Verifying Pufferfish Privacy in Hidden Markov Models

Depeng Liu[1,2(✉)], Bow-Yaw Wang[3], and Lijun Zhang[1,2,4]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
{liudp,zhanglj}@ios.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Institute of Information Science, Academia Sinica, Taipei, Taiwan
bywang@iis.sinica.edu.tw
[4] Institute of Intelligent Software, Guangzhou, China

Abstract. Pufferfish is a Bayesian privacy framework for designing and analyzing privacy mechanisms. It refines differential privacy, the current gold standard in data privacy, by allowing explicit prior knowledge in privacy analysis. In practice, privacy mechanisms often need be modified or adjusted to specific applications. Their privacy risks have to be re-evaluated for different circumstances. Privacy proofs can thus be complicated and prone to errors. Such tedious tasks are burdensome to average data curators. In this paper, we propose an automatic verification technique for Pufferfish privacy. We use hidden Markov models to specify and analyze discrete mechanisms in Pufferfish privacy. We show that the Pufferfish verification problem in hidden Markov models is NP-hard. Using Satisfiability Modulo Theories solvers, we propose an algorithm to verify privacy requirements. We implement our algorithm in a prototypical tool called FAIER, and analyze several classic privacy mechanisms in Pufferfish privacy. Surprisingly, our analysis show that naïve discretization of well-established privacy mechanisms often fails, witnessed by counterexamples generated by FAIER. In discrete *Above Threshold*, we show that it results in absolutely no privacy. Finally, we compare our approach with state-of-the-art tools for differential privacy, and show that our verification technique can be efficiently combined with these tools for the purpose of certifying counterexamples and finding a more precise lower bound for the privacy budget ϵ.

1 Introduction

Differential privacy is a framework for designing and analyzing privacy measures [16, 17]. In the framework, data publishing mechanisms are formalized as randomized algorithms. On any input data set, such mechanisms return randomized answers to queries. In order to preserve privacy, differential privacy aims to ensure that similar output distributions are yielded on similar input data sets. Differential privacy moreover allows data curators to evaluate privacy and utility quantitatively. The framework has attracted lots of attention from academia and industry such as Microsoft [13] and Apple [2].

Pufferfish is a more recent privacy framework which refines differential privacy [23]. In differential privacy, there is no explicit correlation among entries in data

B. Finkbeiner and T. Wies (Eds.): VMCAI 2022, LNCS 13182, pp. 174–196, 2022.
https://doi.org/10.1007/978-3-030-94583-1_9

sets during privacy analysis. The no free lunch theorem [22] in data privacy shows that prior knowledge about data sets is crucial to privacy analysis. The Pufferfish privacy framework hence allows data curators to analyze privacy with prior knowledge about data sets. Under the Bayesian privacy framework, it is shown that differential privacy preserves the same level of privacy if there is no correlation among entries in data sets.

For differential and Pufferfish privacy, data publishing mechanisms are analyzed – often on paper– with sophisticated mathematical tools. The complexity of the problem is high [19], and moreover, it is well-known that such proofs are very subtle and error-prone. For instance, several published variations of differentially private mechanisms are shown to violate privacy [11,26]. In order to minimize proof errors and misinterpretation, the formal method community has also started to develop techniques for checking differentially private mechanisms, such as verification techniques based on approximate couplings [1,5–8,18], randomness alignments [32–34], model checking [24] as well as those with well-defined programming semantics [3,27] and techniques based on testing and searching [9,10,14,35].

Reality nevertheless can be more complicated than mathematical proofs. Existing privacy mechanisms hardly fit their data publishing requirements perfectly. These algorithms may be implemented differently when used in practice. Majority of differentially private mechanisms utilize continuous perturbations by applying the Laplace mechanism. Computing devices however only approximate continuous noises through floating-point computation, which is discrete in nature. Care must be taken lest privacy should be lost during such finite approximations [28]. Moreover, adding continuous noises may yield uninterpretable outputs for categorical or discrete numerical data. Discrete noises are hence necessary for such data. A challenging task for data curators is to guarantee that the implementation (discrete in nature) meets the specification (often continuous distributions are used). It is often time consuming – if not impossible, to carry out privacy analysis for each modification. Automated verification and testing techniques are in this case a promising methodology for preserving privacy.

In this work, we take a different approach to solve the problems above. We focus on Pufferfish privacy, and propose a lightweight but automatic verification technique. We propose a formal model for data publishing mechanisms and reduce Pufferfish privacy into a verification problem for hidden Markov models (HMMs). Through our formalization, data curators can verify their specialized privacy mechanisms without going through tedious mathematical proofs.

We have implemented our algorithm in a prototypical tool called FAIER (the pufferFish privAcy verifIER). We consider privacy mechanisms for bounded discrete numerical queries such as counting. For those queries, classical continuous perturbations may give unusable answers or even lose privacy [28]. We hence discretize privacy mechanisms by applying discrete perturbations on such queries. We report case studies derived from differentially private mechanisms. Our studies show that naïve discretization may induce significant privacy risks. For the *Above Threshold* example, we show that discretization does not have any privacy at all. For this example, our tool generates *counterexamples* for an arbitrary small privacy budget ϵ. Another interesting problem for differential privacy is to find the largest lower bound of ϵ, below which the mechanism will not be differentially private. We discuss how our verification approach can be efficiently combined with testing techniques to solve this problem.

Below we summarize the main contributions of our paper:

1. We propose a verification framework for Pufferfish privacy by specifying privacy mechanisms as HMMs and analyzing privacy requirements in the models (Sect. 4). To our best knowledge, the work of Pufferfish privacy verification had not been investigated before.
2. Then we study the Pufferfish privacy verification problem on HMMs and prove the verification problem to be NP-hard (Sect. 5.1).
3. On the practical side, nevertheless, using SMT solvers, we design a verification algorithm which automatically verifies Pufferfish privacy (Sect. 5.2).
4. The verification algorithm is implemented into the tool FAIER (Sect. 6.1). We then perform case studies of classic mechanisms, such as Noisy Max and Above Threshold. Using our tool, we are able to catch privacy breaches of the specialized mechanisms (Sect. 6.2, 6.3).
5. Compared with the state-of-the-art tools DP-Sniper [10] and StatDP [14] on finding the privacy budget ϵ (or finding privacy violations) for differential privacy, our tool has advantageous performances in obtaining the most precise results within acceptable time for discrete mechanisms. We propose to exploit each advantage to the full to efficiently obtain a precise lower bound for the privacy budget ϵ (Sect. 7).

2 Preliminaries

A *Markov Chain* $K = (S, p)$ consists of a finite set S of *states* and a *transition distribution* $p : S \times S \to [0, 1]$ such that $\sum_{t \in S} p(s, t) = 1$ for every $s \in S$. A *Hidden Markov Model* (HMM) $H = (K, \Omega, o)$ is a Markov chain $K = (S, p)$ with a finite set Ω of *observations* and an *observation distribution* $o : S \times \Omega \to [0, 1]$ such that $\sum_{\omega \in \Omega} o(s, \omega) = 1$ for every $s \in S$. Intuitively, the states of HMMs are not observable. External observers do not know the current state of an HMM. Instead, they have a state distribution (called *information state*) $\pi : S \to [0, 1]$ with $\sum_{s \in S} \pi(s) = 1$ to represent the likelihood of each state in an HMM.

Let $H = ((S, p), \Omega, o)$ be an HMM and π an initial state distribution. The HMM H can be seen as a (randomized) generator for sequences of observations. The following procedure generates observation sequences of an arbitrary length:

1. $t \leftarrow 0$.
2. Choose an initial state $s_0 \in S$ by the initial state distribution π.
3. Choose an observation ω_t by the observation distribution $o(s_t, \bullet)$.
4. Choose a next state s_{t+1} by the transition distribution $p(s_t, \bullet)$.
5. $t \leftarrow t + 1$ and go to 3.

Given an observation sequence $\overline{\omega} = \omega_0\omega_1 \cdots \omega_k$ and a state sequence $\overline{s} = s_0 s_1 \cdots s_k$, it is not hard to compute the probability of observing $\overline{\omega}$ along $\overline{s}$ on an HMM $H = ((S, p), \Omega, o)$ with an initial state distribution π. Precisely,

$$\begin{aligned} &\Pr(\overline{\omega}, \overline{s}|H) = \Pr(\overline{\omega}|\overline{s}, H) \times \Pr(\overline{s}, H) \\ &= [o(s_0, \omega_0) \cdots o(s_k, \omega_k)] \times [\pi(s_0) p(s_0, s_1) \cdots p(s_{k-1}, s_k)] \\ &= \pi(s_0) o(s_0, \omega_0) \cdot p(s_0, s_1) \cdots p(s_{k-1}, s_k) o(s_k, \omega_k). \end{aligned} \tag{1}$$

Since state sequences are not observable, we are interested in the probability $\Pr(\overline{\omega}|H)$ for a given observation sequence $\overline{\omega}$. Using (1), we have $\Pr(\overline{\omega}|H) =$

$\sum_{\overline{s}\in S^{k+1}} \Pr(\overline{\omega}, \overline{s}|H)$. But the summation has $|S|^{k+1}$ terms and is hence inefficient to compute. An efficient algorithm is available to compute the probability $\alpha_t(s)$ for the observation sequence $\omega_0\omega_1 \cdots \omega_t$ with the state s at time t [31]. Consider the following definition:

$$\alpha_0(s) = \pi(s)o(s, \omega_0) \tag{2}$$

$$\alpha_{t+1}(s') = \left[\sum_{s\in S} \alpha_t(s)p(s, s')\right] o(s', \omega_{t+1}). \tag{3}$$

Informally, $\alpha_0(s)$ is the probability that the initial state is s with the observation ω_0. By induction, $\alpha_t(s)$ is the probability that the t-th state is s with the observation sequence $\omega_0\omega_1 \cdots \omega_t$. The probability of observing $\overline{\omega} = \omega_0\omega_1 \cdots \omega_k$ is therefore the sum of probabilities of observing $\overline{\omega}$ over all states s. Thus $\Pr(\overline{\omega}|H) = \sum_{s\in S} \alpha_k(s)$.

