Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)

Privacy Leakage of Search-Based Multi-Agent Planning Algorithms

Michal Štolba, Daniel Fišer, Antonín Komenda
{michal.stolba,daniel.fiser,antonin.komenda}@aic.fel.cvut.cz
Department of Computer Science, Faculty of Electrical Engineering,
Czech Technical University in Prague, Czech Republic

Abstract

Privacy-Preserving Multi-Agent Planning (PP-MAP) has recently gained the attention of the research community, resulting in a number of PP-MAP planners and theoretical works. Many such planners lack strong theoretical guarantees, thus in order to compare their abilities w.r.t. privacy, a versatile and practical metric is crucial. In this work, we propose such a metric, building on the existing theoretical work. We generalize and implement the approach in order to be applicable on real planning domains and provide an evaluation of state-of-the-art PP-MAP planners over the standard set of benchmarks. The evaluation shows that the proposed privacy leakage metric is able to provide a comparison of PP-MAP planners and reveal important properties.

Introduction

Multi-agent planning (MAP) (Durfee 1999) comes in multiple flavors such as Dec-POMDPs (Oliehoek and Amato 2016) or Deterministic MAP (DMAP) (Brafman and Domshlak 2008; Torreño et al. 2017), each focusing on different aspects of MAP. One of the main rationales behind multi-agent planning is that the planning agents have certain private knowledge necessary to solve the planning problem. The agents are not willing to share such information but they have to use it in order to be able to find a joint solution to the PP-MAP problem.

In recent years, a number of privacy-preserving planning techniques have emerged, while in (Tožička, Štolba, and Komenda 2017) the authors have shown that for the most common MAP paradigms (including state-space search, such as MAFS (Nissim and Brafman 2014)) it is not possible to achieve complete (strong) privacy, efficiency, and completeness at the same time. This means that any practical privacy-preserving planner is essentially bound to leak some private information. In order to compare such planners in the context of privacy, it is necessary to quantify such private information leakage.

A definition of privacy leakage metric has been addressed in two theoretical publications. The approach of (Van Der Krogt 2009) bases the leakage metric on the number of plans of an agent compatible with the communicated information, whereas the approach of (Štolba, Tožička, and Komenda 2017) (denoted as STK from now on) uses the difference between the number of transition systems of the agent compatible with the information available to the adversary before and after the planning process. The former approach assumes the knowledge of the agent's problem and therefore can be used only by the agent itself. The latter is more general, as it is based purely on the knowledge available to the adversary. Both approaches were so far proposed only in theory. Both mentioned works are theoretical and thus were never used to actually measure privacy leakage of a planning algorithm on a set of benchmarks. In this work, we fill this gap.

The paper is structured as follows. We first formalize MAP (including a running example) and privacy leakage definition based on STK. Then we apply the privacy leakage on search-based algorithms and formally define the search tree structure, which we use to propose novel algorithms for detecting various sources of privacy leakage. We generalize the computation of privacy leakage to arbitrary sizes of variable domains and propose a novel SAT-based technique to compute possible transition systems of an action given arbitrary combinations of the sources of leakage. We continue by introducing a novel MILP-based technique to compute the actual privacy leakage. Finally, we implement and evaluate the proposed techniques in an established MAP planning framework MAPlan (Fišer, Štolba, and Komenda 2015) and compare privacy leakage of its variants (including MAFS (Nissim and Brafman 2014) and Secure-MAFS (Brafman 2015)) on the CoDMAP (Štolba, Komenda, and Kovacs 2016) benchmark set. This is the first work to practically compare PP-MAP algorithms based on a quantified leakage of private information.

