# Differentially Private Reward Functions for Markov Decision Processes

Alexander Benvenuti[1], Calvin Hawkins[2], Brandon Fallin[2], Bo Chen[2],
Brendan Bialy[3], Miriam Dennis[3], Matthew Hale[1]

***Abstract*— Markov decision processes often seek to maximize a reward function, but onlookers may infer reward functions by observing agents, which can reveal sensitive information. Therefore, in this paper we introduce and compare two methods for privatizing reward functions in policy synthesis for multi-agent Markov decision processes, which generalize Markov decision processes. Reward functions are privatized using differential privacy, a statistical framework for protecting sensitive data. The methods we develop perturb (i) each agent's individual reward function or (ii) the joint reward function shared by all agents. We prove that both of these methods are differentially private and show approach (i) provides better performance. We then develop an algorithm for the numerical computation of the performance loss due to privacy on a case-by-case basis. We also exactly compute the computational complexity of this algorithm in terms of system parameters and show that it is inherently tractable. Numerical simulations are performed on waypoint guidance of an autonomous vehicle which shows that privacy induces only negligible performance losses in practice.**

## I. Introduction

Many autonomous systems share sensitive information to operate, such as teams of robots. As a result, there has arisen interest in privatizing such information when it is communicated. However, these protections can be difficult to provide in systems in which agents are inherently observable, such as in a traffic system in which agents are visible to other vehicles [1] or a power system in which power usage is visible to a utility company [2]. These systems do not offer the opportunity to modify communications to protect information precisely because agents are observed directly. A fundamental challenge is that information about agents is visible to observers, though we would still like to limit the inferences that can be drawn from that information.

In this paper, we consider the problem of protecting the reward function of a Markov decision process, even when its states and actions can be observed. In particular, we model individual agents as Markov Decision Processes (MDPs), and we model collections of agents as Multi-Agent Markov Decision Processes (MMDPs). Given that an MDP is simply an MMDP with a single agent, we focus on MMDPs. In MMDPs, agents' goal is to synthesize a reward-maximizing policy. Accordingly, we develop a method for synthesizing policies that preserve the privacy of the agents' reward function while still approximately maximizing it.

Since these agents can be observed, the actions they take could reveal their reward function or some of its properties, which may be sensitive information. For example, quantitative methods can draw such inferences from agents' trajectories both offline [3], [4] and online [5]. Additionally onlookers may draw qualitative inferences from agents as well, such as their goal state [6]. This past work shows that harmful quantitative and qualitative inferences can be drawn without needing to recover the entire reward function. Therefore, we seek to protect agents' reward functions from both existing privacy attacks and those yet to be developed.

We use differential privacy to provide these protections. Differential privacy is a statistical notion of privacy originally used to protect entries of databases [7]. Differential privacy has been used recently in control systems and filtering [8]–[12], and to privatize objective functions in distributed optimization [13]–[18]. The literature on distributed optimization has already established that agents' objective functions require privacy, and a key focus has been convex minimization problems. In this paper, we consider agents maximizing reward functions and hence are different from that work, though the need to protect individual agents' rewards is just as essential as the protection of objectives in those works.

Differential privacy is appealing because of its strong protections for data and its immunity to post-processing [19]. That is, the outputs of arbitrary computations on differentially private data are still protected by differential privacy. This property provides protections against observers that make inferences about agents' rewards, including through techniques that do not exist yet. We therefore first privatize reward functions, then using dynamic programming to synthesize a policy with the privatized reward function. Since this dynamic programming stage is post-processing on privatized reward functions, the resulting policy preserves the privacy of the reward functions as well, as do observations of an agent executing that policy and any downstream inferences that rely on those observations.

Of course, we expect perturbations to reward functions to affect performance. To assess the impact of privacy on the agents' performance we quantify the sub-optimality of the policy synthesized with the privatized reward function. In particular, it relates the total discounted reward (known

[1]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA USA. Emails: {abenvenuti3,matthale}@gatech.edu.

[2]Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL USA. Emails: {calvin.hawkins,brandonfallin,bo.chen}@ufl.edu.

[3]Munitions Directorate, Air Force Research Laboratory, Eglin AFB, FL USA. Emails: {brendan.bialy,miriam.dennis.1}@us.af.mil.
This work was partially supported by NSF under CAREER grant #1943275, AFRL under grant #FA8651-23-F-A008, ONR under #N00014-21-1-2502, and AFOSR under grant #FA9550-19-1-0169. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of sponsoring agencies.

as the value function) with privacy to that without privacy. We develop an algorithm to compute the cost of privacy, and we compute the exact computational complexity of this algorithm in terms of system parameters. These calculations show the tractability of this algorithm.

To summarize, we make the following contributions:

- We develop two differential privacy mechanisms for reward functions in MMDPs. (Theorems 1 and 2)
- We provide an analytical bound on the accuracy of privatized reward functions and use that bound to trade-off privacy and accuracy. (Theorem 3)
- We provide an algorithm to compute the trade-off between privacy and performance, then quantify its computational complexity. (Theorem 4)
- We validate the impact of privacy upon performance in simulation.