3 Pufferfish Privacy Framework

Differential privacy is a privacy framework for design and analysis of data publishing mechanisms [16]. Let $\mathcal{X}$ denote the set of *data entries*. A *data set* of size n is an element in $\mathcal{X}^n$. Two data sets $\overline{\mathbf{d}}, \overline{\mathbf{d}}' \in \mathcal{X}^n$ are *neighbors* (written $\Delta(\overline{\mathbf{d}}, \overline{\mathbf{d}}') \leq 1$) if $\overline{\mathbf{d}}$ and $\overline{\mathbf{d}}'$ are identical except for at most one data entry. A *data publishing mechanism* (or simply *mechanism*) $\mathcal{M}$ is a randomized algorithm which takes a data set $\overline{\mathbf{d}}$ as inputs. A mechanism satisfies ϵ-differential privacy if its output distributions differ by at most the multiplicative factor e^ϵ on every neighboring data sets.

Definition 1. *Let $\epsilon \geq 0$. A mechanism $\mathcal{M}$ is* ϵ-differentially private *if for all $r \in range(\mathcal{M})$ and data sets $\overline{\mathbf{d}}, \overline{\mathbf{d}}' \in \mathcal{X}^n$ with $\Delta(\overline{\mathbf{d}}, \overline{\mathbf{d}}') \leq 1$, we have $\Pr(\mathcal{M}(\overline{\mathbf{d}}) = r) \leq e^\epsilon \Pr(\mathcal{M}(\overline{\mathbf{d}}') = r)$.*

Intuitively, ϵ-differential privacy ensures similar output distributions on similar data sets. Limited differential information about each data entry is revealed and individual privacy is hence preserved. Though, differential privacy makes no assumption nor uses any prior knowledge about data sets. For data sets with correlated data entries, differential privacy may reveal too much information about individuals. Consider, for instance, a data set of family members. If a family member has contracted a highly contagious disease, all family are likely to have the same disease. In order to decide whether a specific family member has contracted the disease, it suffices to determine whether *any* member has the disease. It appears that specific information about an individual can be inferred from differential information when data entries are correlated. Differential privacy may be ineffective to preserve privacy in such circumstances [22].

Pufferfish is a Bayesian privacy framework which refines differential privacy. Theorem 6.1 in [23] shows how to define differential privacy equivalently in Pufferfish framework. In Pufferfish privacy, a random variable $\overline{\mathbf{D}}$ represents a data set drawn from a distribution $\theta \in \mathbb{D}$. The set $\mathbb{D}$ of distributions formalizes prior knowledge about data sets, such as whether data entries are independent or correlated. Moreover, a set $\mathbb{S}$ of *secrets* and a set $\mathbb{S}_{\text{pairs}} \subseteq \mathbb{S} \times \mathbb{S}$ of *discriminative secret pairs* formalize the information

to be protected. A mechanism $\mathcal{M}$ satisfies ϵ-Pufferfish privacy if its output distributions differ by at most the multiplicative factor e^{ϵ} when conditioned on all the secret pairs.

Definition 2. *Let $\mathbb{S}$ be a set of secrets, $\mathbb{S}_{pairs} \subset \mathbb{S} \times \mathbb{S}$ a set of discriminative secret pairs, $\mathbb{D}$ a set of data set distributions scenarios, and $\epsilon \geq 0$, a mechanism $\mathcal{M}$ is* ϵ-Pufferfish private *if for all $r \in range(\mathcal{M})$, $(s_i, s_j) \in \mathbb{S}_{pairs}$, $\theta \in \mathbb{D}$ with $\Pr(s_i|\theta) \neq 0$ and $\Pr(s_j|\theta) \neq 0$, we have*

$$\Pr(\mathcal{M}(\overline{\mathbf{D}}) = r|s_i, \theta) \leq e^{\epsilon} \Pr(\mathcal{M}(\overline{\mathbf{D}}) = r|s_j, \theta)$$

where $\overline{\mathbf{D}}$ is a random variable with the distribution θ.

In the definition, $\Pr(s_i|\theta) \neq 0$ and $\Pr(s_j|\theta) \neq 0$ ensure the probabilities $\Pr(\mathcal{M}(\overline{\mathbf{D}}) = r|s_i, \theta)$ and $\Pr(\mathcal{M}(\overline{\mathbf{D}}) = r|s_j, \theta)$ are defined. Hence $\Pr(\mathcal{M}(\overline{\mathbf{D}}) = r|s, \theta)$ is the probability of observing r conditioned on the secret s and the data set distribution θ. Informally, ϵ-Pufferfish privacy ensures similar output distributions on discriminative secrets and prior knowledge. Since limited information is revealed from prior knowledge, each pair of discriminative secrets is protected.

4 Geometric Mechanism as Hidden Markov Model

We first recall in Sect. 4.1 the definition of geometric mechanism, a well-known discrete mechanism for differential privacy. In Sect. 4.2, we then recall an example exploiting Markov chains to model geometric mechanisms, followed by our modeling formalism and Pufferfish privacy analysis using HMMs in Sect. 4.3.

4.1 Geometric Mechanism

Consider a simple data set with only two data entries. Each entry denotes whether an individual has a certain disease. Given such a data set, we wish to know how many individuals contract the disease in the data set. More generally, a *counting* query returns the number of entries satisfying a given predicate in a data set $\overline{\mathbf{d}} \in \mathcal{X}^n$. The number of individuals contracting the disease in a data set is hence a counting query. Note that the difference of counting query results on neighboring data sets is at most 1.

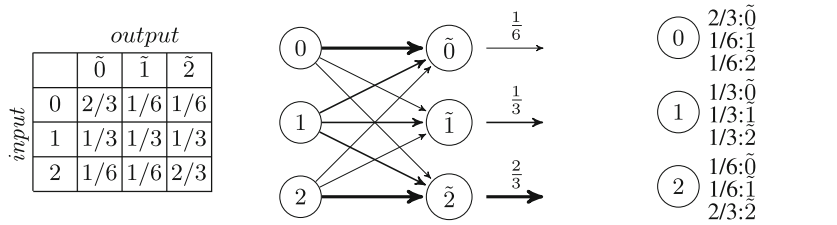
Counting queries may reveal sensitive information about individuals. For instance, suppose we know John's record is in the data set. We immediately infer that John has contracted the disease if the query answer is 2. In order to protect privacy, several mechanisms are designed to answer counting queries.

Consider a counting query $f : \mathcal{X}^n \to \{0, 1, \ldots, n\}$. Let $\alpha \in (0, 1)$. The α-*geometric mechanism* $\mathcal{G}_f$ for the counting query f on the data set $\overline{\mathbf{d}}$ outputs $f(\overline{\mathbf{d}}) + Y$ on a data set $\overline{\mathbf{d}}$ where Y is a random variable with the geometric distribution [20,21]: $\Pr[Y = y] = \frac{1-\alpha}{1+\alpha}\alpha^{|y|}$ for $y \in \mathbb{Z}$. For any neighboring data sets $\overline{\mathbf{d}}, \overline{\mathbf{d}}' \in \mathcal{X}^n$, recall that $|f(\overline{\mathbf{d}}) - f(\overline{\mathbf{d}}')| \leq 1$. If $f(\overline{\mathbf{d}}) = f(\overline{\mathbf{d}}')$, the α-geometric mechanism has the same output distribution for f on $\overline{\mathbf{d}}$ and $\overline{\mathbf{d}}'$. If $|f(\overline{\mathbf{d}}) - f(\overline{\mathbf{d}}')| = 1$, it is easy to conclude that $\Pr(\mathcal{G}_f(\overline{\mathbf{d}}) = r) \leq e^{-\ln \alpha} \Pr(\mathcal{G}_f(\overline{\mathbf{d}}') = r)$ for any neighboring $\overline{\mathbf{d}}, \overline{\mathbf{d}}'$ and $r \in \mathbb{Z}$. The

α-geometric mechanism is $-\ln\alpha$-differentially private for any counting query f. To achieve ϵ-differential privacy, one simply chooses $\alpha = e^{-\epsilon}$.

The range of the geometric mechanism is $\mathbb{Z}$. It may give nonsensical outputs such as negative integers for non-negative queries. The *truncated α-geometric mechanism over* $\{0, 1, \ldots, n\}$ outputs $f(\overline{\mathbf{d}}) + Z$ where Z is a random variable with the distribution:

$$\Pr[Z = z] = \begin{cases} 0 & \text{if } z < -f(x) \\ \frac{\alpha^{f(x)}}{1+\alpha} & \text{if } z = -f(x) \\ \frac{1-\alpha}{1+\alpha}\alpha^{|z|} & \text{if } -f(x) < z < n - f(x) \\ \frac{\alpha^{n-f(x)}}{1+\alpha} & \text{if } z = n - f(x) \\ 0 & \text{if } z > n - f(x) \end{cases}$$



(a) $\frac{1}{2}$-Geometric Mechanism (b) Markov Chain (c) Hidden Markov Model

Fig. 1. Truncated $\frac{1}{2}$-geometric mechanism

Note the range of the truncated α-geometric mechanism is $\{0, 1, \ldots, n\}$. The truncated α-geometric mechanism is also $-\ln\alpha$-differentially private for any counting query f. We will study several examples of this mechanism to get a better understanding of Pufferfish privacy and how we use models to analyze it.

4.2 Differential Privacy Using Markov Chains

We present a simple example taking from [24], slightly adapted for analyzing different models, i.e., the Markov chain and the hidden Markov model.