Multi-Agent Planning

For a set of agents $\mathcal{A}$, a PP-MAP problem $\mathcal{M} = \{\Pi_i\}_{i=1}^{|\mathcal{A}|}$ is a set of agent problems, where for an agent $\alpha_i \in \mathcal{A}$ the agent problem is

$$\Pi_i = \left\langle \mathcal{V}_i = \mathcal{V}^{\mathsf{pub}} \cup \mathcal{V}^{\mathsf{priv}_i}, \mathcal{O}_i = \mathcal{O}^{\mathsf{pub}_i} \cup \mathcal{O}^{\mathsf{priv}_i}, s_I^i, s_\star \right\rangle,$$

where $\mathcal{V}_i$ is a set of variables s.t. each $V \in \mathcal{V}_i$ has a finite domain $\mathsf{dom}(V)$, $\mathcal{V}^{\mathsf{pub}}$ is the set of public variables (with all values public), common to all agents, and $\mathcal{V}^{\mathsf{priv}_i}$ is the set of variables private to α_i (with all values private), such that

$\mathcal{V}^{\mathsf{pub}} \cap \mathcal{V}^{\mathsf{priv}_i} = \emptyset$ and $\mathcal{V}^{\mathsf{priv}_i} \cap \mathcal{V}^{\mathsf{priv}_j} = \emptyset$ for all $i \neq j$. A complete assignment over $\mathcal{V} = \mathcal{V}^{\mathsf{pub}} \cup \bigcup_{i=1}^{|\mathcal{A}|} \mathcal{V}^{\mathsf{priv}_i}$ is a *state*, partial assignment over $\mathcal{V}$ is a *partial state*.

We use $s[V]$ to denote the value of a variable V in the (partial) state s, $s[\mathcal{V}']$ to denote the state s restricted to $\mathcal{V}' \subseteq \mathcal{V}_i$, and $\mathsf{vars}(s)$ to denote the set of variables with value defined in s. The state s_I is the global initial state and $s_I^i = s_I[\mathcal{V}^i]$ is the initial state of agent α_i. A partial state $s_\star$ defined over $\mathcal{V}^{\mathsf{pub}}$ represents the goal condition, i.e., s is a goal state iff $s_\star = s[\mathsf{vars}(s_\star)]$.

The set $\mathcal{O}^i$ is a set of actions of α_i, $\mathcal{O}^{\mathsf{priv}_i}$ is a set of private actions and $\mathcal{O}^{\mathsf{pub}_i}$ is a set of public actions of α_i. The sets $\mathcal{O}^{\mathsf{pub}_i}$, $\mathcal{O}^{\mathsf{priv}_j}$ are pairwise disjoint for all i, j. An action is defined as $a = \langle \mathsf{pre}(a), \mathsf{eff}(a) \rangle$, where $\mathsf{pre}(a)$ and $\mathsf{eff}(a)$ are partial states over $\mathcal{V}_i$ representing the precondition and the effect respectively. An action a is public if $\mathsf{vars}(\mathsf{pre}(a)) \cap \mathcal{V}^{\mathsf{pub}} \neq \emptyset$ or $\mathsf{vars}(\mathsf{eff}(a)) \cap \mathcal{V}^{\mathsf{pub}} \neq \emptyset$, otherwise a is private. A public action may also have private preconditions or effects.

An action a is applicable in a state s if $s[\mathsf{vars}(\mathsf{pre}(a))] = \mathsf{pre}(a)$ and the application of a in s, denoted as $s' = a \circ s$ results in a state s' s.t. $s'[V] = \mathsf{eff}(a)[V]$ if $V \in \mathsf{vars}(\mathsf{eff}(a))$ and $s'[V] = s[V]$ otherwise. Let $\pi = (a_1, ..., a_k)$ be a sequence of actions, if a_1 is applicable in s_0 and for each $1 \leq i \leq k$, a_i is applicable in s_{i-1} and $s_i = a_i \circ s_{i-1}$, then we denote $s_k = \pi \circ s_0$ the application of π on s_0 and s_k is the resulting state.

