Path to a New Secure Reinforcement Learning Model - Explicit Construction as an **IP** and Cryptographic Implications

Abstract

This project defines and studies several structural results of the search variants of Markov Decision Processes (MDPs) and draws explicit connections between reinforcement learning theory, which is based on variants of MDPs, and Interactive Proof Systems. To do so, we first define the search versions of the typical variants of MDPs, and extended the [PT87] structural results about decisions to the functional hierarchy. Then, we define a previously unstudied variant of MDPs that is succinctly defined with an intractible size, called LARGE-FH-MDP-SEARCH, and show that it is **FPSPACE**-complete (in particular, equivalent to Partially Observable MDPs with a tractible size). This structural result helps us conclude the trade-off between sizes and observability, as well new conditions that help prove $\mathbf{P} \neq \mathbf{PSPACE}$ and prove $\mathbf{IP} = \mathbf{PSPACE}$. Then, using LARGE-FH-MDP-SEARCH, we formulate a canonical model for reinforcement learning that is more aligned with complexity theory, called *Statistical Reinforcement Learning System* (**SRLS**). We show explicit reductions both ways between **SRLS** and **IP** algorthmically and conclude how **RL** could be conducted with Zero-Knowledge. Finally, we discuss future potentials of **SRLS**.

Reinforcement Learning Theory, Markov Decision Processes, Interactive Proof Systems, Virtual Black-Box Obfuscation Constructible for SVL, Zero-Knowledge Learning, Total Search Function Complexity.

1 Introduction

In this work, we initiate a new line of research motivated by the following question:

> What are the insights that lie in the intersection of reinfrceoment learning, computational complexity, computational learning theory and cryptography.

While there is a plethora of research on reinforcement learning (**RL**) theory, its ties to computational learning theory, research into ties with computational complexity theory is sparse. Further still, to the best of our knowledge, we find that research regarding the intersection of **RL** and cryptography is non-existent. Despite the lack of such connections, **RL** theory as it has been studied in literature can offer many insights for solving complexity theoretic and cryptographic open questions, and this project is to mention some of such results.

[PT87] showed that the decision version of the central problem of **RL** – that of finding an optimal policy of a Markov decision process (MDP) – is complete for **P** and **PSPACE** in the fully-observable and partially-observable cases, respectively, so we have the necessary foundation to approach our motivating question. Through their results, we can analyze fully-observable (resp. partially-observable) MDPs as time (resp. space) efficient models of computation. Thus, to make concrete ties between the theories of **RL** and those of complexity theory and cryptography, we first start by formalizing our notations and proposed results, categorized into the following directions.

In the first part, we extend the findings of [PT87] by considering *search problems* as opposed to *decision problems*. One of the motivations for doing so is that, by nature of MDPs, such search problems are *total*. To that end, we formalize and study the complexities of the problems of finding optimal value functions for finite-horizon and infinite-horizon cases of both types of MDPs (sections 3.2 and 3.3). Next, we consider the natural problem of finding an optimal-value function for intractably-large fully-observable MDPs (LARGE-FH-MDP-SEARCH), which was *not* explored in literature. We do so by representing the intractably-large fully-observable MDP through a succinct description, and we think we can show that it is **FPSPACE**-complete, same as the finite-horizon partially observable MDPs (theorem 4.2). Thus, what this result shows is that there exists a trade-off between **observability** and **number of states and actions** (corollary 1). Finally, for this part, we show that by assuming *polynomially-secure Virtual Black Box (***VBB***) for* SVL, $\mathbf{P} \neq \mathbf{PSPACE}$ (theorem 4).

In the second part, we introduce the seemingly lacking *canonical computational model* that represents an **RL** system. We utilize our preliminary results to address this deficiency to design a new computational model called a *Statistical* **RL** *Model* (definition 20). We then show that a **SRLS** can be used to find the optimal value functions for both fully-observable and partially-observable MDPs. We show this by explicitly designing a protocol to find the optimal value function of an intractably large, MDP. Finally, we establish ties to cryptography by proving the existence of a correspondence between **IP** systems and **SRLS***s* (theorem 5 and corollary 8). Our result can be rephrased as an alternative proof of $\mathbf{IP} = \mathbf{PSPACE}$.

In the third part, we focus on analyzing **SRLS**s that find optimal policies of fully-observable MDPs and show learning can be achieved in a *privacy-preserving manner*. Specifically, we show that if *one-way functions exist*, then Statistical **RL** for fully-observable, tractable cases can be achieved with *zero-knowledge* (theorem 7). In particular, our result is incomparable to known results like [GMW91a, BJSW20], we are dealing with a class of **FPSPACE**-complete problems with the additional assumptions of OWFs.

Finally, we raise open questions regarding achieving *accuracy boosting* with multiple agents, and quantum analogues for Statistical **RL** models.

This project report is structured in this way:

- Section 2 provides the necessary backgrounds of **RL** as an MDP optimization problem.
- Section 3 gives proofs to some basic results as warm-ups to the first part of our results.
- In section 4 are some of our main results for a new class of problems in **RL**, called LARGE-FH-MDP-SEARCH, and its complexity theoretic and cryptographic hardnesses.
- In section 5 is the second part of our results.
- In section 6 is the third part of our results.
- In section 7, we give some directions for future works extending our new **SRLS** model, including the fourth part of our results.

Note that everything beyond section 3 are original to this report (though some of the results in section 3 follow pretty easily from literature).

2 Preliminaries

2.1 Markove Decision Processes (MDPs)

We begin our preliminary review by analyzing the central object of our study, the MDPs. Informally, an MDP represents a probabilistic computation model. This connection is made obvious by noting that the description of an MDP is syntactically similar to that of a probabilistic Turing Machine. Furthermore, much like a Turing Machine, an MDP starts at a dedicated starting state, and transitions to new states through actions that are available at any given state. Throughout these transitions, a reward may be provided if a certain action is taken at a certain state. Thus, for each state-action pair, we can define a so-called *reward function*. Furthermore, the next state that is entered given a state and an associated action taken can be *stochastic*.

Formally, we first define the base form of an MDP and specify variants of it. Specifically, we can have four variants, based on combinations of two specifications, namely, time bound and observability (see definition 3 for partially observable), so we have:

- Unbounded runtime, fully observable (definition 1).
- Bounded runtime, fully observable (definition 2).
- Unbounded runtime, partially observable (definition 4).
- Bounded runtime, partially observable (definition 5).

Definition 1 (Basic MDP, a.k.a. Infinite Horizon MDP). *An IH-MDP, $x \in \{0,1\}^n$, is a 6-tuple (S, A, R, Ps_0, γ) where*

- *S is a finite set of states.*
- *A is a finite set of actions.*
- *$R \subseteq \mathbb{Q}$ is a bounded set of rewards (see remark 2 for why we take subset of $\mathbb{Q}$).*
- *$P : S \times A \times S \times R \to [0,1]$ is a joint probability distribution describing the transition dynamics of the process. It takes the current state, action on the current state, next state, and the reward for this sequence of actions as input, and output the probability that, given the current state and action, the given next state and reward is what the learner gets. In particular, if a next state or its corresponding reward is impossible given the current state and action, the output would be* 0.
- *$s_0 \in S$ is the dedicated start state.*
- *$\gamma \in [0,1] \cap \mathbb{Q}$ is a discount factor (see remark 2 for why we intersect with $\mathbb{Q}$).*

Remark 1. *The P here differs from the typical literature, as typical literature considers the transition dynamics as a conditional probability given that one is already in the current state C_i. For example, our P here is*

$$P(c_i, a, c_j, r) = \Pr[S = c_i, A = a, S' = c_j, R = r],$$

in contrast to the typical conditional probability

$$P'(c_i, a, c_j, r) = \Pr[S' = c_j, R = r \mid S = c_i, A = a].$$

It's not hard to see that both are valid and equivalent, as

$$P = \Pr[S = c_i, A = a \wedge S' = c_j, R = r] = P' \cdot \frac{\Pr[S = c_i, A = a]}{\Pr[S' = c_j, R = r]},$$

so we say that our joint probability distribution describes the transition dynamics of the MDPs.