Example 1. To see how differential privacy works, consider the truncated $\frac{1}{2}$-geometric mechanism (Fig. 1a). In the table, we consider a counting query $f : \mathcal{X}^2 \to \{0, 1, 2\}$. For any data set $\overline{\mathbf{d}}$, the mechanism outputs j when $f(\overline{\mathbf{d}}) = i$ with probability indicated at the (i, j)-entry in the table. For instance, the mechanism outputs $\tilde{0}$, $\tilde{1}$, and $\tilde{2}$ with probabilities $\frac{2}{3}$, $\frac{1}{6}$, and $\frac{1}{6}$ respectively when $f(\overline{\mathbf{d}}) = 0$.

Let f be the query counting the number of individuals contracting a disease. Consider a data set $\overline{\mathbf{d}}$ whose two members (including John) have contracted the disease. The number of individuals contracting the disease is 2 and hence $f(\overline{\mathbf{d}}) = 2$. From the table in Fig. 1a, we see the mechanism answers $\tilde{0}$, $\tilde{1}$, and $\tilde{2}$ with probabilities $\frac{1}{6}$, $\frac{1}{6}$, and

$\frac{2}{3}$ respectively. Suppose we obtain another data set $\overline{\mathbf{d}}'$ by replacing John with an individual who does not contract the disease. The number of individuals contracting the disease for the new data set is 1 and thus $f(\overline{\mathbf{d}}') = 1$. Then, the mechanism answers $\tilde{0}$, $\tilde{1}$, and $\tilde{2}$ with the probability $\frac{1}{3}$.

The probabilities of observing $\tilde{0}$ on the data sets $\overline{\mathbf{d}}$ and $\overline{\mathbf{d}}'$ are respectively $\frac{1}{6}$ and $\frac{1}{3}$. They differ by the multiplicative factor 2. For other outputs, their observation probabilities are also bounded by the same factor. The truncated $\frac{1}{2}$-geometric mechanism is hence $\ln(2)$-differentially private.

In order to formally analyze privacy mechanisms, we specify them as probabilistic models. Figure 1b shows a Markov chain for the truncated $\frac{1}{2}$-geometric mechanism. We straightly turn inputs and outputs of the table in Fig. 1a into states of the Markov chain and output probabilities into transition probabilities. In the figure, thin arrows denote transitions with probability $\frac{1}{6}$; medium arrows denote transitions with probability $\frac{1}{3}$; thick arrows denote transitions with probability $\frac{2}{3}$. For instance, state 0 can transit to state $\tilde{0}$ with probability $\frac{2}{3}$ while it can transit to the state $\tilde{1}$ with probability $\frac{1}{6}$. ■

The Markov chain model is straightforward but can become hazy for complicated privacy mechanism. We next discuss how to use an HMM to model the mechanism.

4.3 Pufferfish Privacy Using Hidden Markov Models

We denote data sets as states and possible outputs of the mechanism are denoted by observations. The transition distribution stimulates the randomized privacy mechanism performed on data sets. Distributions of data sets are denoted by initial information states. Privacy analysis can then be performed by comparing observation probabilities from the two initial information states. We illustrate the ideas in examples.

Example 2. Fig. 1c gives an HMM for the truncated $\frac{1}{2}$-geometric mechanism. For any counting query f from $\mathcal{X}^2$ to $\{0, 1, 2\}$, it suffices to represent each $\overline{\mathbf{d}} \in \mathcal{X}^2$ by $f(\overline{\mathbf{d}})$ because the mechanism only depends on $f(\overline{\mathbf{d}})$. The order of entries, for instance, is irrelevant to the mechanism. We hence have the states 0, 1 and 2 denoting the set $\{f(\overline{\mathbf{d}}) : \overline{\mathbf{d}} \in \mathcal{X}^2\}$ in the figure. Let $\{\tilde{0}, \tilde{1}, \tilde{2}\}$ be the set of observations. We encode output probabilities into observation probabilities at states. At state 0, for instance, $\tilde{0}$, $\tilde{1}$, $\tilde{2}$ can all be observed with probability $\frac{2}{3}$, $\frac{1}{6}$, $\frac{1}{6}$ respectively. It is obvious that the number of states are reduced by half compared with the Markov chain. Generally, HMMs allow multiple observations to show at one single state, which leads to smaller models.

Fix an order for states, say, $0, 1, 2$. An information state can be represented by an element in $[0, 1]^3$. In differential privacy, we would like to analyze probabilities of every observation from neighboring data sets. For counting queries, neighboring data sets can change query results by at most 1. Let $\overline{\mathbf{d}}$ be a data set. Consider the initial information state $\pi = (0, 0, 1)$ corresponding to $f(\overline{\mathbf{d}}) = 2$. For any neighbor $\overline{\mathbf{d}}'$ of $\overline{\mathbf{d}}$, we have $f(\overline{\mathbf{d}}') = 2$ or $f(\overline{\mathbf{d}}') = 1$. It suffices to consider corresponding information states π or $\tau = (0, 1, 0)$. Let's compare the probability of observing $\omega = \tilde{1}$ from information states π and τ. Starting from π, we have $\alpha_0 = \pi$ and probabilities of $\frac{1}{6}$, $\frac{1}{3}$ and $\frac{1}{6}$ respectively observing $\tilde{1}$ at each state. So the probability of observing ω is $\frac{1}{6}$. On the other hand, we

have $\alpha_0 = \tau$ and the probability of observing ω is $\frac{1}{3}$. Similarly, one can easily check the probabilities of observing $\tilde{0}$ and $\tilde{2}$ on any neighboring data sets and the ratio of one probability over the other one under the same observation will not be more than 2. ■

Differential privacy provides a framework for quantitative privacy analysis. The framework ensures similar output distributions regardless of the information about an arbitrary individual. In other words, if an attacker gets certain prior knowledge about the data sets, chances are that differential privacy will underestimate privacy risks. Since all data entries are correlated, replacing one data entry does not yield feasible data sets with correlated entries. Consequently, it is questionable to compare output distributions on data sets differing in only one entry. Instead, this is the scenario where Pufferfish privacy should be applied.

Example 3. Consider a data set about contracting a highly contagious disease containing John and a family member he lives with. An attacker wishes to know if John has contracted the disease. Since the data set keeps information on the contagious disease about two family members, an attacker immediately deduces that the number of individuals contracting the disease can only be 0 or 2. The attacker hence can infer whether John has the disease by counting the number of individuals contracting the disease.

Suppose a data curator tries to protect John's privacy by employing the truncated $\frac{1}{2}$-geometric mechanism (Fig. 1). We analyze this mechanism formally in the Pufferfish framework. Let the set of data entries $\mathcal{X} = \{0, 1\}$ and there are four possible data sets in $\mathcal{X}^2$. For any $0 < p < 1$, define the data set distribution $\theta_p : \mathcal{X}^2 \to [0, 1]$ as follows. $\theta_p(0, 0) = 1 - p$, $\theta_p(1, 1) = p$, and $\theta_p(0, 1) = \theta_p(1, 0) = 0$. Consider the distribution set $\mathbb{D} = \{\theta_p : 0 < p < 1\}$. Note that infeasible data sets are not in the support of θ_p.

Assume John's entry is in the data set. Define the set of secrets $\mathbb{S} = \{c, nc\}$ where c denotes that John has contracted the disease and nc denotes otherwise. Our set of discriminative secret pairs $\mathbb{S}_{\text{pairs}}$ is $\{(c, nc), (nc, c)\}$. That is, we would like to compare probabilities of all outcomes when John has the disease or not.

When John has not contracted the disease, the only possible data set is $(0, 0)$ by the distribution θ_p. The probability of observing $\tilde{0}$ therefore is $\frac{2}{3}$ (Fig. 1a). When John has the disease, the data set $(0, 0)$ is not possible under the condition of the secret and the distribution θ_p. The only possible data set is $(1, 1)$. The probability of observing $\tilde{0}$ is $\frac{1}{6}$. Now we have $\frac{2}{3} = \Pr(\mathcal{G}_f(\overline{\mathbf{D}}) = \tilde{0} | nc, \theta_p) \not\leq 2 \times \frac{1}{6} = 2 \times \Pr(\mathcal{G}_f(\overline{\mathbf{D}}) = \tilde{0} | c, \theta_p)$. We conclude the truncated $\frac{1}{2}$-geometric mechanism does not conform to $\ln(2)$-Pufferfish privacy. Instead, it satisfies $\ln(4)$-Pufferfish privacy. ■

Table 1. Pufferfish analysis of $\frac{1}{2}$-geometric mechanism

Data Sets\Observations	$\tilde{0}$	$\tilde{1}$	$\tilde{2}$
Without John's record	$\frac{p^2-4p+4}{6}$	$\frac{-2p^2+2p+1}{6}$	$\frac{p^2+2p+1}{6}$
With John's record	$\frac{4-3p}{12-6p}$	$\frac{4-3p}{12-6p}$	$\frac{2}{6-3p}$

With the formal model (Fig. 1c), it is easy to perform privacy analysis in the Pufferfish framework. More precisely, the underlying Markov chain along with observation distribution specify the privacy mechanism on input data sets. Prior knowledge about data sets is nothing but distributions of them. Since data sets are represented by various states, prior knowledge is naturally formalized as initial information states in HMMs. For Pufferfish privacy analysis, we again compare observation probabilities from initial information states conditioned on secret

pairs. The standard algorithm for HMMs allows us to perform more refined privacy analysis. Besides, it is interesting to observe the striking similarity between the Pufferfish privacy framework and HMMs. In both cases, input data sets are unknown but specified by distributions. Information can only be released by observations because inputs and hence computation are hidden from external attackers or observers. Pufferfish privacy analysis with prior knowledge is hence closely related to observation probability analysis from information states. Such similarities can easily be identified in the examples.

Example 4. Consider a non-contagious disease. An attacker may know that contracting the disease is an independent event with probability p. Even though the attacker does not know how many individuals have the disease exactly, he infers that the number of individuals contracting the disease is 0, 1, and 2 with probabilities $(1-p)^2$, $2p(1-p)$, and p^2 respectively. The prior knowledge corresponds to the initial information state $\pi = ((1-p)^2, 2p(1-p), p^2)$ in Fig. 1c. Assume John has contracted the disease. We would like to compare probabilities of observations $\tilde{0}$, $\tilde{1}$, and $\tilde{2}$ given the prior knowledge and the presence or absence of John's record.

