A Rigorous and Customizable Framework for Privacy

Daniel Kifer
Penn State University

Ashwin Machanavajjhala
Yahoo! Research

ABSTRACT

In this paper we introduce a new and general privacy framework called Pufferfish. The Pufferfish framework can be used to create new privacy definitions that are customized to the needs of a given application. The goal of Pufferfish is to allow experts in an application domain, who frequently do not have expertise in privacy, to develop rigorous privacy definitions for their data sharing needs. In addition to this, the Pufferfish framework can also be used to study existing privacy definitions.

We illustrate the benefits with several applications of this privacy framework: we use it to formalize and prove the statement that differential privacy assumes independence between records, we use it to define and study the notion of *composition* in a broader context than before, we show how to apply it to protect unbounded continuous attributes and aggregate information, and we show how to use it to rigorously account for prior data releases.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Statistical Databases; K.4.1 [**Computers and Society**]: Privacy

General Terms

Theory

Keywords

privacy, differential privacy

1. INTRODUCTION

With improvements in data collection technologies, increased emphasis on data analytics, and the increasing need for different parties to share datasets, the field of statistical privacy is seeing an unprecedented growth in importance and diversity of applications. These applications include protecting privacy and confidentiality in (computer) network data collections [32], protecting privacy and identifiability in genome-wide association studies (GWAS) [19], protecting confidentiality in Census data products [25, 15], etc. In each case, the goal is to release useful information (i.e. privacy-preserving query answers or a sanitized version of the dataset) while protecting confidential information.

These applications have diverse characteristics: some data sets are more sensitive in nature than others; some applications must protect aggregate secrets while others must protect secrets about individuals; in some cases, only certain attributes (or attribute values) need to be protected, etc.

Furthermore, properties of the data-generating process, such as correlations between records, play an important role. Assumptionless privacy definitions are a myth: if one wants to publish useful, privacy-preserving sanitized data then one *must* make assumptions about the original data and data-generating process [22, 14].

Thus application domain experts, who are frequently not experts in privacy, cannot simply use a single generic privacy definition – they must develop a new privacy definition or customize an existing one. Without guidance this has resulted in ad-hoc solutions and endless cycles of attacks on privacy, followed by proposed fixes, followed by new attacks.

At the same time, privacy experts need new methods for analyzing new and existing privacy definitions in order to evolve the state of the art. Many of the newer definitions are based on "indistinguishability" – making it difficult for attackers to distinguish between certain pairs of datasets; we show how these approaches frequently obscure the semantics of the privacy guarantees that are provided.

In this paper we present a new privacy framework, called Pufferfish. This framework can be used to study existing privacy definitions like differential privacy [12], to study important concepts like *composition* [17], and to generate privacy definitions that are customized for different application domains. The Pufferfish framework follows modern design guidelines such as adherence to privacy axioms [21, 20] and making assumptions as explicit as possible.

The contributions of this paper are:

- A new privacy framework which provides rigorous statistical semantic privacy guarantees.
- An application of the framework to differential privacy which formalizes and then proves the statement that differential privacy "assumes" independence between records.
- Another application to differential privacy showing how to modify it in response to prior release of non-differentially private information (naive applications of differential privacy can lead to a privacy breach [22]).

- An application of the framework that allows us to expand the notion of composition between privacy definitions and to provide conditions under which certain types of composition hold.
- A general application to datasets with unbounded continuous variables; in particular, we show how to prevent an attacker from accurately inferring the value of an unbounded continuous attribute (both in terms of relative and absolute error). We further show how this is related to the protection of aggregate secrets.

The outline of the paper is as follows. In Section 2 we introduce basic concepts and notation. In Section 3 we present the Pufferfish framework. We discuss related work in Section 4. We show that Pufferfish satisfies fundamental privacy axioms in Section 5. Subsequent sections present a variety of applications of this framework. We use Pufferfish to analyze differential privacy and prove its record-independence assumption in Section 6. We show how to deal with continuous attributes and aggregate secrets in Section 7. We use Pufferfish to study composition in Section 8. We show how to provide privacy while accounting for prior data releases in Section 9. Proofs can be found in the full version of this paper.

2. NOTATION AND TERMINOLOGY

Let $\mathcal{I}$ be the set database instances that are possible for a given application. For datasets that are collections of records, we let $\mathcal{T}$ represent the domain of tuples, we use the symbol t to represent a value in the domain $\mathcal{T}$, and we use the variable r to represent a record.[1] For ease of explanation and to simplify notation, we will assume each individual is associated with at most one record; we note that the Pufferfish framework does not need this assumption.

In the setting we consider, a *data curator* has a dataset $\boldsymbol{\mathfrak{D}ata}$. Let records($\boldsymbol{\mathfrak{D}ata}$) denote the set of records in $\boldsymbol{\mathfrak{D}ata}$. To an *attacker* (who does not know what the dataset is), $\boldsymbol{\mathfrak{D}ata}$ represents a *random variable*. The data curator will choose a *privacy definition* and an *privacy mechanism* (algorithm) $\mathfrak{M}$ that satisfies that privacy definition. The data curator will then apply $\mathfrak{M}$ to the data to obtain a *sanitized output* $\omega \equiv \mathfrak{M}(\boldsymbol{\mathfrak{D}ata})$.

The attacker considers the true dataset $\boldsymbol{\mathfrak{D}ata}$ to be a random variable. We use the letter θ to represent a probability distribution and, for $D \in \mathcal{I}$, we will use the notation $P(\boldsymbol{\mathfrak{D}ata} = D \mid \theta)$ to represent the probability, under θ, that the true dataset is D. For convenience, we summarize the notation used in this paper in Table 1.

$\mathcal{I}$	The set of possible database instances.
D	A dataset belonging to $\mathcal{I}$.
$\boldsymbol{\mathfrak{D}ata}$	A random variable representing the true dataset (which is unknown to the attacker).
$\mathcal{T}$	The domain of tuples.
t	A value in $\mathcal{T}$.
r_i	The i^{th} record in a dataset.
$\mathcal{H}$	The set of all individuals. $\mathcal{H} = \{h_1, h_2, \dots\}$
$\mathbb{S}$	Set of potential secrets. Revealing s or $\neg s$ may be harmful if $s \in \mathbb{S}$.
σ_i	$r_i \in$ records($\boldsymbol{\mathfrak{D}ata}$): The statement that the record r_i belonging to individual h_i is in the data.
$\sigma_{(i,t)}$	$r_i \in$ records($\boldsymbol{\mathfrak{D}ata}$)$\wedge r_i = t$: The statement that the record r_i belonging to individual h_i has value $t \in \mathcal{T}$ and is in the data.
$\mathbb{S}_{\text{pairs}}$	Discriminative pairs. $\mathbb{S}_{\text{pairs}} \subset \mathbb{S} \times \mathbb{S}$.
$\mathbb{D}$	The set of evolution scenarios: a conservative collection of plausible data generating distributions.
θ	A probability distribution. The probability, under θ, that the data equals D_i is $P(\boldsymbol{\mathfrak{D}ata} = D_i \mid \theta)$.
$\mathfrak{M}$	A privacy mechanism: a deterministic or randomized algorithm (often used in the context of a privacy definition)

Table 1: Table of Notation

3. THE PUFFERFISH FRAMEWORK

The Pufferfish framework requires a domain expert to specify three crucial components: a set of *potential secrets* $\mathbb{S}$, a set of *discriminative pairs* $\mathbb{S}_{\text{pairs}} \subseteq \mathbb{S} \times \mathbb{S}$, and a collection of *data evolution scenarios* $\mathbb{D}$. We now describe these components in detail.

The set of potential secrets $\mathbb{S}$, is an explicit specification of what we would like to protect.[2] Examples include statements such as *"the record for individual h_i is in the data"*, *"the record for individual h_i is <u>not</u> in the data"*, *"the query volume is $1-5$ million queries"*, etc. Additional examples can be found in Sections 3.2, 6, 7, and 9. A statement $s \in \mathbb{S}$ need not be true for the actual dataset – an attacker will form his own opinions about which statements are likely to be true or not. In general, a domain expert should add a statement s to the potential secrets $\mathbb{S}$ if either the claim that s is true or the claim that s is false can be harmful. The role of $\mathbb{S}$ is to provide a domain for the *discriminative pairs*, a subset of $\mathbb{S} \times \mathbb{S}$ which we discuss next.

The set of discriminative pairs $\mathbb{S}_{\text{pairs}}$ is a subset of $\mathbb{S} \times \mathbb{S}$. The role of $\mathbb{S}_{\text{pairs}}$ is to tell us *how* to protect the potential secrets $\mathbb{S}$. The main intuition is that for any discriminative pair (s_i, s_j), where $s_i \in \mathbb{S}$ and $s_j \in \mathbb{S}$, we would like to guarantee that attackers are unable to distinguish between the case where s_i is true of the actual data and the case where s_j is true of the actual data.[3] For this reason, s_i and s_j must be **mutually exclusive** but not necessarily exhaustive (it could be the case that neither is true). One example of a discriminative pair is ("Bob is in the table", "Bob is not in the table").

The discriminative pairs allow highly customizable privacy guarantees. For example, we can specify discriminative pairs such ("Bob has Cancer", "Bob has AIDS") to prevent inference about what disease Bob has (assuming only one disease is recorded). If, additionally we **avoid** specifying ("Bob is not healthy", "Bob is healthy"), then overall we are

[1]We treat records as random variables and tuples are the values they can take.

[2]After all, we cannot provide semantic guarantees if we do not know what the guarantees should be about.

[3]In general, this cannot be simulated by specifying all pairs of databases (D_k, D_ℓ) where s_i is true of D_k and s_j is true of D_ℓ – the resulting privacy definition (instantiated by Definition 3.1) will often be too strong because such D_k, D_ℓ pairs can differ almost arbitrarily.

allowing disclosure of whether Bob is sick or healthy, but if he is sick we are not allowing inference about the disease. For continuous attributes we can specify discriminative pairs of the form ("Bob's salary is $x \pm 10,000$", "Bob's salary is $[x + 20,000] \pm 10,000$") for all x to say that it is not ok to allow inference about Bob's salary to within an absolute error of $10,000$. We can similarly use this trick for aggregates to specify that it is not ok to allow inference about total number of sales to within $20,000$ units. Additional examples are given throughout the paper. We illustrate the particular importance of using discriminative pairs when we discuss continuous attributes in Section 7.