*Related Work*

Privacy has previously been considered for Markov decision processes, both in planning [20]–[22] and reinforcement learning [23], [24]. Privacy has also been considered for Markov chains [25] and for general symbolic systems [26]. The closest work to ours is in [20] and [27]. In [20], privacy is applied to transition probabilities, while we apply it to reward functions. In [27], the authors use differential privacy to protect rewards that are learned. We differ since we consider planning problems with a known reward.

This paper is organized as follows: Section II provides background and problem statements, and Section III presents two methods for privatizing reward functions. Then, Section IV formalizes accuracy-privacy trade-offs, and Section V presents a method of computing the cost of privacy. Section VI provides simulations, and Section VII concludes.

*Notation*

For $N \in \mathbb{N}$, we use $[N]$ to denote $\{1, 2, \dots, N\}$, we use $\Delta(B)$ to be the set of probability distributions over a finite set $B$, and we use $|\cdot|$ for the cardinality of a set. We also use $\lceil \cdot \rceil$ as the ceiling function. We use $\pi$ both as the usual constant and as a policy for an MDP since both uses are standard. The meaning will be clear from context.

## II. Preliminaries and Problem Formulation

This section reviews Markov decision processes and differential privacy, then provides formal problem statements.

### A. Markov Decision Processes

Consider a collection of $N$ agents indexed over $i \in [N]$. We model agent $i$ as a Markov decision process.

**Definition 1** (Markov Decision Process)**.** Agent $i$'s Markov Decision Process (MDP) is the tuple $\mathcal{M}^i = (\mathcal{S}^i, \mathcal{A}^i, r^i, \mathcal{T}^i)$, where $\mathcal{S}^i$ and $\mathcal{A}^i$ are agent $i$'s finite sets of local states and actions, respectively. Additionally, let $\mathcal{S} = \mathcal{S}^1 \times \cdots \times \mathcal{S}^N$ be the joint state space of all agents. Then $r^i : \mathcal{S} \times \mathcal{A}^i \to \mathbb{R}$ is agent $i$'s reward function, and $\mathcal{T}^i : \mathcal{S}^i \times \mathcal{A}^i \to \Delta(\mathcal{S}^i)$ is agent $i$'s transition probability function. $\lozenge$

With $\mathcal{T}^i : \mathcal{S}^i \times \mathcal{A}^i \to \Delta(\mathcal{S}^i)$, we see that $\mathcal{T}^i(s^i, a^i) \in \Delta(\mathcal{S}^i)$ is a probability distribution over the possible next states when taking action $a^i \in \mathcal{A}^i$ in state $s^i \in \mathcal{S}^i$. We abuse notation and let $\mathcal{T}^i(s^i, a^i, y^i)$ be the probability of transitioning from state $s^i$ to state $y^i \in \mathcal{S}^i$ when agent $i$ takes action $a^i \in \mathcal{A}^i$. We now model the collection of agents as a Multi-Agent Markov Decision Process (MMDP).

**Definition 2** (Multi-Agent Markov Decision Process [28])**.** A Multi-Agent Markov Decision Process (MMDP) is the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \gamma, \mathcal{T})$, where $\mathcal{S} = \mathcal{S}^1 \times \cdots \times \mathcal{S}^N$ is the joint state space, $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$ is the joint action space, $r(s, a) = \frac{1}{N} \sum_{i \in [N]} r^i(s, a^i)$ is the joint reward function value for joint state $s = (s^1, \dots, s^N) \in \mathcal{S}$ and joint action $a = (a^1, \dots, a^N) \in \mathcal{A}$, the constant $\gamma \in (0, 1]$ is the discount factor, and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the joint transition probability distribution. That is, $\mathcal{T}(s, a, y) = \prod_{i=1}^{N} \mathcal{T}^i(s^i, a^i, y^i)$ denotes the probability of transitioning from joint state $s$ to joint state $y = (y^1, \dots, y^N) \in \mathcal{S}$ given joint action $a$, for all $s,\ y \in \mathcal{S}$ and $a \in \mathcal{A}$. $\lozenge$

Given a joint action $a_j \in \mathcal{A}$, agent $i$ takes the local action $a^i_{I_j(i)} \in \mathcal{A}^i$, where we use $I_j(i)$ to denote the index of agent $i$'s local action corresponding to joint action $j$. That is, for some action $a_j \in \mathcal{A}$ we have $a_j = \left(a^1_{I_j(1)}, a^2_{I_j(2)}, \dots, a^N_{I_j(N)}\right)$. For $r(s_j, a_k) = \frac{1}{N} \sum_{i \in [N]} r^i(s_j, a^i_{I_k(i)})$ we define the mapping $J$ such that

$$r(s_j, a_k) = J(\{r^i(s_j, a^i_{I_k(i)})\}_{i \in [N]}). \tag{1}$$

A joint policy $\pi : \mathcal{S} \to \mathcal{A}$, represented as $\pi = (\pi^1, \dots, \pi^N)$, where $\pi^i : \mathcal{S} \to \mathcal{A}^i$ is agent $i$'s policy for all $i \in [N]$, is a set of policies which commands agent $i$ to take action $\pi^i(s)$ in joint state $s$. The control objective for an MMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \gamma, \mathcal{T})$ then is: given a joint reward function $r$, develop a joint policy that solves

$$\max_{\pi} V_\pi(s) = \max_{\pi} \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t))\Big], \tag{2}$$

where we call $V_\pi$ the "value function".