The public projection of a (partial) state s is $s^\triangleright = s[\mathcal{V}^{\mathsf{pub}}]$. The public projection of an action $a \in \mathcal{O}^{\mathsf{pub}}$ is $a^\triangleright = \langle \mathsf{pre}(a)^\triangleright, \mathsf{eff}(a)^\triangleright \rangle$. The public projection of Π_i is

$$\Pi_i^\triangleright = \left\langle \mathcal{V}^{\mathsf{pub}}, \mathcal{O}_i^\triangleright = \{a^\triangleright | a \in \mathcal{O}^{\mathsf{pub}_i}\}, s_I^\triangleright, s_\star^\triangleright \right\rangle$$

The solution to Π_i is a sequence π_i of actions from $\mathcal{O}_i \cup \bigcup_{j \neq i} \mathcal{O}_j^\triangleright$, s.t. $s_k = \pi_i \circ s_I$ and s_k is a goal state. We define $\pi^\triangleright = (a_1^\triangleright, ..., a_k^\triangleright)$ with all private actions omitted to be the public projection of π_i. The global solution to $\mathcal{M}$ is a set of plans $\{\pi_i\}_i^{|\mathcal{A}|}$ such that each π_i is a local solution to Π_i and all π_i are equivalent w.r.t. the public actions, formally $\pi_i^\triangleright = \pi_j^\triangleright$ for all i, j.

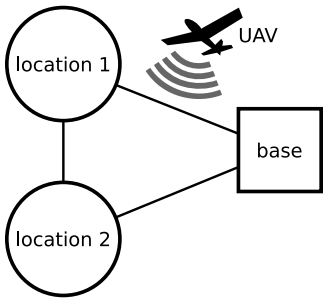


Figure 1: The UAV example PP-MAP problem.

Example. Here we introduce a running example problem, originally conceived by STK, and use it throughout the paper to illustrate important concepts. The example is based on a military scenario where a UAV must survey locations and refuel at the base of a coalition partner. The partners do not want to share the information about surveyed locations and about base fuel supply. The problem is modeled using the following variables:

- $\mathcal{V}^{\mathsf{pub}}$: $\mathbf{f}$ UAV has fuel, $\mathbf{c}$ mission is complete
- $\mathcal{V}^{\mathsf{priv}}_{\mathsf{UAV}}$: l1,l2 location 1,2 is complete
- $\mathcal{V}^{\mathsf{priv}}_{\mathsf{base}}$: s base has enough supplies

All variables are binary, i.e., with values from $\{\mathsf{true}, \mathsf{false}\}$. In the initial state, all variables are false except for $\mathsf{s} = \mathsf{true}$, $\mathbf{c} = \mathsf{true}$ must hold in the goal state. The public actions of α_{UAV} can be formulated as follows:

- Survey Location 1:
  $\mathsf{SL1} = \langle \{\mathbf{f} = \mathsf{true}\}, \{\mathsf{l1} = \mathsf{true}, \mathbf{f} = \mathsf{false}\} \rangle$
- Survey Location 2:
  $\mathsf{SL2} = \langle \{\mathbf{f} = \mathsf{true}\}, \{\mathsf{l2} = \mathsf{true}, \mathbf{f} = \mathsf{false}\} \rangle$
- Complete mission:
  $\mathsf{C} = \langle \{\mathsf{l1} = \mathsf{true}, \mathsf{l2} = \mathsf{true}\}, \{\mathbf{c} = \mathsf{true}\} \rangle$

And the public actions of α_{base} are:

- Refuel:
  $\mathsf{R} = \langle \{\mathbf{f} = \mathsf{false}, \mathsf{s} = \mathsf{true}\}, \{\mathbf{f} = \mathsf{true}, \mathsf{s} = \mathsf{false}\} \rangle$
- Refuel and Resupply:
  $\mathsf{RR} = \langle \{\mathbf{f} = \mathsf{false}, \mathsf{s} = \mathsf{false}\}, \{\mathbf{f} = \mathsf{true}, \mathsf{s} = \mathsf{true}\} \rangle$

The public variables are $\mathbf{c}, \mathbf{f}$ (bold) and thus the public projections of actions are:

- $\mathsf{SL1}^\triangleright = \mathsf{SL2}^\triangleright = \langle \{\mathbf{f} = \mathsf{true}\}, \{\mathbf{f} = \mathsf{false}\} \rangle$
- $\mathsf{C}^\triangleright = \langle \{\}, \{\mathbf{c} = \mathsf{true}\} \rangle$
- $\mathsf{R}^\triangleright = \mathsf{RR}^\triangleright = \langle \{\mathbf{f} = \mathsf{false}\}, \{\mathbf{f} = \mathsf{true}\} \rangle$

Because $\mathsf{SL1}^\triangleright, \mathsf{SL2}^\triangleright$ and $\mathsf{R}^\triangleright, \mathsf{RR}^\triangleright$cannot be distinguished, we denote the projected actions simply as $\mathsf{SL}^\triangleright$ and $\mathsf{R}^\triangleright$ respectively.

Privacy Leakage

Let us first state a number of assumptions we place on the agents and their interaction, as is usual in the Secure Multiparty Computation literature.

Semi-honest agents. The agents adhere to the algorithm and the communication protocol but try to infer as much private knowledge as possible.

Knowledge of the algorithm. The adversary knows the planning algorithm of the agent, but we do not assume any particular heuristic and thus the adversary does not exploit any information gained from the heuristic computation except for the heuristic value itself.

FIFO communication. Each communication channel between a pair of agents is first-in, first-out, i.e., the messages are received in the order they were sent.

Colluding adversaries. We refer to the agent trying to hide information as agent $\alpha = \alpha_i$ for some i (the agent). We model all other agents as a single agent (the adversary), which is common in Secure Multiparty Computation, as all the agents can collude and combine their knowledge in order to infer more private information. We omit the index i when referring to the agent is clear from the context.

Publicly known bounds. We assume that there are publicly known bounds on the size of the agent's problem, in

particular, $|\mathcal{V}^{\mathsf{priv}_i}| \leq p$ and $|\mathsf{dom}(V)| \leq d$ for all $V \in \mathcal{V}^{\mathsf{priv}_i}$.

We adopt (and later generalize) the leakage metric of STK, which is itself based on (Smith 2009), as follows. First, it is important to understand what information is the adversary given and what additional information it obtains through the planning process. The a priori information is a tuple $I_{\mathsf{apriori}} = \langle \Pi^{\triangleright}, \pi^{\triangleright}, p, b \rangle$ where $\Pi^{\triangleright}$ is the public projection of the problem and $\pi^{\triangleright}$ of the solution plan, p and b are the publicly known bounds on the number of private variables and on the size of private variable domains respectively.

The information obtained by the adversary is a sequence of messages exchanged between the agents $M = (m_1, ..., m_k)$. The posterior information, that is, the additional information available to the adversary after the execution of the planning process, is a tuple $I_{\mathsf{post}} = \langle \Pi^{\triangleright}, \pi^{\triangleright}, p, b, M \rangle$.

Based on STK in order to quantify the information, we associate the a priori information I_{apriori} with a random variable representing the uncertainty of the adversary about the agent's input of the planning algorithm (i.e., the problem Π_i and its respective transition system. We denote $\tau(I_{\mathsf{apriori}})$ the number of transition systems compatible with I_{apriori}. Assuming uniform distribution of the inputs the probability of any particular transition system being the input of α is $1/\tau(I_{\mathsf{apriori}})$. Converting to an information measure ((Smith 2009) uses min-entropy) yields $\log \frac{1}{P(I_{\mathsf{apriori}})} = \log \tau(I_{\mathsf{apriori}})$, where $\log$ denotes base-2 logarithm.

Similarly, we associate the public output, i.e., the communicated messages with a random variable I_{post} representing the information including new observations. The conditional probability $P(I_{\mathsf{apriori}}|I_{\mathsf{post}})$ then corresponds to the probability of a transition system being the hidden input with respect to all available information I_{post}. The respective information measure is then conditional min-entropy $\log \frac{1}{P(I_{\mathsf{apriori}}|I_{\mathsf{post}})} = \log \tau(I_{\mathsf{post}})$ where $\tau(I_{\mathsf{post}})$ is the number of transition systems compatible with I_{post}.