Remark 2. *While not standard in pre-existing literature (to the best of our knowledge), we ensure that any rational values present in x are efficiently computable and bounded by a time-constructible function $T(|x|)$, where $T : \mathbb{N} \to \mathbb{N}$.*

Definition 2 (Finite-Horizon MDP). *An FH-MDP, $x \in \{0,1\}^n$, is a 7-tuple $(S, A, R, Ps_0, \gamma, 1^H)$ that has the same first six elements as definition 1 with the additional 7-th element being an explicit time-bound that the process is allowed to run.*

- *1^H is the horizon, where $H \in \mathbb{N}_{>0}$.*

Definition 3 (Partially Observable)**.** *Next, we want to further generalize MDPs by allowing the current states to not be observed with perfect certainty. In particular, in the partially observable model, we partition all the states into some subsets,* $\{O_1, \ldots, O_k\}$*, which we call observations, and, at each state, we only know which observation we are in (we generally consider the elements within an observation is distributed uniformly by default, but it could also specify what probability it is for each state). In particular, we consider the transition dynamics as*

$$Pr[S' = s | O = o, A = a].$$

Definition 4 (Infinite-Horizon Partially Observable MDP)**.** *An IH-POMDP,* $x \in \{0,1\}^n$*, is a* 7*-tuple* $(S, A, O, R, P, s_0, \gamma)$ *that has the same six elements as definition 1 with the additional* 3*-rd element being a partition of* S *as observations:*

- *O is a set of observations (see definition 3 for how this works).*

Definition 5 (Finite-Horizon Partially Observable MDP)**.** *An FH-POMDP,* $x \in \{0,1\}^n$*, is an* 8*-tuple* $(S, A, O, R, P, s_0, \gamma, 1^H)$ *which adds both of the following to the* 6 *elements of definition 1:*

- *O is a set of observations (see definition 3 for how this works).*
- 1^H *is the horizon where* $H \in \mathbb{N}_{>0}$.

2.2 Policies, Value Functions and Bellman-Optimality

It is important to remind again that the goal of any variant of MDPs is to find optimal policies, i.e. the policy that maximizes reward. So, to define policies, we must first formalize what reward functions are.

Definition 6 (Reward Function)**.** *Let* $r : S, A \to \mathbb{Q}$ *be a reward function, and it is defined as:*

$$r(s, a) = \sum_{r \in R} \left(r \cdot \Pr[R = r | S = s, A = a] \right),$$

or, more compactly, we write

$$r(s, a) = \mathbb{E}\left[R | S = s, A = a\right]. \tag{1}$$

Even though each policy itself is not related to the reward function, the optimal policy is defined as having the maximal reward over the system's course of actions. Notice that, since we consider stochastic MDPs (as in, so far, given current state and action under our joint probability distribution that describes the transition dynamics, the next state and reward are probabilistic), we need to allow the policies to be probabilistic too. The purpose of allowing probabilistic policies is analogous to how allowing mixed strategies in Nash-Equilibrium problems makes it a total search problem.

Definition 7 (Policies for MDPes)**.** *Let x be our MDP input tuple. A policy for x, π, is a conditional probability distribution over actions given a state. Thus, $\forall s \in S$, π is*

$$\pi(s) = [\Pr[A = a_1|S = s], \Pr[A = a_2|S = s], \ldots, \Pr[A = a_{|A|}|S = s]].$$

For short-hand, we say

$$\pi(a|s) = \Pr[A = a|S = s].$$

With each specific policy as defined above, we let $\Pi := \{\pi : \pi \text{ is a valid policy for } x\}$.

Note that as a policy, π, prescribes a distribution of actions over a state, we can substitute it into equation (1) for actions to compute the *expected reward* in some $s \in S$ as:

$$r(s, \pi(s)) = \sum_{a \in A} \pi(a|s)\mathbb{E}[R|S = s, A = a].$$

Though we can just take these rewards as they are at each state s, and then add them up over the entire course of the **RL** system, we can consider adding a discount to the rewards from states before the current state. This discount is not necessary for Finite-Horizon problems but is vital for infinite horizon problems for them to be meaningful, which we discuss below.

Definition 8 (Discounted Expected Rewards, Value of a Policy)**.** *Given that a policy also causes a transition into a new state, we can consider adding discounted expected rewards that are incurred through the policy. This quantity is known as the value of the policy, and is defined as the following: for every $s \in S$ and some $\gamma \in [0, 1]$,*

$$V^\pi(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S, r \in R} \Pr\left[S' = s', R = r|S = s, A = a\right](r(s, a) + \gamma \cdot V^\pi(s')),$$

where the "$r(s, a)$" component is just broken down from the original $\mathbb{E}[R|S = s, A = a]$, and "$\gamma \cdot V^\pi(s')$" is the value of the policy till before the next state at a discount.

Finally, as we mentioned before, an optimal policy is an element of Π such that it maximizes the (discounted) expected rewards over the course of the **RL** system. We formally define *optimal policy* for an MDP below.

Definition 9 (Optimal Policy for MDPs)**.** *An optimal policy, denoted π^*, is one that maximizes the value over all states. In other words, $\forall (s, \pi) \in S \times \Pi$, the following is true:*

$$V^{\pi^*}(s) \geq V^\pi(s)$$

Now, this definition may appear surprising, particular because it asserts that an optimal policy must optimizes the (discounted) expected value at each state, s, not just the final value. But, this is true by invoking *Bellman-Optimality*:

Theorem 1 (Bellman-Optimality [SB18]). *An optimal policy must take the action that maximizes its immediate expected reward. That is, an optimal policy is both globally and locally optimal. Formally, Bellman-Optimality states that*

$$V^{\pi^*}(s) = \max_a \sum_{r,s'} P(S = s', R = r|S = s, A = a)(r(s,a) + \gamma V^*(s')).$$

Remark 3. *In particular, the Bellman-optimality equation ensures that our verification of policies remain efficient. Furthermore, it is straightforward to note that these definitions are applicable in both the finite-horizon and infinite horizon case of fully-observable MDPs.*

2.3 Belief State for Partially-observable cases

Next, we define the completely analogous concepts for the partially-observable case. Recall that in the partially-observable case, next-states cannot be directly observed. Instead, an observation signal is provided given an action. To that end, we can compute a probability distribution over all possible next states, given an action that was taken and an observation that was received. We call this conditional probability distribution a *belief state* and it is formally defined in the following manner.