Often, it is necessary to evaluate how well a non-optimal policy performs on a given MMDP. Accordingly, we state the following proposition that the Bellman operator is a contraction mapping, which we will use in Section V to evaluate any policy on a given MMDP.

**Proposition 1** (Policy Evaluation [29])**.** Fix an MMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \gamma, \mathcal{T})$. Let $V(s')$ be the value function at state $s'$, and let $\pi$ be a joint policy. Define the Bellman operator $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^n$ as $\mathcal{L}V := \sum_{a \in \mathcal{A}} \pi(s) \left(r(s, a) + \sum_{s' \in \mathcal{S}} \gamma \mathcal{T}(s, a, s') V(s')\right)$. Then $\mathcal{L}$ is a $\gamma$-contraction mapping with respect to the $\infty$-norm. That is, $\|\mathcal{L}V_1 - \mathcal{L}V_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty$ for all $V_1, V_2 \in \mathbb{R}^n$.

Solving an MMDP is $P$-Complete and is done efficiently via dynamic programming [29], which we use in this paper.

### B. Differential Privacy

We now describe the application of differential privacy to vector-valued data and this will be applied to agents' rewards

represented as vectors. The goal of differential privacy is to make "similar" pieces of data appear approximately indistinguishable. The notion of "similar" is defined by an adjacency relation. Many adjacency relations exist, and we present the one used this paper; see [19] for additional background.

**Definition 3** (Adjacency)**.** Fix an adjacency parameter $b > 0$ and two vectors $v, w \in \mathbb{R}^n$. Then $v$ and $w$ are adjacent if the following conditions hold: (i) There exists some $j \in [n]$ such that $v_j \neq w_j$ and $v_k = w_k$ for all $k \in [n] \setminus \{j\}$ and (ii) $\|v - w\|_1 \leq b$, where $\|\cdot\|_1$ denotes vector 1-norm. We use $\mathrm{Adj}_b(v, w) = 1$ to say $v$ and $w$ are adjacent. $\Diamond$

Differential privacy is enforced by a randomized map called a "mechanism." For a function $f : \mathbb{R}^n \to \mathbb{R}^m$, a mechanism $\mathscr{M}$ approximates $f(x)$ for all inputs $x$ according to the following definition.

**Definition 4** (Differential Privacy [19])**.** Fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Let $b > 0$, $\epsilon > 0$, and $\delta \in [0, \frac{1}{2})$ be given. A mechanism $\mathscr{M} : \mathbb{R}^n \times \Omega \to \mathbb{R}^m$ is $(\epsilon, \delta)$-differentially private if for all $x, x' \in \mathbb{R}^n$ adjacent in the sense of Definition 3, and for all measurable sets $S \subseteq \mathbb{R}^m$, we have $\mathbb{P}[\mathscr{M}(x) \in S] \leq e^{\epsilon}\mathbb{P}[\mathscr{M}(x') \in S] + \delta$. $\Diamond$

In Definition 4, the strength of privacy is controlled by the privacy parameters $\epsilon$ and $\delta$. In general, smaller values of $\epsilon$ and $\delta$ imply stronger privacy guarantees. Here, $\epsilon$ can be interpreted as quantifying the leakage of sensitive information and $\delta$ can be interpreted as the probability that differential privacy leaks more information than allowed by $\epsilon$. Typical values of $\epsilon$ and $\delta$ are $0.1$ to $3$ [30] and $0$ to $0.05$ [31], respectively. Differential privacy is calibrated using the "sensitivity" of the function being privatized.

**Definition 5** (Sensitivity)**.** Fix an adjacency parameter $b > 0$. The $\ell_2$-sensitivity of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is $\Delta_2 f = \sup_{x,x':\mathrm{Adj}_b(x,x')=1} \|f(x) - f(x')\|_2$. $\Diamond$

A larger sensitivity implies that higher variance noise is needed to mask differences in adjacent data when generating private outputs. We next define the Gaussian mechanism for enforcing differential privacy.

**Lemma 1** (Gaussian Mechanism; [7])**.** Let $b > 0$, $\epsilon > 0$, and $\delta \in [0, 1/2)$ be given, and fix the adjacency relation from Definition 3. The Gaussian mechanism takes sensitive data $f(x) \in \mathbb{R}^m$ as an input and outputs $\mathcal{G}(x) = f(x) + z$, where $z \sim \mathcal{N}(0, \sigma^2 I)$. It is $(\epsilon, \delta)$-differentially private if $\sigma \geq \frac{\Delta_2 f}{2\epsilon}\kappa(\epsilon, \delta)$, where $\kappa(\epsilon, \delta) = (\mathcal{Q}^{-1}(\delta) + \sqrt{\mathcal{Q}^{-1}(\delta)^2 + 2\epsilon})$, with $\mathcal{Q}(y) = \frac{1}{2\pi}\int_y^{\infty} e^{-\frac{\theta^2}{2}} d\theta$.