The final information leakage is computed as

$$\log \tau(I_{\mathsf{apriori}}) - \log \tau(I_{\mathsf{post}})$$

By observing that each action contributes to the number of possible transition systems independently, we can focus on the number of transition systems $\tau_{\mathsf{post}}(a)$ represented by each individual action $a \in \mathcal{O}^{\mathsf{pub}}$ and compute the number of transition system as

$$\tau(I_{\mathsf{post}}) = \prod_{a^{\triangleright} \in \mathcal{O}^{\triangleright}} \tau_{\mathsf{post}}(a) \qquad (1)$$

and analogously for $\tau(I_{\mathsf{apriori}})$.

Let us now define private information leakage based on the properties of actions, originally introduced by STK and generalized in this work. A single projected action $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ represents all actions $a_k \in \mathcal{O}^{\mathsf{pub}}$ such that $a_k[\mathcal{V}^{\mathsf{pub}}] = a^{\triangleright}$, as in Figure 2. We define the possible properties of $a^{\triangleright}$ as follows.

Definition 1. Let $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ and $\mathcal{O}^{\triangleright}_a = \{a' \in \mathcal{O}^{\mathsf{pub}} | a^{\triangleright} = a'^{\triangleright}\}$, we say $a^{\triangleright}$ is

Init-applicable (ia) if there exists $a \in \mathcal{O}^{\triangleright}_a$ such that $\mathsf{pre}(a) = s_I[\mathsf{vars}(\mathsf{pre}(a))]$.

Not-init-applicable (nia) if for all $a \in \mathcal{O}^{\triangleright}_a$ holds $\mathsf{pre}(a) \neq s_I[\mathsf{vars}(\mathsf{pre}(a))]$.

Privately-independent (pi) if there exist $V \in \mathcal{V}^{\mathsf{priv}}$ and $a \in \mathcal{O}^{\triangleright}_a$ such that $V \notin \mathsf{vars}(\mathsf{pre}(a))$, or for all values $v \in \mathsf{dom}(V)$ there is some $a' \in \mathcal{O}^{\triangleright}_a$ such that $v = \mathsf{pre}(a')[V]$.

Privately-dependent (pd) if there exist $V \in \mathcal{V}^{\mathsf{priv}}$ and $a \in \mathcal{O}^{\triangleright}_a$ such that $V \in \mathsf{vars}(\mathsf{pre}(a))$ and there exists some value $v \in \mathsf{dom}(V)$ such that $v \neq \mathsf{pre}(a')[V]$ for all $a' \in \mathcal{O}^{\triangleright}_a$.

Privately-nondeterministic (pn) if there exist $V \in \mathcal{V}^{\mathsf{priv}}$ and $a, a' \in \mathcal{O}^{\triangleright}_a$ such that $V \in \mathsf{vars}(\mathsf{eff}(a))$, $V \in \mathsf{vars}(\mathsf{eff}(a'))$, and $\mathsf{eff}(a)[V] \neq \mathsf{eff}(a')[V]$.

Informally, an action is init-applicable if it is applicable in the initial state (where the value of every $V \in \mathcal{V}^{\mathsf{priv}_i}$ is not known but fixed to some v_0). An action is privately-independent if for some V it is applicable on a state regardless of the value of V, privately-dependent if it is applicable only for some values of V and not applicable for some other values of V. Finally, an action is privately-nondeterministic if its application can result in two different values of V.

Clearly, the defined properties are not exhaustive and thus the proposed leakage metric is a lower bound and considering additional properties (or different techniques of possible transition systems reduction) might increase the computed leakage.

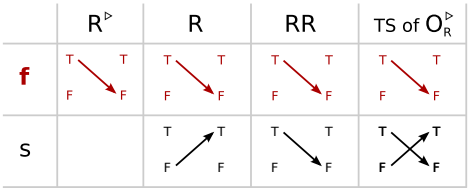


Figure 2: Example of public projection and related transition systems of the resupply action. The values true, false are shown as nodes T, F, the edges represent a transition for the given variable. Column $\mathsf{R}^{\triangleright}$ shows the public (projected) transition system, $\mathsf{R}, \mathsf{RR} \in \mathcal{O}^{\triangleright}_{\mathsf{R}}$ are the actions with equal public projections and their individual transition systems. Column $\mathcal{O}^{\triangleright}_{\mathsf{R}}$ shows the combined transition system of the projected action.