Definition 10 (Belief States). *A belief state is a conditional probability distribution over states given an action taken and an observation received. It is defined as*

$$b(s') = \frac{P(O = o|S' = s, A = a)}{P(O = o|B = b, A = a)} \cdot \sum_{s \in S} P(S' = s|S = s, A = a) \cdot b(s),$$

where $P(O = o|B = b, A = a)$ is defined as

$$P(O = o|B = b, A = a) = \sum_{s' \in S} P(O = o|S' = s, A = a) \sum_{s \in S} P(S' = s'|S = s, A = a) \cdot b(s).$$

Furthermore, we define b as

$$b = [b(s_1), \ldots, b(s_{|S|})].$$

2.4 Bellman-Optimality for Partially-Observable Case

Note that each belief b defines a completely Markovian signal. However, each belief state is not longer discrete, but rather continuous and defined over the $|S| - 1$ dimensional simplex. Let $\mathcal{B}$ denote the set of all belief states and can define policies for the partially observable case.

Definition 11 (Policies for Partially Observable MDPs). *A policy π for a partially-observable MDP is a conditional probability distribution over actions given beliefs. Thus, $\forall b \in \mathcal{B}$, it is defined as*

$$\pi(a|b) = Pr[A = a|B = b].$$

For short-hand, we can say the following

$$\pi(b) = [Pr[A = a_1|B = b], \ldots, Pr[A = a_{|A|}|B = b]].$$

In a manner that is similar to the one described above, we can now define the Bellman-Optimality equation as follows:

$$V^*(b) = \max_a \left\{ \sum_o P(o \mid b, a) \left[\sum_{s \in S} R(s, a) \cdot b(s) + \gamma \sum_{s'} P(s' \mid b, a, o) \cdot V^*(b') \right] \right\}.$$

2.5 Every MDP has an Optimal Policy

Notice that every MDP, be it fully-observable or partially-observable, finite-horizon or infinite-horizon, has an optimal policy, formally stated and proved in the following lemma.

Lemma 1 (MDPs are Total). *Every MDP has an optimal policy.*

Proof. Let x be an MDP. If there exists a state $s \in S$ such that there is no associated action available to it, introduce a single action $a \in A$ and let $\Pr[S' = s | S = s, A = a] = 1$. Let Π_x be the set of policies associated with x. Due to the fact that $|\Pi_x| \geq 1$ and every $\pi \in \Pi$ can be partially ordered by the value function V^π, there exists at an optimal policy $\pi^* \in \Pi$. ■

3 Basic Hardness Results of MDPs

3.1 Formal Search Problems

Now that we know the variants of MDPs that we are considering are all total, we first define and analyze the following search problems (as relations) called: FH-MDP-SEARCH, IH-MDP-SEARCH, FH-POMDP-SEARCH, IH-POMDP-SEARCH. Keep in mind that they are simple search equivalent of the decision problems that we already introduced in section 2.1.

Definition 12 (FH-MDP). *FH-MDP is the kind of MDP optimal policy search problems where the function machine is given an instance of a finite-horizon, fully-observable MDP whose total relation can be defined by:*

$$\textit{FH-MDP} = \{\langle (S, A, P, R, s_0, \gamma, 1^H), V \rangle : \forall (s, \pi) \in S \times \Pi, V(s) \geq V^\pi(s)\}.$$

Thus, when the function machine is given an input $x = (S, A, P, R, s_0, \gamma, 1^H)$ it outputs an optimal value function, $y = V*$, such that $(x, y) \in$ FH-MDP.

Definition 13 (IH-MDP). *IH-MDP is the kind of MDP optimal policy search problems where the function machine is given an instance of an infinite-horizon, fully-observable MDP whose total relation can be defined by:*

$$\textit{IH-MDP} = \{\langle (S, A, P, R, s_0, \gamma), V \rangle : \forall (s, \pi) \in S \times \Pi, V(s) \geq V^\pi(s)\}.$$

In a similar manner to FH-MDP, the function machine is given an input $x = (S, A, P, R, s_0, \gamma)$ and outputs an optimal value function, V such that $(x, y) \in FHMDP$.

Definition 14 (FH-POMDP)**.** *FH-POMDP is the kind of MDP optimal policy search problems where the function machine is given an instance of a finite-horizon, partially-observable MDP that can be defined by:*

$$\textsf{FH-POMDP} = \{\langle (S, A, P, R, O, s_0, \gamma, 1^H), V\rangle : \forall(s, \pi) \in S \times \Pi, V(s) \geq V^{\pi}(s)\}.$$

The goal of the underlying function machine is to, given an input instance $x = (S, A, P, R, O, s_0, \gamma, 1^H)$, find the optimal value $y = V$ such that $(x, y) \in$ FH-POMDP.

Definition 15 (IH-POMDP)**.** *Similar to FH-POMDP, IH-POMDP is the kind of MDP optimal policy search problems where the function machine is given an instance of an infinite-horizon, partially observable MDP that can be defined by:*

$$\textsf{FH-POMDP} = \{\langle (S, A, P, R, \Omega, s_0, \gamma, 1^H), V\rangle : \forall(s, \pi) \in S \times \Pi, V(s) \geq V^{\pi}(s)\}.$$

The uncertainty of the underlying hidden state is encoded in the same manner as that of FH-POMDP. However, given an input instance $x = \langle S, A, P, R, \Omega, s_0, \gamma, 1^H\rangle$, the task of the machine is to find an optimal value function $y = V$ such that is maximized the expected discounted sum of rewards, which implies that $(x, y) \in$ IH-POMDP.

3.2 FH-MDP and IH-MDP are FP-Complete

To prove these claims, we show that both search problems are (1) computable by deterministic function machines and (2) have decision variants that are **P**-hard. For now, we focus on proving these claims for FH-MDP. The proofs of this section naturally follow from the foundational work as laid by Papadimitriou and Tsitsklis [PT87].

Proof. (FH-MDP is **FP**-Complete). Let $x = \langle S, A, P, R, s_0, \gamma, 1^H\rangle$ be an instance of FH-MDP. Note that there exist various methods to compute the optimal policy π^*, one such method is to use linear programming ([PT87] showed several variants of FH-MDP are in **P** using linear programming). To ensure that linear programming terminates within H steps, we write H in unary. Thus, utilizing linear programming to compute the optimal policy π^* for x inadvertently utilizes a reduction to an **FP**-Complete problem. We let M be a function machine that solves FH-MDP. Hence, FH-MDP $\in$ **FP**.

To show that FH-MDP is complete for **FP**, we show that the decision variant is **P**-hard. To do so, we adapt [PT87] (nearly verbatim) and show a reduction from the circuit evaluation problem.

Definition 16 (Circuit Evaluation Problem)**.** *Let $C = ((a_i, b_i, c_i))_{i\in[k]}$ where for each $i \in [k]$, $a_i \in \{\wedge, \vee, 0, 1\}$, and $b_i, c_i \in [i-1]$. If $a_i \in \{0, 1\}$, then $b_i = c_i = 0$ and the triple (a_i, b_i, c_i) is denoted an input. Furthermore, if $a_i \in \{\wedge, \vee\}$ then it must be the case that $b_i, c_i < i$, and the triple (a_i, b_i, c_i) is denoted a gate. The circuit evaluation problem is to decide if (a_k, b_k, c_k) evaluates to 1, given a circuit $C = ((a_i, b_i, c_i))_{i\in[k]}$.*

The reduction from a circuit C to an FH-MDP is as follows:

Algorithm 1 Reduction $R(C = ((a_i, b_i, c_i))_{i\in[k]})$

```
$S \leftarrow \{t\}, A \leftarrow \{0, 1\}$
for $(a_i, b_i, c_i) \in C$ do
    $S \leftarrow S \cup \{s_i\}$
    if $a_i = 1$ then
        $P(S' = t, R = 1 \mid S = s_i, A = 0) = 1$
    else if $a_i = 0$ then
        $P(S' = t, R = 0 \mid S = s_i, A = 0) = 1$
    else if $a_i = \wedge$ then
        $P(S' = s_{b_i}, R = 0 \mid S = s_i, A = 0) = 1$
        $P(S' = s_{c_i}, R = 0 \mid S = s_i, A = 1) = 1$
    else if $a_i = \vee$ then
        $P(S' = s_{b_i}, R = 0 \mid S = s_i, A = 0) = 1/2$
        $P(S' = s_{c_i}, R = 0 \mid S = s_i, A = 0) = 1/2$
        $P(S' = s_{b_i}, R = 0 \mid S = s_i, A = 1) = 1/2$
        $P(S' = s_{c_i}, R = 0 \mid S = s_i, A = 1) = 1/2$
    end if
end for
```

Then, $\exists$ a deterministic Turing machine D that decides the circuit-value problem, utilizing R and M.