In this work, we sometimes consider the identity query $f(x) = x$, which has $\Delta_2 f = b$, where $b$ is from Definition 3.

**Lemma 2** (Immunity to Post-Processing; [19])**.** Let $\mathscr{M} : \mathbb{R}^n \times \Omega \to \mathbb{R}^m$ be an $(\epsilon, \delta)$-differentially private mechanism. Let $h : \mathbb{R}^m \to \mathbb{R}^p$ be an arbitrary mapping. Then the composition $h \circ \mathscr{M} : \mathbb{R}^n \to \mathbb{R}^p$ is $(\epsilon, \delta)$-differentially private.

This lemma implies that we can first privatize rewards and then compute a decision policy from those privatized rewards, and the execution of that policy also keeps the rewards differentially private because it is post-processing.

### C. Problem Statements

Consider the policy synthesis problem in (2). Computing $\pi^*$ depends on the sensitive reward function $r^i$ for all $i \in [N]$. An adversary observing agents execute $\pi^*$ may attempt to infer $r^i$ itself or its properties. Therefore, we seek a framework for multi-agent policy synthesis that preserves the privacy of $r^i$ while still performing well. This will be done by solving the following problems.

**Problem 1.** *Develop privacy mechanisms to privatize individual agents' reward functions in MMDPs.*

**Problem 2.** *Develop bounds on the accuracy of privatized rewards that will be used to trade off privacy and accuracy.*

**Problem 3.** *Determine the computational complexity of evaluating the loss in utility from using a policy generated on the privatized rewards.*

## III. Private Policy Synthesis

In this section, we solve Problem 1. First, we illustrate how we represent reward functions to apply differential privacy. Then, we present two mechanisms for applying privacy to our representation of reward functions. Let $n_i = |\mathcal{S}^i|$ and $m_i = |\mathcal{A}^i|$ be the numbers of local states and local actions, respectively, for agent $i$. The joint state and action spaces then have $n = \prod_{i\in[N]} n_i$ states and $m = \prod_{i\in[N]} m_i$ actions.

### A. Privacy Setup

To use Lemma 1 to enforce differential privacy, we first express the reward function as a vector. We represent the mapping $r^i$ as a vector $R^i \in \mathbb{R}^{nm_i}$, where the entries correspond to $r^i$ being evaluated on all inputs. We define $R^i$ as the vector with entries $r^i(s, a^i)$ for all $s \in \mathcal{S}$ and $a^i \in \mathcal{A}^i$.

We use the following convention for representing $R^i$. Denote the joint states by $s_1, s_2, \ldots, s_n$ and denote the local actions by $a^i_1, a^i_2, \ldots, a^i_{m_i}$. Then we set $R^i = [r^i(s_1, a^i_1), \ldots, r^i(s_1, a^i_{m_i}), r^i(s_2, a^i_1), \ldots, r^i(s_n, a^i_{m_i})]^T$, where $R^i_j$ denotes the $j^{th}$ entry of the vector $R^i$. This can be repeated to represent the joint reward $R \in \mathbb{R}^{nm}$. Using Definition 3, we say two reward functions belonging to agent $i$, denoted $R^i$ and $\hat{R}^i$, are adjacent if they differ in one entry, with their absolute difference bounded above by $b$.

We note that the solution to Problem 1 is not unique, and we consider two means of enforcing privacy. The two setups we consider differ in where privacy is implemented in them. First, we apply the Gaussian mechanism to agent $i$'s list of rewards, $R^i$, referred to as "input perturbation". In input perturbation, the aggregator receives privatized rewards, denoted $\tilde{R}^i$ from agent $i$ for all $i \in [N]$, and the aggregator uses those to generate a policy $\tilde{\pi}^*$ for the MMDP. In the second setup, we instead apply the Gaussian mechanism to the list of joint rewards, $R$, referred to as "output perturbation". In output perturbation, the aggregator receives sensitive rewards $r^i$ from agent $i$ for all $i \in [N]$, computes the vector of joint

**Algorithm 1:** Private Policy Synthesis via Input Perturbation

**Inputs**: $\{r^i\}_{i\in[N]}$, $\mathcal{S}$, $\mathcal{A}$, $\gamma$, $\mathcal{T}$, $\epsilon$, $\delta$, $b$, $N$;
**Outputs**: Privacy-preserving policies $\{\tilde{\pi}^{*,i}\}_{i\in[N]}$;

Agents set $\sigma = \frac{b}{2\epsilon}\kappa(\epsilon,\delta)$;
**for all** $i \in [N]$ *in parallel* **do**
  Agent $i$ generates its private reward function with the Gaussian mechanism, $\tilde{R^i} = R^i + w^i$;
  Agent $i$ sends $\tilde{R}^i$ to the aggregator;
**end**
Aggregator generates joint reward:
$\tilde{r}(s_j, a_k) = \frac{1}{N}\sum_{i\in[N]} \tilde{r}^i(s, a^i)\ \forall j \in [n], \forall k \in [m]$;
Aggregator generates optimal joint policy,
$\tilde{\pi}^* = \arg\max_\pi E\left[\sum_{t=0}^{\infty} \gamma^t \tilde{r}(s_t, \pi(s_t))\right]$;
Aggregator sends $\tilde{\pi}^{*,i}$ to agent $i$ for all $i \in [N]$

rewards $R$, and applies privacy to it to generate $\tilde{R}$. Then it uses $\tilde{R}$ to generate a policy $\tilde{\pi}^*$ for the MMDP [19].