Example. According to Definition 1, the example action $\mathsf{R}^{\triangleright}$ shown in Figure 2 is ia, pi, pn in s. WLOG we can assume that $s_I[\mathsf{s}] = \mathsf{true}$, thus $\mathsf{R}^{\triangleright}$ is ia. Moreover, $\mathsf{R}^{\triangleright}$ is pi in s because $\mathsf{pre}(\mathsf{R})[\mathsf{s}] = \mathsf{true}$ and $\mathsf{pre}(\mathsf{RR})[\mathsf{s}] = \mathsf{true}$, and $\mathsf{R}^{\triangleright}$ is pn in s because $\mathsf{eff}(\mathsf{R})[\mathsf{s}] \neq \mathsf{eff}(\mathsf{RR})[\mathsf{s}]$.

Leakage of Search-Based Algorithms

MAFS is an adaptation of the classical heuristic forward-search to the multi-agent setting. Each agent $\alpha_i \in \mathcal{A}$ expands its search space defined over the variables in $\mathcal{V}_i$. The agent α_i expands a state s using all applicable actions in $\mathcal{O}_i$. If s is expanded using $a \in \mathcal{O}^{\mathsf{priv}_i}$, the resulting state

$s' = a \circ s$ is added only to the open list of agent α_i as it has changed only in private variables $\mathcal{V}^{\mathsf{priv}_i}$. If s is expanded using $a \in \mathcal{O}^{\mathsf{pub}_i}$, the resulting state $s' = a \circ s$ is sent to (all) other agents, as it may influence their search. In order to keep information private, only the public projection $s'^{\triangleright}$ is sent to other agents. Nevertheless, to be able to reconstruct the private parts of received states, each agent α_i must include some reference to the i-parent state, defined as follows.

Definition 2. Let s, s', s'' be states defined over $\mathcal{V}_i$ in the search space of agent $\alpha_i \in \mathcal{A}$ such that $s' = a_0 \circ s$ where $a_0 \in \mathcal{O}^{\mathsf{pub}_i}$ and s'' is the result of application of a sequence $(a_1, ..., a_k)$ on s', where for all $a_2, ..., a_{k-1} \notin \mathcal{O}_i$. We say, that s' is the **i-parent** of s''.

Informally, the i-parent of s'' is the last state s' expanded by α_i on the path leading to s''. In order to reconstruct the private part $s''[\mathcal{V}^{\mathsf{priv}_i}]$ of the state s'', the agent α_i needs to know the i-parent state s' (or, at least, its private part). Let S^i be the state space of agent α_i, we define $\delta_i : S^i \to \mathbb{N}$ as the function assigning an agent-specific identifier (id) to each state $s \in S^i$. The $\delta_i(s)$ for each agent is sent as part of each state message $m = \langle s^{\triangleright}, \delta_1(s), ..., \delta_n(s) \rangle$ where $s^{\triangleright}$ is a public projection of the sent or received state and $\delta_1, ..., \delta_n$ are the id functions representing $s[\mathcal{V}^{\mathsf{priv}_i}]$ for each $i \in 1, ..., n$. In order to be complete, it must hold that $\delta_i(s) \neq \delta_i(s')$ if $s[\mathcal{V}^{\mathsf{priv}_i}] \neq s'[\mathcal{V}^{\mathsf{priv}_i}]$ for each agent, otherwise the agent would not be able to reconstruct the private part correctly.

The message m might also contain the global heuristic value h or projected heuristic value $h^{\triangleright}$ computed by the sending agent and the g value determining the cost for getting from s_I to s. The projected heuristic $h^{\triangleright}$ is computed by each agent α_i on Π_i separately, thus the computation itself leaks no information and the resulting value depends only on the problem of agent α_i. In contrast, the global heuristic is computed in a distributed way with all agents participating. We assume that the global heuristic is privacy-preserving, such as MA-Pot (Štolba, Fišer, and Komenda 2016), and therefore the computation itself leaks no private information as well.