Algorithm 2 Distinguisher $D(C = ((a_i, b_i, c_i))_{i\in[k]})$

```
$\langle S, A, P, R, \gamma, s_0 \rangle \leftarrow R(C)$
$\pi^* \leftarrow M(x)$
if $V^{\pi^*}(s_0) \geq 0$ then
    Return 1
else
    Return 0
end if
```

Analogously, a similar construction to this works for proving that IH-MDP is **FP**-Complete. So, both FH-MDP and IH-MDP are **FP**-Complete. ■

3.3 FH-POMDP and IH-POMDP are FPSPACE-Complete

For the scope of this paper, we specify the following definition of **FPSPACE**:

Definition 17 (**FPSPACE**). **FPSPACE** *is the search problem where both the work tape and the output tape need to be poly-space bounded. In particular, this differs from common definitions of* **FPSPACE** *where the output tape is allowed to be unbounded. While* **FP** $\subsetneq$

FPSPACE *is known for the common definition, it is not known whether or not* **FP** *is strictly contained in* **FPSPACE** *in our definition and such a strict containment would imply a strict inequality between* **P** *and* **PSPACE***.*

In a similar manner, we can show that FH-POMDP and IH-POMDP are **FPSPACE**-Complete, by reducing TQBF to FH-POMDP. We note that this section is also taken verbatim from [PT87].

Proof. We want to show that, given a totally quantified boolean formula

$$\exists x_1 \forall x_2, ..., \forall x_n \phi(x_1, x_2, ..., x_n),$$

we can construct an FH-POMDP instance and show that $\phi \in$ TQBF if and only if $V^{\pi^*}(s_0) \geq 0$.

Firstly, we assume that ϕ contains n boolean variables and m clauses. Next, for each pair $(i, j) \in [m] \times [n]$, we introduce the states $A_{i,j}, A'_{i,j}, T_{i,j}, T'_{i,j}$ and $F_{i,j}, F'_{i,j}$. Furthermore, we introduce $2m$ states $A_{i,n+1}, A'_{i,n+1}$. Essentially, we require each of three states A_{ij}, T_{ij}, F_{ij} to denote the value of x_j in C_i. Furthermore, the additional three states, $A^{'}_{ij}, T^{'}_{ij}, F^{'}_{ij}$ will denote scenarios in which we set a value of x_j in C_i such that C_i fails to be satisfied. We will go into more detail of this construction below. Finally, we introduce a dedicated starting state s_0.

Secondly, we construct observations Ω. We will construct the following observations. For each variable x_j, let

$$A_j = \{A_{ij} : i \in [m]\}, A'_j = \{A'_{ij} : i \in [m]\},$$
$$T_j = \{T_{ij} : i \in [m]\}, T^{'}_j = \{T^{'}_{ij} : i \in [m]\},$$
$$F_j = \{F_{ij} : i \in [m]\}, F^{'}_j = \{F^{'}_{ij} : i \in [m]\}.$$

Thus, we let

$$\Omega = \{s_0\} \bigcup_{j \in [n]} \{A_j\} \cup \{T^{'}_j\} \cup \{F_j\}.$$

Secondly, we introduce observations, Ω, such that taking a particular action a at a partition of states yields the same observation. ■

4 Complexity Theoretic and Cryptographic Implications of LARGE-FH-MDP-SEARCH

4.1 Large MDPs

We now introduce a new, related search problem that encodes fully-observable, finite-horizon MDPs with exponential states and actions. The aim of this search problem is capture a real-world scenario that arises often. One in which the optimization problem involves a state and action space that (possibly)cannot be fully traversed in an efficient manner. We define the LARGE-FH-MDP-SEARCH as follows.

Definition 18 (LARGE-FH-MDP-SEARCH). *LARGE-FH-MDP-SEARCH is the kind of MDP optimal policy search problems where the function machine is given an instance of a finite-horizon, fully-observable MDP whose total relation can be defined by:*

$$\textit{LARGE-FH-MDP-SEARCH} = \{\langle (|S|, |A|, P, |R|, s_0, \gamma, H), \pi^* \rangle : \forall (s,\pi) \in S \times \Pi, V^{\pi^*}(s) \geq V^{\pi}(s)\}.$$

Here,

- *$|S|, |A|$ are allowed to be exponential size as they are succinctly defined by the P circuit, as explained below for P. Notice that, since they have $2^{O(n)}$ sizes, their lengths encoded in binary takes only $O(n)$ sizes for input.*
- *$R = \{0,1\}^r \subseteq \mathbb{Q}$ is the set of all rewards.*
- *$P : S \times A \times S \times R \to [0,1] \cap \mathbb{Q}$ is the joint transition dynamics. Note that we define S and A implicitly through the P circuit here. Like other succinct definitions, one can think of P as listing inputs and outputs to P as a table (with the inputs even implied by some order instead of explicitly written down in the table. For example, to write down a function that maps 00 to 0, 01 to 1, 10 to 1 and 11 to 0, one only needs to write down 0110, as the order of the input from 00 to 11 can be implied). This allows us to implicitly define S and A with exponential sizes of $|x|, x \in$ LARGE-FH-MDP-SEARCH.*
- *$s_0 \in \{0,1\}^n$.*
- *$\gamma \in [0,1] \cap \mathbb{Q}$.*
- *H is the horizon and is written in binary.*

Remark 4. *The main difference between FH-MDP and LARGE-FH-MDP-SEARCH is that the latter succinctly described an exponentially-sized MDP. Furthermore, P succinctly describes the transition dynamics as it is represented as a circuit with rational valued, bounded outputs in $[0,1]$. Finally, we write H in binary to ensure that the runtime does not become pseudo-polynomial.*

4.2 Complexity Theoretic Hardness of LARGE-FH-MDP-SEARCH

Solving this problem is comparable to solving FH-MDP-SEARCH but with exponentially many variables.

Theorem 2. *LARGE-FH-MDP-SEARCH is* **FPSPACE***-Complete.*

Proof. Follows from lemmas 2 and 3. ∎

Lemma 2. *LARGE-FH-MDP-SEARCH is* **FPSPACE***-hard.*

Proof. We first prove that LARGE-FH-MDP-SEARCH can decide **PSPACE**-Complete languages. Let $L \subseteq \{0,1\}^*$ be decidable by a space $S(n) = poly(n)$ deterministic Turing Machine D. Let $x \in \{0,1\}^n$ be an input for D and let $G_{D,x}$ denote the configuration graph of D. We now describe the reduction to an instance $x' \in$ LARGE-FH-MDP-SEARCH as follows.

