|)$. *Mechanism* $\mathcal{M}^*$ *from Example 4.3 satisfies* ϵ*-Pufferfish privacy. The expected number of runs of* $\mathcal{M}^*$ *to reveal the true secret is 3.*

Again, when all possible priors are added in Θ, Pufferfish becomes ϵ-differential privacy, and hence composes linearly. However, the above example implies that such a composition would only happen when enough priors are added, such that for any set S of databases with the same secret, Θ must include priors where S has arbitrarily low probabilities and also include priors where S has arbitrarily high probabilities. While such a case may improve composition, it will ruin utility as such a comprehensive set of priors will include priors that the data curator views as unrealistic.

In the next section, we study necessary conditions for Pufferfish privacy to compose, and show that it needs differential-privacy-like constraints on mechanisms. Greatly expanding the set of priors will be an overkill, i.e., expanding the set of priors until composition is achieved would be equivalent to adding the necessary conditions that we found as well as many others (unnecessarily).

5 NECESSARY CONDITIONS FOR COMPOSITION AND POST-PROCESSING INVARIANCE

The previous section demonstrated that posterior-based privacy definitions, including Pufferfish, can suffer privacy collapse. In this section, we study how to avoid this situation. Specifically, suppose we have a collection of mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$ where each $\mathcal{M}_i$ satisfies Pufferfish with privacy parameter $\epsilon_{\text{puffer}_i}$. What conditions do these mechanisms need to satisfy in order to linearly compose with each other, so that the combined release of all of their outputs satisfies Pufferfish with privacy parameter $\leq \sum_{i=1}^{k} \epsilon_{\text{puffer}_i}$?

The necessary conditions that we prove (Theorem 4) identify a class of constraints that we call the NfC (necessary-for-composition) constraints. For each dataset $\mathcal{D}$, one or more NfC constraints need to be added (in addition to the Pufferfish constraints); otherwise, linear composition will fail. As we will show, the differential privacy constraints, $\log \Pr(\mathcal{M}(\mathcal{D}) = \omega) \leq \epsilon + \log \Pr(\mathcal{M}(\mathcal{D}') = \omega)$, are special cases of NfC constraints, which have the general form of: $\log \Pr(\mathcal{M}(\mathcal{D}) = \omega) \leq \epsilon + \sum_{i=1}^{m} \beta_i \log \Pr(\mathcal{M}(\mathcal{D}_i) = \omega)$.

Then, we study another important property of privacy definitions called *post-processing invariance*. Post-processing invariance means that if $\mathcal{M}_i$ satisfies $\epsilon_{\text{puffer}_i}$-Pufferfish and f_i is a possibly randomized algorithm, then the algorithm $f_i \circ \mathcal{M}_i$, which releases $f_i(\mathcal{M}_i(\mathcal{D}))$, still satisfies Pufferfish with privacy parameter $\epsilon_{\text{puffer}_i}$. Now that extra NfC constraints must be added, one needs to ensure that those constraints are also post-processing invariant. That is, if $\mathcal{M}$ satisfies the extra constraints, then $f_i \circ \mathcal{M}_i$ should also satisfy the constraints. Or, put another way, if $\mathcal{M}_1, \ldots, \mathcal{M}_k$ compose linearly, then so should $f_1 \circ \mathcal{M}_1, \ldots, f_k \circ \mathcal{M}_k$.

Our second result (Theorem 5) states that a non-redundant subset of NfC constraints is post-processing invariant if and only if every constraint in that subset is a differential-privacy-style constraint. Together, these results imply that if one wants to design Pufferfish mechanisms that compose with each other, those mechanisms should also satisfy a form of differential privacy. In Section 6, we show how to design such algorithms for tabular datasets (i.e., sufficient conditions for composition).

5.1 Necessary Condition for Composition

We present necessary conditions for a set of Pufferfish mechanisms to compose, when defending a set of pairs of potential secrets S against priors in Θ. The conditions use the concept of a *convex combination vector* β, which is a vector whose entries are nonnegative and sum up to 1. Entry ℓ in the vector is represented as β_ℓ.

One condition in the following theorem is that the set of possible databases is finite (but arbitrarily large). In practice, this is not a restrictive condition because attribute domains are typically bounded (e.g., a 500-character limit for URLs, 64 bits for numeric values) and only finitely many records can be collected in a finite time (e.g., no database will have more than 2^{64} records). This simplifies the proof as it avoids dealing with corner cases that only arise with infinite-dimensional vectors. Similarly, the necessary conditions consider a set of mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_m$, whose ranges are finite. Mechanisms with infinite output spaces would also need to satisfy these and possibly additional conditions.

Theorem 4 (Necessary Condition for Composition (NfC)). *Let* Θ *be a set of priors over a finite (but arbitrarily large) set of possible datasets. Let* S *be a set of secret pairs, and let* $0 \leq$ *epsilon* $< \infty$ *be the Pufferfish privacy parameter. Let* $\mathcal{M}_1, \ldots, \mathcal{M}_m$ *be a set of mechanisms, with finite ranges, that satisfy* ϵ*-Pufferfish and linearly compose with each other. Then for all ordered pairs* $(s_1, s_2) \in S$ *and*

datasets $\mathcal{D}$ for which s_1 is true, there exists a convex combination vector β over datasets $\mathcal{D}_\ell$ for which s_2 is true, such that:

$$\log \Pr(\mathcal{M}(\mathcal{D}) = \omega) \le \epsilon + \sum_{\mathcal{D}_\ell : s_2(\mathcal{D}_\ell) \text{ is true}} \beta_\ell \log \Pr(\mathcal{M}(\mathcal{D}_\ell) = \omega), \quad (1)$$

for each mechanism and each of its possible outputs ω. Note the convex combination depends on s_1, s_2 and $\mathcal{D}$, and the reverse secret pair (s_2, s_1) may have its own convex combination.

We use the term **NfC constraints** to refer to all inequalities having the form shown in Equation 1. Before presenting the proof sketch of Theorem 4, note that Equation 1 can also be written as:

$$\Pr(\mathcal{M}(\mathcal{D}) = \omega) \le e^\epsilon \prod_{\mathcal{D}_\ell : s_2(\mathcal{D}_\ell) \text{ is true}} \Pr(\mathcal{M}(\mathcal{D}_\ell) = \omega)^{\beta_\ell}.$$

This makes the connection to differential privacy clearer: if $\beta_\ell = 1$ for some index ℓ and 0 for the rest, we get $\Pr(\mathcal{M}(\mathcal{D}) = \omega) \le e^\epsilon \Pr(\mathcal{M}(\mathcal{D}_\ell) = \omega)$, where s_1 is true of $\mathcal{D}$ and s_2 is true of $\mathcal{D}_\ell$. Hence, differential privacy inequalities are a subset of NfC constraints.

PROOF SKETCH (SEE [2] FOR THE FULL PROOF). The overall idea is that we convert the statement "linear composition holds" into a collection of linear programs and take the duals of the linear programs. Since the number of possible outputs is finite, we can write them as $\omega_1, \ldots, \omega_k$. Without loss of generality, we can assume that the ranges of the mechanisms are disjoint, i.e., given an output ω_i, we can tell which mechanism it came from. Therefore, we can use the shorthand $P(\omega_i \mid \mathcal{D}_\ell)$ to mean $P(\mathcal{M}_j(\mathcal{D}_\ell) = \omega_i)$, where $\mathcal{M}_j$ is the mechanism whose range contains ω_i.

The data curator selects r mechanisms with replacement from $\{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ (mechanisms can be repeated), runs them, and records their outputs. Suppose that the fraction of times that output ω_i occurs is α_i. Under linear composition, the Pufferfish privacy cost would be bounded by $r\epsilon$, and for every $\theta \in \Theta$ and secret pair (s_1, s_2), we get the equation:

$$r\epsilon \ge \log \frac{\left(\sum_\ell \Pr_\theta(\mathcal{D}_\ell \mid s_1) \prod_{i=1}^k \Pr(\omega_i \mid \mathcal{D}_\ell)^{r\alpha_i}\right)}{\left(\sum_\ell \Pr_\theta(\mathcal{D}_\ell \mid s_2) \prod_{i=1}^k \Pr(\omega_i \mid \mathcal{D}_\ell)^{r\alpha_i}\right)}.$$

If you divide by r and let $r \to \infty$, this becomes:

$$\epsilon \ge \left(\max_{\{\ell : s_1(\mathcal{D}_\ell) = true\}} \sum_{i=1}^k \alpha_i \log(\Pr(\omega_i \mid \mathcal{D}_\ell))\right) - \left(\max_{\{\ell : s_2(\mathcal{D}_\ell) = true\}} \sum_{i=1}^k \alpha_i \log(\Pr(\omega_i \mid \mathcal{D}_\ell))\right).$$

Such an equation must hold for any $(s_1, s_2) \in S$ and any choice of nonnegative $\alpha_1, \ldots, \alpha_k$ that add up to 1. Then, for every secret pair (s_1, s_2) and each $\mathcal{D}_{\ell^*}$ for which s_1 is true, we create a linear program as follows. We show that the above constraints are true if and only if all of the following linear programs have a solution $\le \epsilon$.