Suppose John's record is indeed in the data set. Since John has the disease, the number of individuals contracting the disease cannot be 0. By the prior knowledge, one can easily obtain the initial information state $\pi = (0, \frac{2p(1-p)}{2p(1-p)+p^2}, \frac{p^2}{2p(1-p)+p^2}) = (0, \frac{2-2p}{2-p}, \frac{p}{2-p})$. If John's record is not in the data set, the initial information state remains as $\tau = ((1-p)^2, 2p(1-p), p^2)$. Then one can compute all the observation probabilities starting from π and τ respectively, which are summarized in Table 1:

For the observation $\tilde{0}$, it is not hard to check $\frac{1}{2} \times \frac{4-3p}{12-6p} \leq \frac{p^2-4p+4}{6} \leq 2 \times \frac{4-3p}{12-6p}$ for any $0 < p < 1$. Similarly, we have $\frac{1}{2} \times \frac{4-3p}{12-6p} \leq \frac{-2p^2+2p+1}{6} \leq 2 \times \frac{4-3p}{12-6p}$ and $\frac{1}{2} \times \frac{2}{6-3p} \leq \frac{p^2+2p+1}{6} \leq 2 \times \frac{2}{6-3p}$ for observations $\tilde{1}$ and $\tilde{2}$ respectively. Therefore, the truncated $\frac{1}{2}$-geometric mechanism satisfies $\ln(2)$-Pufferfish privacy when contracting the disease is *independent*. ■

The above example demonstrates that certain prior knowledge, such as independence of data entries, is indeed not harmful to privacy under the Pufferfish framework. In [23], it is shown that differential privacy is subsumed by Pufferfish privacy (Theorem 6.1) under independence assumptions. The above example is also an instance of the general theorem but formalized in an HMM.

5 Pufferfish Privacy Verification

In this section, we formally define the verification problem for Pufferfish privacy and give the computation complexity results in Sect. 5.1. Then we propose an algorithm to solve the problem in Sect. 5.2.

5.1 Complexity of Pufferfish Privacy Problem

We model the general Pufferfish privacy problems into HMMs and the goal is to check whether the privacy is preserved. First, we define the *Pufferfish verification problem*:

Definition 3. *Given a set of secrets* $\mathbb{S}$*, a set of discriminative secret pairs* $\mathbb{S}_{pairs}$*, a set of data evolution scenarios* $\mathbb{D}$*,* $\epsilon > 0$*, along with mechanism* $\mathcal{M}$ *in a hidden Markov model* $H = (K, \Omega, o)$*, where probability distributions are all discrete. Deciding whether* $\mathcal{M}$ *satisfies* ϵ*-Pufferfish privacy under* $(\mathbb{S}, \mathbb{S}_{pairs}, \mathbb{D})$ *is the* Pufferfish verification problem.

The modeling intuition for H is to use states and transitions to model the data sets and operations in the mechanism $\mathcal{M}$, obtain initial distribution pairs according to prior knowledge $\mathbb{D}$ and discriminative secrets $\mathbb{S}_{\text{pairs}}$, and set outputs as observations in states. Then the goal turns into checking whether the probabilities under the same observation sequence are mathematically similar, i.e., differ by at most the multiplicative factor e^{ϵ}, for every distribution pair and every observation sequence. Therefore, our task is to find the observation sequence and distribution pair that make the observing probabilities differ the most. That is, in order to satisfy Pufferfish privacy, for every observation sequence $\overline{\omega} = \omega_1\omega_2 \ldots$, secret pair $(s_i, s_j) \in \mathbb{S}_{\text{pairs}}$ and $\theta \in \mathbb{D}$, one should have

$$\max_{\overline{\omega},(s_i,s_j),\theta} \Pr(\mathcal{M}(\overline{\mathbf{D}}) = \overline{\omega}|s_i, \theta) - e^{\epsilon} \Pr(\mathcal{M}(\overline{\mathbf{D}}) = \overline{\omega}|s_j, \theta) \tag{4}$$

$$\max_{\overline{\omega},(s_i,s_j),\theta} \Pr(\mathcal{M}(\overline{\mathbf{D}}) = \overline{\omega}|s_j, \theta) - e^{\epsilon} \Pr(\mathcal{M}(\overline{\mathbf{D}}) = \overline{\omega}|s_i, \theta) \tag{5}$$

no more than 0. However, by showing a reduction from the classic Boolean Satisfiability Problem [30], this problem is proved to be NP-hard (in the full version [25]):

Theorem 1. *The Pufferfish verification problem is NP-hard.*

To the best of our knowledge, this is the first complexity result for the Pufferfish verification problem. Note that differential privacy is subsumed by Pufferfish privacy. Barthe et al. [3] show undecidability results for differential privacy mechanisms with continuous noise. Instead, we focus on Pufferfish privacy with discrete state space in HMMs. The complexity bound is lower if more simple models such as Markov chains are used. However some discrete mechanisms in differential privacy, such as Above Threshold, can hardly be modeled in Markov chains [24].

5.2 Verifying Pufferfish Privacy

Given the complexity lower bound in the previous section, next goal is to develop an algorithm to verify ϵ-Pufferfish privacy on any given HMM. We employ Satisfiability Modulo Theories (SMT) solvers in our algorithm. For all observation sequences of length k, we will construct an SMT query to find a sequence violating ϵ-Pufferfish privacy. If no such sequence can be found, the given HMM satisfies ϵ-Pufferfish privacy for all observation sequences of length k.

Let $H = ((S, p), \Omega, o)$ be an HMM, π, τ two initial distributions on S, $c \geq 0$ a real number, and k a positive integer. With a fixed observation sequence $\overline{\omega}$, computing the probability $\Pr(\overline{\omega}|\pi, H)$ can be done in polynomial time [31]. To check if $\Pr(\overline{\omega}|\pi, H) > c \cdot \Pr(\overline{\omega}|\tau, H)$ for any fixed observation sequence $\overline{\omega}$, one simply computes the respective probabilities and then checks the inequality.

Our algorithm exploits the efficient algorithm of HMMs for computing the probability of observation sequences. Rather than a fixed observation sequence, we declare k

Algorithm 1. Pufferfish Check

Require: $H = ((S, p), \Omega, o)$: a hidden Markov model; π, τ: state distributions on S; c: a non-negative real number; k: a positive integer

Ensure: An SMT query q such that q is unsatisfiable iff $\Pr(\overline{\omega}|\pi, H) \leq c \cdot \Pr(\overline{\omega}|\tau, H)$ for every observation sequences $\overline{\omega}$ of length k

1: **function** PUFFERFISHCHECK(H, π_0, π_1, c, k)
2: **for** $s \in S$ **do**
3: $\alpha_0(s) \leftarrow$ PRODUCT($\pi(s)$, SELECT($\mathsf{w}_0, \Omega, o(s, \bullet)$))
4: $\beta_0(s) \leftarrow$ PRODUCT($\tau(s)$, SELECT($\mathsf{w}_0, \Omega, o(s, \bullet)$))
5: **for** $t \leftarrow 1$ **to** $k - 1$ **do**
6: **for** $s' \in S$ **do**
7: $\alpha_t(s') \leftarrow$ PRODUCT(DOT($\alpha_{t-1}, p(\bullet, s')$), SELECT($\mathsf{w}_t, \Omega, o(s', \bullet)$))
8: $\beta_t(s') \leftarrow$ PRODUCT(DOT($\beta_{t-1}, p(\bullet, s')$), SELECT($\mathsf{w}_t, \Omega, o(s', \bullet)$))
9: **return** GT(SUM(α_{k-1}), PRODUCT(c, SUM(β_{k-1}))) $\wedge \bigwedge_{t=0}^{k-1} \mathsf{w}_t \in \Omega$

SMT variables $\mathsf{w}_0, \mathsf{w}_1, \ldots, \mathsf{w}_{k-1}$ for observations at each step. The observation at each step is determined by one of the k variables. Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$ be the set of observations. We define the SMT expression SELECT (w, $\{\omega_1, \omega_2, \ldots, \omega_m\}$, $o(s, \bullet)$) equal to $o(s, \omega)$ when the SMT variable w is $\omega \in \Omega$. It is straightforward to formulate by the SMT ite (if-then-else) expression:

$$\mathsf{ite}(\mathsf{w} = \omega_1, o(s, \omega_1), \mathsf{ite}(\mathsf{w} = \omega_2, o(s, \omega_2), \ldots, \mathsf{ite}(\mathsf{w} = \omega_m, o(s, \omega_m), \mathsf{w}) \ldots))$$

Using SELECT($\mathsf{w}, \{\omega_1, \omega_2, \ldots, \omega_m\}, o(s, \bullet)$), we construct an SMT expression to compute $\Pr(\overline{\mathsf{w}}|\pi, H)$ where $\overline{\mathsf{w}}$ is a sequence of SMT variables ranging over the observations Ω (Algorithm 1). Recall the Eqs. (2) and (3). We simply replace the expression $o(s, \omega)$ with the new one SELECT(w, $\{\omega_1, \omega_2, \ldots, \omega_m\}$, o $(s, \bullet)$) to leave the observation determined by the SMT variable w. In the algorithm, we also use auxiliary functions. PRODUCT($smtExp_0, \ldots, smtExp_m$) returns the SMT expression denoting the product of $smtExp_0, \ldots, smtExp_m$. Similarly, SUM($smtExp_0, \ldots, smtExp_m$) returns the SMT expression for the sum of $smtExp_0, \ldots, smtExp_m$. GT($smtExp_0$, $smtE$ xp_1) returns the SMT expression for $smtExp_0$ greater than $smtExp_1$. Finally, DOT ($[\mathsf{a}_0, \mathsf{a}_1, \ldots, \mathsf{a}_n], [\mathsf{b}_0, \mathsf{b}_1, \ldots, \mathsf{b}_n]$) returns the SMT expression for the inner product of the two lists of SMT expressions, namely, SUM(PRODUCT($\mathsf{a}_0, \mathsf{b}_0$), ..., PRODUCT($\mathsf{a}_n, \mathsf{b}_n$)).