Here is its corresponding instance of LARGE-FH-MDP-SEARCH:

- For each configuration C_i in the graph $G_{M,x}$ assign a state, so $S = \{0,1\}^{O(S(n))}$ and $|S| = O(S(n))$
- $A = \{0\}$, so $|A| = 0$.
- $s_0 = G_{start}$
- $\gamma = 1$
- $H = O(S(n))$, as the horizon can be $2^{O(S(n))}$

We now describe both the reward function $R(s, a = 0)$

$$R(C_i, A = 0) = \begin{cases} 1 & \text{if there exists a legal transition from } C_i \to C_{accept} \\ 0 & \text{otherwise} \end{cases}.$$

Similarly, we describe the transition dynamics using the overall joint probability distribution (rather than the specific transition dynamics at each state, so they don't add up to 1 at each state. See remark 1 about this nuance) as follows

$$P(S = C_i, A = 0, S' = C_j, R = R(C_i, 0)) = \begin{cases} \frac{1}{2^{O(S(n))}} \text{ if there exists a legal transition from } C_i \to C_j \\ 0 \text{ otherwise} \end{cases}.$$

Note that the optimal policy π^* has a value of 1 if and only if there exists a path from $C_{start} \overset{p}{\rightsquigarrow} C_{accept}$. Therefore, the machine for LARGE-FH-MDP-SEARCH provides a solution of $(\pi^*, 1)$ if and only if $x \in L$. Thus, if we run the corresponding LARGE-FH-MDP-SEARCH constructed from x, either we find a π^* that has a value of 1, and with that we can recover the configuration history of x in poly-space and thus find an $x \in L$, or the value of the optimal policy found is not 1, then we say $x \notin L$. We have the power to reuse the poly-space and search through all x, so LARGE-FH-MDP-SEARCH is **FPSPACE**-hard. ■

Lemma 3. *LARGE-FH-MDP-SEARCH* $\in$ **FPSPACE**.

Proof. In this proof, we explicitly show that *Value iteration* can be used to find the optimal value of each state $s \in S$ in polynomial space. The resulting optimal value function can then be used to extract the optimal policy $\pi(s)$ for each $s \in S$ also in polynomial space.

The key issue with using Value iteration for the space claims above is that the original algorithm *explicitly stores* the value of each state $s \in S$ at any given point in the horizon. Unfortunately, if we store explicitly here with exponentially many variables, we will exceed the polynomial space limit. To overcome this issue, we succinctly represent the value function through a half-space, by writing our value function to be of form

$$V_t : \mathbb{R}^n \times \{0,1\}^n \to \mathbb{R}$$

for each value $t \in [H]$. The following algorithm shows value iteration based on this succinct representation explicitly.

Algorithm 3 ValueIteration($x = \langle |S|, |A|, P, R, s_0, \gamma, H \rangle$

```
$\alpha_0 \leftarrow \mathbf{0}$                                ▷ Initialize next parameter vector
for $t \in [2^H]$ do
    $\alpha_t \leftarrow \mathbf{0}$
    for $a \in A$ do                          ▷ Compute temporary vector for each action
        $\beta_a \leftarrow \mathbf{0}$                        ▷ Initialize temporary vector
        for $i \in [n]$ do
            for $s \in [2^{i-1}, 2^i - 1]$ do
                for $s' \in S$ do
                    $\beta_{a_i} \leftarrow \beta_{a_i} + P(s'|s,a) \cdot (R(s,a) + \gamma\alpha_{t-1} \cdot s')$
                end for
            end for
        end for
        $\alpha_t \leftarrow \max(\alpha_{t-1}, \beta_a)$              ▷ Update next parameter vector
    end for
end for
return $\alpha_H$
```

Algorithm 3 provably finds the optimal value of each state in $s \in S$. At $t = 0$, the algorithm initializes $\alpha_0 = \mathbf{0}$. Therefore,

$$V_0(s) = \alpha_0 \cdot s = 0 \quad \text{for all } s \in S,$$

which satisfies the Bellman equation as there are no further returns to be accumulated. Then, inductively, suppose that, at some horizon t, $V_t(s) = \alpha_t \cdot s$ is computed correctly. We show that $V_{t+1}(s) = \alpha_{t+1} \cdot s$ also holds. From the algorithm,

$$\alpha_{t+1} = \max_{a \in A} \sum_{s' \in \{0,1\}^n} P(s'|s,a) \cdot (R(s,a) + \gamma\alpha_t \cdot s'). \tag{2}$$

Plugging equation (2) into the Bellman equation, we get:

$$V_{t+1}(s) = \max_{a \in A} \left(\sum_{s' \in S} P(s'|s,a) \cdot [R(s,a) + \gamma V_t(s')] \right) = \alpha_{t+1} \cdot s.$$

Thus, by induction, the algorithm computes $V_t(s)$ correctly for each horizon t.

Claim 1. *The proposed value iteration algorithm needs only polynomial space.*

Proof (Claim 1). Everything in algorithm 3 only needs to be stored locally and each takes up no more than polynomial space trivially (as they are just input elements), other than β_{α_i} and α_t. This is because β_{α_i} and α_t are associated and need to be explicitly discussed because they are summations of exponentially many rewards, which are passed in as inputs. Let's consider the extreme case, where we take the max reward, whose length cannot exceed the size of n as it is passed in as input, and the corresponding value of the reward would be 2^n. Then, we also consider the extreme case where such a reward is never discounted and is added every iteration of our algorithm. So, it will be added for a total of $|S| \cdot 2^n \cdot |A| \cdot 2^H$ times. So, the largest possible value of each coordinate of β_{α_i} is

$$\begin{aligned}&\leq |S| \cdot 2^n \cdot |A| \cdot 2^H \cdot (2^n)\\ &\leq 2^{5n} \text{ because } |S|, |A| \leq 2^n, H \leq n,\end{aligned}$$

which means that, when represented in binary, this value is $\log(2^{5n}) = 5n = O(n)$-bit long. Since β_{α_i} and α_t are n-tuples with each position taking a max of $O(n)$ bits, we have the space complexity of these intermediate values to take up at most $O(n^2)$ space. ∎

Finally, it is easy to see that tracing out the optimal policy by just following the usual way based on the above value iteration algorithm is correct and uses no more than polynomial space (as we can reuse space after each state). ∎

Corollary 1. *Since both* LARGE-FH-MDP-SEARCH *and* FH-POMDP *are* **FPSPACE***-complete, we have shown that*

$$\textsf{LARGE-FH-MDP-SEARCH} = \textsf{FH-POMDP},$$

meaning there is a trade-off between observability (i.e. uncertainty given by the partial observability) and complexity (i.e. increase of time and space complexity resulted from the increase of sizes of states and actions in the LARGE *case).*

4.3 Cryptographic Hardness of LARGE-FH-MDP-SEARCH

Since LARGE-FH-MDP-SEARCH is **PSPACE**-complete, we must need some stronger cryptographic assumption than unbounded $i\mathcal{O}$ to say something interesting about the case where LARGE-FH-MDP-SEARCH can be computed efficiently. This is because of the following theorem:

Theorem 3 ([MR13]). $\mathbf{P} = \mathbf{NP} \implies \exists i\mathcal{O}$ *(unbounded). Intuitively, this is because the lexicographically first circuit that computes the same function can be found efficiently.*

Corollary 2. *In our case, if* LARGE-FH-MDP-SEARCH *can be computed efficiently, then* $\mathbf{P} = \mathbf{PSPACE} \implies \mathbf{P} = \mathbf{NP} \implies \exists i\mathcal{O}$. *In particular, just the existence of* $i\mathcal{O}$ *is not strong enough to separate* **P** *and* **PSPACE**.