For each $(s_1, s_2) \in S$ and $\mathcal{D}_{\ell *}$ for which $s_1(\mathcal{D}_{\ell^*})$=true:

$$\max_{\epsilon_0, \alpha_1, \ldots, \alpha_k} \epsilon_0$$
$$\text{s.t. } \alpha_1 \ge 0, \ldots, \alpha_k \ge 0$$
$$\alpha_1 + \cdots + \alpha_k = 1$$
$$\text{For all } \mathcal{D}_\ell \text{ having } s_2(\mathcal{D}_\ell) = true:$$
$$\left(\sum_{i=1}^k \alpha_i \log\left(\frac{\Pr(\omega_i \mid \mathcal{D}_{\ell^*})}{\Pr(\omega_i \mid \mathcal{D}_\ell)}\right)\right) - \epsilon_0 \ge 0.$$

We convert each such linear program into the corresponding dual program for each $(s_1, s_2) \in S$ and $\mathcal{D}_{\ell *}$ for which $s_1(\mathcal{D}_{\ell^*})$=true:

$$\min_{\epsilon_0', \text{ convex combination vector } \beta} \epsilon_0'$$
$$\text{s.t. } \epsilon_0' - \log(\Pr(\omega_i \mid \mathcal{D}_{\ell^*})) + \sum_\ell \beta_\ell \log(\Pr(\omega_i \mid \mathcal{D}_\ell)) \ge 0 \text{ for all } \omega_i,$$

where the summation is over all ℓ for which $s_2(\mathcal{D}_\ell) = true$. We conclude that, for composition, all of these dual programs must have a solution $\le \epsilon$. If that is true, each of these programs will come up with a linear combination vector β, which becomes Equation 1. □

5.2 Enforcing Post-processing Invariance

Theorem 4 tells us that if we want composition properties inside Pufferfish, we need to add additional NfC constraints (inequalities with the form of Equation 1). However, such constraints are not necessarily post-processing invariant. That is, after post-processing, a mechanism may fail to satisfy the constraints, and the post-processed mechanisms may not compose. We next show that if a set of NfC constraints is "non-redundant" and post-processing invariant, then each one of those constraints must have the form of: $\log \Pr(\mathcal{M}(\mathcal{D}_i) = \omega) \le \epsilon + \log \Pr(\mathcal{M}(\mathcal{D}_j) = \omega)$, i.e., each one of those constraints must be a differential privacy-style constraint.

To understand redundancy, consider the following two constraints with two different convex combination vectors $\beta^{(0)}$, $\beta^{(1)}$:

$$\log \Pr(\mathcal{M}(\mathcal{D}^*) = \omega) \le \epsilon + \sum_{\mathcal{D}_\ell : s_2(\mathcal{D}_\ell) \text{ is true}} \beta_\ell^{(0)} \log \Pr(\mathcal{M}(\mathcal{D}_\ell) = \omega),$$
$$\log \Pr(\mathcal{M}(\mathcal{D}^*) = \omega) \le \epsilon + \sum_{\mathcal{D}_\ell : s_2(\mathcal{D}_\ell) \text{ is true}} \beta_\ell^{(1)} \log \Pr(\mathcal{M}(\mathcal{D}_\ell) = \omega).$$

Then, for any $c \in (0, 1)$, one can define $\beta^{(c)} = c\beta^{(1)} + (1 - c)\beta^{(0)}$ to get the following redundant constraint:

$$\log \Pr(\mathcal{M}(\mathcal{D}^*) = \omega) \le \epsilon + \sum_{\mathcal{D}_\ell : s_2(\mathcal{D}_\ell) \text{ is true}} \beta_\ell^{(c)} \log \Pr(\mathcal{M}(\mathcal{D}_\ell) = \omega).$$

In general, a constraint is redundant if it is implied by the other constraints that we have. By iteratively removing redundant constraints, we obtain a non-redundant set of constraints.

THEOREM 5 (NFC AND POST-PROCESSING INVARIANCE). *Suppose there are finitely many datasets and m non-redundant NfC constraints, where constraint i involves some dataset $\mathcal{D}_i$, some secret pair $(s_1^{(i)}, s_2^{(i)})$, and a convex combination vector $\beta^{(i)}$, and has the form:*

$$\log \Pr(\mathcal{M}(\mathcal{D}_i) = \omega) \le \epsilon + \sum_{\mathcal{D}_\ell : s_2^{(i)}(\mathcal{D}_\ell) \text{ is true}} \beta_\ell^{(i)} \log \Pr(\mathcal{M}(\mathcal{D}_\ell) = \omega),$$

for all ω. If all of the convex combination vectors have only 1 nonzero entry (i.e., these are all DP-style conditions), this set of constraints is post-processing invariant. If at least one of the convex combination vectors has 2 or more nonzero entries, then this set of constraints is not post-processing invariant.

PROOF SKETCH (SEE [2] FOR THE FULL PROOF). The proof is based on the observations (1) that the feasible region (allowable real-valued vectors of the form $[\Pr(\mathcal{M}(\mathcal{D}_1) = \omega), \ldots, \Pr(\mathcal{M}(\mathcal{D}_n) = \omega)]$) for the system of constraints is only convex when each $\beta^{(i)}$ has a single non-zero component and (2) post-processing acts like a convex operation on those allowable vectors. Hence, we construct a mechanism for which the constraints are tight but are violated after postprocessing. □

The conclusion we reach is that, if we want a set of Pufferfish mechanisms to compose with each other and be post-processing invariant, we should design those mechanisms to satisfy differential privacy. The next section presents sufficient conditions—how to construct those differential privacy mechanisms and choose their privacy parameter ϵ_{DP}, so that they also satisfy Pufferfish with a desired privacy parameter ϵ_{puffer} and compose linearly. Our results are specific to ϵ-Pufferfish (Definition 5), which mirrors ϵ-differential privacy (Definition 1). For other variations of Pufferfish that are analogous to (ϵ, δ)-differential privacy or Renyi-DP [38], we conjecture that similar results would hold; e.g., composable (ϵ, δ)-Pufferfish must also satisfy (ϵ, δ)-DP.

6 CONSTRUCTING GENERAL PUFFERFISH MECHANISMS WITH COMPOSITION

Theorem 4 and Theorem 5 show that a set of linearly composable Pufferfish mechanisms that support post-processing invariance must satisfy a set of constraints that whittle down to the types that are used by differential privacy. These insights provide formal support for prior work [7, 40, 41] that designed linearly composable Pufferfish mechanisms by taking ϵ_{DP}-differential privacy mechanisms and showing that they also satisfy ϵ_{puffer}-Pufferfish for some $\epsilon_{\text{puffer}} \geq \epsilon_{\text{DP}}$. Specifically, Song et al. [40, 41] proposed an extension of the Laplace Mechanism (called MQM) to defend against attacker priors that all share the same Markov network structure. Meanwhile, Cao et al. [7] extend arbitrary DP mechanisms to Markov chains and defend against finitely many Markov chain priors while allowing the Pufferfish privacy parameter to compose.

In this section, we generalize these ideas to show that in any Pufferfish setting for tabular data (including finite Markov chains), the same recipe holds. Any collection of per-entry ϵ_{DP}-differential privacy mechanisms can be translated into ϵ_{puffer}-Pufferfish privacy, and the Pufferfish parameters linearly compose. Furthermore, the translation between ϵ_{DP} and ϵ_{puffer} is guided by a new concept we call an $a(b)$-influence curve. The $a(b)$-influence curve only depends on the set of secret pairs S and the priors in Θ. Although the $a(b)$-influence curve can be computationally expensive to compute exactly, one only needs an upper bound to get privacy guarantees. The tighter the upper bound is, the better the resulting utility. This is analogous to sensitivity computation in differential privacy, which is easy to upper bound (providing privacy guarantees), but can be computationally difficult to estimate exactly.