Algorithm 1 is summarized in the following theorem.

Theorem 2. *Let $H = ((S, p), \Omega, o)$ be a hidden Markov model, π, τ state distributions on S, $c > 0$ a real number, and $k > 0$ an integer. Algorithm 1 returns an SMT query such that the query is unsatisfiable iff $\Pr(\overline{\omega}|\pi, H) \leq c \cdot \Pr(\overline{\omega}|\tau, H)$ for every observation sequence $\overline{\omega}$ of length k.*

In practice, the integer k depends on the length of observation sequence we want to make sure to satisfy Pufferfish privacy. For instance, in the model of Fig. 1c, the maximal length of observation sequence is 1 and thus $k = 1$. If there exist cycles in

models such as Fig. 3, which implies loops in the mechanisms, k should keep increasing (and stop before a set value) in order to examine outputs of different lengths.

6 Pufferfish Privacy Verifier: FAIER

We implement our verification tool and present experimental results in Subsect. 6.1. For the well-known differential privacy mechanisms Noisy Max and Above Threshold, we provide modeling details in HMMs and verify the privacy wrt. several Pufferfish privacy scenarios in Subsect. 6.2 and 6.3, accordingly.

6.1 Evaluation for FAIER

We implement our verification algorithm (Algorithm 1) into the tool FAIER, which is the pufferFish privAcy verifIER. It is implemented in **C++** environment with the SMT solver $Z3$ [29] and we performed all experiments on an Intel(R) Core i7-8750H @ 2.20GHz CPU machine with 4 GB memory and 4 cores in the virtual machine. All the examples in this paper have been verified.

The inputs for our tool include an HMM H of the mechanism to be verified, distribution pair (π,τ) on states in H, a non-negative real number c indicating the privacy budget and an input k specifying the length of observation sequences. Note that unknown parameters are also allowed in the SMT formulae, which can encode certain prior knowledge or data sets distributions.

Table 2. Experiment results: ✓ indicates the property holds, and ✗ not.

Mechanism	Privacy scenario	Result	
		Query answer	Counterexample
Truncated $\frac{1}{2}$-geometric Mechanism	$\ln(2)$-differential privacy (Ex. 2)	✓	
	$\ln(2)$-pufferfish privacy (Ex. 3)	✗	$\tilde{2}$
	$\ln(2)$-pufferfish privacy (Ex. 4)	✓	
Discrete Noisy Max (Algorithm 2)	$\ln(2)$-pufferfish privacy (Ex. 5)	✓	
	$\ln(2)$-pufferfish privacy (Ex. 6)	✗	$\perp, \tilde{3}; p_A = p_B = p_C = \frac{1}{2}$
Above Threshold Algorithm (Algorithm 3)	$4\ln(2)$-differential privacy	✗	$\sqcup, 01, \perp, 12, \perp, 12, \perp, 12, \perp, 21, \top$

We summarize the experiment results in this paper for pufferfish privacy, as well as differential privacy in Table 2. FAIER has the following outputs:

– *Counterexample:* If the privacy condition does not hold (marked by ✗), FAIER will return a witnessing observation sequence leading to the violation.

– *Parameter Synthesis:* If there exist unknown parameters in the model, such as the infection rate p for some disease, a value will be synthesized for the counterexample. See Example 6 where counterexample is found when p_A, p_B, p_C are equal to $\frac{1}{2}$; Or, no value can be found if the privacy is always preserved. See Example 5.
– ✓ is returned if the privacy is preserved.

Note that if there exists a loop in the model, the bound k should continue to increase when an 'UNSAT' is returned. Specially, the bound is set at a maximum of 15 for Above Threshold. It may happen that FAIER does not terminate since some nonlinear constraints are too complicated for $Z3$, such as Example 5, which cannot solved by $Z3$ within 60 min. Thus we encode them into a more powerful tool REDLOG for nonlinear constraints [15]. For every experiment in the table, the time to construct the HMM model and SMT queries is less than 1 s; the time for solving SMT queries are less than 2 s, except for Example 5.

Among the mechanisms in Table 2, Algorithm 2, 3 need our further investigation. We examine these algorithms carefully in the following subsections.

6.2 Noisy Max

Noisy Max is a simple yet useful data publishing mechanism in differential privacy [14, 16]. Consider n queries of the same range, say, the number of patients for n different diseases in a hospital. We are interested in knowing which of the n diseases has the maximal number of patients in the hospital. A simple privacy-respecting way to release the information is to add independent noises to every query result and then return the index of the maximal noisy results.

Algorithm 2. Discrete Noisy Max

Require: $0 \leq v_1, v_2, \ldots, v_n \leq 2$
Ensure: The index r with the maximal $\tilde{v}_r$ among $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_n$

```
1:  function DiscreteNoisyMax(v_1, v_2, ..., v_n)
2:      M, r, c ← −1, 0, 0
3:      for each v_i do
4:          match v_i with                                  ▷ apply 1/2-geometric mechanism
5:              case 0: ṽ_i ← 0, 1, 2 with probability 2/3, 1/6, 1/6
6:              case 1: ṽ_i ← 0, 1, 2 with probability 1/3, 1/3, 1/3
7:              case 2: ṽ_i ← 0, 1, 2 with probability 1/6, 1/6, 2/3
8:          if M = ṽ_i then
9:              c ← c + 1
10:             r ← i with probability 1/c
11:         if M < ṽ_i then
12:             M, r, c ← ṽ_i, i, 1
13:     return r
```

In [16], Noisy Max algorithm adds continuous Laplacian noises to each query result. The continuous Noisy Max algorithm is proved to effectively protect privacy for neighboring data sets [14]. In practice continuous noises however are replaced by discrete

noises using floating-point numbers. Technically, the distribution of discrete floating-point noises is different from the continuous distribution in mathematics. Differential privacy can be breached [28]. The proof for continuous Noisy Max algorithm does not immediately apply. Indeed, care must be taken to avoid privacy breach.

We introduce our algorithm and model. The standard algorithm is modified by adding discrete noises to query results (Algorithm 2). In the algorithm, the variables M and r contain the maximal noisy result and its index respectively. We apply the truncated $\frac{1}{2}$-geometric mechanism to each query with the corresponding discrete range. To avoid returning a fixed index when there are multiple noisy results with the same value, the discrete algorithm explicitly returns the index of the maximal noisy value with an equal probability (Line. 8–14).

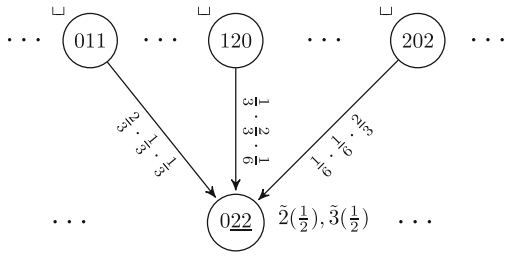


Fig. 2. Hidden Markov model for noisy max

The HMM model with $n = 3$ queries is illustrated in Fig. 2. The top states labeled 011 and 120 correspond to three query results (on neighboring data sets) and $\sqcup$, i.e. nothing, is observed in the initial states. Both states have a transition to the state $0\underline{22}$, representing the perturbed query results obtained with different probabilities. The index of the maximal result will be observed, which is 2 or 3 with probability $\frac{1}{2}$. Next we analyze Algorithm 2 under the Pufferfish framework.

Example 5. Consider three counting queries f_A, f_B, and f_C for the number of individuals contracting the diseases A, B, and C respectively in the data set $\mathcal{X}^2$ with $\mathcal{X} = \{(0,0,0),(0,0,1),\ldots,(1,1,1)\}$. An element $(a,b,c) \in \mathcal{X}$ denotes whether the data entry contracts the diseases A, B, and C respectively. Assume that the contraction of each disease is independent among individuals and the probabilities of contracting the diseases A, B, and C are p_A, p_B, and p_C respectively. The prior knowledge induces an information state for the model in Fig. 2. For example, the state 120 has the probability $2p_A(1-p_A) \cdot p_B^2 \cdot (1-p_C)^2$.

Suppose John is in the data set and whether John contracts the disease A is a secret. We would like to check if the discrete Noisy Max algorithm can protect the secret using the Pufferfish privacy framework. Let us compute the initial information state π given that John has not contracted disease A. For instance, the initial probability of the state 120 is $\frac{2p_A(1-p_A)}{(1-p_A)^2+2p_A(1-p_A)} \cdot p_B^2 \cdot (1-p_C)^2$. The initial information state π is obtained by computing the probabilities of each of the 3^3 top states. Given that John has the disease A, the initial information state τ is computed similarly. In this case, the initial probability of the state 120 becomes $\frac{2p_A(1-p_A)}{2p_A(1-p_A)+p_A^2} \cdot p_B^2 \cdot (1-p_C)^2$. Probabilities of the 3^3 top states form the initial information state τ. From the initial information state π and τ, we compute the probabilities of observing $\sqcup\tilde{1}$, $\sqcup\tilde{2}$, and $\sqcup\tilde{3}$ in the formal model (Fig. 2). The formulae for observation probabilities are easy to compute. However, the SMT solver Z3 cannot solve the non-linear formulae generated by our algorithm. In

order to establish Pufferfish privacy automatically, we submit the non-linear formulae to the constraint solver REDLOG. This time, the solver successfully proves the HMM satisfying $\ln(2)$-Pufferfish privacy. ■

Algorithm 2 is $\ln(2)$-Pufferfish private when the contraction of diseases is independent for data entries. Our next step is to analyze the privacy mechanism model when the contraction of the disease A is correlated among data entries.