The evolution scenarios $\mathbb{D}$ can be viewed as a set of conservative assumptions about how the data evolved (or were generated) and about knowledge of potential attackers. Note that assumptions are absolutely necessary – privacy definitions that can provide privacy guarantees without making any assumptions provide little utility beyond the default approach of releasing nothing at all [22, 14]. Since the data curator wants to release useful information, the role of the domain expert will be to identify a reasonable set of assumptions; in many cases, they already do this informally [35]. More specifically, $\mathbb{D}$ is a set of probability distributions over $\mathcal{I}$ (the possible database instances). Each probability distribution $\theta \in \mathbb{D}$ corresponds to an attacker that we want to protect against and represents that attacker's belief in how the data were generated (incorporating any background knowledge and side information). We illustrate the importance of specifying these distributions in Section 6 where we analyze differential privacy. Below we give some examples of possible choices of $\mathbb{D}$ and their interpretations.

EXAMPLE 3.1 (NO ASSUMPTIONS). *$\mathbb{D}$ can consist of all possible probability distributions over database instances (i.e. including those with arbitrary correlations between records). This corresponds to making no assumptions. We explore this choice in Section 3.2.*

EXAMPLE 3.2 (I.I.D. DATA). *$\mathbb{D}$ can consist of all probability distributions over tables that generate records i.i.d. That is, for every f that is a distribution over $\mathcal{T}$ (domain of tuples), we have $\theta_f \in \mathbb{D}$ where the distribution θ_f is defined as: $P(\boldsymbol{\mathfrak{Data}} = \{r_1, \dots, r_n\} \mid \theta_f) = f(r_1)\times\cdots\times f(r_n)$. Thus each attacker can have a widely different opinion about the probability a random individual has cancer, etc. These attackers, however, do not have knowledge about specific individuals (for this, see Example 3.3).*

EXAMPLE 3.3 (INDEPENDENT BUT NOT I.I.D.). *$\mathbb{D}$ can consist of all probability distributions over tables where records are independent but may have different distributions. That is, $\mathbb{D}$ consists of all θ for which $P(\boldsymbol{\mathfrak{Data}} = \{r_1, \dots, r_n\} \mid \theta)$ equals $f_1(r_1)\times f_2(r_2)\times\cdots\times f_n(r_n)$ for arbitrary $f_1, f_2, \dots, f_n$. That is, an attacker may know that the first individual is a smoker and so the corresponding record r_1 will have a different distribution than record r_2 which corresponds to an individual who is known to be a non-smoker. This is an extension of Example 3.2 where now attackers may have additional information about all individuals. Many additional variations are possible (i.e. attackers only have information about k individuals, etc.). We shall see in Section 6 the close connection between this example and differential privacy. Note that records here are still independent, so this choice of $\mathbb{D}$ may not be appropriate to social networks where correlations between records exist.*

Role of the domain expert. The goal of our framework is to make assumptions explicit. Thus the domain expert needs to specify the potential secrets $\mathbb{S}$ and discriminative pairs $\mathbb{S}_{\text{pairs}}$ (i.e. what should be protected) and evolution scenarios (data assumptions) $\mathbb{D}$ – are data records independent, what correlation structures exist, are attributes independent, etc. Thus to use the Pufferfish framework, the domain expert simply does what he or she does best. Most importantly, the domain expert is no longer required to be a privacy expert.

DEFINITION 3.1 (PUFFERFISH PRIVACY). *Given set of potential secrets $\mathbb{S}$, a set of discriminative pairs $\mathbb{S}_{pairs}$, a set of data evolution scenarios $\mathbb{D}$, and a privacy parameter $\epsilon > 0$, a (potentially randomized) algorithm $\mathfrak{M}$ satisfies ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$ privacy if*

- *for all possible outputs $\omega \in \text{range}(\mathfrak{M})$,*
- *for all pairs $(s_i, s_j) \in \mathbb{S}_{pairs}$ of potential secrets,*
- *for all distributions $\theta \in \mathbb{D}$ for which $P(s_i \mid \theta) \neq 0$ and $P(s_j \mid \theta) \neq 0$*

the following holds[4]*:*

$$P(\mathfrak{M}(\boldsymbol{\mathfrak{Data}}) = \omega \mid s_i, \theta) \leq e^{\epsilon} P(\mathfrak{M}(\boldsymbol{\mathfrak{Data}}) = \omega \mid s_j, \theta) \quad (1)$$
$$P(\mathfrak{M}(\boldsymbol{\mathfrak{Data}}) = \omega \mid s_j, \theta) \leq e^{\epsilon} P(\mathfrak{M}(\boldsymbol{\mathfrak{Data}}) = \omega \mid s_i, \theta) \quad (2)$$

The probabilities in Equations 1 and 2 depend on possible randomness in $\mathfrak{M}$ and as well as the randomness in the data. Note that $P(s_i \mid \theta) \neq 0$ is a technical condition which ensures that the conditional probability $P(\cdot \mid s_i, \theta)$ is defined. Operationally, it means we should focus on attackers (and their associated θ) who still have uncertainty about s_i and s_j (i.e. $P(s_i \mid \theta) \neq 0$ and $P(s_j \mid \theta) \neq 0$)).

The Pufferfish framework differs from differential privacy [12] and its variants [11, 30, 5, 3, 22, 21, 25, 27, 18, 41, 33] in that it can provide precise guarantees about precisely what is being protected and under what conditions the protections hold. This enables a wide range of applications, such as those described in Sections 6, 7, 8, and 9. Other distinctions are discussed with related work in Section 4.1.

3.1 Semantic Guarantees

The semantic guarantees of the Pufferfish framework are best interpreted in terms of *odds* and *odds ratios*. If E_1 and E_2 are mutually exclusive events, then the *prior odds* of E_1 and E_2 is the fraction $\frac{P(E_1)}{P(E_2)}$. When the odds are equal to α, this simply means that E_1 is α times as likely as E_2. If we are given a piece of information A, it may alter the beliefs in the probabilities that E_1 or E_2 are true. For this reason we call $\frac{P(E_1 \mid A)}{P(E_2 \mid A)}$ the *posterior odds* of E_1 and E_2. If the prior odds and posterior odds are approximately equal, $\frac{P(E_1 \mid A)}{P(E_2 \mid A)} / \frac{P(E_1)}{P(E_2)} \approx 1$, then the event A did not provide information that was useful in discriminating between the case where E_1 was true or E_2 was true. The quantity $\frac{P(E_1 \mid A)}{P(E_2 \mid A)} / \frac{P(E_1)}{P(E_2)}$ is known as the *odds ratio* and reflects how much more likely event E_1 has become relative to E_2 after observing A.

[4]In the case of continuous outputs, these conditions are interpreted in terms of the density function or the Radon-Nikodym derivative and are required to hold *almost everywhere* - the ω for which the conditions are violated must have probability 0.

In the Pufferfish framework, each probability distribution $\theta \in \mathbb{D}$ corresponds to an attacker and reflects the attacker's probabilistic beliefs and background knowledge. For all $(s_i, s_j) \in \mathbb{S}_{\text{pairs}}$, all $\theta \in \mathbb{D}$ for which $P(s_i \mid \theta) \neq 0$ and $P(s_j \mid \theta) \neq 0$, and all $\omega \in \text{range}(\mathfrak{M})$, a simple calculation shows that Equations 1 and 2 in Definition 3.1 are equivalent to the condition:

$$e^{-\epsilon} \leq \frac{P(s_i \mid \mathfrak{M}(\boldsymbol{\mathfrak{D}ata}) = \omega, \theta)}{P(s_j \mid \mathfrak{M}(\boldsymbol{\mathfrak{D}ata}) = \omega, \theta)} \Big/ \frac{P(s_i \mid \theta)}{P(s_j \mid \theta)} \leq e^{\epsilon}$$

This is the odds ratio of s_i to s_j and has the following interpretation: **if an attacker thinks s_i is α times as likely as s_j then after seeing the sanitized output the attacker will believe s_i is at most $e^{\epsilon}\alpha$ times and at least $e^{-\epsilon}\alpha$ times as likely as s_j.** In other words, for small values of ϵ, seeing the sanitized output ω provides nearly no information gain to attackers who are trying to distinguish between whether s_i or s_j is true.

3.2 Example: Privacy with no Assumptions

As a warmup, we use Pufferfish to create a privacy definition with no assumptions (a re-interpretation of no-free-lunch privacy [22]). Let $\mathcal{T}$ be the domain of tuples and let $\mathcal{H} = \{h_1, \ldots, h_N\}$ be the set of all individuals. Define σ_i to be the statement *"the record belonging to individual h_i is in the data"* and define $\sigma_{(i,t)}$ to be the statement *"the record belonging to individual h_i is in the data and has value t"*. Define the set of potential secrets $\mathbb{S}$ and discriminative pairs $\mathbb{S}_{\text{pairs}}$ to be:

$$\mathbb{S} = \{\sigma_1, \ldots, \sigma_N\} \cup \{\sigma_{i,t} \; : \; i = 1, \ldots, N \wedge t \in \mathcal{T}\} \quad (3)$$

$$\mathbb{S}_{\text{pairs}} = \left\{(\sigma_{(i,t_a)}, \sigma_{(i,t_b)}) \; : \; i = 1, \ldots, N \wedge t_a, t_b \in \mathcal{T}\right\} \cup \{(\sigma_i, \neg\sigma_i) \; : \; i = 1, \ldots, N\} \quad (4)$$

with the interpretation that for every individual h_i we want to avoid leaking information about whether or not the record of h_i is in the data (this is specified by the discriminative pair $(\sigma_i, \neg\sigma_i)$), and if an attacker already believes h_i is in the data, we want to avoid leaking information about the value of the corresponding record (this is specified by the discriminative pairs $(\sigma_{(i,t_a)}, \sigma_{(i,t_b)})$ where t_a and t_b range over all possible tuple values).

To get privacy with no assumptions, we must make the set of distributional assumptions $\mathbb{D}$ as large as possible. That is, we set $\mathbb{D}$ to be the collection of all probability distributions over database instances (i.e. records in a database need not be independent), as in Example 3.1.