### *B. Input Perturbation*

In the case of input perturbation each agent privatizes the identity map of its own reward. Formally, for all $i \in [N]$ agent $i$'s reward vector is privatized by taking $\tilde{R}^i = R^i + w^i$, where $w^i \sim \mathcal{N}(0, \sigma^2 I)$ for all $i \in [N]$. Then, we use $\tilde{r}(s_j, a_k) = J(\{\tilde{r}^i(s_j, a^i_{I_k(i)})\}_{i\in[N]})$ where $J$ is from (1) for all $j \in [n]$ and $k \in [m]$ to compute $\tilde{r}$, which is the private form of the joint reward $r$ from Definition 2. The private reward $\tilde{r}$ is then used in place of $r$ to synthesize the agents' joint policy. After privatization, policy synthesis is post-processing of differentially private data, which implies that the policy also keeps each agent's reward function differentially private due to Lemma 2. Algorithm 1 presents this method of determining policies from private agent reward functions using input perturbation.

**Theorem 1** (Solution to Problem 1)**.** *Given privacy parameters $\epsilon > 0$, $\delta \in [0, 0.5)$, and adjacency parameter $b > 0$, the mapping from $\{r^i\}_{i\in[N]}$ to $\{\pi^{*,i}\}_{i\in[N]}$ defined by Algorithm 1 keeps each $r^i$ $(\epsilon, \delta)$-differentially private with respect to the adjacency relation in Definition 3.*

*Proof.* See authors' technical report in [32]. □

Using Algorithm 1, each agent enforces the privacy of its own reward function before sending it to the aggregator. Performing input perturbation this way enforces differential privacy on a per-agent basis, which is referred to as "local differential privacy" [33]. The main advantage of input perturbation is that the aggregator does not need to be trusted since it is only sent privatized information. Another advantage is that agents may select differing levels of privacy. However, to provide a fair comparison with output perturbation, we consider each agent using the same $\epsilon$ and $\delta$.

### *C. Output Perturbation*

In output perturbation, for all $i \in [N]$ agent $i$ sends its sensitive (non-private) vector of rewards, $R^i$, to the aggregator. Then the aggregator uses these rewards to form the joint reward $R$. For privacy, noise is added to the joint reward

**Algorithm 2:** Private Policy Synthesis via Output Perturbation

**Inputs**: $\{r^i\}_{i\in[N]}$, $\mathcal{S}$, $\mathcal{A}$, $\gamma$, $\mathcal{T}$, $\epsilon$, $\delta$, $b$, $N$;
**Outputs**: Privacy-preserving policies $\{\tilde{\pi}^{*,i}\}_{i\in[N]}$;

**for all** $i \in [N]$ *in parallel* **do**
  Agent $i$ sends reward $r^i$ to the aggregator
**end**
Aggregator generates joint reward function:
$r(s_j, a_k) = \frac{1}{N}\sum_{i\in[N]} r^i(s, a^i)\ \forall j \in [n], \forall k \in [m]$;
Aggregator sets $\sigma = \frac{b}{2\epsilon N}\kappa(\epsilon,\delta)\mu$;
Aggregator generates private reward function with the Gaussian mechanism, $\tilde{R} = R + w$;
Aggregator generates optimal joint policy,
$\tilde{\pi}^* = \arg\max_\pi E\left[\sum_{t=0}^{\infty} \gamma^t \tilde{r}(s_t, \pi(s_t))\right]$;
Aggregator sends $\tilde{\pi}^{*,i}$ to agent $i$ for all $i \in [N]$;

vector, namely $\tilde{R} = R + w$, where $w \sim \mathcal{N}(0, \sigma^2 I)$. Similar to the input perturbation setup, computing the joint policy using the privatized $\tilde{R}$ is differentially private because computation of the policy is post-processing of private data. Algorithm 2 presents this method of computing policies when reward functions are privatized using output perturbation.

**Theorem 2** (Alternative Solution to Problem 1)**.** *Fix privacy parameters $\epsilon > 0$, $\delta \in [0, 0.5)$, and adjacency parameter $b > 0$. Set $\mu = \max_{j\in[n]} \prod_{\substack{\ell=1 \\ \ell\neq j}}^{N} m_\ell$. The mapping from $J(\{r^i\}_{i\in[N]})$ to $\{\pi^{*,i}\}_{i\in[N]}$ defined by Algorithm 2 keeps each $r^i$ $(\epsilon, \delta)$-differentially private with respect to the adjacency relation in Definition 3.*

*Proof.* See authors' technical report in [32]. □

Unlike input perturbation, output perturbation requires that agents trust the aggregator with their sensitive information. Additionally, all agents will have the same level of privacy.