Example 6. Assume that the data set consists of 2 family members, including John, and there are 5 queries which ask the number of patients of 5 diseases in the data set. To protect privacy, Algorithm 2 is applied to query results. Now assume an attacker has certain prior knowledge: 1. Disease 1 is so highly contagious that either none or both members infect the disease; 2. Disease 2 to Disease 5 are such diseases that every person has the probability of p_k to catch Disease k; and 3. The attacker knows the values of probabilities: $p_k = \frac{k}{10}$ for $k \in \{3, 4, 5\}$, but does not know the value of p_2. Suppose the secret is whether John has contracted Disease 1 and we wonder whether there exists such a p_2 that $\ln(2)$-Pufferfish private is violated. We can compute the initial distribution pair π and τ given the above information. For instance, if John has contracted Disease 1, then the initial probability for state 21110 is $p_2(1-p_2)\cdot(\frac{3}{10})(1-\frac{3}{10})\cdot(\frac{4}{10})(1-\frac{4}{10})(1-\frac{5}{10})^2$. Similarly, we obtain the initial information state given that John has not contracted the disease. Then FAIER verifies the mechanism does not satisfy $\ln(2)$-Pufferfish private with the synthesized parameter $p_2 = \frac{1}{2}$. ■

Provably correct privacy mechanisms can leak private information by seemingly harmless modification or assumed prior knowledge. Ideally, privacy guarantees of practical mechanisms need be re-established. Our verification tool can reveal ill-designed privacy protection mechanisms easily.

6.3 Above Threshold

Above threshold is a classical differentially private mechanism for releasing numerical information [16]. Consider a data set and an *infinite* sequence of counting queries $f_1, f_2, \ldots$. We would like to know the index of the first query whose result is above a given threshold. In order to protect privacy, the classical algorithm adds continuous noises on the threshold and each query result. If the noisy query result is less than the noisy threshold, the algorithm reports $\perp$ and continues to the next counting query. Otherwise, the algorithm reports $\top$ and stops.

We consider counting queries with range $\{0, 1, 2\}$ and apply the truncated geometric mechanism for discrete noises. The discrete above threshold algorithm is shown in Algorithm 3. The algorithm first obtains the noisy threshold $\tilde{t}$ using the truncated $\frac{1}{4}$-geometric mechanism. For each query result r_i, it computes a noisy result $\tilde{r}_i$ by applying the truncated $\frac{1}{2}$-geometric mechanism. If $\tilde{r}_i < \tilde{t}$, the algorithm outputs $\perp$ and continues. Otherwise, it halts with the output $\top$.

Algorithm and Model. To ensure ϵ-differential privacy, the classical algorithm applies the $\frac{2}{\epsilon}$- and $\frac{4}{\epsilon}$-Laplace mechanism to the threshold and each query result respectively.

The continuous noisy threshold and query results are hence $\frac{\epsilon}{2}$- and $\frac{\epsilon}{4}$-differentially private. In Algorithm 3, the discrete noisy threshold and query results are $2\ln(2)$- and $\ln(2)$-differentially private. If the classical proof still applies, we expect the discrete above threshold algorithm is $4\ln(2)$-differentially private for $\frac{\epsilon}{2} = 2\ln(2)$.

Figure 3 gives an HMM for Algorithm 3. In the model, the state $t_i r_j$ represents the input threshold $t = i$ and the first query result $r = f_1(\overline{\mathbf{d}}) = j$ for an input data set $\overline{\mathbf{d}}$. From the state $t_i r_j$, we apply the truncated $\frac{1}{4}$-geometric mechanism. The state $\tilde{t}_i r_j$ hence means the noisy threshold $\tilde{t} = i$ with the query result $r = j$. For instance, the state $t_0 r_1$ transits to $\tilde{t}_1 r_1$ with probability $\frac{3}{20}$. After the noisy threshold is obtained, we compute a noisy query result by the truncated $\frac{1}{2}$-geometric mechanism. The state $\tilde{t}_i \tilde{r}_j$ represents the noisy threshold $\tilde{t} = i$ and the noisy query result $\tilde{r} = j$. In the figure, we see that the state $\tilde{t}_1 r_0$ moves to $\tilde{t}_1 \tilde{r}_0$ with probability $\frac{2}{3}$. At the state $\tilde{t}_i \tilde{r}_j$, $\top$ is observed if $j \geq i$; otherwise, $\bot$ is observed. From the state $\tilde{t}_i \tilde{r}_j$, the model transits to the states $\tilde{t}_i r_0, \tilde{t}_i r_1, \tilde{t}_i r_2$ with uniform distribution. This simulates the next query result in Algorithm 3. The model then continues to process the next query.

Algorithm 3. Input: private database $\overline{\mathbf{d}}$, counting queries $f_i : \overline{\mathbf{d}} \to \{0, 1, 2\}$, threshold $t \in \{0, 1, 2\}$; Output: $a_1, a_2, \ldots$

procedure ABOVETHRESHOLD($\overline{\mathbf{d}}$, $\{f_1, f_2, \ldots\}$, t)
  match t **with** ▷ apply $\frac{1}{4}$-geometric mechanism
    case 0: $\tilde{t} \leftarrow 0, 1, 2$ with probability $\frac{4}{5}, \frac{3}{20}, \frac{1}{20}$
    case 1: $\tilde{t} \leftarrow 0, 1, 2$ with probability $\frac{1}{5}, \frac{3}{5}, \frac{1}{5}$
    case 2: $\tilde{t} \leftarrow 0, 1, 2$ with probability $\frac{1}{20}, \frac{3}{20}, \frac{4}{5}$
  for each query f_i **do**
    $r_i \leftarrow f_i(\overline{\mathbf{d}})$
    match r_i **with** ▷ apply $\frac{1}{2}$-geometric mechanism
      case 0: $\tilde{r}_i \leftarrow 0, 1, 2$ with probability $\frac{2}{3}, \frac{1}{6}, \frac{1}{6}$
      case 1: $\tilde{r}_i \leftarrow 0, 1, 2$ with probability $\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$
      case 2: $\tilde{r}_i \leftarrow 0, 1, 2$ with probability $\frac{1}{6}, \frac{1}{6}, \frac{2}{3}$
    if $\tilde{r}_i \geq \tilde{t}$ **then halt** with $a_i = \top$ **else** $a_i = \bot$

The bottom half of Fig. 3 is another copy of the model. All states in the second copy are underlined. For instance, the state $\underline{\tilde{t}_2 r_0}$ represents the noisy threshold is 2 and the query result is 0. Given an observation sequence, the two copies are used to simulate the mechanism conditioned on the prior knowledge with the two secrets. In the figure, we define the observation set $\Omega = \{\sqcup, \bot, \top, 00, 01, 10, 11, 12, 21, 22, \spadesuit, \heartsuit, \diamondsuit, \clubsuit\}$. At initial states $t_i r_j$ and $\underline{t_i r_j}$, only $\sqcup$ can be observed. When the noisy threshold is greater than the noisy query result ($\tilde{t}_i \tilde{r}_j$ and $\underline{\tilde{t}_i \tilde{r}_j}$ with $i > j$), $\bot$ is observed. Otherwise, $\top$ is observed at states $\tilde{t}_i \tilde{r}_j$ and $\underline{\tilde{t}_i \tilde{r}_j}$ with $i \leq j$. Other observations are used to "synchronize" query results for neighboring data sets. More details are explained in [25].

Differential Privacy Analysis. We can now perform differential privacy analysis using the HMM in Fig. 3. By construction, each observation corresponds to a sequence of

queries on neighboring data sets and their results. If the proof of continuous above threshold mechanism could carry over to our discretized mechanism, we would expect differences of observation probabilities from neighboring data sets to be bounded by the multiplicative factor of $e^{4\ln(2)} = 16$. Surprisingly, our tool always reports larger differences as the number of queries increases. After generalizing finite observations found by $Z3$, we obtain an observation sequence of an arbitrary length described below.

Fix $n > 0$. Consider a data set $\overline{\mathbf{d}}$ such that $f_i(\overline{\mathbf{d}}) = 1$ for $1 \leq i \leq n$ and $f_{n+1}(\overline{\mathbf{d}}) = 2$. A neighbor $\overline{\mathbf{d}}'$ of $\overline{\mathbf{d}}$ may have $f_i(\overline{\mathbf{d}}') = 2$ for $1 \leq i \leq n$ and $f_{n+1}(\overline{\mathbf{d}}') = 1$. Note that $|f_i(\overline{\mathbf{d}}) - f_i(\overline{\mathbf{d}}')| \leq 1$ for $1 \leq i \leq n+1$. f_i's are counting queries. Suppose the threshold $t = 2$. Let us compute the probabilities of observing $\bot^n\top$ on $\overline{\mathbf{d}}$ and $\overline{\mathbf{d}}'$.