Before going into the technical details, we first explain the intuition behind the $a(b)$-influence curve. Suppose there is an infectious disease dataset and a person A in this dataset. Person A has a set $\mathcal{H}_1$ containing b_1 people who are highly likely to get infected if A gets sick (e.g., $\mathcal{H}_1$ is the set of close family members), a set $\mathcal{H}_2$ (with $\mathcal{H}_1 \subset \mathcal{H}_2$) containing b_2 people who are moderately likely to get infected if A gets sick (e.g., $\mathcal{H}_2$ contains $\mathcal{H}_1$ and close friends), and a third set $\mathcal{H}_3$ containing b_3 people who are only slightly likely to get infected from A (e.g., $\mathcal{H}_3$ contains $\mathcal{H}_2$ and acquaintances).

Now, suppose A can increase the chance of infections of people not in $\mathcal{H}_1$ by at most a factor of a_1, can increase the chance of infection of people not in $\mathcal{H}_2$ (resp., $\mathcal{H}_3$) by at most a factor of $a_2 < a_1$ (resp. $a_3 < a_2$), and so on. These tuples $(a_1, b_1), (a_2, b_2), (a_3, b_3)$ form the $a(b)$-influence curve. That is, we can think of the a value as a function of b. When considering b number of people most closely associated with A (or the number of cells in a tabular dataset most closely associated with a secret), $a(b)$ upper bounds the effect of A on anyone outside of this close group of size b.

This $a(b)$-influence curve provides translations between ϵ_{DP} and ϵ_{puffer} as follows. An algorithm satisfying ϵ_{DP}-differential privacy also satisfies Pufferfish with privacy parameter $\epsilon_{\text{puffer}} = b\epsilon_{\text{DP}} + a(b)$ for any choice of b. This means that the best ϵ_{puffer} parameter is $\min_b b\epsilon_{\text{DP}} + a(b)$. Thus, if one has the $a(b)$-influence curve, or knows at least a few points on the curve, they can compute the upper bound on the Pufferfish privacy parameter.

In order to show how the $a(b)$-influence curve is a generalization of the work of Song et al. [41], we briefly describe the Markov Quilt Mechanism and *max-influence* concept in Section 6.1. Then, we formally define $a(b)$-influence, prove the translation between ϵ_{DP} and ϵ_{puffer}, and present the slightly-better-than-additive composition results in Section 6.2. We discuss practical considerations in estimating the $a(b)$-curve, including robustness to mis-specifying priors in Section 6.3. We also give worked out examples of the $a(b)$-influence curve in Section 6.4.

6.1 The MQM and Max-Influence [41]

Song et al. [41] showed that applying the Laplace mechanism of differential privacy to a database generated by a Markovian structure can realize composable Pufferfish privacy. As we will show later, this finding is a special instance of our generalized result. We briefly describe the result from [41] as a background.

Song et al. [41] focused on a specific case where each entry of the database has a dependency described by a Markovian-structure graph G. Figure 1 shows an example where the database $X = \{X_1, \ldots, X_n\}$ is generated by a Markov chain, i.e., the value of X_i only depends on X_{i-1}. They introduced a concept called a *Markov Quilt*, which is an extension of the Markov Blanket [28]. A set of nodes X_Q is a Markov Quilt for a node X_i if: (1) conditioning on X_Q, the graph G is partitioned into two independent node sets, X_N and X_R, such that $X = X_N \cup X_Q \cup X_R$, and $X_i \in X_N$, and (2) for all $x_R \in \mathcal{X}^{card(X_R)}, x_Q \in \mathcal{X}^{card(X_Q)}, x_i \in \mathcal{X}$, we have $\Pr[X_R = x_R | X_Q = x_Q, X_i = x_i] = \Pr[X_R = x_R | X_Q = x_Q]$. In other words, the graph is partitioned in a way that the values of nodes X_R only depend on X_Q, and not on X_N (see Figure 1).

Song et al. [41] proposed a composable Pufferfish privacy mechanism called a Markov Quilt Mechanism (MQM) that can hide the value of each node X_i in the Markov chain. In fact, the MQM mechanism with ϵ_{puffer}-Pufferfish privacy is equivalent to a Laplace mechanism of differential privacy of a certain ϵ_{DP} (i.e., a Laplace noise of scale $\frac{L}{\epsilon_{\text{DP}}}$ is added to an L-Lipschitz query output). The relationship between ϵ_{puffer} and ϵ_{DP} is as follows. First, they define *max-influence* as below:

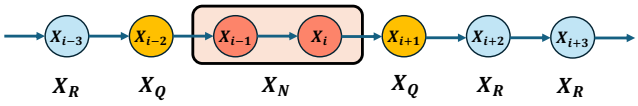

Figure 1: An illustration of X_R, X_Q, X_N in Markov Chain.

Definition 6 (Max-Influence [41]). *The max-influence of a variable X_i on a set of variables X_A under Θ is defined as:*

$$e_\Theta(X_A|X_i) = \sup_{\theta\in\Theta} \max_{a,b\in\mathcal{X}, x_A\in\mathcal{X}^{card(X_A)}} \log \left| \frac{\Pr(X_A = x_A|X_i = a, \theta)}{\Pr(X_A = x_A|X_i = b, \theta)} \right|.$$

Then, Song et al. [41] showed that applying a Laplace mechanism with ϵ_{DP}-differential privacy in fact achieves composable ϵ_{puffer}-Pufferfish privacy, where $\epsilon_{\text{DP}} = \frac{\epsilon_{\text{puffer}} - e_\Theta(X_Q|X_i)}{card(X_N)}$. Here, $card(X_N)$ is the cardinality of X_N. MQM only works when there exists a partition such that $\epsilon_{\text{puffer}} > e_\Theta(X_Q|X_i)$, and the proposal [41] was to enumerate all the possible partitions and choose the partition with the largest ϵ_{DP} for utility.

6.2 Sufficient Condition for Composition

The MQM mechanism [41] only works under databases whose generation follows a Markovian structure and is restricted to Laplace noise. In this section, we introduce a generalized framework that can translate an arbitrary per-entry ϵ_{DP}-differential privacy mechanism into a composable Pufferfish mechanism. We consider a general tabular dataset with m records and n attributes without loss of generality. Let $I = \{(i, j) : i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}\}$ be the entry index set of the entire dataset ($|I| = mn$). We do not restrict the generation of the dataset (i.e., it doesn't have to come from a Markov chain, etc.).

Assume an arbitrary index set $\mathcal{L} \subseteq I$, which we consider to be *low-influence* region, and a complement set $\mathcal{H} = I \setminus \mathcal{L}$, which we consider as *high-influence* region. $\mathcal{D}_\mathcal{L}$ is the entry values of $\mathcal{D}$ restricted to indices in $\mathcal{L}$, and the corresponding random variable is $Data_\mathcal{L}$. While the separation of $\mathcal{H}$ and $\mathcal{L}$ can be arbitrary, we are interested in finding a separation where the information leakage about the secret is minimal through $\mathcal{L}$, when the size of $\mathcal{H}$ is restricted to a certain value, b. The worst-case leakage through $\mathcal{L}$ in such optimal $(\mathcal{H}, \mathcal{L})$ pairs for all possible b values can be summarized in a single curve, which we call the $a(b)$-influence curve. We formally define *$a(b)$-influence* below:

Definition 7 ($a(b)$-influence). *Assume an arbitrary split of the index set I, $\mathcal{L} \subseteq I$ and $\mathcal{H} = I \setminus \mathcal{L}$. Given any $(s_i, s_j) \in S$ and any possible $Data_\mathcal{L}$, let $\mathcal{U}_\mathcal{L} = \{\mathcal{D}_{\mathcal{L}1}, \mathcal{D}_{\mathcal{L}2}, \ldots\}$ denote the set of all such shared subsets supported by Θ. The worst-case information leakage through $\mathcal{L}$ for secret pair (s_i, s_j) is,*

$$a^\dagger_{(s_i,s_j)}(\mathcal{L}) = \sup_{\theta\in\Theta} \max_{\mathcal{D}_\mathcal{L}\in\mathcal{U}_\mathcal{L}} \left| \log \frac{\Pr[Data_\mathcal{L} = \mathcal{D}_\mathcal{L} \mid s_i, \theta]}{\Pr[Data_\mathcal{L} = \mathcal{D}_\mathcal{L} \mid s_j, \theta]} \right|.$$

If we define $a^\star_{(s_i,s_j)}(b)$ for (s_i, s_j) as:

$$a^\star_{(s_i,s_j)}(b) = \inf_{\mathcal{L}\subseteq I:\, |\mathcal{H}|\le b} a^\dagger_{(s_i,s_j)}(\mathcal{L}),$$

the $a(b)$-influence curve is an upper-bounding curve for all $a^\star_{(s_i,s_j)}(b)$:

$$a(b) \ge \max_{(s_i,s_j)\in S} a^\star_{(s_i,s_j)}(b),$$

*and we say that (Θ, S) satisfies $a(b)$-*influence. *Note that by definition, the curve is monotonically non-increasing.*

Essentially, the $a(b)$-influence curve is a succinct summary of the risk of revealing a counterfactual database, where $\mathcal{L}$ entries stay the same, but at most b number of $\mathcal{H}$ entries are replaced.