Theorem 3.1. *Let $\epsilon > 0$, let $\mathbb{S}$ and $\mathbb{S}_{pairs}$ be specified as in Equations 3 and 4 and let $\mathbb{D}$ be the set of all possible distributions over database instances. Then an algorithm $\mathfrak{M}$ satisfies ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$ if and only if for **every** pair of databases D_1 and D_2 and every $\omega \in \text{range}(\mathfrak{M})$,*

$$e^{-\epsilon} P(\mathfrak{M}(D_2) = \omega) \leq P(\mathfrak{M}(D_1) = \omega) \leq e^{\epsilon} P(\mathfrak{M}(D_2) = \omega)$$

Note that the randomness here only depends on $\mathfrak{M}$ because each of the D_1 and D_2 are given databases (and hence not random). From this warmup example, we see that if we make no assumptions then the goal is to prevent an attacker from distinguishing between any possible input datasets; the result is a total lack of utility. Note that this warmup shows a formal equivalence between Pufferfish with no assumptions and a strawman privacy definition called no-free-lunch privacy that was used in [22] as an example of a privacy definition without utility.

4. RELATED WORK

4.1 Relation to Differential Privacy Variants

One can view the Pufferfish framework as a substantial generalization of differential privacy. Thus we explain the differences between Pufferfish and differential privacy [12] and its variants [11, 30, 5, 3, 22, 21, 25, 27, 18, 41, 33].

A related framework, known as adversarial privacy [33] allows domain experts to plug in various data generating distributions. While there is no known equivalence between adversarial privacy and differential privacy, Rastogi et al. [33] have proved an equivalence between a certain instantiation of adversarial privacy and a variant of differential privacy known as ϵ-indistinguishability (the main difference with differential privacy is the requirement that neighboring databases have the same size) [13]. Adversarial privacy also only seeks to protect the presence/absence of a tuple in the dataset. In contrast, Pufferfish is more flexible. We can prove equivalences between instantiations of Pufferfish and differential privacy as well as ϵ-indistinguishability. Pufferfish also allows more fine-grained protection of a tuple (such as preventing inference of continuous attributes to within a certain absolute error) and can handle aggregate secrets as well. As a result, Pufferfish supports a wide variety of applications such as those in Sections 6, 7, 8, and 9.

Zhou et al. [41] present two variants of differential privacy (δ-constrained and ZLW distributional) whose privacy semantics are unclear. We show approximate equivalences to instantiations of the Pufferfish framework in Section 7.3, thereby providing approximate semantics for those definitions as well.

In general, the difference between Pufferfish and related work is the wide variety of applications it can support, such as those discussed in this paper. This flexibility is the result of a different philosophy about how to phrase privacy definitions. Differential privacy uses concepts like neighboring databases (i.e. those that differ on only one tuple) and tries to avoid making any statements about data probabilities. As we show in Sections 6, 7.3, and 9, that approach can obscure what semantic guarantees are provided and the conditions under which they are provided. Pufferfish takes the opposite approach by focusing directly on what an attacker can infer.

As with [11, 3, 41, 33], Pufferfish explicitly uses data-generating probabilities. Furthermore, like adversarial privacy [33], Pufferfish allows more general probability distributions (records do not have to be independent). This allows us to compare Pufferfish to differential privacy and to show that differential privacy is an instantiation of the Pufferfish framework under an assumption of independence between records (Section 6). The difference with those pieces of work is the following.

The concept of neighboring databases arose in differential privacy from the desire to protect individual records; thus two databases are neighbors if they differ on one record. In subsequent variations, the concept of neighbors (or some generalization) plays a key role [11, 30, 5, 22, 21, 25, 27, 41, 18]. Of the remaining work, adversarial privacy [33] only seeks to protect inference about the presence of a tuple, while BLR distributional privacy [3] does not indicate what secrets it protects. One of the distinguishing approaches of the Pufferfish framework is that it does not search for "the right notion of neighbors" (which can obscure what it is we are trying to protect). Instead, it focuses on the actual secrets to be protected $\mathbb{S}$ and how they are to be protected via the discriminative pairs $\mathbb{S}_{\text{pairs}}$.

The combination of distributional assumptions and allowing explicit (and arbitrary) choices of what to protect give Pufferfish its flexibility and allow the applications described in this paper.

4.2 Relationship to Other Work

Other privacy frameworks also exist. A large class, of which k-anonymity [34] is perhaps the most well known, are known as *syntactic methods* because they are mainly concerned with the syntactic form of the outputs $\omega \in \text{range}(\mathfrak{M})$ rather than the probabilities that govern the relationship between inputs to outputs: $P(\mathfrak{M}(D_i) = \omega)$. Because of this, the resulting privacy definitions tend to be less secure and are often subject to new kinds of attacks (see the surveys [6, 16, 1] for more details). It is also often not clear what data assumptions those definitions are making.

The protection of aggregate information gets less attention than the problem of protecting individual records. The protection of aggregates is most relevant to business data where aggregates reflect different kinds of business secrets. Much of the work in this area is called *knowledge hiding* and focuses on hiding association rules, frequent itemsets, and classification rules (e.g., [9, 7, 31, 38, 37, 28, 8, 2, 39, 10, 36, 29]) that are deemed to be sensitive. Work in this area generally follows the syntactic paradigm and it is often unclear what rigorous privacy guarantees can be provided or what data assumptions are needed for ensuring privacy.

5. PUFFERFISH AND PRIVACY AXIOMS

Research in statistical privacy has been moving away from ad-hoc privacy definitions and towards formal and rigorous privacy definitions. The reason is that rigorous privacy definitions offer the promise of ending the endless cycle of discovering a vulnerability in a privacy definition, proposing a fix, finding a vulnerability in the "fixed" version, etc. (see [6] for some examples).

To this end, recent research has started examining the properties that privacy definitions need to have [21, 20]. Modern design guidelines for privacy definitions include 2 fundamental axioms known as *transformation invariance* and *convexity* [20]. While the ideas contained in the axioms have been accepted by the privacy community for a long time, only recently has there been an insistence that privacy definitions actually satisfy them (in fact, many of the vulnerabilities associated with definitions such as k-anonymity are a direct result of not satisfying those axioms [24]).

In this section we show that every privacy definition in the Pufferfish framework satisfies both fundamental axioms, thus ensuring that it satisfies modern design guidelines.

The axioms are:

AXIOM 5.1. (Transformation Invariance [20]). *If an algorithm $\mathfrak{M}$ satisfies a privacy definition and $\mathcal{A}$ is any algorithm such that (1) its domain contains the range of $\mathfrak{M}$ and (2) its random bits (if any) are statistically independent from the random bits (if any) of $\mathfrak{M}$, then the algorithm $\mathcal{A} \circ \mathfrak{M}$, which first runs $\mathfrak{M}$ on the data and then runs $\mathcal{A}$ on the output should also satisfy the same privacy definition.*

The justification for the transformation invariance axiom is that $\mathcal{A}$ is an algorithm whose only input is the output of $\mathfrak{M}$ and so it simulates a data analyst who is performing a statistical analysis using the output of $\mathfrak{M}$; thus it would be strange if a privacy definition implied the output ω of $\mathfrak{M}$ was safe to release, but the results of the statistical analysis on this output ω were not (many existing privacy definitions fail to satisfy this property [20]).

AXIOM 5.2 (CONVEXITY [20]). *If $\mathfrak{M}_1$ and $\mathfrak{M}_2$ satisfy a privacy definition, and $p \in [0, 1]$, then the algorithm $\mathfrak{M}^p$ which runs $\mathfrak{M}_1$ with probability p and $\mathfrak{M}_2$ with probability $1 - p$ should also satisfy the privacy definition.*

The convexity axiom says that a data curator is allowed to choose any algorithm $\mathfrak{M}$ that satisfies the curator's chosen privacy definition and that this choice can be randomized (thus adding further uncertainty into the creation of sanitized data). Again, most proposed existing privacy definitions fail to satisfy this property.

The following theorem confirms that the Pufferfish framework satisfies modern privacy design guidelines.

THEOREM 5.1. *For every $\mathbb{S}$, $\mathbb{S}_{pairs}$, $\mathbb{D}$, and $\epsilon > 0$, the privacy definition ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$ satisfies the axioms of convexity and transformation invariance.*

6. PUFFERFISH ANALYSIS OF DIFFERENTIAL PRIVACY

Differential privacy [12] is a state of the art privacy definition which has been very influential in modern privacy research. It is formally defined as:

DEFINITION 6.1 (DIFFERENTIAL PRIVACY [12]). *Given a privacy parameter $\epsilon > 0$, an algorithm $\mathfrak{M}$ satisfies ϵ-differential privacy if for all $\omega \in \text{range}(\mathfrak{M})$ and all pairs of datasets D_i and D_j that differ on the presence of one tuple (i.e. D_i can be derived from D_j by either adding or deleting exactly one tuple), the following holds:*

$$P(\mathfrak{M}(D_i) = \omega) \leq e^{\epsilon} P(\mathfrak{M}(D_j) = \omega)$$

where the probability only depends on the randomness in $\mathfrak{M}$.

This definition has several interpretations: (1) for small ϵ, changing the value of any tuple is unlikely to change the output of $\mathfrak{M}$; (2) an attacker who knows all but one tuples will not learn much about the remaining tuple.

Because neither the definition of differential privacy nor its interpretations mentioned any data-generating distributions, many believed that it was applicable to any setting and that it made no assumptions about the data. These misconceptions were challenged in recent work [18, 22].

With the Pufferfish framework, we can address these claims in a more precise manner. We can answer questions such as: *for what data evolution scenarios $\mathbb{D}$ is ϵ-differential privacy equal to the privacy definition ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$?*. We will see that the appropriate $\mathbb{D}$ requires independence between records[5], thus formalizing the claim that differential privacy assumes tuple independence.

We will do this by formulating a probabilistic model of record independence to define the data evolution scenarios $\mathbb{D}$. With this choice of $\mathbb{D}$ we show that ϵ-differential privacy is an instantiation of the Pufferfish framework (Theorem 6.1). We also show that if $\mathbb{D}$ contains distributions

[5] i.e. the assumption (depending on the application) of independence between medical records, edges in a social network, queries in a search log, etc.

with correlated records then the resulting Pufferfish instantiation is strictly stronger than ϵ-differential privacy (Theorem 6.2); alternatively, under correlations between records, ϵ-differential privacy is not strong enough to guarantee that the changes in attacker's beliefs (*viz.* the semantic guarantees represented by the odds ratio in Section 3.1) are bounded by a factor of e^ϵ (hence the bounds on an attacker's inference that result from uses of ϵ-differential privacy degrade under correlations).