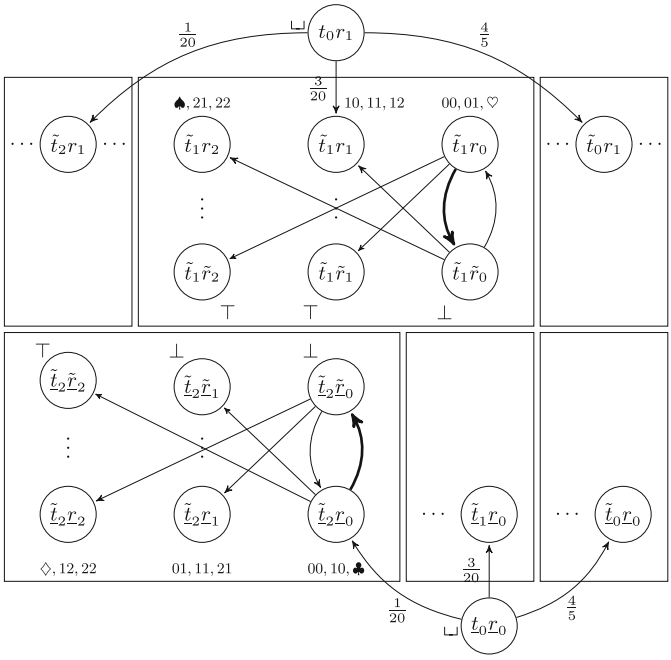


Fig. 3. Hidden Markov model for above threshold

If $\tilde{t} = 0$, $\tilde{f}_1 \geq \tilde{t}$. The algorithm reports $\top$ and stops. We cannot observe $\bot^n\top$: recall the assumption that $n > 0$. It suffices to consider $\tilde{t} = 1$ or 2. When $\tilde{t} = 1$, $\tilde{f}_i(\overline{\mathbf{d}}) = 0$ for $1 \leq i \leq n$ and $\tilde{f}_{n+1}(\overline{\mathbf{d}}) \geq 1$. Recall $f_i(\overline{\mathbf{d}}) = 1$ for $1 \leq i \leq n$ and $f_{n+1}(\overline{\mathbf{d}}) = 2$. The probability of observing $\bot^n\top$ is $(\frac{1}{3})^n \cdot \frac{5}{6}$. When $\tilde{t} = 2$, $\tilde{f}_1(\overline{\mathbf{d}}) \leq 1$ for $1 \leq i \leq n$ and $\tilde{f}_{n+1}(\overline{\mathbf{d}}) = 2$. The probability of observing $\bot^n\top$ is thus $(\frac{2}{3})^n \cdot \frac{2}{3}$. In summary, the probability of observing $\bot^n\top$ with $\overline{\mathbf{d}}$ when $t = 2$ is $\frac{3}{20} \cdot (\frac{1}{3})^n \cdot \frac{5}{6} + \frac{4}{5} \cdot (\frac{2}{3})^n \cdot \frac{2}{3}$. The case for $\overline{\mathbf{d}}'$ is similar. When $\tilde{t} = 1$, the probability of observing $\bot^n\top$ is $(\frac{1}{6})^n \cdot \frac{2}{3}$. When $\tilde{t} = 2$, the probability of observing the same sequence is $(\frac{1}{3})^n \cdot \frac{1}{3}$. Hence the probability of observing $\bot^n\top$ with $\overline{\mathbf{d}}'$ when $t = 2$ is $\frac{3}{20} \cdot (\frac{1}{6})^n \cdot \frac{2}{3} + \frac{4}{5} \cdot (\frac{1}{3})^n \cdot \frac{1}{3}$. Now,

$$\frac{\Pr(\omega = \bot^n\top|\overline{\mathbf{d}}, t=2)}{\Pr(\omega = \bot^n\top|\overline{\mathbf{d}}', t=2)} = \frac{\frac{3}{20}\cdot(\frac{1}{3})^n\cdot\frac{5}{6} + \frac{4}{5}\cdot(\frac{2}{3})^n\cdot\frac{2}{3}}{\frac{3}{20}\cdot(\frac{1}{6})^n\cdot\frac{2}{3} + \frac{4}{5}\cdot(\frac{1}{3})^n\cdot\frac{1}{3}}$$
$$> \frac{\frac{4}{5}\cdot(\frac{2}{3})^n\cdot\frac{2}{3}}{\frac{3}{20}\cdot(\frac{1}{3})^n\cdot\frac{2}{3} + \frac{4}{5}\cdot(\frac{1}{3})^n\cdot\frac{1}{3}} = \frac{\frac{8}{15}(\frac{2}{3})^n}{\frac{11}{30}(\frac{1}{3})^n} = \frac{16}{11}\cdot 2^n.$$

We see that the ratio of $\Pr(\omega = \bot^n\top|\overline{\mathbf{d}}, t=2)$ and $\Pr(\omega = \bot^n\top|\overline{\mathbf{d}}', t=2)$ can be arbitrarily large. Unexpectedly, the discrete above threshold cannot be ϵ-differentially private for any ϵ. Replacing continuous noises with truncated discrete noises does not preserve any privacy at all. This case emphasizes the importance of applying verification technique to practical implementations.

7 Combining Techniques for Differential Privacy

In this section, we investigate into two state-of-the-art tools for detecting violations of differential privacy, namely StatDP [14] and DP-Sniper [10], to compare with our tool. We decide to choose these tools as baselines since they support programs with arbitrary loops and arbitrary sampling distributions. On the contrary, DiPC [3,4], DP-Finder [9] and CheckDP [32] et al. do not support arbitrary loops or only synthesize proofs for privacy budget ϵ when Laplace distributions are applied. In order to compare with our tool FAIER, the discrete mechanisms with truncated geometric distributions are implemented in these tools. We present comparisons in Subsect. 7.1, and moreover, in Subsect. 7.2, we discuss how testing and our verification technique can be combined to certify counterexamples and find the precise lower bound for privacy budget.

7.1 Comparison

Different Problem Statements. As all the tools can be used to find the privacy budget ϵ for differential private mechanisms, the problem statements they address are different: I. With a fixed value of ϵ, StatDP runs the mechanism repeatedly and tries to report the output event that makes the mechanism violate ϵ-differential privacy, with a p-value as the confidence level. If the p-value is below 0.05, StatDP is of high confidence that ϵ-differential privacy is violated; Otherwise the mechanism is very likely (depending on the p-value) to satisfy. II. On the other hand, DP-Sniper aims to learn for the optimal attack that maximizes the ratio of probabilities for certain outputs on all the neighboring inputs. Therefore it returns the corresponding "optimal" witness (neighboring inputs) along with a value ϵ such that the counterexample violates ϵ-differential privacy with ϵ as large as possible. III. Differently, FAIER makes use of the HMM model and examines all the pairs of neighboring inputs and outputs to make sure that ϵ-differential privacy is satisfied by all cases, or violated by an counterexample, with a fixed value of ϵ. IV. Note that FAIER is aimed at Pufferfish privacy verification where prior knowledge can affect the data sets distributions and unknown parameters are allowed, which are not involved in the other tools. Meanwhile, the others support continuous noise while FAIER does not (unless an HMM with finite state space can be obtained).

Efficiency and Precision. We make comparison of the tools in terms of efficiency and precision by performing experiments on Discrete Noisy Max (Algorithm 2) with $n = 5$ queries. The lower bound [9] of the privacy budget, i.e., the largest ϵ that the mechanism is not ϵ-differential privacy, is 1.372 up to a precision of 0.001. I. Fix an ϵ, StatDP takes 8 s on average to report an event 0 along with a p-value under the usual setting of $100k/500k$ times for event selection/counterexample detection. However, there is a need for specifying the range of ϵ in advance and more values of ϵ to test will consume more time. We first select ϵ increasingly with a step of 0.1 in the range of [0, 2]. Then the range is narrowed down according to the p-values and we select ϵ in the range with a smaller step 0.01 and so on. The similar process also applies for FAIER. Altogether StatDP takes around 600 s to get an overview of the results. Fast enough, though, it has the drawback of instability and the precision is lower than the other tools. It reports the mechanism satisfies 1.344-differential privacy in the first execution, which is incorrect, and reports it violates 1.353-differential privacy in the second execution.

Table 3. Heuristic input patterns used in StatDP and DP-Sniper, from [14]

Category	D_1	D_2
One Above	[1, 1, 1, 1, 1]	[2, 1, 1, 1, 1]
One Below	[1, 1, 1, 1, 1]	[0, 1, 1, 1, 1]
One Above Rest Below	[1, 1, 1, 1, 1]	[2, 0, 0, 0, 0]
One Below Rest Above	[1, 1, 1, 1, 1]	[0, 2, 2, 2, 2]
Half Half	[1, 1, 1, 1, 1]	[0, 0, 2, 2, 2]
All Above & All Below	[1, 1, 1, 1, 1]	[2, 2, 2, 2, 2]
X Shape	[1, 1, 0, 0, 0]	[0, 0, 1, 1, 1]

II. DP-Sniper returns a witness [0, 2, 2, 2, 2] and [1, 1, 1, 1, 1] with $\epsilon = 1.371$ for three times, which is correct, stable and the result is almost the true lower bound. However, it takes around 4600 s on average to train a multi-layer perceptron with $10000k$ samples and get this result. Unlike the evaluation in [10], DP-Sniper performs much slower than StatDP when it comes to discrete random noise. The reason is that DP-Sniper cannot use high-efficient sampling commands such as numpy.random.laplace to get all the samples at once. It has to calculate and sample different distributions according to different inputs. We've tried to use numpy.random.choice to sample different distributions, but it is inefficient for small vectors and wouldn't terminate for more than 10 h in our experiment. We've also tried to reduce the number of samples to $1000k$. This time it terminates with 308 s with an imprecise $\epsilon = 1.350$.

III. FAIER takes less than 1 s to build the HMM model and 160 s to compute SMT query for every data set (possible initial state), which will be later used to compute on neighboring data sets if an ϵ is assigned. The results returned by FAIER are the most precise ones. It takes Z3 523 s to verify that 1.373-differential privacy is satisfied and 234 s that 1.372-differential privacy is violated witnessed by the input pair [0, 2, 2, 2, 2] and [1, 1, 1, 1, 1] and output event 1. It takes only 40 s to verify when $\epsilon = 1.34$, a little far away from the true lower bound. Altogether it takes around 1600 s to assure the true bound, which is acceptable.

7.2 Combining Verification and Testing

The findings during experiments inspire us to combine verification (FAIER) and testing (DP-Sniper, StatDP) together to efficiently make use of each tool. First, we can see that the witnesses found by FAIER and DP-Sniper are the same one. Actually, if heuristic

searching strategies for input pairs are used, i.e., Table 3 used in DP-Sniper and StatDP, FAIER will quickly find the violation pairs, which saves huge time in the occasions of privacy violations. Second, since the witness returned by DP-Sniper is the optimal input pair that maximize the probability difference, FAIER can precisely verify whether the "optimal" witness satisfies ϵ-differential privacy, whereby FAIER will more likely to find the true lower bound as ϵ increase in short time. Third, since StatDP returns an imprecise result quickly given an ϵ, we can combine StatDP and FAIER to efficiently get a precise lower bound. The pseudo-code is in Algorithm 4.