A natural next thing stronger than $i\mathcal{O}$ is virtual black-box, defined as follows:

Definition 19 (Virtual Black-Box, **VBB**). *Informally,* **VBB** *gives the kind of obfuscation that nothing about the program other than inputs and outputs can be learned by anyone interacting with it (while the program works exactly the same as when it was unobfuscated). Formally, an $\mathcal{O}$ is a* **VBB** *if*

- ***Functionality:*** *For any $C \in \mathcal{C}$,*

$$\Pr_x[\mathcal{O}(C)(x) = C(x)] = 1.$$

- ***Security:*** *For any PPT D, there exists a PPT S_D such that*

$$\left|\Pr\left[D(\mathcal{O}(C)) = 1\right] - \Pr\left[S_D^C\left(1^\lambda\right) = 1\right]\right| \leq \varepsilon.$$

Remark 5. *Unfortunately, [BGI+01] showed that many variants of* **VBB** *are impossible to construct, thus proving that* **VBB** *cannot exist in general. However, lucky for us, there are still plausible special cases of* **VBB** *whose constructibilities are open questions. In particular, it is not known whether or not* **VBB** *is constructible for circuits used in* SVL *construction, which is part of the reason why it is still considered in a much later paper [BPR15]. Fortunately, such a* **VBB** *is sufficient for our purposes.*

Theorem 4 (Some Plausible **VBB** $\implies$ **P** $\neq$ **PSPACE**). *Polynomially secure* **VBB** *construtible for circuits in the* SVL *construction* $\implies$ **P** $\neq$ **PSPACE**.

Proof. Let D be a deterministic Turing Machine that, given an instance $x \in$ LARGE-FH-MDP-SEARCH, finds an optimal policy and value (π^*, k) for it. Then, we can say that

$$Pr[D(x) = (\pi^*, k)] = 1.$$

As before, $x = (|S|, |A|, P, R, s_0, \gamma, H)$ where:

- $S = \{0,1\}^n$, so $|S| = n$.
- $A = \{0\}$, so $|A| = 0$.
- $s_0 = 0^n$.
- $\gamma = 1$.
- $H = n$, so the horizon is 2^n.

Next, choose a value t uniformly at random from $\{0,1\}^n$. We now define both the reward function $R(s, a = 0)$

$$R(s, A = 0) = \begin{cases} 1 & \text{if } t > 0^n \text{ and } s = t - 1 \\ 1 & \text{if } t = 0^n \text{ and } s = 2^n - 1 \\ 0 & \text{otherwise} \end{cases},$$

and the description P of the transition dynamics as a joint probability distribution

$$P(S = s, A = 0, S' = s', R = R(s,0)) = \begin{cases} \frac{1}{2^n} \text{ if } s' = s+1 \\ 0 \text{ otherwise} \end{cases} .$$

Let $\mathcal{O}$ be some polynomially secure **VBB** constructible for circuits in the SVL construction. Suppose D is now challenged by the instance $c = (|S|, |A|, \mathcal{O}(P), \mathcal{O}(R), s_0, \gamma, H)$ and let E be the event that D queries a state s such that $R(s,0) = 1$. The reason why such an $\mathcal{O}$ is strong enough is because the P and R are in essence comparable in functionality to the successor circuits in an SVL instance. Note that by construction, there is only one such s. In particular,

$$\Pr[D(c) = (\pi^*, 1)] = \Pr[E] \cdot \Pr[D(c) = (\pi^*, 1)|E] + \Pr[\overline{E}] \cdot \Pr[D(c) = (\pi^*, 1)|\overline{E}].$$

Now, note that

$$\Pr[D(c) = (\pi^*, 1)|\overline{E}] = 0,$$

because if D does not query the value s such that $R(s,0) = 1$, then D does not return $(\pi^*, 1)$. Therefore, we can say that $\Pr[D(c) = (\pi^*, 1)|\overline{E}] = 0$. As a result, we can say that

$$\Pr[D(c) = (\pi^*, 1)] \le \Pr[E].$$

Since $D \in \mathbf{P}$, and $\mathcal{O}(R), \mathcal{O}(P)$ are secure against any polynomial-time machine, the information relating to t cannot be recovered from the descriptions of $\mathcal{O}(R), \mathcal{O}(P)$ with non-negligible probability. As a result, we can say that

$$\Pr[E] \le \frac{poly(n)}{2^n} < 1, \text{for sufficiently large } n.$$

However, we already showed that $Pr[D(c) = (\pi^*, 1)] = 1$ is necessary, so we have arrived at a contradiction. Therefore, if polynomially-secure $\mathcal{O}$ exists, there does not exist a poly-time machine for LARGE-FH-MDP-SEARCH. As LARGE-FH-MDP-SEARCH is **FPSPACE**-Complete, this implies that **FP** $\neq$ **FPSPACE**, which in turn implies that **P** $\neq$ **PSPACE** (see definition 17 for the nuance such that this implication is true). ■

5 Duality between IP and Statistical RL System (SRLS)

We now define our notion of a **SRLS** as follows.

Definition 20 (Statistical **RL** System)**.** *A* **SRLS** *is a 3-tuple* (x, A, E)*, where:*

- *x is an instance of LARGE-FH-MDP-SEARCH*
- *A is an efficient, probabilistic Turing Machine, representing the agent. On input x, A must compute the optimal Value function $V(s)$ for all $s \in S$ with probability $\ge 2/3$. Furthermore, A is able to make statistical queries to E from the query-set $Q = \{0,1,*\}^n \times A \cup \{*\} \times \{0,1,*\}^n \times R \cup \{*\} \to [0,1]$.*

- *E is a computationally unbounded Turing Machine that is given access to x representing the environment. Furthermore, when E recieves a query $q \in Q$, it computes the associated probability of the query and returns it to A*

Before proceeding, it is worthwhile to consider what a query $q \in Q$ represents. When we see any $*$, a wildcard character, appear in the query, we aggregate probability measures over the values $\{0, 1\}$ in the associated position. Thus, a query of the form $q = (1 \circ *^{n-1}, a, 0 \circ^* n - 1, r)$ encodes $Pr(S \in [2^{n-1}, 2^n - 1], A = a, S' \in [0, 2^{n-1} - 1], R = r)$. Furthermore, note that we can make much more refined queries than the example provided.

Thus, a response to single query is capable of encoding an exponential number of calls to P. We ensure that this, potentially exponential-time, computation is off-loaded to E, the unbounded Turing Machine. We will see this through the following lemma.

Lemma 4 (Coalescing Exponential Queries). *$R(v, a)$ and $P(v'|v, a)$ can be efficiently computed for any values $v, v \in \{0, 1, *\}^n$.*

5.1 Main Theorem: Explicit Duality between SRLS and IP

Theorem 5. *Let L be a language and S be its associated search problem. An Interactive-Proof system decides L if and only if there exists a* **SRLS** *that solves S.*

Proof. Follows from lemmas 5, 6, and 7. ■

Now, we proceed to prove the lemmas. Note that what we are trying to show with 5 is algorithmic in nature as, complexity theoretically, LARGE-FH-MDP-SEARCH has already been shown to be **FPSPACE**-complete as an intermediate between both (). So, the lemmas here are to have an explicit construction between the two frameworks that we have shown to be equivalent.