To proceed, we must first specify the potential secrets $\mathbb{S}$ and discriminative pairs $\mathbb{S}_{\text{pairs}}$. Let $\mathcal{T}$ be the domain of tuples. Let $\mathcal{H} = \{h_1, h_2, \dots, h_N\}$ be the set of all individuals in a population of size N. Define σ_i to be the statement $r_i \in \text{records}(\mathfrak{Data})$ (i.e. *"record r_i belonging to individual h_i is in the data"*) and let $\sigma_{(i,t)}$ be the statement $r_i \in \text{records}(\mathfrak{Data}) \wedge r_i = t$ (i.e. *"record r_i belonging to individual h_i has value t and is in the data"*). Let

$$\mathbb{S} = \left\{\sigma_{(i,t)} \ : \ h_i \in \mathcal{H}, \ t \in \mathcal{T}\right\} \cup \{\neg\sigma_i \ : \ h_i \in \mathcal{H}\} \quad (5)$$

$$\mathbb{S}_{\text{pairs}} = \left\{(\sigma_{(i,t)}, \neg\sigma_i) \ : \ h_i \in \mathcal{H}, t \in \mathcal{T}\right\} \quad (6)$$

Thus for any individual h_i in the population $\mathcal{H}$ and any possible tuple value $t \in \mathcal{T}$, the goal is to prevent an attacker from distinguishing whether the record r_i belonging to h_i is in the data and has value t vs. the data has no record about individual h_i (this is our mathematical translation of the goals in [12]).

For the probabilistic model, suppose each individual h_i is associated with distributions π_i and f_i in the following roles:

- The probability that record r_i belonging to individual h_i is in the data is $P(r_i \in \text{records}(\mathfrak{Data})) \equiv P(\sigma_i) = \pi_i$.
- $P(r_i = t \mid r_i \in \text{records}(\mathfrak{Data})) \equiv P(\sigma_{(i,t)} \mid \sigma_i) = f_i(t)$

With this notation, the model is:

$$\theta \equiv \{\pi_1, \dots, \pi_N, f_1, \dots, f_N\} \quad (7)$$

$$P(\mathfrak{Data} \mid \theta) = \prod_{r_i \in \text{records}(\mathfrak{Data})} f_i(r_i)\pi_i \prod_{r_j \notin \text{records}(\mathfrak{Data})} (1 - \pi_j)$$

In other words, the presence/absence/record-value of each individual is independent of the presence/absence/record-values of other individuals. We set $\mathbb{D}$ to be the set of all possible probability distributions of the form given in Equation 7 (i.e. for all possible choices of the π_i and f_i).

The following theorem says that under this probabilistic model, ϵ-differential privacy becomes an instantiation of the Pufferfish framework.

THEOREM 6.1. *Let $\mathbb{S}$ and $\mathbb{S}_{pairs}$ be defined as in Equations 5 and 6. Let $\mathbb{D}^*$ be the set of all distributions of the form specified in Equation 7. With these choices, ϵ-differential privacy is equivalent to ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D}^*)$.*

The following theorem says that if we have any correlations between records, then some differentially private algorithms leak more information than is allowable (under the odds ratio semantics in Section 3.1), in which case an attacker's posterior beliefs may differ significantly from the prior beliefs depending on the strength of the correlation.

THEOREM 6.2. *Let $\mathbb{S}$ and $\mathbb{S}_{pairs}$ be defined as in Equations 5 and 6. Let $\mathbb{D}^*$ be the set of all distributions of the form specified in Equation 7. If we choose the data evolution scenarios $\mathbb{D}_{other}$ such that $\mathbb{D}_{other} \not\subseteq \mathbb{D}^*$ then ϵ-differential privacy is not equivalent to ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D}_{other})$ (i.e. with the same ϵ-parameter) and hence does not bound the odds-ratio to the interval $[e^{-\epsilon}, e^\epsilon]$.*

Similar results can be shown for ϵ-indistinguishability (defined in [13]).

7. CONTINUOUS ATTRIBUTES AND AGGREGATE SECRETS

One of the difficult problems in privacy-preserving data publishing is protecting the values of continuous variables that are not a priori bounded or which are bounded but can take very large values (such as income). For example, many algorithms for differential privacy do not work in the first case (i.e. no a priori bound) [13] and provide poor utility in the second case.

In this section we use the Pufferfish framework to provide a solution to this problem (Section 7.1). This application shows the importance of specifying discriminative pairs $\mathbb{S}_{\text{pairs}}$. We then show how this solution can be used to protect aggregate secrets (Section 7.2). Finally, we use Pufferfish to provide approximate privacy semantics for δ-constrained α-differential privacy [41] and ZLW distributional privacy [41] (Section 7.3); those two definitions were also designed for continuous attributes but their semantic guarantees and conditions under which those guarantees hold were not clear.

7.1 Protecting Continuous Attributes

As we saw in Section 6, differential privacy is designed to make it difficult to distinguish between the case when an individual's record was included in the data with value t or whether it was not in the data at all. This has to be true whether $t = 1$ or $t = 1,000,000$ (e.g., in the case where the tuple domain $\mathcal{T} = [0, 10^6]$). To account for the possibility that the record of one individual could dominate an aggregate statistic such as $SUM(t)$, an algorithm such as the Laplace mechanism [12] needs to add noise with standard deviation proportional to 10^6 in order to satisfy differential privacy (thus potentially masking out the signal in the data).

If this loss of utility is unacceptable, the data curator may want to relax privacy by stating requirements such as (1) an attacker should not be able to infer any individual's salary to within an absolute error of less than $1,000$, or (2) an attacker should not be able to infer any individual's salary to within a relative error of 10%. Both of these requirements can be handled in the Pufferfish framework.

7.1.1 *Privacy via Absolute Error*

For ease of explanation, suppose that records belonging to individuals $h_1, \dots, h_n$ are known to be in the data and suppose the data curator only collects an individual's income so that the domain of tuples is $\mathbb{R}^+$, the set of nonnegative real numbers (hence the domain is an unbounded set). If we are interested in preventing inference about income that is within absolute error k, then we can proceed as follows. Let $\sigma_{i,[x-k,x+k)}$ be the statement "the income of individual h_i is in the range $[x-k, x+k)$". Define

$$\mathbb{S} = \left\{\sigma_{i,[x-k,x+k)} \ : \ i = 1, \dots, n, \ x \in [0, \infty)\right\} \quad (8)$$

We set discriminative pairs to be neighboring intervals (note that the intervals are half-open so that each discriminative pair consists of mutually exclusive statements). With this setting, we are requiring that the attackers should have difficulty in distinguishing between whether someone's income is between $y-k$ and $y+k$ or a neighboring interval $[y+k, y+3k)$ or $[y-3k, y-k)$ thus ensuring that inference to within $\pm k$ is not possible. Formally,

$$\mathbb{S}_{\text{pairs}} = \left\{ (\sigma_{i,[x-k,x+k)}, \sigma_{i,[x+k,x+3k)}) \;:\; {}^{i=1,\dots,n}_{x\in[0,\infty)} \right\} \quad (9)$$

We can set the evolution scenarios $\mathbb{D}$ to be the set of all probability distributions that assign incomes to individuals $h_1, \dots, h_n$ independently. Thus the model is:

$$\begin{aligned} \theta &\equiv [f_1, \dots, f_n] & (10) \\ P(\mathfrak{D}\boldsymbol{ata} \mid \theta) &= \prod_{r_i \in \text{records}(\mathfrak{D}\boldsymbol{ata})} f_i(r_i) \end{aligned}$$

where the interpretation of the probability (as a probability mass function or density function) depends on whether the f_i are continuous or not. We set $\mathbb{D}$ to be all probability distributions of the form given in Equation 10 (i.e. for all choices of $f_1, \dots, f_n$).

With the resulting instantiation ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D})$, the following lemma shows that we can answer the query "what is the sum of the salaries" as follows. Let $t_1, \dots, t_n$ be the tuple values. Compute $X + \sum_{i=1}^n t_i$ where X is a random variable drawn from the Laplace$(4k/\epsilon)$ distribution with density function $\frac{\epsilon}{8k} e^{-\epsilon|x|/4k}$. Note that when the data set size n is large, the true average salary $\frac{\sum_{i=1}^n t_i}{n}$ and the noisy average (i.e. this noisy sum divided by n) are very close to each other with high probability. In contrast, satisfying differential privacy by adding noise to the sum would require a distribution with infinite variance (i.e. not possible) since there is no upper bound on tuple values; thus the relaxation created using Pufferfish allows more utility while clearly describing the privacy lost (i.e. income is inferable to an absolute error of $\geq k$ but to no smaller range).

LEMMA 7.1. *With $\mathbb{S}$ and $\mathbb{S}_{pairs}$ defined in Equations 8 and 9 let $\mathbb{D}$ be the set of all probability distributions having the form specified in Equation 10. The algorithm $\mathfrak{M}$ which returns $X + \sum_{i=1}^n t_i$ where X has density $\frac{\epsilon}{8k} e^{-\epsilon|x|/4k}$ satisfies ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$.*

7.1.2 *Privacy via Relative Error*

We can extend these ideas to protect against inference to within a prespecified *relative error* as well. One way to approach this problem is to choose $c \in (0, 1)$ and define $\sigma_{i,[cy,y/c)}$ to be the statement that "the income of individual h_i is in the range $[cy, y/c)$". Thus, for example, to express inference to within 10% we set $c = 0.1$. We define the set of potential secrets to be:

$$\mathbb{S} \;=\; \left\{ \sigma_{i,[cy,y/c)} \;:\; i = 1, \dots, n,\; y > 0 \right\}$$

The discriminative pairs are again neighboring intervals. With this setting, we are requiring that the attackers should have difficulty in distinguishing between whether someone's income is in the interval $[cy, y/c)$ – whose center in terms of geometric mean is y – or a neighboring interval $[y/c, y/c^3)$ or $[cy, c^3 y)$, thus limiting the attacker's ability to infer the income to within a factor of c. Formally,

$$\mathbb{S}_{\text{pairs}} = \left\{ (\sigma_{i,[cy,y/c)}, \sigma_{i,[y/c,y/c^3)}) \;:\; i = 1, \dots, n,\; y > 0 \right\}$$

As in the case of absolute error, we can set the evolution scenarios $\mathbb{D}$ to be the set of all data-generating distributions that generate records independently (but not necessarily i.i.d.):

$$\begin{aligned} \theta &\equiv [f_1, \dots, f_n] & (11) \\ P(\mathfrak{D}\boldsymbol{ata} \mid \theta) &= \prod_{r_i \in \text{records}(\mathfrak{D}\boldsymbol{ata})} f_i(r_i) \end{aligned}$$

Now note that $t \in [cy, y/c)$ if and only if $\log t \in [y + \log c, y - \log c)$ and so protecting y for relative error becomes the same as protecting $\log y$ for absolute error $\pm \log c$. Thus this version of relative error is reduced to the case of absolute error, and so we can protect tuples by applying additive noise to the logarithm, or, equivalently by using multiplicative noise.