Theorem 6 (Translation between DP and Pufferfish). *Fix a family of priors Θ, and a set of secret pairs S. Suppose (Θ, S) satisfies $a(b)$-influence (Definition 7), and (a, b) is a point on the $a(b)$-influence curve. Suppose a randomized mechanism $\mathcal{M}$ achieves per-entry ϵ_{DP}-differential privacy, with $\epsilon_{\text{DP}} \le \frac{\epsilon_{\text{puffer}} - a}{b}$ and $a < \epsilon_{\text{puffer}}$. Then, $\mathcal{M}$ satisfies ϵ_{puffer}-Pufferfish privacy with respect to the secret set S and set of priors Θ.*

Theorem 6 can be written as $\epsilon_{\text{puffer}} = b\epsilon_{\text{DP}} + a$, which can also be interpreted as saying that the Pufferfish parameter uses differential privacy's group privacy with a group size of b (accounting for the $b\epsilon_{\text{DP}}$ part of the formula) plus a penalty a for cross-record correlation that extends beyond the group. Our next theorem, about composition, shows that this extra penalty only needs to be paid once, so that the composition of the Pufferfish parameter is still linear but is better than just adding up the Pufferfish parameters:

Theorem 7 (Sufficient Condition for Composition). *Fix a family of priors Θ and a set of secret pairs S. Consider k mechanisms $\mathcal{M}_1, ..., \mathcal{M}_k$ applied sequentially to the same dataset, each satisfying $\epsilon_{\text{DP}1}, ..., \epsilon_{\text{DP}k}$-differential privacy, where $\epsilon_{\text{DP}\ell} \le \frac{\epsilon_{\text{puffer}_\ell} - a_\ell}{b_\ell}$. Here, (a_ℓ, b_ℓ) are arbitrary points on the $a(b)$-influence curve of (Θ, S) with $a_\ell < \epsilon_{\text{puffer}_\ell}$. That is, each $\mathcal{M}_\ell$ achieves $\epsilon_{\text{puffer}_\ell}$-Pufferfish according to Theorem 6. Then, the joint release of the outputs satisfies $(\max_\ell a_\ell + \sum_{\ell=1}^k \epsilon_{\text{puffer}_\ell} - \sum_{\ell=1}^k a_\ell)$-Pufferfish privacy. Note that the composition is sub-additive due to $(\max_\ell a_\ell - \sum_{\ell=1}^k a_\ell)$ being always negative.*

Theorem 6 and Theorem 7 together provide us with a powerful tool to translate differential privacy mechanisms into composable Pufferfish mechanisms. With an $a(b)$-influence curve, we can use any popular DP algorithms, e.g., randomized response [44], exponential mechanism [32], etc., depending on the use case, and not be restricted to the Laplace mechanism [40, 41], for example. Our theorems ensure that all of them would be valid Pufferfish mechanisms, and they would compose linearly.

Our results have several key implications. First, this reduces the manual effort needed to design Pufferfish mechanisms for different use cases and applications. Currently, there are not many existing (composable) Pufferfish mechanisms to choose from, unlike differential privacy, which has a rich set of well-understood mechanisms. Currently, using Pufferfish to create privacy-preserving data products is complex: the data curator needs to design new mechanisms, prove they satisfy Pufferfish individually, and then manually analyze their joint privacy leakage (compositional properties). This process, especially the last step, requires significant expenditure of manual effort. However, our result implies that one can re-purpose existing differential privacy mechanisms to achieve Pufferfish. It requires a one-time (per application) computation of

the $a(b)$-curve, and then, any DP algorithm and mature DP frameworks like OpenDP [37] can be used. In fact, our theoretical results imply that any composable Pufferfish mechanism must meet DP-like constraints, so adapting differential privacy is essentially inevitable and not a heuristic decision. We show the practical benefit of this in our evaluation (Section 7). Finally, our result provides a retrospective understanding of what Pufferfish privacy is achieved for data that were already released under differential privacy.

6.3 Practical Considerations

We next consider the following practical questions. Are there efficient ways to upper-bound the $a(b)$-influence curve? How can we compute the curve when there is uncertainty about the data prior (e.g., the prior is mis-specified)? What happens if, instead of specifying the priors Θ directly, the data curator specifies them indirectly as "the set of all priors having a given $a(b)$-influence curve," but this curve is incorrect for the true distribution?

6.3.1 Efficient Curve Estimation. Computing the $a(b)$-influence curve *exactly* may require a brute-force search over all index set splits $(\mathcal{L}, \mathcal{H})$ and all secret pairs (s_i, s_j) in Definition 7 (for some well-defined priors, the compute can be much simpler, as we show in Section 6.4). When this is infeasible, the goal is to efficiently compute an upper bound $\hat{a}(b)$ on the curve to get an upper bound on ϵ_{puffer}. A simple strategy is to sample a subset of the index set splits and only calculate the $a(b)$-influence curve over that subset, which produces an upper bound because the true curve is an infimum over all the possible index splits. While choosing a good subset may not always be straightforward, we present an efficient heuristic for graph-structured priors, where each secret corresponds to the value of a node in a Markov or Bayesian network (e.g., [40]). For a secret pair (s_i, s_j) associated with a node X_s, and a small value of b, we construct candidate splits by placing X_s and its $b-1$ nearest nodes into the high-influence region. Formally, let $\mathcal{N}_{b-1}(X_s)$ denote the collection of all possible sets of $b-1$ nearest nodes to X_s in the graph. For each $N \in \mathcal{N}_{b-1}(X_s)$, we define $\mathcal{H}_b^N(X_s) = \{X_s\} \cup N$ and $\mathcal{L}_b^N(X_s) = I \setminus \mathcal{H}_b^N(X_s)$. We use these candidate sets (with $a^\dagger$ defined as in Definition 7) to compute $\hat{a}(b)$ as follows:

$$\hat{a}^\star_{(s_i,s_j)}(b) = \min_{N \in \mathcal{N}_{b-1}(X_s)} a^\dagger_{(s_i,s_j)}(\mathcal{L}_b^N(X_s))$$
$$\text{and } \hat{a}(b) = \max_{(s_i,s_j)\in S} \hat{a}^\star_{(s_i,s_j)}(b).$$

This computation is manageable for small values of b. The intuition behind this heuristic is that in graph-structured priors, nodes that are closest to X_s are typically the most strongly correlated with it, and therefore their values are more likely to reveal information about the secret. We show an example of applying this efficient curve estimation heuristic to a Bayesian network in Section 6.4.