Contrary to some other privacy literature, we attain significantly better performance using input perturbation over output perturbation. For output perturbation, the standard deviation $\sigma$ used to calibrate the noise added for privacy essentially grows exponentially with the number of agents, which can be seen in the term $\mu$ in Theorem 2. This is because we consider the joint state $s$ in evaluating $r^i(s, a^i)$ and each joint state-local action pair will appear many times in $\Delta_2 J$. For input perturbation, since the joint reward is computed from privatized rewards, the standard deviation of privacy noise does not depend on the number of agents. Given the essentially exponential dependency on agent number in output perturbation, we focus on input perturbation going forward unless stated otherwise.

## IV. Accuracy Analysis

In this section, we solve Problem 2. Specifically, we analyze the accuracy of the reward functions that are privatized using input perturbation. To do so, we compute an upper bound on the expected maximum absolute error between the sensitive, non-private joint reward $r$ and the privatized reward $\tilde{r}$. Then we use this bound to develop guidelines for calibrating privacy to obey a bound on allowable error.

**Theorem 3** (Solution to Problem 2)**.** *Fix privacy parameters $\epsilon > 0$, $\delta \in [0, 0.5)$, adjacency parameter $b > 0$, and MMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \gamma, \mathcal{T})$ with $N$ agents, $n$ joint states, and $m$ joint actions. Let $\tilde{r}$ be defined as in Algorithm 1. Then $\mathbb{E}\left[\max_{k,j} |\tilde{r}(s_k, a_j) - r(s_k, a_j)|\right] \leq \frac{Cb}{2\epsilon}\kappa(\epsilon, \delta)$ where $C = \sqrt{2/(N\pi)} + \sqrt{(1 - 2/\pi)\,(nm - 1)/N}$.*

*Proof.* See authors' technical report in [32]. □

**Corollary 1.** *Fix $\delta \in [0, 0.5)$ and let the conditions from Theorem 3 hold. A sufficient condition to achieve $\mathbb{E}\left[\max_{k,j} |\tilde{r}(s_k, a_j) - r(s_k, a_j)|\right] \leq A$ is given by $\epsilon \geq \frac{2C^2b^2}{4A^2} + \frac{Cb\mathcal{Q}^{-1}(\delta)}{A}$, where $C = \sqrt{\frac{2}{N\pi}} + \sqrt{\left(1 - \frac{2}{\pi}\right)\frac{(nm-1)}{N}}$.*

*Proof.* See authors' technical report in [32]. □

Theorem 3 shows that the accuracy of private rewards depends on the size of the MMDP $nm$, the number of agents $N$, and the privacy parameters $\epsilon$, $\delta$, and $b$. The $\sqrt{nm-1}$ term indicates that while the error grows with the number of agents, the error does not grow exponentially. Corollary 1 provides a trade-off that allows users to tune $\epsilon$ to balance privacy and accuracy.

## V. Cost of Privacy

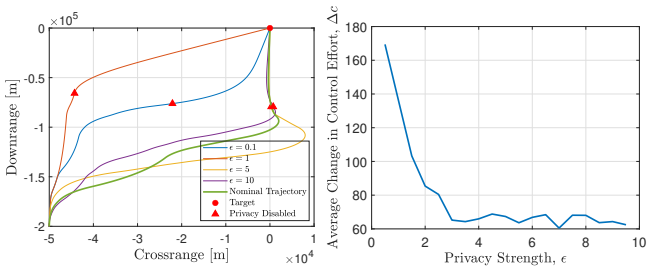
In this section, we solve Problem 3. To do so, we compare (i) an optimal policy generated on the original, non-private reward functions, denoted $\pi^*$, and (ii) an optimal policy generated on the privatized reward functions, denoted $\tilde{\pi}^*$. Beginning at some state $s_0 \in \mathcal{S}$, the function $V_{\pi^*}(s_0)$ encodes the performance of the MMDP given the optimal policy and $V_{\tilde{\pi}^*}(s_0)$ encodes the performance of the MMDP given the policy generated on private rewards. We analyze the performance loss due to privacy using the "cost of privacy" metric introduced in [20], namely $|V_{\tilde{\pi}^*}(s_0) - V_{\pi^*}(s_0)|$. We must compute $V_{\pi^*}(s_0)$ and $V_{\tilde{\pi}^*}(s_0)$ to do so, and these terms do not have a closed form in general.

Proposition 1 provides a method for empirically evaluating the value function for a policy on a given MMDP by repeatedly applying the Bellman operator. We can then compute the cost of privacy numerically for any problem using an off-the-shelf policy-evaluation algorithm. Next, we state a theorem on the number of operations required to compute the cost of privacy in this way.