Lemma 5 (**IP** $\leq$ LARGE-FH-MDP-SEARCH). *Every $x \in L \in$ **IP** can be reduced to the problem of finding the optimal value function for a LARGE-FH-MDP-SEARCH as follows:*

Proof. Let $L \subseteq \{0, 1\}^*$ be a language that is decided by an **IP** system V, P. Then, for any $x \in \{0, 1\}^*$,

- $x \in L \implies Pr[(V, P)(x) = 1] \geq 2/3$
- $x \notin L \implies Pr[(V, P)(x) = 1] \leq 1/3$

Notice that if V runs in time $T(n)$, where $n = |x|$, then V utilizes at most $T(n)$ space, even if V makes queries to P.

To that end, let $G_{V,x}$ denote the configuration graph of V on input x, C^x_{start} be the start configuration and let C^x_{accept} be the unique accepting configuration. Note that any configuration can be represented with $O(T(n))$ bits. Now, let us construct a LARGE-FH-MDP-SEARCH instance x as follows:

- $S = \{0,1\}^{O(T(n))}$, so $|S| = n$
- $A = \{0\}$
- $R(S = C_i, A = 0) = \begin{cases} 1 \text{ if there exists a legal transition from } C_i \text{ to } C^x_{accept} \\ 0 \text{ otherwise} \end{cases}$
- $P(S = C_i, A = 0, C_j, R = (C_i, A = 0)) = \begin{cases} \frac{1}{2^{O(T(n)}} \cdot \delta(C_i, C_j) \text{ where } \delta(C_i, C_j) \text{ is defined by } \delta \\ 0 \text{ otherwise} \end{cases}$
- $s_0 = C^x_{start}$
- $\gamma = 1$
- $H = O(T(n))$

To clarify, when we say $\delta(C_i, C_j)$ is defined by δ, we mean that δ is the transition function defined by the verifier V. Thus, we abuse notation in this instance to encode the probability that C_i moves to C_j.

Claim 2. *$x \in L \implies V^*_H(s_0) \geq 2/3$, and that $x \notin L \implies V^*_H(s_0) \leq 1/3$.*

Note that if $x \in L$, then at least 2/3 of all computation paths from C^x_{start} lead to C^x_{accept}. Due to our construction, we have a discount factor of 1, which means that the optimal value of s_0 will be at least 2/3.

Furthermore, if $x \notin L$, then $\leq 1/3$ of all computation paths from C^x_{start} lead to C^x_{accept}. Once again, by way of our construction, this imples that the optimal value $V^*_H(s_0) \leq 1/3$. Thus, every Interactive-Proof system loaded with an input x can be encoded, in log-space, as a LARGE-FH-MDP-SEARCH instance. ■

We are now ready to define a protocol to find the optimal value function of any instance of LARGE-FH-MDP-SEARCH instance x. When combined with the reduction above, we can reduce every decision by interactive-proof to that of finding the optimal value of a large, finite-horizon MDP through **SRLS**.

Lemma 6 (LARGE-FH-MDP-SEARCH $\leq$ **SRLS**)**.** *Every $x \in$ LARGE-FH-MDP-SEARCH can be reduced to computing an* **SRLS***.*

> NOTE: While we believe it is possible to solve LARGE-FH-MDP-SEARCH through a **SRLS** due to the previous structural result, we do not believe that our algorithm outlined is correct because we are outlining a deterministic protocol here, rather than a randomized protocol (which would imply **dIP** = **NP** = **PSPACE**).

Proof. Note that we have quite a few hurdles to overcome. The primary hurdle is the exponential number of states, as $S = \{0,1\}^n$. Thus, utilizing Value Iteration in the manner described earlier will not be viable, as A must be an efficient, probabilistic Turing Machine. The second hurdle, is that H is written in binary, and so we cannot iterate through the horizon iteratively. At a high level, our strategy will be as follows. We will aggregate the state space $S = \{0,1\}^n$ into the following:

$$V = \{*^{i-1} \circ 1 \circ *^{n-i}\}_{i=0}^{n}.$$

Thus, each element $v \in V$ represents a set of 2^{n-1} many states $s \in S$. Thus, $v = *^{i-1} \circ 1 \circ *^{n-i}$ represents the set of all states $s \in S$ with 1 in the i^{th} bit position. We will then utilize queries to E to offload exponentially long computations to E. For the explicit algorithm, see algorithm 4 in Appendix A.

Algorithm 4 correctly computes LARGE-FH-MDP-SEARCH. To compute the optimal value of any particular state $s \in S$, we can utilize $\{\alpha_{vH}\}_{v \in V}$ in the following way:

$$V_H^*(s) = \max_{a \in A}[R(s,a) + \gamma \sum_{v \in V} P(v|s,a) \cdot \alpha_{vH}]$$

Thus, when given continued access to E, A is able compute the optimal value for any $s \in S$ efficiently. To see why this is correct, we will once again use induction over $t \in H$ and show that the algorithm satisfies the Bellman-Optimality equation for every $t \in [H]$.

To show the correctness, observe that, when $t = 1$, in this case for all $s \in S$, we have that

$$V_1^*(s) = \max_{a \in A}[R(s,a) + \gamma \sum_{v \in V} P(v|s,a) \cdot \alpha_{v0}] = \max_{a \in A}[R(s,a)].$$

Then, for any horizon $t \in [H]$, notice that

$$V_t(s) = \max_{a \in A}[R(s,a) + \gamma \sum_{v \in V} Pr(v|t,a) \cdot \alpha_{v,t-1}],$$

which can be rewritten as

$$V_t(s) = \max_{a \in A}[R(s,a) + \gamma(\sum_{v \in V} \sum_{s \in S} Pr(S' = s'|S = s, A = a)) \cdot \alpha_{v,t-1}].$$

Now, inductively, assuming that $\alpha_{v,t-1}$ accurately captures the value of superstate $v \in V$ up to time $t-1$, we can substitute its value and see that

$$V_t(s) = \max_{a \in A}[R(s,a) + \gamma \sum_{s' \in S} Pr(S' = s'|S = s, A = a) \cdot V_{t-1}(s')],$$

where the final equation holds true because the aggregate value $V_t(u)$ captures the notion of an "averaged" value across all $s \in u$. Thus, A finds the optimal value function.

Algorithm 4 is efficient. Note that as A performs binary search over the horizon H(that is written in binary) to find when to stop value iteration. Thus, there are $O(\log_2(H)) = O(|H|)$ many value iterations performed. The time complexity of value iteration is itself $O(n^2 \cdot |A|)$, because we consider pairwise discounted return over superstates $v, v' \in V$ in each iteration. Thus, the total time complexity is $O(|H| \cdot n^2 \cdot |A|)$. ■

Lemma 7 (**SRLS** $\leq$ **IP**). *Any* **SRLS** *can be formulated as an* **IP** *system.*

> NOTE: We are still not confident with this result, because this lemma shows that there exists a deterministic IP-system that decides LARGE-FH-MDP-SEARCH. This would imply that **NP** = **PSPACE**, which is unlikely.

Proof. Note that with a small change to the construction we have above, we can prove this statement in the following manner. Suppose we wish to decide if $x \in L$, where $L \in$ **PSPACE**. Then, we take its corresponding $S(n)$-space machine and reduce it to an instance of LARGE-FH-MDP-SEARCH as described before. Then, we can use the **SRLS** to find if the optimal value at C^x_{start} is 1 by computing $V_H(s_0)$ of the associated LARGE-FH-MDP-SEARCH. As the additional step allows this protocol to decide strings, we can convert the SRL system into an IP-system. ■

Lemma 8. *The correspondence between Interactive-Proof systems and* **SRLS***s implies that* **IP** = **PSPACE**.