6.3.2 Uncertain or mis-specified prior. Suppose a practitioner specifies a θ that differs from the true prior θ^*, but is not too far off—for example, the probabilities have up to 20% error (i.e., $0.8\Pr[Data_{\mathcal{L}} = \mathcal{D}_{\mathcal{L}} \mid s, \theta] \le \Pr[Data_{\mathcal{L}} = \mathcal{D}_{\mathcal{L}} \mid s, \theta^*] \le 1.2\Pr[Data_{\mathcal{L}} = \mathcal{D}_{\mathcal{L}} \mid s, \theta]$). We can still upper bound the $a(b)$-influence curve in the presence of such uncertainties. In general, suppose there is some lower bound $l(\mathcal{D}_{\mathcal{L}}, s)$ (e.g., $0.8\Pr[Data_{\mathcal{L}} = \mathcal{D}_{\mathcal{L}} \mid s, \theta]$) and some upper bound $u(\mathcal{D}_{\mathcal{L}}, s)$ (e.g., $1.2\Pr[Data_{\mathcal{L}} = \mathcal{D}_{\mathcal{L}} \mid s, \theta]$) on $\Pr[Data_{\mathcal{L}} = \mathcal{D}_{\mathcal{L}} \mid s, \theta^*]$ for all $s \in S$ and $Data_{\mathcal{L}}$. Then, the value $a^\dagger_{(s_i,s_j)}$ from Definition 7 can be replaced by:

$$\bar{a}^\dagger_{(s_i,s_j)}(\mathcal{L}) = \max_{\mathcal{D}_{\mathcal{L}}} \max\left\{ \left| \log \frac{u(\mathcal{D}_{\mathcal{L}}, s_i)}{l(\mathcal{D}_{\mathcal{L}}, s_j)} \right|, \left| \log \frac{l(\mathcal{D}_{\mathcal{L}}, s_i)}{u(\mathcal{D}_{\mathcal{L}}, s_j)} \right| \right\}. \quad (2)$$

Using this upper bound $\bar{a}^\dagger_{(s_i,s_j)}$ in place of $a^\dagger_{(s_i,s_j)}$ provides an upper bound on the true $a(b)$-influence curve. As an example, we apply this technique to a Bayesian network in Section 6.4.

6.3.3 Mis-specified $a(b)$-influence Curve. Suppose the practitioner specifies the prior set Θ as "the set of all priors whose $a(b)$-influence curve is upper bounded by a given function $a_1(b)$", but the actual influence curve for the true prior is some other function $a_2(b)$, which is not upper bounded by $a_1(b)$. To get a target Pufferfish parameter ϵ_{puffer}, the practitioner will incorrectly set $\epsilon_{\text{DP}} = \frac{\epsilon_{\text{puffer}} - a_1(b^\star)}{b^\star}$ for some integer $b^\star$. What is the true Pufferfish parameter $\epsilon'_{\text{Puffer}}$ that is achieved by this ϵ_{DP}-mechanism? Using Theorem 6,

$$\begin{aligned}\epsilon'_{\text{puffer}} &= \inf_{b>0} \left(a_2(b) + b \cdot \frac{\epsilon_{\text{puffer}} - a_1(b^\star)}{b^\star}\right) \\ &\le \left(a_2(b^\star) + b^\star \cdot \frac{\epsilon_{\text{puffer}} - a_1(b^\star)}{b^\star}\right) \\ &= \epsilon_{\text{puffer}} + (a_2(b^\star) - a_1(b^\star)).\end{aligned}$$

Thus, if the practitioner fails to specify a correct upper bound on the influence curve, the actual privacy parameter might be weaker by at most the difference between the two curves at $b^\star$: $a_2(b^\star) - a_1(b^\star)$.

6.4 Example $a(b)$-influence Curves

Dependent DP [30], which assumes an item is correlated with at most L-1 other items, is a special case of the $a(b)$-influence curve where $a(b) = \infty$ for $b < L$ and $a(b) = 0$ for $b \ge L$. The connection between differential privacy and Pufferfish with an independence assumption [27] means that per-entry DP is equivalent to setting $a(0) = \infty$ and $a(b) = 0$ for $b \ge 1$. Similarly, when there are n attributes, using group DP to protect groups of m' records is equivalent to setting $a(b) = \infty$ for $b < m'n$ and $a(b) = 0$ otherwise. In other applications, priors are often specified as Markov and Bayesian networks [4, 40, 41]. Thus, we next provide examples of $a(b)$-influence curve derivations for Markov chains, multivariate Gaussian priors, and real-world Bayesian networks with prior uncertainty.

Markov Chain. The $a(b)$-influence curve for a simple Markov chain can be calculated analytically as the following. The proofs can be found in the full version [2].

Example 6.1. We consider a binary and time-homogeneous chain with each node's value being 0 or 1. We consider the value of each node as a secret. Its transition matrix is:

$$P = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}, \qquad 0 < p, q < 1,$$

and the prior distribution of the Markov chain is its stationary distribution. We set the chain has length T, and T is large enough so we only consider $b << T$.

In this case, we can have a closed-form solution for $a(b)$-influence curve. Let us consider an arbitrary i-th node X_i's value as a secret. for a fixed b, we partition the nodes into $\mathcal{H}$ and $\mathcal{L}$, such that $\mathcal{H}$

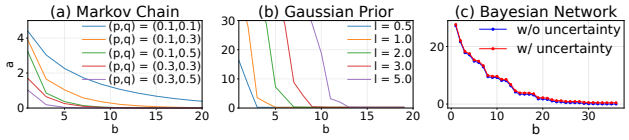

Figure 2: $a(b)$-influence curves under different priors: (a) binary Markov chains and (b) multivariate Gaussian priors. (c): $a(b)$ influence curve for ALARM network. The blue curve is calculated from the exact prior, and the red curve is an upper bound by assuming 20% prior uncertainty.

contains X_i and $b-1$ surrounding nodes. Let d_L (resp., d_R) denote the distance from X_i to the closest node of $\mathcal{L}$ on the left (resp., right), so that $d_L + d_R - 1 = b$. By the Markov property, conditioning on all variables in $\mathcal{L}$ influences X_i only through these two closest low-influence nodes, and the leakage through $\mathcal{L}$ reduces to the leakage through those two nodes.

Thus, finding an optimal $a(b)$-influence curve becomes a one-dimensional optimization problem of finding (d_L, d_R) that maximizes the leakage. The maximum leakage occurs when these nodes both have the same value of 0 or 1. When $q > p$, the chain tends to stay at 1 more strongly than at 0, so the leakage is maximum when these nodes have values 0. When $q \le p$, the worst case is instead achieved when both have values 1. Finally, from midpoint convexity, the influence is minimized when the two nearest low-influence nodes are placed as evenly as possible on both sides of X_i, giving $d_L^\star = \lfloor (b+1)/2 \rfloor$ and $d_R^\star = \lceil (b+1)/2 \rceil$. In summary, the optimal $a(b)$-influence curve can be calculated as:

Theorem 8. *Given the binary Markov chain in Example 6.1, let $\lambda = q + p - 1$, the $a(b)$-influence curve is calculated by*

$$a^\star(b) = \left|\log \frac{\pi_0 + \lambda^{d_L}(1-\pi_0)}{\pi_0 - \lambda^{d_L}\pi_0}\right| + \left|\log \frac{\pi_0 + \lambda^{d_R}(1-\pi_0)}{\pi_0 - \lambda^{d_R}\pi_0}\right| \text{ if } q > p,$$

$$a^\star(b) = \left|\log \frac{\pi_1 + \lambda^{d_L}(1-\pi_1)}{\pi_1 - \lambda^{d_L}\pi_1}\right| + \left|\log \frac{\pi_1 + \lambda^{d_R}(1-\pi_1)}{\pi_1 - \lambda^{d_R}\pi_1}\right| \text{ if } q \le p,$$

where $d_L = \left\lfloor \frac{b+1}{2} \right\rfloor$ and $d_R = \left\lceil \frac{b+1}{2} \right\rceil$. The star ($\star$) indicates that this curve is optimum (tight).

Figure 2(a) plots the resulting $a(b)$-influence curves for different values of p and q. When p and q are close to 1/2 (i.e., the chain exhibits weak temporal correlation), the resulting $a(b)$-curve lies closer to the origin and quickly decays—indicating that the secret does not propagate across nodes much. As p or q decreases towards 0 (or increases towards 1, which we omit due to symmetry), the chain exhibits stronger temporal correlation, and the $a(b)$-influence curve also spreads out over b.

Multivariate Gaussian. Next, we illustrate how we can derive the $a(b)$-influence curve when the prior distribution θ is a multivariate Gaussian. While a closed-form solution is not possible, we can still efficiently calculate the optimal $a(b)$-influence.