**Theorem 4** (Solution to Problem 3)**.** *Fix privacy parameters $\epsilon > 0$ and $\delta \in [0, 1/2)$. Let $\tilde{r}$ be the output of Algorithm 1. Given an MMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \gamma, \mathcal{T})$, the number of computations required to compute the cost of privacy $|V_{\tilde{\pi}^*}(s_0) - V_{\pi^*}(s_0)|$ to within $\eta$ of the exact value is is $nm(K_1 + K_2)$, where, $K_1 = \lceil \log(\frac{4R_{max}}{\eta(1-\gamma)^2}) / \log(\frac{1}{\gamma}) \rceil$, $K_2 = \lceil \log(\frac{4\tilde{R}_{max}}{\eta(1-\gamma)^2}) / \log(\frac{1}{\gamma}) \rceil$, and $R_{max} = \max_{s,a} |r(s,a)|$, $\tilde{R}_{max} = \max_{s,a} |\tilde{r}(s,a)|$.*

*Proof.* See authors' technical report in [32]. □

Theorem 4 shows that the computational complexity of computing the cost of privacy grows bilinearly with number of joint states and joint actions, indicating that this method is tractable for computing the cost of privacy. If $\pi^*$ is generated using value iteration, the user already has $V_{\pi^*}$ and does not need to compute it again for the cost of privacy, which would reduce the complexity of computing the cost of privacy.




(a) (b)

Fig. 1: Simulation results for Example 1. (a) high-speed vehicle trajectories given varying privacy levels from Example 1. When privacy is weaker, e.g., at $\epsilon = 10$, the trajectory of the vehicle is a close match to the nominal trajectory, while when privacy is stronger, there are larger deviations from the nominal trajectory. (b) average control effort for 600 private policies generated with $\epsilon \in [0.1, 10]$. As the strength of privacy decreases (and $\epsilon$ increases), the change in control effort decreases. The optimal policy induces the least amount of turning in the vehicle, which requires less control effort, but private policies may require sharp turns to reach the target, inducing more control effort as a result.

## VI. Numerical Simulations

In this section, we consider a large MDP with a single agent to assess how differential privacy changes a state trajectory over time.

**Example 1** (Waypoint Guidance)**.** To highlight the applicability to a single agent, consider a high-speed vehicle using waypoints to navigate towards a target. We consider an MDP with a $20 \times 20$ set of waypoints as the states with the actions $\mathcal{A} = \{\texttt{north}, \texttt{south}, \texttt{east}, \texttt{west}, \texttt{arrived}\}$. We consider a 3 degree-of-freedom vehicle model, with waypoints that exist in the plane of the initial condition. The initial position of the vehicle is the first waypoint and the initial state of the MDP. The action for that state yields a new waypoint for the vehicle to navigate towards. Once the vehicle is closer to the new waypoint than the original, we say the new waypoint is now the current state of the vehicle, and the policy again commands the vehicle to navigate to a new waypoint. This continues until the vehicle is less than 80 km from the target, at which point the vehicle is considered to have "arrived" at its target. As a result, within 80 km it is commanded to navigate directly to the target (as opposed to another waypoint) without privacy.

The optimal policy gives the optimal set of waypoints given an initial starting state. We implemented differential privacy for the reward function of this MDP using Algorithm 1. Note that since there is only one agent, the outputs of Algorithm 1 and Algorithm 2 are identical.

Trajectories generated using various $\epsilon$ values are presented in the crossrange-downrange plane in Figure 1a. We simulated 600 privatized policies for each $\epsilon \in [0.1, 10]$. While on average we recover the nominal trajectory as epsilon grows,

note that individual trajectories are random draws, and here the reward privatized with $\epsilon = 1$ yielded a more off-nominal trajectory than the one with $\epsilon = 0.1$.

Treating the trajectory generated from the sensitive policy as the nominal trajectory, we assess the performance of the trajectories using the change in the cumulative acceleration commands, which we refer to as "control effort" and denote by $\Delta c$, between the trajectories generated using the sensitive, non-private policy and the privatized policy. That is, $\Delta c = \int_0^{t_f} \|u\|_2^2 dt - \int_0^{\tilde{t}_f} \|\tilde{u}\|_2^2 dt$, where $u$ is the commanded acceleration of the vehicle operating with a policy developed on the non-private rewards, and $\tilde{u}$ is defined analogously for the vehicle operating with a policy developed on the private rewards. Here $t_f$ is the flight duration of the vehicle when using a policy generated on sensitive, non-private reward $r$ and $\tilde{t}_f$ is the flight duration of the vehicle when using a policy generated on privatized rewards $\tilde{r}$. Note that under privacy, trajectories may differ in length, which means that comparing the accelerations at each time step may not yield a meaningful comparison. As a result, we compare the total commanded accelerations over the entire trajectory, shown in Figure 1b. On average, with stronger privacy (i.e., smaller $\epsilon$), the policy commands the vehicle to off-nominal trajectory waypoints requiring larger overall control efforts. We see $\Delta c$ steadily declines up to $\epsilon = 2$, indicating that there is minimal performance gain by decreasing the strength of privacy any further, which implies that $\epsilon = 2$ effectively trades off privacy and performance in this case.

## VII. Conclusion

We have developed two methods for protecting reward functions in MMDPs from observers by using differential privacy, and we identified input perturbation as the more tractable method. We also examined the accuracy versus privacy trade-off and the computational complexity of computing the performance loss due to privacy, and showed the success of these methods in simulation. Future work will be focused on developing guidelines for designing reward functions that are amenable to privacy.

## References


[1] D. J. Glancy, "Privacy in autonomous vehicles," *Santa Clara L. Rev.*, vol. 52, p. 1171, 2012.