6 Implications of the Duality Theorem

6.1 Zero-Knowledge Proof Systems for RL Systems

Definition 21 (Zero-Knowledge Proof Systems, **ZKPS**). *A* **ZKPS** *for $L \in$* **IP** *is one such that all of the following are satisfied:*

- *(Completeness) If $x \in L$,* $\Pr[(P, V)[x] = 1] \geq 1 - \mathsf{neg}$.
- *(Soundedness) If $x \notin L$,* $\Pr[(P, V)[x] = 1] \leq \frac{1}{2}$.
- *(Zero Knowledge)* $\forall V', \exists S_{\mathsf{PPT}}, \forall x \in L, \mathsf{VIEW}_{P,V'}[x] = S_{\mathsf{PPT}}[x]$. *In other words, the transcript of the interactions between provers and verifiers need to convey no knowledge, where knowledge is defined as anything that cannot be simulated using a probabilistic poly-time machine (*PPT*).*

By definition 21, a **ZKPS** is defined on an **IP** system. So, a natural question to ask is, now that we have defined a **SRLS** as a learning system complexity theoretically equivalent to **IP**, how we can learn with **ZK**. To do so, we need to understand specific properties that make an **IP** problem **ZKPS**, such as the one below:

Theorem 6 ([GMW91b]). *Suppose one-way functions exist (or any other ways to commit a secret bit), the corresponding* **IP** *system of any* **NP** *problems can be conducted with* **ZK**.

Theorem 7. *If one-way functions exist (or any other ways to commit a secret bit), then the optimal policy for fully-observable MDPs can be learned with zero-knowledge (or, in general, any* **NP** *problems' equivalent* **SRLS** *can be learned with* **ZK***).*

Proof. As FH-MDP and IH-MDP are **FP**-Complete, their associated decision problem of finding whether $V^*(s_0) \geq k$ for some k is in **P**. Thus, there exists a **ZKPS** to decide this problem by [GMW91b]. Then, on an input instance x, the can then be converted into a LARGE-FH-MDP-SEARCH instance, whose learning / optimization can be learned by an **SRLS** carrying over the same **ZK** guarantee. ■

Remark 6. *By [GMW91b], we would already know that the associated decision problems of FH-MDP and IH-MDP being in* **P** *yield an associated* **ZK** *system. What's interesting about theorem 7 is that we can also formulate any of these problems into a* **SRLS** *based on LARGE-FH-MDP-SEARCH, as well. In particular, we take the verifier of the* **ZK** *system as an input to LARGE-FH-MDP-SEARCH, and by the explicit construction above, we say that the verifier wouldn't learn anything in its equivalent* **SRLS** *by the* **ZK** *guarantee. In other words, theorem 7 shows that* **RL** *can be achieved in a privacy-preserving manner, despite seemingly counter-intuitive.*

7 Conclusions and Future Directions

7.1 Works / Errors Remained within This Project

The most immediate concern is to validate the protocol laid out by the **SRLS**. The authors of this paper are highly suspicious of it as it is a deterministic protocol that solves a PSPACE-hard problem. We believe that there is some randomization that is missing in the protocol, and integrating it to correct the protocol would be the first step.

Once this correction is made, the lemmas that follow will hold. We have shown that tractable cases for fully-observable **RL** problems can be shown in a privacy-preserving manner by utilizing the reduction to LARGE-FH-MDP-SEARCH. However, designing an explicit zero-knowledge proof system for FH-MDP and IH-MDP will be important to complete this task.

7.2 SRLS Connection with Computational Learning Theory

The statistical **RL** model allows the agent A to make statistical queries to E with perfect accuracy. While this scenario is acceptable in a theoretical realm, it is not viable in a real-world setting. To make the **SRLS** viable in a real-world setting, we will need to introduce a tolerance parameter $\tau \in [0, 1]$ that encodes an error range of acceptable queries. When E is given (q, τ) it will have to compute q with said tolerance. To implement this functionality, we will no longer be able to think of E as being computationally unbounded. To cope with this issue, E will have to take samples from the distribution imposed by P and compute an estimate $\hat{p}$ of q. The study of bounding the computational abilities of E will require a further-integrated study of the intersection of reinforcement-learning theory and computational-complexity theory. One direction the authors suggest is studying the similarilities between our proposed **SRLS** and Statistical Query Models.

7.3 SRLS for Quantum

Another direction is to consider E as being a **BQP** machine that allows *quantum statistical queries*. This direction would require a slight modification to the query-set, where states (represented as qubits) in super-position may be queried along with the $*$ character.

There is also a known result about Quantum **ZKPS** that states that **ZKPS** exists for problems in **QMA** by [BJSW16], so similar things about **ZK** learning with **SRLS** can possibly said about quantum analogues.

7.4 SRLS for an RL Analogue of Boosting

Finally, a natural extension to the **SRLS** maybe considered, where multiple agents $\{A_i\}_{i\in[k]}$ are allowed to interact with each other. We suspect that natural analogues to boosting, in **PAC**-Learning, can be derived by invoking parallel-repetition theorems.

8 Acknowledgements

All the work in this written report other than the ones cited are originally conducted by the two authors within the scope of COMS 6261 Advanced Cryptography (Spring 2024), without other contributions from outside of the class.

References

[BGI$^+$01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual international cryptology conference*, pages 1–18. Springer, 2001.

[BJSW16] Anne Broadbent, Zhengfeng Ji, Fang Song, and John Watrous. Zero-knowledge proof systems for qma. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 31–40. IEEE, 2016.

[BJSW20] Anne Broadbent, Zhengfeng Ji, Fang Song, and John Watrous. Zero-knowledge proof systems for qma. *SIAM Journal on Computing*, 49(2):245–283, 2020.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1480–1498. IEEE, 2015.

[GMW91a] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, jul 1991.

[GMW91b] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[MR13] Tal Moran and Alon Rosen. There is no indistinguishability obfuscation in pessiland. *Cryptology ePrint Archive*, 2013.

[PT87] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

A Appendix A: Explicit Algorithm that Reduces from LARGE-FH-MDP-SEARCH to SRLS

Algorithm 4 $A^E(x = \langle |S|, A, P, R, s_0, \gamma, H \rangle)$

```
$L = 0$, $U = 2^{\lceil \log_2(H) \rceil}$    ▷ $U$ is the smallest power of 2 larger than $H$, and both $L, U$ are written in binary
Initialize $V$ as defined above.
for $v \in V$ do
    Let $\alpha_{v0} = 0$
end for
while $L < U$ do
    $M = \frac{L+U}{2}$
    for $t \in [M]$ do
        for $a \in A$ do
            for $v_i \in V$ do
                $\beta_{\alpha v} \leftarrow \mathbf{0}$
                for $v_j \in V$ do
                    $\beta_{av_i} \leftarrow \beta_{av_i} + P(v_j | v_i, a) \cdot (R(v_i, a) + \gamma \alpha_{t-1} \cdot v_j)$
                end for
                if $t = M/2$ then
                    for $v \in V$ do
                        Set global $\alpha_{v,t/2}$
                    end for
                end if
                $\alpha_{vt} = max(\alpha_{v,t-1}, \beta_{\alpha v})$    ▷ Element-wise comparison between two vectors.
            end for
        end for
    end for
    if $\max_{v \in V} |\alpha_{vt} - \alpha_{v(t/2)}| = 0$ then
        set $U = M$
    else
        $L = M + 1$
    end if
end while
Return $\{\alpha_{vH}\}_{v \in V}$
```