Example 6.2. Consider a single-record continuous dataset represented by a vector $\mathbf{x} \in \mathbb{R}^n$. Throughout this example, the prior distribution is a truncated multivariate Gaussian,

$$\mathbf{x} \sim \mathcal{N}(0, \Sigma) \;\; \mathbf{x} \in [-\gamma, \gamma]^n,$$

where $\gamma > 0$ bounds the support. The covariance matrix Σ is:

$$\Sigma_{jk} \;=\; \exp\left(-\frac{(j-k)^2}{\ell}\right), \qquad j,k \in [n].$$

The secret is defined as *region events* on each coordinate. Specifically, the secret is whether each coordinate x_i's value is in a certain range:

$$s_i(r) \;=\; \{x_i \in [r, r+\delta], \qquad r \in [-\gamma, \gamma-\delta]\}.$$

A secret pair on i is given by two disjoint region events $(s_i(r),\ s_i(r'))$, where $[r, r+\delta] \cap [r', r'+\delta] = \emptyset$.

Among all the entries, the middle coordinate $x_{i^\star}$ has the greatest information leakage, where $i^\star = \lceil n/2 \rceil$. Therefore, we only need to consider secrets defined on $x_{i^\star}$, $s(r) \equiv \{x_{i^\star} \in [r, r+\delta]\}$. For a fixed b, the optimal low-influence index set $\mathcal{L}^\star$ with size $n-b$ is the set of indices with the farthest distance to $i^\star$. Under $\mathcal{L}^\star$, we calculate $a^\star(b) = \max_{s(r),s(r'),x_{\mathcal{L}}} \log \frac{\Pr[x_{\mathcal{L}} | x_{i^\star} \in [r,r+\delta]]}{\Pr[x_{\mathcal{L}} | x_{i^\star} \in [r',r'+\delta]]}$, which is the optimal (tight) $a(b)$. Although $\mathbf{x}_{\mathcal{L}}$ is high-dimensional, we have:

$$\Pr(\mathbf{x}_{\mathcal{L}} \mid x_{i^\star}) \propto \Pr(x_{i^\star} \mid \mathbf{x}_{\mathcal{L}}) / \Pr(x_{i^\star}).$$

The conditional distribution $\Pr(x_{i^\star} \mid \mathbf{x}_{\mathcal{L}})$ is a one-dimensional Gaussian distribution, denoted as $\mathcal{N}(\mu_{i^\star,\mathcal{L}},\ \sigma^2_{i^\star,\mathcal{L}})$. As $\sigma^2_{i^\star,\mathcal{L}}$ does not depend on $\mathbf{x}_{\mathcal{L}}$, for fixed $\mathcal{L}$, the maximization over $\mathbf{x}_{\mathcal{L}}$ reduces to sweeping over $\mu_{i^\star,\mathcal{L}}$.

Theorem 9. *Consider Example 6.2 where $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$ is truncated to $[-\gamma, \gamma]^n$ and Σ is induced by the Gaussian-process kernel $\Sigma_{jk} = \exp(-(j-k)^2/\ell)$. Fix a region length $\delta << \gamma$ and define region secrets as $s_i(r) \equiv \{x_i \in [r, r+\delta]\}, r \in [-\gamma, \gamma-\delta]$ on any coordinate i. The effective set of secret pairs to search for is defined on the center coordinate $x_{i^\star}$ with $i^\star = \lceil n/2 \rceil$. Moreover, for each fixed b: (1) the optimal low-influence index set $\mathcal{L}$ with size $n-b$ is the set of indices with the farthest distance to $i^\star$, and (2) the max influence of $\mathbf{x}_{\mathcal{L}}$ over $x_{i^\star}$ reduces to a one-dimensional sweep over the mean on the conditional distribution $x_{i^\star} | \mathbf{x}_{\mathcal{L}}$.*

We report $a(b)$-curve for decay factor $\ell = \{0.5, 1, 2, 3, 5\}$, and pick $\gamma = 5, \delta = 0.1$. The results are shown in Figure 2(b). As ℓ grows larger, the correlation among entries becomes stronger, and the corresponding $a(b)$-curve is farther away from the origin.

Bayesian Network with Uncertain Prior. Finally, we study a realistic use-case having a general Bayesian network as the prior, with uncertainty added as in Section 6.3.2. ALARM [4] is a benchmark for patient monitoring. It uses an expert-constructed Bayesian network, which models the medical knowledge of dependencies between diagnostic hypotheses, intermediate physiological states, and observable monitoring variables through 37 nodes, 46 directed edges, 16 finding variables, and 13 intermediate variables. All nodes are categorical variables representing clinical conditions, physiological states, or measurements. In our setup, the secret is the value of an individual node in a patient record.

Figure 2(c) plots the calculated $a(b)$-influence curve (blue line), using the upper-bound estimation method from Section 6.3.1 (there are too many nodes to calculate the exact curve). Additionally, we calculated a curve for the case of prior uncertainty using the techniques of Section 6.3.2 (red line). Specifically, we used 20% prior uncertainty, i.e., $u(\mathcal{D}_{\mathcal{L}}, s) = 1.2p(\mathcal{D}_{\mathcal{L}}, s), l(\mathcal{D}_{\mathcal{L}}, s) = 0.8p(\mathcal{D}_{\mathcal{L}}, s)$. Even with this moderate amount of fluctuation, the $a(b)$-influence curve does not change much to account for potential uncertainty.

	Ours-Exp w/o Uncertainty			MQM w/o Uncertainty			**Ours-Exp** w/ Uncertainty			MQM w/ Uncertainty		
ϵ_{puffer}	Acc@1	Acc@2	Acc@3	Acc@1	Acc@2	Acc@3	Acc@1	Acc@2	Acc@3	Acc@1	Acc@2	Acc@3
0.5	**79.0%**	**63.1%**	**58.5%**	36.3%	26.7%	20.83%	**42.4%**	**34.0%**	**28.4%**	16.2%	11.1%	8.1%
1	**88.5%**	**76.7%**	**72.5%**	46.4%	36.3%	30.6%	**81.8%**	**66.9%**	**62.4%**	38.6%	28.9%	23.0%
2	**94.0%**	**85.7%**	**82.8%**	60.8%	47.5%	43.0%	**92.8%**	**82.9%**	**79.6%**	55.2%	43.6%	38.8%
3	**95.7%**	**89.6%**	**87.7%**	72.1%	55.3%	50.8%	**95.1%**	**88.3%**	**86.1%**	68.1%	52.3%	47.9%
4	**96.7%**	**91.7%**	**90.4%**	79.1%	61.7%	57.0%	**96.4%**	**91.0%**	**89.5%**	76.7%	59.2%	54.6%
5	**97.3%**	**93.0%**	**92.1%**	83.4%	66.9%	62.0%	**97.1%**	**92.6%**	**91.5%**	81.9%	65.0%	60.1%

Table 2: Acc@k for evaluating the Top-3 popular visited locations using the Foursquare check-in dataset, under various ϵ_{puffer}. The metric is higher the better, and the best mechanism for each row is highlighted in bold. We also report the numbers when 20% uncertainty is set around the prior ("w/ Uncertainty"). Our new mechanism (Ours-Exp) outperforms MQM in all cases.

	Ours-Exp w/o Uncertainty			MQM w/o Uncertainty			**Ours-Exp** w/ Uncertainty			MQM w/ Uncertainty		
ϵ_{puffer}	HR (↑)	NDCG (↑)	ℓ_1 (↓)	HR (↑)	NDCG (↑)	ℓ_1 (↓)	HR (↑)	NDCG (↑)	ℓ_1 (↓)	HR (↑)	NDCG (↑)	ℓ_1 (↓)
0.5	**78.7%**	**0.937**	**765**	37.6%	0.726	5500	**45.6%**	**0.763**	**3838**	18.6%	0.606	17842
1	**89.3%**	**0.977**	**295**	49.8%	0.791	3025	**81.7%**	**0.952**	**634**	40.4%	0.742	4746
2	**94.4%**	**0.987**	**105**	64.3%	0.868	1554	**93.1%**	**0.986**	**147**	59.3%	0.841	1976
3	**96.3%**	**0.993**	**69**	73.3%	0.917	913	**95.5%**	**0.991**	**83**	70.1%	0.900	1125
4	**97.2%**	**0.994**	**60**	79.2%	0.946	571	**96.9%**	**0.994**	**61**	77.1%	0.933	684
5	**97.8%**	**0.994**	**56**	83.4%	0.964	386	**97.6%**	**0.994**	**57**	81.9%	0.957	449

Table 3: Hit Rate@K (HR), NDCG@K (NDCG), and ℓ_1 Count Error (ℓ_1) for evaluating the Top-3 popular visited locations using the Foursquare check-in dataset, under various ϵ_{puffer}. Hit Rate@K and NDCG@K are higher the better (indicated by the ↑ symbol), and ℓ_1 Count Error is lower the better (indicated by the ↓ symbol). The best among the mechanisms are highlighted in bold. We also report the numbers when 20% uncertainty is set around the prior ("w/ Uncertainty"). Our new mechanism (Ours-Exp) outperforms MQM in all privacy parameters and all metrics.