The adversary agent cannot observe the global search tree as it includes private actions of the agent and when receiving a state, the adversary has no information about the action applied by the agent. The adversary can easily encode the id of the i-parent state in the communication by setting $\delta_i(s)$ to a unique id of the state s. When the adversary receives $m = \langle s^{\triangleright}, \delta_1(s), ..., \delta_n(s) \rangle$ the information in $\delta_i(s)$ encodes exactly the i-parent.

Another crucial information the adversary needs to determine is whether two states s, s' such that $s^{\triangleright} = s'^{\triangleright}$ received from the agent are in fact two different states.

Definition 3. Let s, s' be two states. We say that s, s' are **publicly equivalent but different** (PEBD) if $s^{\triangleright} = s'^{\triangleright}$ and $s \neq s'$. We use $s \triangleq s'$ to denote PEBD states s, s'.

As proposed by STK, the PEBD states can be determined based on g and h values and the sets of successor states. Let α_i be the agent and all other agents the colluding adversaries.

Proposition 4. *(STK, generalized for $|\mathcal{A}| > 2$) Let s, s' be states s.t. $s^{\triangleright} = s'^{\triangleright}$. Let h, g be the heuristic function and g the cost function. If $s[V^{\mathsf{priv}_i}] = s[V^{\mathsf{priv}_j}]$ for all $j \neq i$ and $h(s) \neq h(s') \vee g(s) \neq g(s')$ then $s \triangleq s'$.*

We introduce a new proposition which is applicable when the agents use a projected heuristic computation.

Proposition 5. *Let s, s' be states s.t. $s^{\triangleright} = s'^{\triangleright}$. Let $h^{\triangleright}$ be a projected heuristic function computed by the agent α_i. If $h^{\triangleright}(s) \neq h^{\triangleright}(s')$ then $s \triangleq s'$.*

Proof. Trivial as $h^{\triangleright}$ depends only on Π_i and thus if $h^{\triangleright}(s) \neq h^{\triangleright}(s')$ and $s[\mathcal{V}^{\mathsf{pub}}] = s'[\mathcal{V}^{\mathsf{pub}}]$ it must hold that $s[\mathcal{V}_i] \neq s'[\mathcal{V}_i]$. □

Finally, for each edge in the reconstructed search tree, the adversary needs to determine the actions possibly responsible for the edge. Let $s^{\triangleright}, s'^{\triangleright}$ be two states such that $s^{\triangleright}$ is the i-parent of $s'^{\triangleright}$. If $\mathsf{pre}(a^{\triangleright})[V] = s^{\triangleright}[V]$ for all $V \in \mathsf{vars}(\mathsf{pre}(a^{\triangleright}))$ and $\mathsf{eff}(a^{\triangleright})[V] = s'^{\triangleright}[V]$ for all $V \in \mathsf{vars}(\mathsf{eff}(a^{\triangleright}))$ then $a^{\triangleright}$ is possibly responsible for the edge from $s^{\triangleright}$ to $s'^{\triangleright}$. Multiple actions might be possibly responsible for a single edge and a single action might be possibly responsible for multiple edges but for each state $s^{\triangleright}$, there is only a single edge ending at $s^{\triangleright}$ because the i-parent is determined unambiguously. Based on all the above information, the colluding adversary agent builds the reconstructed search tree structure. Let α_i be the agent, the reconstructed search tree is formally defined as follows.