7.2 Aggregate Secrets

In this section we discuss how Pufferfish can be used to protect aggregate information (as in business data). The discussion is brief because this is a simple extension of the ideas in Sections 7.1.1 and 7.1.2; yet the discussion is necessary because there is little focus in the literature on rigorous and formal privacy guarantees for business data.[6]

In some cases a business may have a large dataset $\mathfrak{D}\boldsymbol{ata} = \{r_1, \dots, r_n\}$ that can be considered to be an i.i.d. sample from some distribution. The business may decide that letting the public learn about this distribution is acceptable. The business may reason as follows: if an attacker had a large dataset generated independently from that distribution, the attacker may learn that distribution and use it to make inferences about the business's own data. For example, the attacker may use the gained knowledge about the true distribution to infer $\sum_{i=1}^n r_i$ up to an additive error of $O(\sqrt{n})$ with high probability (i.e. sampling error). Thus the business may consider it acceptable to create a data release as long as one cannot infer certain sums to within $\pm\sqrt{n}$.

Let μ be a metric over datasets. We define $\sigma_{[\mu,D,\delta]}$ to be the statement that $\mu(\mathfrak{D}\boldsymbol{ata}, D) \leq \delta$ and $\sigma^*_{[\mu,D,\delta]}$ to be the statement that $2\delta \geq \mu(\mathfrak{D}\boldsymbol{ata}, D) > \delta$. The set of potential secrets and discriminative pairs could then be defined as:

$$\begin{aligned} \mathbb{S}_\delta &= \left\{ \sigma_{[\mu,D,\delta]} \;:\; D \in \mathcal{I} \right\} \cup \left\{ \sigma^*_{[\mu,D,\delta]} \;:\; D \in \mathcal{I} \right\} & (12) \\ \mathbb{S}_{\text{pairs}_\delta} &= \left\{ (\sigma_{[\mu,D,\delta]}, \sigma^*_{[\mu,D,\delta]}) \mid D \in \mathcal{I} \right\} & (13) \end{aligned}$$

where δ could be set to $\sqrt{n}$ and the data evolution scenarios can be the set of all distributions over $\mathcal{I}$ in which records are generated i.i.d. or just independently.

Note that one can union the potential secrets in Equations 12 and 8 and union the discriminative pairs in Equations 13 and 9 to protect both aggregate information and secrets about individuals.

7.3 δ-Constrained and Distributional Privacy

Zhou et al. [41] also proposed two privacy definitions that can be used with continuous variables. Those definitions were introduced solely for the study of utility and their precise privacy semantics (i.e. what inferences do they protect against) were not explained nor explored. As with differential privacy, they are phrased in terms of databases that should be indistinguishable. As a result the privacy guarantees and conditions under which they hold are not clear. We show an approximate equivalence between those privacy definitions and instantiations of Pufferfish, so that the Pufferfish framework (approximately) subsumes those definitions

[6]Even nonrigorous approaches are rare for business data.

and gives them clearer privacy semantics. We start with those definitions:

Definition 7.1. (Constrained and ZLW-Distributional Privacy [41]) *Let μ be a metric and let $\delta > 0$ and $\epsilon > 0$ be constants. Let $\mathfrak{M}$ be an algorithm. For all $\omega \in \text{range}(\mathfrak{M})$, if the following constraints hold*

$$e^{-\epsilon}P(\mathfrak{M}(D_2) = \omega) \leq P(\mathfrak{M}(D_1) = \omega) \leq e^{\epsilon}P(\mathfrak{M}(D_2) = \omega)$$

- *whenever D_1 and D_2 differ on the value of 1 tuple and $\mu(D_1, D_2) \leq \delta$ then algorithm $\mathfrak{M}$ satisfies $\underline{\delta\text{-constrained } \epsilon\text{-differential privacy.}}$*
- *If those conditions hold when (1) D_1 and D_2 belong to a prespecified countable subset $S \subset \mathcal{I}$ and (2) $\mu(D_1, D_2) < \delta$ and (3) $D_1 \cap D_2 \neq 0$ then algorithm $\mathfrak{M}$ satisfies $\underline{\text{ZLW } (\epsilon, \delta)\text{-distributional privacy}}$[7].*

First, note that δ-constrained ϵ-differential privacy with metric μ is equal to ZLW (ϵ, δ^*)-distributional privacy with a properly chosen metric μ^* that combines Hamming distance with the metric μ, an appropriate choice of δ^*, and setting $S = \mathcal{I}$. Thus, we can focus only on ZLW distributional privacy. Second, the condition $D_1 \cap D_2 = \emptyset$ is also not necessary since it can also be achieved by proper choice of metric (we therefore drop this condition to increase generality). Third, it is not clear what (if any) privacy semantics are produced by the condition $D_1, D_2 \in S$. Thus we remove this condition (i.e. set $S = \mathcal{I}$) and use Pufferfish to provide approximate semantics to the resulting definition:

Definition 7.2. (Modified ZLW-Privacy). *Let μ be a metric and let $\delta > 0$ and $\epsilon > 0$ be constants. An algorithm $\mathfrak{M}$ satisfies $\underline{(\epsilon, \delta)\text{-modified ZLW privacy}}$ if for every $\omega \in \text{range}(\mathfrak{M})$ and every pair of databases D_1, D_2 such that $\mu(D_1, D_2) \leq \delta$, the following conditions hold:*

$$e^{-\epsilon}P(\mathfrak{M}(D_2) = \omega) \leq P(\mathfrak{M}(D_1) = \omega) \leq e^{\epsilon}P(\mathfrak{M}(D_2) = \omega)$$

The approximate equivalence (subject to a mild condition on μ) to a Pufferfish instantiation is the following:

Theorem 7.1. *Let μ be a metric over database instances such that whenever $\mu(D_1, D_2) \leq \delta$ there exists[8] a $D^* \in \mathcal{I}$ with $\mu(D_1, D^*) \leq \delta$ and $\mu(D_2, D^*) > \delta$. Let $\epsilon > 0$ and $\delta > 0$. Set $\mathbb{S}_\delta$ as in Equation 12 and $\mathbb{S}_{pairs_\delta}$ as in Equation 13. Define $\mathbb{D}$ to be the set of distributions over dataset instances with n records where record values are independent (e.g., all distributions of the form given in Equation 11). If $\mathfrak{M}$ satisfies ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}_\delta, \mathbb{S}_{pairs_\delta}, \mathbb{D})$ then it also satisfies (ϵ, δ)-modified ZLW privacy; conversely, if $\mathfrak{M}$ satisfies (ϵ, δ)-modified ZLW privacy then it satisfies the definition 4ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}_\delta, \mathbb{S}_{pairs_\delta}, \mathbb{D})$ (i.e. up to a four-fold degradation of semantic guarantees in terms of odds-ratio).*

Thus although the precise privacy semantics of the ZLW privacy definitions [41] are unknown, the ideas discussed in Sections 7.1.1, 7.1.2, and 7.2 combined with Theorem 7.1 show that the Pufferfish framework can give the ZLW privacy definitions some approximate semantics in terms of an odds-ratio bound of $4e^{\epsilon}$ when protecting secrets about a database to within absolute error (as defined by the metric μ) of $\pm\delta$.

[7]We use the prefix ZLW to distinguish it from the distributional privacy definition introduced in [3].

[8]This condition is achieved, for example, by the L_1 norm, L_2 norm, etc. as long as δ is less than the radius of $\mathcal{I}$.

8. COMPOSITION

Given a privacy definition, the notion of *composition* [17] refers to the degradation of privacy due to two independent data releases. For example, Alice may choose two algorithms $\mathfrak{M}_1$ and $\mathfrak{M}_2$ (whose random bits are independent of each other), run both on the same data and publish both results. Alternatively, Alice and Bob may have overlapping datasets; they can each run an algorithm $\mathfrak{M}_{Alice}$ and $\mathfrak{M}_{Bob}$ (possibly satisfying different privacy definitions) on their own dataset and output the result. It is known that privacy can degrade in those instances. For example, two independent releases of k-anonymous tables can lead to a privacy breach [17]. Also, a differentially private release combined with a release of deterministic statistics can also lead to a breach [22]. On the other hand, differential privacy composes well with itself: if Alice uses an ϵ_1-differentially private algorithm and Bob uses an ϵ_2-differentially private algorithm, the result (from an attacker's perspective) is the same as if Alice and Bob pooled their data and used an $(\epsilon_1 + \epsilon_2)$-differentially private algorithm.

Properly handling multiple data releases requires a mixture of policy and technological solutions. If Alice is planning multiple data releases, Alice needs to account for information leaked by her prior data releases. If Alice and Bob are planning to release sanitized data, they need to coordinate their efforts if their raw data are correlated. That is, they need to agree on a technological solution that guarantees that the combination of their data releases will not breach privacy.

8.1 Pufferfish View of Composition

Since the Pufferfish framework provides a wide variety of privacy definitions, it allows us to study composition more generally. The key point is that, as before, we need to specify probabilistic assumptions about how datasets are related.

Suppose Alice has a dataset $\mathfrak{Data}_{Alice}$ and Bob has a dataset $\mathfrak{Data}_{Bob}$. Alice announces that she will use a privacy definition $\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D})$ to publish a sanitized version of her data and Bob announces that he will use a privacy definition $\mathfrak{Priv}$ (which need not belong to the Pufferfish framework). Alice would like to examine the consequences to privacy that can occur when they both release sanitized data using their chosen privacy definitions.