Algorithm 4 first feeds mechanism M as input to the testing tool StatDP, to obtain an interval I whose left end point is ϵ with p-value <0.05 and right end point with p-value $= 1$. StatDP can conclude if p-value<0.05, the mechanism doesn't satisfy ϵ-differential privacy with high confidence and if p-value$= 1$, the mechanism satisfies for sure. However, for other p-values, StatDP is not confident to give useful conclusions. Here is where our tool can work out—FAIER can determine whether M satisfies ϵ-differential privacy, given any ϵ. As a result we can combine to efficiently get arbitrarily close to the lower bound ϵ wrt. a given precision by binary search. For instance, we apply StatDP on Algorithm 2 to get an interval $I = [1.34, 1.38]$ according to the p-value graph, and then apply our tool FAIER to verify ϵ-differential privacy. Consequently, our tool reports the lower bound is 1.372 (up to a precision of 0.001).

Algorithm 4. Pseudo-code to compute the lower bound

```
1:  procedure COMPUTE LOWER BOUND(Mechanism M)
2:      Use StatDP with input M to get an interval I        ▷ the left end point is an ϵ with
    p-value< 0.05 and the right one is one with p-value= 1
3:      Apply binary search on I, in each iteration the value is ϵ
4:      repeat
5:          Use FAIER with input M and ϵ
6:          if result is SAT then                           ▷ not satisfy ϵ-differential privacy
7:              left end point = ϵ
8:          else                                            ▷ satisfy ϵ-differential privacy
9:              right end point = ϵ
10:     until reaching required precision
11:     return ϵ
```

8 Related Work

Methods of proving/testing differential privacy. Barthe et al. [7,8] proposed to prove differential privacy at the beginning. Then a number of work [1,5,6] extended probabilistic relational Hoare logic and applied approximate probabilistic couplings between programs on adjacent inputs. They successfully proved differential privacy for several algorithms, but cannot disprove privacy. Zhang et al. [32–34] proposed to apply randomness alignment to evaluate privacy cost and implemented CheckDP that could

rewrite classic privacy mechanisms involving Laplacian noise to verify differential privacy. Bichsel et al. [9], Ding et al. [14] and Zhang et al. [35] used testing and searching to find violations for differential privacy mechanisms, the results of which may be too coarse or imprecise. Liu et al. [24] chose Markov chains and Markov decision processes to model deferentially private mechanisms and verify privacy properties in extended probabilistic temporal logic. McIver et al. [27] applied Quantitative Information Flow to analyze Randomized response mechanism in differential privacy. We note that all the automated tools above for proving or testing differential privacy, plus ours, have not been well studied in privacy mechanisms with considerably large data sets.

Complexity in Verifying Differential Privacy. Gaboardi et al. [19] studied the problem of verifying differential privacy for probabilistic loop-free programs. They showed that to decide ϵ-differential privacy is $\mathbf{coNP}^{\#\mathbf{P}}$-complete and to approximate the level of differential privacy is both $\mathbf{NP}$-hard and $\mathbf{coNP}$-hard. Barthe et al. [3] first proved that checking differential privacy is undecidable. The difference with our work lies in that we study verification problems for mechanisms modeled in HMMs in Pufferfish privacy. Chistikov et al. [12] proved that the big-O problem for labeled Markov chains (LMCs) is undecidable, which is similar to deciding the ratio of two probabilities in differential privacy. Though, their proof does not apply here since HMMs in our paper do not have the same non-deterministic power as LMCs.

Acknowledgements. We would like to thank the anonymous reviewers for their valuable suggestions and comments about this paper. The work is supported by Ministry of Science and Technology of Taiwan under the Grant Number 108-2221-E-001-010-MY3; the Data Safety and Talent Cultivation Project AS-KPQ-109-DSTCP and NSFC under the Grant Number 61836005.

References

1. Albarghouthi, A., Hsu, J.: Synthesizing coupling proofs of differential privacy. Proc. ACM Program. Lang. **2**(POPL), 1–30 (2017)
2. Apple: About privacy and location services in IOS and IPADOS (2020). https://support.apple.com/en-us/HT203033/. Accessed 9 Sept 2021
3. Barthe, G., Chadha, R., Jagannath, V., Sistla, A., Viswanathan, M.: Deciding differential privacy for programs with finite inputs and outputs, pp. 141–154 (2020). https://doi.org/10.1145/3373718.3394796
4. Barthe, G., Chadha, R., Jagannath, V., Sistla, A.P., Viswanathan, M.: Automated methods for checking differential privacy. CoRR abs/1910.04137 (2019). http://arxiv.org/abs/1910.04137
5. Barthe, G., Gaboardi, M., Grégoire, B., Hsu, J., Strub, P.Y.: Proving differential privacy via probabilistic couplings (2016)
6. Barthe, G., Gaboardi, M., Hsu, J., Pierce, B.: Programming language techniques for differential privacy. ACM SIGLOG News **3**(1), 34–53 (2016). https://doi.org/10.1145/2893582.2893591
7. Barthe, G., Köpf, B., Olmedo, F., Zanella Béguelin, S.: Probabilistic relational reasoning for differential privacy. In: POPL '12, pp. 97–110 (2012). https://doi.org/10.1145/2103656.2103670
8. Barthe, G., Köpf, B., Olmedo, F., Zanella Béguelin, S.: Probabilistic relational reasoning for differential privacy. SIGPLAN Not. **47**(1), 97–110 (2012). https://doi.org/10.1145/2103621.2103670

9. Bichsel, B., Gehr, T., Drachsler-Cohen, D., Tsankov, P., Vechev, M.: Dp-finder: finding differential privacy violations by sampling and optimization, pp. 508–524 (2018). https://doi.org/10.1145/3243734.3243863
10. Bichsel, B., Steffen, S., Bogunovic, I., Vechev, M.: DP-sniper: black-box discovery of differential privacy violations using classifiers. In: SP'21, pp. 391–409 (2021). https://doi.org/10.1109/SP40001.2021.00081
11. Chen, Y., Machanavajjhala, A.: On the privacy properties of variants on the sparse vector technique. CoRR abs/1508.07306 (2015). http://arxiv.org/abs/1508.07306
12. Chistikov, D., Kiefer, S., Murawski, A.S., Purser, D.: The Big-O problem for labelled markov chains and weighted automata. In: CONCUR 2020. Leibniz International Proceedings in Informatics (LIPIcs), vol. 171, pp. 41:1–41:19 (2020). https://doi.org/10.4230/LIPIcs.CONCUR.2020.41
13. Ding, B., Kulkarni, J., Yekhanin, S.: Collecting telemetry data privately. In: NIPS'17, pp. 3574–3583 (2017)
14. Ding, Z., Wang, Y., Wang, G., Zhang, D., Kifer, D.: Detecting violations of differential privacy. In: Backes, M., Wang, X. (eds.) CCS, pp. 475–489 (2018)
15. Dolzmann, A., Sturm, T.: REDLOG: computer algebra meets computer logic. SIGSAM Bull. **31**(2), 2–9 (1997). https://doi.org/10.1145/261320.261324
16. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theoret. Comput. Sci. **9**(3–4), 211–407 (2014)
17. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006). https://doi.org/10.1007/11787006_1
18. Farina, G.P., Chong, S., Gaboardi, M.: Coupled relational symbolic execution for differential privacy. In: Programming Languages and Systems, pp. 207–233 (2021)
19. Gaboardi, M., Nissim, K., Purser, D.: The complexity of verifying loop-free programs as differentially private. In: ICALP 2020. Leibniz International Proceedings in Informatics (LIPIcs), vol. 168, pp. 129:1–129:17 (2020). https://doi.org/10.4230/LIPIcs.ICALP.2020.129
20. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. In: STOC, pp. 351–360. ACM, New York (2009)
21. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. SIAM J. Comput. **41**(6), 1673–1693 (2012)
22. Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: SIGMOD, pp. 193–204 (2011)
23. Kifer, D., Machanavajjhala, A.: Pufferfish: a framework for mathematical privacy definitions. ACM Trans. Database Syst. **39**(1), 3:1–3:36 (2014)
24. Liu, D., Wang, B.-Y., Zhang, L.: Model checking differentially private properties. In: Ryu, S. (ed.) APLAS 2018. LNCS, vol. 11275, pp. 394–414. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02768-1_21
25. Liu, D., Wang, B., Zhang, L.: Verifying pufferfish privacy in hidden Markov models. CoRR abs/2008.01704 (2020). https://arxiv.org/abs/2008.01704
26. Lyu, M., Su, D., Li, N.: Understanding the sparse vector technique for differential privacy. Proc. VLDB Endow. **10**(6), 637–648 (2017). https://doi.org/10.14778/3055330.3055331
27. McIver, A., Morgan, C.: Proving that programs are differentially private. In: Lin, A.W. (ed.) APLAS 2019. LNCS, vol. 11893, pp. 3–18. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34175-6_1
28. Mironov, I.: On significance of the least significant bits for differential privacy. In: CCS'12, pp. 650–661 (2012)

29. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
30. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. Math. Oper. Res. **12**(3), 441–450 (1987)
31. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)
32. Wang, Y., Ding, Z., Kifer, D., Zhang, D.: CheckDP: an automated and integrated approach for proving differential privacy or finding precise counterexamples. In: CCS '20, pp. 919–938 (2020). https://doi.org/10.1145/3372297.3417282
33. Wang, Y., Ding, Z., Wang, G., Kifer, D., Zhang, D.: Proving differential privacy with shadow execution. In: PLDI'19, pp. 655–669 (2019)
34. Zhang, D., Kifer, D.: LightDP: towards automating differential privacy proofs. In: POPL'17, vol. 52, pp. 888–901 (2017)
35. Zhang, H., Roth, E., Haeberlen, A., Pierce, B.C., Roth, A.: Testing differential privacy with dual interpreters. Proc. ACM Program. Lang. **4**(OOPSLA) (2020). https://doi.org/10.1145/3428233