7 EXPERIMENTS

Our experiments highlight the benefit of our approach: (1) they rely on composition, (2) they take into account uncertainty about data-generating priors, and (3) they demonstrate that the freedom of adapting existing DP algorithms often provides a quick way of improving over hand-crafted Pufferfish mechanisms.

7.1 Experimental Setup

7.1.1 Datasets. We evaluate the mechanisms with two representative datasets where strong correlations exist between datapoints.

Foursquare check-in dataset [47, 48] contains users' location data with a corresponding timestamp across 77 countries. Each location is associated with a semantic label. We merged similar labels (e.g., "Cocktail Bar" with "Bar") and only chose the 77 most common labels worldwide, mapping all other locations as "Other". Each datapoint of the resulting dataset is location information of a particular user at a particular time. The rules for merging labels are in the Appendix of [2]. Clearly, locations of a user within a short time period are expected to be highly correlated.

Capture24 dataset [8] contains 151 participants' activity data monitored across 24 hours. Participants are grouped by age (18–29, 30–37, 38–52, 53+). The dataset annotated activities in multiple ways, and we use the `WillettsMET2018` annotation, where participants' activities are labelled into 11 activity categories. Again, activities of a participant within a nearby time period are expected to be highly correlated.

7.1.2 Query of Interest. As a demonstration, we study *noisy Top-K* queries. In particular, for the Foursquare check-in dataset, the query we evaluate is: *"Within each country, what are the Top-3 most frequently visited locations?"*. For the Capture24 dataset. the query we evaluate is: *"Within each age group, what are the Top-3 most frequent activities?"*. These are deliberately chosen to evaluate the types of queries that prior work [41] did not consider. For the queries that prior works studied (e.g., querying the average value [41]), mechanisms from prior works can be used.

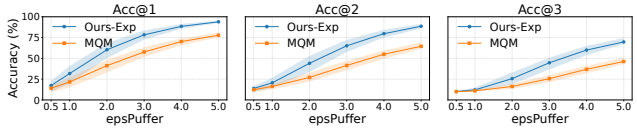

Figure 3: Acc@k for evaluating the Top-3 most frequent activities for each participant in the Capture24 dataset under various ϵ_{puffer}. Ours-Exp outperforms MQM across all privacy parameters. Shaded regions indicate the utility range when assuming 20% uncertainty around the prior.

7.1.3 Evaluated Mechanisms. We compare an approach (**Ours-Exp**) based on our framework to MQM [41].

Ours-Exp uses our framework's flexibility in choosing appropriate mechanisms by adapting the exponential mechanism [32] of differential privacy and running it k times. The item returned in the first run is treated as the top item. The second run samples among

the remaining items to produce the estimated second-ranked item, etc. Given a target Pufferfish privacy parameter ϵ_{puffer}, the privacy budgets are allocated evenly among the k runs so that the overall parameter via composition is ϵ_{puffer}.

MQM is a Pufferfish mechanism proposed by prior work [41] . It adapts the Laplace mechanism of differential privacy to achieve Pufferfish privacy with composition. Noise is added to the m categories and the top-k noisy categories are selected.

7.1.4 Prior Distribution Construction. Using MQM or our method (Ours-Exp) requires specifying a prior θ. Following prior works [41], we assume the temporal data are generated by a first-order Markov chain, and use the data to fit a Markov model with a stationary transition matrix. For the Foursquare check-in dataset, each user's location is a state, and a location change is considered a state transition. We truncate the dataset to only leave the check-in data for the first 500 days. We built a 78×78 transition matrix using the first 400 out of 500 days of data, and used the remaining 100 days for testing. For the Capture24 dataset, each participant's activity is a state, and activity change is considered a state transition. We hold out 8 participants from each age group for testing and use the rest to fit the 11×11 transition matrix. Note that MQM requires the prior θ to be a Markov chain, while our general approach from Section 6.2 works with an arbitrary prior. We additionally ran experiments with 20% uncertainty added around the prior using the results of Section 6.3.2 (i.e., $u(\mathcal{D}_{\mathcal{L}}, s) = 1.2p(\mathcal{D}_{\mathcal{L}}, s), l(\mathcal{D}_{\mathcal{L}}, s) = 0.8p(\mathcal{D}_{\mathcal{L}}, s)$.). We denote results with uncertainty as "w/ Uncertainty" and results with no uncertainty as "w/o Uncertainty".

7.1.5 Utility Metrics. We evaluate with the following metrics:

Acc@k measures the accuracy of predicting the k-th popular category correctly. The exact ranking (i.e., k) must be predicted correctly (e.g., if the most popular location is predicted as second-most popular, it is regarded as incorrect).

Hit Rate@K measures the precision of the correct Top-K categories appearing in the predicted Top-K. The order does not matter, as long as the correct category appears in the predicted Top-K.

NDCG@K, or Normalized Discounted Cumulative Gain, is a frequently used metric in ranking systems [22]. NDCG accounts for the order of the answer and rewards putting higher-ranked categories near the top, even if the exact rank was incorrect. Precisely, NDCG is calculated by:

$$\text{DCG@}K = \sum_{k=1}^{K} \frac{\text{rel}(\hat{\pi}_k)}{\log_2(k+1)}, \quad \text{NDCG@}K = \frac{\text{DCG@}K}{\text{IDCG@}K},$$

where $\text{rel}(i)$ is the ground-truth score for item i (we use true counts), and IDCG is the DCG of the ideal ranking. By normalizing to the best possible ranking for the same query (IDCG), scores lie in $[0, 1]$, and a higher score means better prediction.

ℓ_1 **Count Error** is the sum of the absolute difference between the predicted Top-K's counts and the true Top-K's counts. We additionally included this metric because it can give us a sense of how different the actual counts are when misprediction occurs. For example, if the most and second-most popular categories have similar counts, this metric does not penalize swapping their ranking in prediction; however, if their counts differ significantly, getting the order wrong would significantly increase the error. Specifically, the error is calculated by $\text{L1} = \sum_{k=1}^{K} |\tilde{c}_k - c_k|$, where $\tilde{c}_k$ is the count of the category that the noisy mechanism places at rank k, and c_k is the count of the category that truly belongs at rank k.

Majority We additionally show another example of what the accuracy would be if we use the Top-1/2/3 labels from the train data as the prediction to evaluate the complexity of the question. The Majority only achieved 18.61% and 16.67% accuracy averaged across 3 positions. A full overview of the datasets, prior constructions, and implementation choices is provided in Appendix in [2].

7.2 Results: Foursquare Check-in Dataset

Table 2 (Acc@k) and Table 3 (Hit Rate@K, NDCG@K, ℓ_1 Count Error) show the utility results for MQM and our new mechanism (Ours-Exp). Both Table 2 and Table 3 show that, across all privacy parameters and metrics, Ours-Exp outperformed MQM both with and without prior uncertainty. Ours-Exp outperforms the Laplace-based MQM method because it uses the exponential mechanism of differential privacy, which is known to perform better for Top-k queries [15], while using our Theorem 6 to translate it into a composable Pufferfish mechanism. Also, incorporating 20% uncertainty into the computation of the worst-case leakage $a(b)$-influence curve only mildly affects utility in most cases. Except for $\epsilon_{\text{puffer}} = 0.5$, the utility decrease is modest, suggesting that our framework can preserve useful accuracy even when the prior is only approximately specified. It is worth stating explicitly that the utility improvement in our experiments comes from our framework enabling the use of alternative DP mechanisms that are better matched to the query (the exponential mechanism for Top-K queries).

7.3 Results: Capture24 Dataset

Figure 3 shows the utility of Ours-Exp for various privacy parameters and metrics for the Capture24 dataset. Again, in all the tested setups, Ours-Exp outperformed or showed on-par utility compared to the baselines (MQM; Ours-Exp also outperformed group DP, which is omitted). Note that this dataset only has 11 activities, so 9–10% accuracy is almost random guessing.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we study how to add composition properties to the Pufferfish framework. We analyzed privacy collapses under composition and provided necessary and sufficient conditions for linear composition. We conclude that the constraints on mechanisms required by the ϵ_{puffer}-Pufferfish framework should be augmented with ϵ_{DP}-differential privacy constraints. We introduced the $a(b)$-influence curve to translate between those two privacy parameters. Future work includes algorithm design under various classes of priors and extending the necessary and sufficient conditions for privacy definitions like Renyi-Pufferfish.