In order for Alice to study how her privacy definition composes with possible data releases from Bob, she needs to consider *all* the different plausible relationships between her dataset and Bob's dataset. Thus she also needs to specify a $\underline{\text{set}}$ $\mathcal{C}$ond of conditional probability distributions of Bob's data given her own.[9] Each $\phi \in \mathcal{C}$ond specifies a conditional probability distribution $P(\mathfrak{Data}_{\text{Bob}} = D' \mid \mathfrak{Data}_{\text{Alice}} = D, \phi)$. The distributions $\phi \in \mathcal{C}$ond and $\theta \in \mathbb{D}$ combine to form a joint distribution:

$$\begin{aligned} &P(\mathfrak{Data}_{\text{Bob}} \wedge \mathfrak{Data}_{\text{Alice}} \mid \phi, \theta) \\ &\quad = \quad P(\mathfrak{Data}_{\text{Bob}} \mid \mathfrak{Data}_{\text{Alice}}, \phi) P(\mathfrak{Data}_{\text{Alice}} \mid \theta) \end{aligned}$$

Alice can then reason as follows. For the moment, suppose Bob has already chosen to apply an algorithm $\mathcal{A}$ to his data. Alice can study the effect of releasing the output of $\mathfrak{M}(\mathfrak{Data}_{\text{Alice}})$ by considering the distributions in $\theta \in \mathbb{D}$ and

[9]Thus we decompose the joint distribution of $\mathfrak{Data}_{Alice}$ and $\mathfrak{Data}_{Bob}$ into the marginal distribution of $\mathfrak{Data}_{Alice}$ and the conditional distribution of $\mathfrak{Data}_{Bob}$ given $\mathfrak{Data}_{Alice}$.

$\phi \in \mathcal{C}\text{ond}$. Simulating an adversary's reasoning, for each $(s_i, s_j) \in \mathbb{S}_{\text{pairs}}$, $\omega \in \text{range}(\mathfrak{M})$, $\omega^* \in \text{range}(\mathcal{A})$, she derives:

$$P(\mathcal{A}(\boldsymbol{\mathfrak{Data}}_{\text{Bob}}) = \omega^* \wedge \mathfrak{M}(\boldsymbol{\mathfrak{Data}}_{\text{Alice}}) = \omega \mid s_i, \phi, \theta)$$

$$= \int \begin{pmatrix} P(\mathfrak{M}(D){=}\omega)P(\boldsymbol{\mathfrak{Data}}_{Alice}{=}D) \mid s_i,\theta) \\ \times E[P(\mathcal{A}(\boldsymbol{\mathfrak{Data}}_{\text{Bob}}){=}\omega^*) \mid \boldsymbol{\mathfrak{Data}}_{Alice}{=}D,\phi] \end{pmatrix} dD \qquad (14)$$

where, $E\left[P(\mathcal{A}(\boldsymbol{\mathfrak{Data}}_{Bob}) = \omega^*) \mid \boldsymbol{\mathfrak{Data}}_{\text{Alice}} = D, \phi\right]$

$$= \int P\big(\mathcal{A}(D') = \omega^*\big) P\left(\boldsymbol{\mathfrak{Data}}_{\text{Bob}} = D' \middle| \boldsymbol{\mathfrak{Data}}_{\text{Alice}} = D, \phi\right) dD'$$

is the averaged conditional probability (using distribution $\phi \in \mathcal{C}\text{ond}$) of seeing Bob's sanitized output ω^* given that Alice's dataset is D.

The significance of Equation 14 is that an attacker who uses the distributions $\theta \in \mathbb{D}$ and $\phi \in \mathcal{C}\text{ond}$ to reason about the joint data release would reason in the exact same way in an alternate universe in which Bob releases nothing and Alice releases information about her dataset using an algorithm $\mathfrak{M}^{\star}_{\phi,\mathcal{A},\mathfrak{M}}$ with $\text{range}(\mathfrak{M}) \times \text{range}(\mathcal{A})$ and which outputs the pair (ω, ω^*) with probability $P[\mathfrak{M}^{\star}_{\phi,\mathcal{A},\mathfrak{M}}(D) = (\omega, \omega^*)] = P(\mathfrak{M}(D) = \omega) E\big[P(\mathcal{A}(\boldsymbol{\mathfrak{Data}}_{Bob}) = \omega^*) \big| \boldsymbol{\mathfrak{Data}}_{\text{Alice}} = D, \phi\big]$. Thus to study the privacy properties of this joint data release, Alice only needs to study the algorithms of the form $\mathfrak{M}^{\star}_{\phi,\mathcal{A},\mathfrak{M}}$ (for all choices of $\phi \in \mathcal{C}\text{ond}$, all $\mathfrak{M}$ satisfying her privacy definition, and all $\mathcal{A}$ satisfying Bob's privacy definition). In particular, she can ask when $\mathfrak{M}^{\star}_{\phi,\mathcal{A},\mathfrak{M}}$ (for all choices of $\phi, \mathfrak{M}, \mathcal{A}$) satisfies ϵ'-$\mathcal{P}\text{uffer}\mathcal{F}\text{ish}(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D})$ (i.e. her privacy definition with a different privacy parameter ϵ').

8.2 Self-composition

In this section, we study a special case of the discussion in the previous section. We call this special case *self-composition*. This is a helpful property for privacy definitions to have since it is useful in the design of algorithms. In self-composition, Alice plans multiple independent releases of her own data (i.e. the Alice and Bob from the previous section are the same person, and $\mathcal{C}\text{ond}$ consists of the trivial conditional probability where $P(\boldsymbol{\mathfrak{Data}}_{Bob} = D \mid \boldsymbol{\mathfrak{Data}}_{Alice} = D) = 1$).

Thus, Alice has a dataset $\boldsymbol{\mathfrak{Data}}$, announces a privacy definition ϵ-$\mathcal{P}\text{uffer}\mathcal{F}\text{ish}(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D})$ and chooses two algorithms $\mathfrak{M}_1$ and $\mathfrak{M}_2$ (with independent sources of randomness) that satisfy this definition. Alice computes $\omega_1 = \mathfrak{M}_1(\boldsymbol{\mathfrak{Data}})$ and $\omega_2 = \mathfrak{M}_2(\boldsymbol{\mathfrak{Data}})$ and releases the sanitized output (ω_1, ω_2) to the public.

From the previous section, we see that this is the same as if Alice had used an algorithm $\mathfrak{M}^*_{\mathfrak{M}_1,\mathfrak{M}_2}$ whose range equals $\text{range}(\mathfrak{M}_1) \times \text{range}(\mathfrak{M}_2)$ and with the probabilistic behavior $P\big[\mathfrak{M}^*_{\mathfrak{M}_1,\mathfrak{M}_2}(D) = (\omega_1, \omega_2)\big] = P(\mathfrak{M}_1(D) = \omega_1)P(\mathfrak{M}_2(D) = \omega_2)$. Ideally, $\mathfrak{M}^*_{\mathfrak{M}_1,\mathfrak{M}_2}$ still satisfies Alice's chosen instantiation of the Pufferfish framework with some privacy parameter ϵ'. This brings up the notion of *linear self-composition*:

Definition 8.1. (Linear Self-composition). *We say that* $\mathcal{P}\text{uffer}\mathcal{F}\text{ish}(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$ *self-composes linearly if* $\forall \epsilon_1, \epsilon_2 > 0$, *all* $\mathfrak{M}_1$ *satisfying* ϵ_1-$\mathcal{P}\text{uffer}\mathcal{F}\text{ish}(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$ *and all* $\mathfrak{M}_2$ *satisfying* ϵ_2-$\mathcal{P}\text{uffer}\mathcal{F}\text{ish}(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$*, the algorithm* $\mathfrak{M}^*_{\mathfrak{M}_1,\mathfrak{M}_2}$ *satisfies* $(\epsilon_1 + \epsilon_2)$-$\mathcal{P}\text{uffer}\mathcal{F}\text{ish}(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$*, where* $\mathfrak{M}^*_{\mathfrak{M}_1,\mathfrak{M}_2}$ *is the algorithm with* $\text{range}(\mathfrak{M}^*_{\mathfrak{M}_1,\mathfrak{M}_2}) = \text{range}(\mathfrak{M}_1) \times \text{range}(\mathfrak{M}_2)$ *such that for all* $D \in \mathcal{I}$, $P\big[\mathfrak{M}^*_{\mathfrak{M}_1,\mathfrak{M}_2}(D) = (\omega_1, \omega_2)\big] = P[\mathfrak{M}_1(D) = \omega_1]P[\mathfrak{M}_2(D) = \omega_2]$

Linear self-composition is useful for algorithm design because it allows Alice to split a complicated algorithm $\mathfrak{M}$ into a collection of simpler algorithms $\mathfrak{M}_1, \ldots, \mathfrak{M}_k$ and allocate her overall privacy budget ϵ among them [26].

8.2.1 Sufficient conditions for self-composition

In general, not all instantiations of the Pufferfish framework will self-compose linearly. Furthermore, it is not always easy to tell if a particular instantiation will self-compose linearly. However, we provide an important class of sufficient conditions called *universally composable evolution scenarios.*

When domain experts create instantiations of the Pufferfish framework, they add more and more probability distributions θ into the evolution scenarios $\mathbb{D}$ to create a reasonable (yet conservative) set of data-generating distributions. Each evolution scenario θ adds an additional constraint that an algorithm $\mathfrak{M}$ must satisfy.

If we have a privacy definition $\mathcal{P}\text{uffer}\mathcal{F}\text{ish}(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D})$ that self-composes linearly, it can happen that adding more evolution scenarios (i.e. replacing $\mathbb{D}$ with a strict superset $\mathbb{D}'$) will break the composition property. However, there are some θ that we can always add without worrying about breaking composition. We refer to these special θ as *universally composable evolution scenarios.*

Definition 8.2. (Universally composable evolution scenarios). *Given* $\mathbb{S}$ *and* $\mathbb{S}_{pairs}$*, we say that* θ *is a* universally composable evolution scenario for $\mathbb{S}_{\text{pairs}}$ *if the privacy definition* $\mathcal{P}\text{uffer}\mathcal{F}\text{ish}(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D} \cup \{\theta\})$ *self-composes linearly whenever* $\mathcal{P}\text{uffer}\mathcal{F}\text{ish}(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$ *self-composes linearly.*

Universally composable evolution scenarios have a very special form.

Theorem 8.1. *Given* $\mathbb{S}$ *and* $\mathbb{S}_{pairs}$*, the probability distribution* θ *is a universally composable evolution scenario for* $\mathbb{S}_{pairs}$ *if and only if for all* $(s_i, s_j) \in \mathbb{S}_{pairs}$ *having* $P(s_i \mid \theta) \neq 0$ *and* $P(s_j \mid \theta) \neq 0$ *there exist datasets* $D_i, D_j \in \mathcal{I}$ *such that* $P(\boldsymbol{\mathfrak{Data}} = D_i \mid s_i, \theta) = 1$ *and* $P(\boldsymbol{\mathfrak{Data}} = D_j \mid s_j, \theta) = 1$

The interesting aspect of Theorem 8.1 is that when the set of evolution scenarios $\mathbb{D}$ consists solely of universally composable evolution scenarios, then the resulting instantiation of the Pufferfish framework is a neighbor-based definition similar to differential privacy.