[2] Z. Guan, G. Si, X. Zhang, L. Wu, N. Guizani, X. Du, and Y. Ma, "Privacy-preserving and efficient aggregation based on blockchain for power grid communications in smart communities," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 82–88, 2018.

[3] A. Y. Ng, S. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[4] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[5] T. Zhi-Xuan, J. Mann, T. Silver, J. Tenenbaum, and V. Mansinghka, "Online bayesian goal inference for boundedly rational planning agents," *Advances in neural information processing systems*, vol. 33, pp. 19 238–19 250, 2020.

[6] M. Ramırez and H. Geffner, "Goal recognition over pomdps: Inferring the intention of a pomdp agent," in *IJCAI*. IJCAI/AAAI, 2011, pp. 2009–2014.

[7] C. Dwork, "Differential privacy," *Automata, languages and programming*, pp. 1–12, 2006.

[8] J. Le Ny and G. J. Pappas, "Differentially private filtering," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 341–354, 2013.

[9] J. Cortés, G. E. Dullerud, S. Han, J. Le Ny, S. Mitra, and G. J. Pappas, "Differential privacy in control and network systems," in *55th IEEE Conference on Decision and Control (CDC)*, 2016, pp. 4252–4272.

[10] S. Han and G. J. Pappas, "Privacy in control and dynamical systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 309–332, 2018.

[11] K. Yazdani, A. Jones, K. Leahy, and M. Hale, "Differentially private lq control," *IEEE Transactions on Automatic Control*, 2022.

[12] C. Hawkins and M. Hale, "Differentially private formation control," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 6260–6265.

[13] Y. Wang, M. Hale, M. Egerstedt, and G. E. Dullerud, "Differentially private objective functions in distributed cloud-based optimization," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 3688–3694.

[14] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proceedings of the 16th International Conference on Distributed Computing and Networking*, 2015, pp. 1–10.

[15] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 50–64, 2016.

[16] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via objective perturbation," in *2016 American control conference (ACC)*. IEEE, 2016, pp. 2061–2066.

[17] R. Dobbe, Y. Pu, J. Zhu, K. Ramchandran, and C. Tomlin, "Customized local differential privacy for multi-agent distributed optimization," *arXiv preprint arXiv:1806.06035*, 2018.

[18] Y.-W. Lv, G.-H. Yang, and C.-X. Shi, "Differentially private distributed optimization for multi-agent systems via the augmented lagrangian algorithm," *Information Sciences*, vol. 538, pp. 39–53, 2020.

[19] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[20] P. Gohari, M. Hale, and U. Topcu, "Privacy-preserving policy synthesis in markov decision processes," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 6266–6271.

[21] B. Chen, C. Hawkins, M. O. Karabag, C. Neary, M. Hale, and U. Topcu, "Differential privacy in cooperative multiagent planning," in *Uncertainty in Artificial Intelligence*. PMLR, 2023, pp. 347–357.

[22] P. Venkitasubramaniam, "Privacy in stochastic control: A markov decision process perspective," in *51st Annual Allerton Conference on Communication, Control, and Computing*, 2013, pp. 381–388.

[23] X. Zhou, "Differentially private reinforcement learning with linear function approximation," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 6, no. 1, pp. 1–27, 2022.

[24] P. Ma, Z. Wang, L. Zhang, R. Wang, X. Zou, and T. Yang, "Differentially private reinforcement learning," in *International Conference on Information and Communications Security*. Springer, 2019, pp. 668–683.

[25] B. Fallin, C. Hawkins, B. Chen, P. Gohari, A. Benvenuti, U. Topcu, and M. Hale, "Differential privacy for stochastic matrices using the matrix dirichlet mechanism," in *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2023, pp. 5067–5072.

[26] B. Chen, K. Leahy, A. Jones, and M. Hale, "Differential privacy for symbolic systems with application to markov chains," *Automatica*, vol. 152, p. 110908, 2023.

[27] D. Ye, T. Zhu, W. Zhou, and S. Y. Philip, "Differentially private malicious agent avoidance in multiagent advising learning," *IEEE transactions on cybernetics*, vol. 50, no. 10, pp. 4214–4227, 2019.

[28] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *TARK*, vol. 96. Citeseer, 1996, pp. 195–210.

[29] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[30] J. Hsu, M. Gaboardi, A. Haeberlen, S. Khanna, A. Narayan, B. C. Pierce, and A. Roth, "Differential privacy: An economic method for choosing epsilon," in *2014 IEEE 27th Computer Security Foundations Symposium*. IEEE, 2014, pp. 398–410.

[31] C. Hawkins, B. Chen, K. Yazdani, and M. Hale, "Node and edge differential privacy for graph laplacian spectra: Mechanisms and scaling laws," *IEEE Transactions on Network Science and Engineering*, 2023.

[32] A. Benvenuti, C. Hawkins, B. Fallin, B. Chen, B. Bialy, M. Dennis, and M. Hale, "Differentially private reward functions for markov decision process," *arXiv preprint arXiv:2309.12476*, 2023.

[33] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 2013, pp. 429–438.