ACKNOWLEDGMENTS

This work was supported by the US National Science Foundation under Awards CNS-2349610 and CNS-2317232. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] John M Abowd, Robert Ashmead, Ryan Cumings-Menon, Simson Garfinkel, Micah Heineck, Christine Heiss, et al. 2022. The 2020 census disclosure avoidance system topdown algorithm. *arXiv preprint arXiv:2204.08986* (2022).

[2] Jiamu Bai, Guanlin He, Xin Gu, Daniel Kifer, and Kiwan Maeng. 2026. Composition for Pufferfish Privacy. *arXiv preprint arXiv:2602.02718* (2026).

[3] Raef Bassily, Adam Groce, Jonathan Katz, and Adam Smith. 2013. Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 439–448.

[4] Ingo A Beinlich, Henri Jacques Suermondt, R Martin Chavez, and Gregory F Cooper. 1989. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *AIME 89: Second European Conference on Artificial Intelligence in Medicine*. Springer, 247–256.

[5] Raghav Bhaskar, Abhishek Bhowmick, Vipul Goyal, Srivatsan Laxman, and Abhradeep Thakurta. 2011. Noiseless database privacy. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 215–232.

[6] Guy Blanc, William Pires, and Toniann Pitassi. 2025. Differential privacy from axioms. *arXiv preprint arXiv:2511.21876* (2025).

[7] Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong. 2017. Quantifying differential privacy under temporal correlations. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 821–832.

[8] Shing Chan, Yuan Hang, Catherine Tong, Aidan Acquah, Abram Schonfeldt, Jonathan Gershuny, and Aiden Doherty. 2024. CAPTURE-24: A large dataset of wrist-worn activity tracker data collected in the wild for human activity recognition. *Scientific Data* 11, 1 (2024), 1135.

[9] Damien Desfontaines, Esfandiar Mohammadi, Elisabeth Krahmer, and David Basin. 2019. Differential privacy with partial knowledge. *arXiv preprint arXiv:1905.00650* (2019).

[10] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., 3574–3583.

[11] Ni Ding. 2022. Kantorovich mechanism for pufferfish privacy. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 5084–5103.

[12] Ni Ding. 2024. Approximation of pufferfish privacy for Gaussian priors. *IEEE Transactions on Information Forensics and Security* 19 (2024), 5630–5640.

[13] Irit Dinur and Kobbi Nissim. 2003. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 202–210.

[14] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 486–503.

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[16] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. ACM, 1054–1067.

[17] Farhad Farokhi. 2021. Noiseless privacy: Definition, guarantees, and applications. *IEEE Transactions on Big Data* 9, 1 (2021), 51–62.

[18] Arpita Ghosh and Robert Kleinberg. 2016. Inferential privacy guarantees for differentially private mechanisms. *arXiv preprint arXiv:1603.01508* (2016).

[19] Google. [n.d.]. Tensorflow Privacy Github. https://github.com/tensorflow/privacy. Last accessed June 24, 2026.

[20] Xi He, Ashwin Machanavajjhala, and Bolin Ding. 2014. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1447–1458.

[21] Ron S Jarmin, John M Abowd, Robert Ashmead, Ryan Cumings-Menon, Nathan Goldschlag, Michael B Hawes, Sallie Ann Keller, Daniel Kifer, Philip Leclerc, Jerome P Reiter, et al. 2023. An in-depth examination of requirements for disclosure risk assessment. *Proceedings of the National Academy of Sciences* 120, 43 (2023), e2220558120.

[22] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.

[23] Noah Johnson, Joseph P. Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proc. VLDB Endow.* 11, 5 (Jan. 2018), 526–539.

[24] Noah M. Johnson, Joseph P. Near, Joseph M. Hellerstein, and Dawn Song. 2018. Chorus: Differential Privacy via Query Rewriting. *CoRR* abs/1809.07750 (2018). arXiv:1809.07750

[25] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2015. The composition theorem for differential privacy. In *International conference on machine learning*. PMLR, 1376–1385.

[26] Daniel Kifer and Ashwin Machanavajjhala. 2011. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 193–204.

[27] Daniel Kifer and Ashwin Machanavajjhala. 2014. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems (TODS)* 39, 1 (2014), 1–36.

[28] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[29] Wenjuan Liang, Hong Chen, Ruixuan Liu, Yuncheng Wu, and Cuiping Li. 2020. A pufferfish privacy mechanism for monitoring web browsing behavior under temporal correlations. *Computers & Security* 92 (2020), 101754.

[30] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. 2016. Dependence makes you vulnberable: Differential privacy under dependent tuples.. In *NDSS*, Vol. 16. 21–24.

[31] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. 2008. Privacy: Theory meets Practice on the Map. In *2008 IEEE 24th International Conference on Data Engineering*. 277–286.

[32] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 94–103.

[33] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 19–30.

[34] Casey Meehan and Kamalika Chaudhuri. 2021. Location trace privacy under conditional priors. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2881–2889.

[35] Solomon Messing, Bogdan State, Chaya Nayak, Gary King, and Nate Persily. 2018. URLs Dataset for RFP.pdf. In *Facebook URL Shares*. Harvard Dataverse.

[36] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 263–275.

[37] OpenDP. 2026. OpenDP. https://github.com/opendp/opendp. Last accessed June 24, 2026.

[38] Clément Pierquin, Aurélien Bellet, Marc Tommasi, and Matthieu Boussard. 2024. Rényi pufferfish privacy: General additive noise mechanisms and privacy amplification by iteration via shift reduction lemmas. In *International Conference on Machine Learning (ICML 2024)*.

[39] Masoumeh Shafieinejad, Huseyin Inan, Marcello Hasegawa, and Robert Sim. 2021. On privacy and confidentiality of communications in organizational graphs. *arXiv preprint arXiv:2105.13418* (2021).

[40] Shuang Song and Kamalika Chaudhuri. 2017. Composition properties of inferential privacy for time-series data. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 814–821.

[41] Shuang Song, Yizhen Wang, and Kamalika Chaudhuri. 2017. Pufferfish privacy mechanisms for correlated data. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1291–1306.

[42] Apple Differential Privacy Team. 2017. Learning with Privacy at Scale. *Apple Machine Learning Journal* 1, 8 (2017).

[43] Michael Carl Tschantz, Shayak Sen, and Anupam Datta. 2020. SoK: Differential privacy as a causal property. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 354–371.

[44] Stanley L Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American statistical association* 60, 309 (1965), 63–69.

[45] Zheng Xu, Yanxiang Zhang, Galen Andrew, Christopher Choquette, Peter Kairouz, Brendan Mcmahan, Jesse Rosenstock, and Yuanbo Zhang. 2023. Federated Learning of Gboard Language Models with Differential Privacy. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*. Association for Computational Linguistics, 629–639.

[46] Bin Yang, Issei Sato, and Hiroshi Nakagawa. 2015. Bayesian differential privacy on correlated data. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of Data*. 747–762.

[47] Dingqi Yang, Daqing Zhang, Longbiao Chen, and Bingqing Qu. 2015. Nationtelescope: Monitoring and visualizing large-scale collective behavior in lbsns. *Journal of Network and Computer Applications* 55 (2015), 170–180.

[48] Dingqi Yang, Daqing Zhang, and Bingqing Qu. 2016. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7, 3 (2016), 1–23.

[49] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. 2021. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298* (2021).

[50] Tao Zhang, Bradley A Malin, Netanel Raviv, and Yevgeniy Vorobeychik. 2025. Differential Confounding Privacy and Inverse Composition. In *2025 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 1–6.

[51] Tao Zhang and Yevgeniy Vorobeychik. 2025. Sliced Rényi Pufferfish Privacy: Directional Additive Noise Mechanism and Private Learning with Gradient Clipping. *arXiv preprint arXiv:2512.01115* (2025).

[52] Wanrong Zhang, Olga Ohrimenko, and Rachel Cummings. 2022. Attribute privacy: Framework and mechanisms. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. 757–766.