That is, let $\theta \in \mathbb{D}$ be a universally composable evolution scenario and let $(s_i, s_j) \in \mathbb{S}$ be a discriminative pair with nonzero probability under θ and $D_i, D_j \in \mathcal{I}$ be the datasets associated with θ by Theorem 8.1. Then D_i and D_j can be considered "neighbors" and the Pufferfish constraints (Equations 1 and 2 in Definition 3.1) become:

$$\begin{aligned} P(\mathfrak{M}(D_i) = \omega) &\leq e^{\epsilon} P(\mathfrak{M}(D_j) = \omega) \\ P(\mathfrak{M}(D_j) = \omega) &\leq e^{\epsilon} P(\mathfrak{M}(D_i) = \omega) \end{aligned}$$

with randomness only depending on $\mathfrak{M}$.

In the case of differential privacy, those universally composable evolution scenarios θ are those for which there exist databases D_1 and D_2 that differ only on one tuple and have $P(\boldsymbol{\mathfrak{Data}} = D_1 \mid \theta) + P(\boldsymbol{\mathfrak{Data}} = D_2 \mid \theta) = 1$. Note that Theorem 6.1 says that we can further increase $\mathbb{D}$ to include all of the other distributions that generate records independently without change to the privacy guarantees, and Theorem 6.2 says that differentially private algorithms may leak too much information if we include any other distributions (i.e. those with correlated records).

9. DIFFERENTIAL PRIVACY WITH DETERMINISTIC CONSTRAINTS

It was shown in [22] that differential privacy does not compose well with deterministic data constraints such as those caused by previous deterministic releases of information. That is, when a data curator provides exact query answers about the data and subsequently publishes additional information using ϵ-differential privacy, the combined data releases can leak much more information than each of the 2 releases in isolation. Constraints caused by deterministic releases of information are often the result of legal or contractual obligations (e.g., the U.S. Decennial Census).

Prior work [22] proposed a modification of differential privacy, called *induced neighbors privacy* [22] to account for prior deterministic data releases. As with many variants of differential privacy, it was a "neighbors-based" based definition that tried to make certain pairs of databases (i.e. neighbors) indistinguishable from each other. As we have shown in Sections 6 and 7, this can obscure privacy guarantees and the conditions under which the guarantees hold. Viewed through the Pufferfish lens, induced neighbors privacy does not properly bound the attacker's odds ratio (the main reason being that neighbor-based privacy definitions often cannot explicitly mention what secrets to protect).

In this section we extend the discussion of composition from Section 8 to show how to use Pufferfish to modify differential privacy in a way that takes into account arbitrary deterministic constraints (not just those caused by prior deterministic releases of data). The result is a privacy definition with precise semantic guarantees and clearly specified assumptions under which they hold. We also show some conditions under which induced neighbors privacy [22] is actually equivalent to an instantiation of the Pufferfish framework, thus providing induced neighbors privacy with precise semantic guarantees in those situations.

9.1 Preliminaries

Several types of constraints are common (some of them result from deterministic releases of data):

- ***Counts***: The number of tuples in the dataset or the number of AIDS patients with age less than 25 are pieces of knowledge that impose count constraints. These constraints are often called *marginal constraints*, and can be represented as a constraint like $\sum_{r\in\text{records}(\mathfrak{Data})} g(r) = C$, where g is a function from the tuple domain $\mathcal{T}$ to $\{0,1\}$, and C is an integer. For instance, to encode a constraint about the number of AIDS patients with age less than 25, one can choose g such that $g(t) = 1$ only when t is a tuple with AIDS and age less than 25, and $g(t) = 0$ otherwise.
- ***Univariate Histograms***: These are a special kind of count constraints $\sum_{r\in\text{records}(\mathfrak{Data})} g_i(r) = C_i$ (for $i = 1,\dots,k$) where the g_i have disjoint supports (i.e. if for some $t \in \mathcal{T}$ we have $g_i(t) = 1$ then $g_j(t) = 0$ for all other j). Such constraints capture a variety of statistics that might be known about a database, including the total number of rows, number of tuples satisfying a set of mutually exclusive properties, as well as the results of bucketization algorithms that are common in the k-anonymization and ℓ-diversity literature.
- ***General Deterministic Constraints***: In general, deterministic constraints eliminate some of the databases from the domain of database instances $\mathcal{I}$. Such a general constraint $\mathcal{Q}$ can be formally described as a function $\mathcal{Q} : \mathcal{I} \to \{0,1\}$ such that $\mathcal{Q}(D) = 0$ means D is not possible (i.e. does not satisfy the constraint) and $\mathcal{Q}(D) = 1$ means D is a possible.

Example 9.1. *Let us give an example of a general constraint that will help illustrate the benefits of Pufferfish and distinguish it from neighbor-based privacy definitions. Suppose there are n students with ID numbers ranging from 1 to n. They are scheduled take an oral exam in the order determined by their id numbers. The dataset $\mathfrak{Data}$ tracks whether or not a student has taken the exam. Initially $\mathfrak{Data}$ consists of n records with value 0. After student i takes the exam, the i^{th} record is set to 1. At any point in time, the database $\mathfrak{Data}$ can be in only one of the $n+1$ states defined by the constraint $\mathcal{Q}$: $\exists k \forall i \le k,\ r_i = 1 \bigwedge \forall i > k,\ r_i = 0$, as shown below.*

D_0	D_1	D_2	...	D_{n-2}	D_{n-1}	D_n
0	1	1	...	1	1	1
0	0	1	...	1	1	1
...	...	...	...	...	...	...
0	0	0	...	0	1	1
0	0	0	...	0	0	1

Suppose at some time point, the data curator wants to release the current number of students who have taken the exam without revealing who has or has not taken the test. The Laplace mechanism, which satisfies ϵ-differential privacy, would add noise with density $f(x) = \frac{1}{2\epsilon}e^{-|x|/\epsilon}$ to the current number of students who have taken the exam. When n, the number of total students, is large, this strategy leaks too much information. For example, if the noisy answer is close to 0, the true value was probably not n and so the n^{th} student probably has not yet taken the exam. As we shall see, it is not possible to release meaningful information in this situation, but differential privacy does not warn us about it (neither will induced-neighbors privacy, Definition 9.2).

Induced neighbors privacy [22] uses the following definitions:

Definition 9.1. (Move [22]). *Given a database D, a* move *m is a process that adds or deletes a tuple from D, resulting in a database $m(D)$.*

Definition 9.2. (Induced Neighbors $\mathcal{N}_{\mathcal{Q}}$ [22]). *Given a general constraint $\mathcal{Q}$, let $\mathcal{I}_{\mathcal{Q}}$ be the set of databases satisfying those constraints. Let D_a and D_b be two databases. Let n_{ab} be the smallest number of moves necessary to transform D_a into D_b and let $\{m_1,\dots,m_{n_{ab}}\}$ be the set of those moves. We say that D_a and D_b are* neighbors induced by *$\mathcal{Q}$, denoted as $\mathcal{N}_{\mathcal{Q}}(D_a, D_b)$ =true, if the following holds.*

- *$D_a \in \mathcal{I}_{\mathcal{Q}}$ and $D_b \in \mathcal{I}_{\mathcal{Q}}$.*
- *No subset of $\{m_1,\dots,m_{n_{ab}}\}$ can transform D_a into some $D_c \in \mathcal{I}_{\mathcal{Q}}$.*

Definition 9.3. (Induced Neighbors Privacy). *An algorithm $\mathfrak{M}$ satisfies induced neighbor privacy with constraint $\mathcal{Q}$, if for each output $\omega \in \text{range}(\mathfrak{M})$ and for every pair D_1, D_2 of neighbors induced by Q, the following holds:*

$$P(\mathfrak{M}(D_1) = \omega) \le e^{\epsilon} P(\mathfrak{M}(D_2) = \omega)$$

It is easy to see that the Laplace mechanism from Example 9.1 also satisfies induced neighbor privacy for this particular scenario since all induced neighbors are pairs (D_i, D_{i+1}). We compute the amount of information leakage (Example 9.2) after considering the Pufferfish view.

9.2 Pufferfish with Deterministic Constraints

Following the notation from Section 6 (also see Table 1 containing our notation), define:

$$\mathbb{S} = \{\sigma_{(i,t))} \ : \ h_i \in \mathcal{H}, \ t \in \mathcal{T}\} \cup \{\neg\sigma_i \ : \ h_i \in \mathcal{H}\} \quad (15)$$

$$\begin{aligned}\mathbb{S}_{\text{pairs}} &= \{(\sigma_{(i,t)}, \neg\sigma_i) \ : \ h_i \in \mathcal{H}, t \in \mathcal{T}\} \quad (16)\\ &\cup \{(\sigma_{(i,t)}, (\sigma_{t,t'})) \ : \ h_i \in \mathcal{H}, t, t' \in \mathcal{T}\}\end{aligned}$$

Thus, the goal is to prevent an attacker from (a) learning whether the record of individual h_i is in the data, and (b) distinguishing between two possible values of r_i (in case h_i is known to be in the data). The data evolution scenarios $\mathbb{D}^*_{\mathcal{Q}}$ is the set of all probability distributions with the following form (i.e., they generate records independently conditioned on $\mathcal{Q}$):

$$\begin{aligned}\theta &\equiv \{\pi_1, \ldots, \pi_N, f_1, \ldots, f_N, \mathcal{Q}\} \quad (17)\\ P(\mathfrak{Data} \mid \theta) &= 0, \text{ if } \mathcal{Q}(\mathfrak{Data}) = 0\\ &= \frac{1}{Z_{\mathcal{Q}}} \prod_{r_i \in \text{records}(\mathfrak{Data})} f_i(r_i)\pi_i \prod_{r_j \notin \text{records}(\mathfrak{Data})} (1-\pi_j), \text{ otherwise}\end{aligned}$$

where the normalization constant $Z_{\mathcal{Q}} = P(\mathcal{Q}(\mathfrak{Data}) = 1)$.

We show that ϵ-induced neighbors privacy is a necessary condition for guaranteeing ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D}^*_{\mathcal{Q}})$, for any general constraint $\mathcal{Q}$.

THEOREM 9.1. (Necessary Condition). *Given a general constraint $\mathcal{Q}$, if $\mathfrak{M}$ satisfies ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D}^*_{\mathcal{Q}})$ then $\mathfrak{M}$ satisfies ϵ-induced neighbors privacy with respect to $\mathcal{Q}$.*

However, the next example shows ϵ-induced neighbors privacy is not sufficient, hence does guarantee an attacker's odds ratio is bounded within $[e^{-\epsilon}, e^{\epsilon}]$.

EXAMPLE 9.2. *Continuing Example 9.1, we show that the Laplace mechanism, which satisfies both ϵ-differential privacy and ϵ-induced neighbors privacy, does not satisfy ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D}^*_{\mathcal{Q}})$.*

Consider a θ of the form given in Equation 17 such that for all i, $f_i(0) = f_i(1) = 0.5$, $\pi_i = 1$. Thus all the allowable datasets $D_0, \ldots, D_n$ from Example 9.1 are equally likely under θ. Consider the discriminative pair $(\sigma_{(1,0)}, \sigma_{(1,1)})$ and note that if record 1 has value 0 then, according to our constraints, so do all records r_ℓ for $\ell > 1$, so D_0 is the only dataset for which $\sigma_{(1,0)}$ can be true.

$$\begin{aligned}&P(\mathfrak{M}(\mathfrak{Data}) = n \mid \sigma_{(1,1)}, \theta)\\ &= \sum_{j=1}^{n} \frac{P(\mathfrak{M}(D_j) = n)}{n} = \sum_{j=1}^{n} \frac{\epsilon}{2n} e^{-\epsilon(n-j)}\\ &> e^{\epsilon} \cdot \frac{\epsilon}{2} e^{-\epsilon n} = e^{\epsilon} \cdot P(\mathfrak{M}(\mathfrak{Data}) = n \mid \sigma_{(1,0)}, \theta)\end{aligned}$$

Therefore satisfying ϵ-differential privacy or induced neighbors privacy in this situation does not bound an attacker's odds-ratio to the range $[e^{-\epsilon}, e^{\epsilon}]$.

In this situation, ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D}^*_{\mathcal{Q}})$ requires

$$\forall D_i, D_j, \ P(\mathfrak{M}(D_i) = \omega) \ \leq \ e^{\epsilon} \cdot P(\mathfrak{M}(D_j) = \omega) \quad (18)$$

The condition is clearly sufficient. Necessity can be shown by considering the output $\omega = n$, and different priors θ_i that assign $f_j(1) = 1 - \delta$ for $j \leq i$, and $f_j(1) = \delta$ otherwise, where δ tends to zero. These priors capture different adversaries who believe strongly (with high prior probability) that i students have taken the exam.

Pufferfish tells us (via Equation 18) that we cannot release meaningful data in this situation because it reduces to the condition that attackers should not be able to distinguish between any pair of valid datasets. However, we next show that it is possible to release meaningful data for a broad class of typical constraints.

9.3 Pufferfish with Univariate Histograms

Univariate histograms, as defined in Section 9.1, form an important subclass of constraints. For instance, the Census Bureau is legally obliged to publish the exact number of people living in each state [4]. A search engine is contractually bound to report to an advertiser the number of users who have clicked on ads using mutually exclusive predefined ranges (e.g., $100 - 200$ clicks). Another interesting use case occurs when there has been a prior release of data using a mechanism $\mathfrak{M}_{uni}$ based on statistical disclosure limitation techniques like partitioning and microaggregation [1], or bucketization algorithms based on syntactic privacy notions like k-anonymity (with say $k = 10,000$), ℓ-diversity, etc. [6, 23, 40]. The output of $\mathfrak{M}_{uni}$ in all the above cases is a univariate histogram. In all these cases, we can provide additional releases of information by using Pufferfish to limit any further inference an attacker could make.

In fact, for those cases, ϵ-induced neighbor privacy becomes an instantiation of the Pufferfish framework (Theorems 9.1 and 9.2).

THEOREM 9.2. (SUFFICIENT CONDITION FOR UNIVARIATE HISTOGRAMS). *Given a univariate histogram constraint $\mathcal{Q}_{uni}$: $\{\sum_{t \in \mathfrak{Data}} g_i(t) = C\}$, define $\mathbb{S}$ and $\mathbb{S}_{pairs}$ as in Equations 15 and 16 and let $\mathbb{D}^*_{\mathcal{Q}}$ be the set of all distributions with form specified in Equation 17. Then $\mathfrak{M}$ satisfies ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D}^*_{\mathcal{Q}_{uni}})$ if $\mathfrak{M}$ satisfies ϵ-induced neighbors privacy with respect to $\mathcal{Q}_{uni}$.*

Thus the algorithms proposed in [22] can be used in this case to bound an attacker's inference.

An important question that was left open in [22] is whether induced neighbor privacy is linear self composable. Theorems 9.1 and 9.2 allow us to answer this question. Since ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D}^*_{\mathcal{Q}_{uni}})$ and induced neighbor privacy (for univariate histograms) are equivalent definitions, it is easy to see that the former can be written solely in terms of universally composable evolution scenarios, proving that ϵ-$\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D}^*_{\mathcal{Q}_{uni}})$ composes with itself linearly. This means that, for a database with a prior univariate histogram release $\mathcal{Q}_{uni}$, and further releases using $\mathfrak{M}_1$ and $\mathfrak{M}_2$ that satisfy $\mathcal{P}$uffer$\mathcal{F}$ish$(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D}^*_{\mathcal{Q}_{uni}})$ with parameters ϵ_1 and ϵ_2, respectively, the combined mechanism $\mathfrak{M}_{\mathfrak{M}_1,\mathfrak{M}_2}$ guarantees $(\epsilon_1 + \epsilon_2)$-$\mathcal{P}$uffer$\mathcal{F}ish(\mathbb{S}, \mathbb{S}_{\text{pairs}}, \mathbb{D}^*_{\mathcal{Q}_{uni}})$.

10. CONCLUSIONS

We presented the Pufferfish framework, a new and general framework that allows application domain experts to develop rigorous privacy definitions for their data sharing needs. The framework allows the domain experts to customize privacy to the specific set of secrets and data evolution scenarios that are typical in that domain. We used our general framework to prove the statement that differential privacy assumed independence between records, and define and study notions of composition in a broader context than before. We also applied the framework to derive rigorous definitions for handling unbounded continuous attributes, protecting aggregate information, and prior deterministic data releases.

11. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1054389 and a gift from Yahoo!

12. REFERENCES

[1] N. Adam and J. Worthmann. Security-control methods for statistical databases. *ACM Computing Surveys*, 21(4):515–556, 1989.

[2] C. C. Aggarwal, J. Pei, and B. Zhang. On privacy preservation against adversarial data mining. In *KDD*, 2006.

[3] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[4] P. J. Cantwell, H. Hogan, and K. M. Styles. The use of statistical methods in the u.s. census: Utah v. evans. *The American Statistician*, 58(3):203–212, 2004.

[5] K. Chaudhuri and N. Mishra. When random sampling preserves privacy. In *CRYPTO*, 2006.

[6] B.-C. Chen, D. Kifer, K. LeFevre, and A. Machanavajjhala. Privacy-preserving data publishing. *Foundations and Trends in Databases*, 2(1-2):1–167, 2009.

[7] C. Clifton. Using sample size to limit exposure to data mining. *Journal of Computer Security*, 8(4), 2000.

[8] C. Clifton, M. Kantarcioglu, and J. Vaidya. Defining privacy for data mining. In *Proc. of the NSF Workshop on Next Generation Data Mining*, 2002.

[9] C. Clifton and D. Marks. Security and privacy implications of data mining. In *Proceedings of the ACM SIGMOD Workshop on Data Mining and Knowledge Discovery*, 1996.

[10] A. Delis, V. S. Verykios, and A. A. Tsitsonis. A data perturbation approach to sensitive classification rule hiding. In *SAC*, 2010.

[11] Y. Duan. Privacy without noise. In *CIKM*, 2009.

[12] C. Dwork. Differential privacy. In *ICALP*, 2006.

[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[14] C. Dwork and M. Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *JPC*, 2(1), 2010.

[15] S. E. Fienberg. Confidentiality and disclosure limitation methodology: Challenges for national statistics and statistical research. Technical Report 668, CMU, 1997.

[16] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Computing Surveys*, 42(4), 2010.

[17] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, 2008.

[18] J. Gehrke, E. Lui, and R. Pass. Towards privacy for social networks: A zero-knowledge based definition of privacy. In *TCC*, 2011.

[19] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8), 08 2008.

[20] D. Kifer and B.-R. Lin. An axiomatic view of statistical privacy and utility. To appear in Journal of Privacy and Confidentiality.

[21] D. Kifer and B.-R. Lin. Towards an axiomatization of statistical privacy and utility. In *PODS*, 2010.

[22] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *SIGMOD*, 2011.

[23] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, 2006.

[24] B.-R. Lin and D. Kifer. A framework for extracting semantic guarantees from privacy definitions. Technical report, Penn State University, 2012.

[25] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: From theory to practice on the map. In *ICDE*, 2008.

[26] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30, 2009.

[27] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. Computational differential privacy. In *CRYPTO*, 2009.

[28] G. V. Moustakides and V. S. Verykios. A maxmin approach for hiding frequent itemsets. *Data Knowl. Eng.*, 65(1), 2008.

[29] J. Natwichai, X. Li, and M. E. Orlowska. A reconstruction-based algorithm for classification rules hiding. In *ADC*, 2006.

[30] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.

[31] S. R. M. Oliveira and O. R. Zaiane. Algorithms for balancing privacy and knowledge discovery in association rule mining. In *International Database Engineering and Applications Symposium*, 2003.

[32] PREDICT (protected repository for the defense of infrastructure against cyber threats). http://www.cyber.st.dhs.gov/predict/.

[33] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: Output perturbation for queries with joins. In *PODS*, pages 107–116, 2009.

[34] P. Samarati. Protecting respondents' identities in microdata release. *TKDE*, 13(6), 2001.

[35] S. Sankararaman, G. Obozinski, M. I. Jordan, and E. Halperin. Genomic privacy and limits of individual detection in a pool. *Nature genetics*, 41(9):965–967, September 2009.

[36] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Rec.*, 33(1), 2004.

[37] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE Trans. Knowl. Data Eng.*, 16(4), 2004.

[38] E. T. Wang and G. Lee. An efficient sanitization algorithm for balancing information privacy and knowledge discovery in association patterns mining. *Data & Knowledge Engineering*, 65(3), 2008.

[39] K. Wang, B. Fung, and P. Yu. Template-based privacy preservation in classification problems. In *ICDM*, 2005.

[40] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *VLDB*, 2006.

[41] S. Zhou, K. Ligett, and L. Wasserman. Differential privacy with compression. In *ISIT*, 2009.