Definition 6. The search tree is defined as a tuple $\mathsf{ST} = \langle S_{\mathsf{rec}}, S_{\mathsf{sen}}, S_I, i\text{-par}, \mathsf{ap\text{-}res}, E_{\mathsf{rec}}, E_{\mathsf{sen}}, h, g \rangle$ where

- $S_{\mathsf{rec}}, S_{\mathsf{sen}}$ is the set of public states $s^{\triangleright}$ received from the agent α_i and sent to the agent α_i respectively.
- S_I is the set of public states $s^{\triangleright}$ such that s is reachable from s_I without using actions of the agent α_i.
- $i\text{-par} : S_{\mathsf{rec}} \to S_{\mathsf{sen}} \cup \{s_I^{\triangleright}\}$ assigns to each received state $s^{\triangleright}$ its i-parent $s'^{\triangleright}$.
- $\mathsf{ap\text{-}res} : \mathcal{O}^{\triangleright} \to 2^{S_{\mathsf{rec}}}$ assigns to each action $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ the set of received states possibly resulting from the application of $a^{\triangleright}$, i.e., $a^{\triangleright}$ is possibly responsible for the edge ending at each such state.
- $E_{\mathsf{rec}} \subseteq S_{\mathsf{rec}} \times S_{\mathsf{rec}}$ and $E_{\mathsf{sen}} \subseteq S_{\mathsf{sen}} \times S_{\mathsf{sen}}$ are partial PEBD relations on received and sent states respectively, based on I_{post} and the above propositions.
- h, g are the heuristic and cost functions for each state. Alternatively $h^{\triangleright}$ represents the projected heuristic function received from the agent.

Note that the search tree can be reliably reconstructed only up to the last state for which all successors were received. Based on the reconstructed search tree and Definition 1 the adversary determines the properties of actions. A property (or a combination of properties) might be associated with a set of actions, meaning that the set of properties applies on at least one of the actions but it is not possible to determine which one. To associate the actions with their possible properties, the adversary constructs a function

$$\mathsf{prop} : 2^{\mathcal{O}^{\triangleright}} \to 2^{\{\mathsf{ia},\mathsf{nia},\mathsf{pi},\mathsf{pd},\mathsf{pn}\}} \tag{2}$$

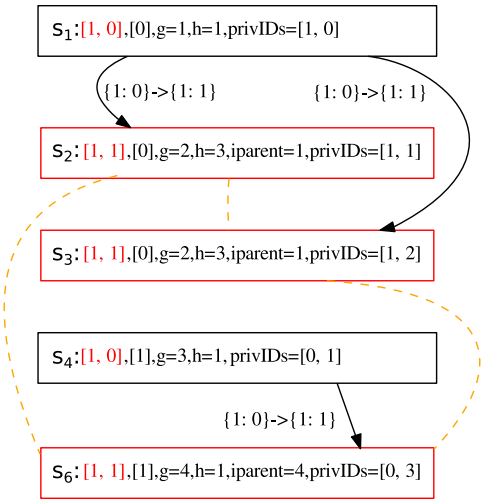


Figure 3: Relevant excerpt of a search tree for the running example as reconstructed by α_{UAV}. Variables are encoded as $[\mathbf{c}, \mathbf{f}]$, [s] where true and false is encoded as 0 and 1 respectively. The privIDs field represents private part identifiers $[\delta_{\alpha_{\mathsf{base}}}(s), \delta_{\alpha_{\mathsf{UAV}}}(s)]$ for the respective agents.

Example. An example of (partial) search tree is shown in Figure 3. Received states $S_{\mathsf{rec}} = \{s_2, s_3, s_6\}$ are red, the i-par function is represented by a solid arrow from i-par(s) to s. The partial PEBD relation $E_{\mathsf{rec}} = \{(s_2, s_3), (s_2, s_6), (s_3, s_6)\}$ is represented by dashed lines, all the PEBD states are identified based on the private identifiers $\delta_{\alpha_{\mathsf{UAV}}}$. Similarly, $(s_1, s_4) \in E_{\mathsf{sen}}$ because $\delta_{\alpha_{\mathsf{UAV}}}(s_1) \neq \delta_{\alpha_{\mathsf{UAV}}}(s_4)$. Based on the change in the public variables, the ap-res is constructed as ap-res$(\mathsf{SL}) = \{s_2, s_3, s_6\}$ and ap-res$(\mathsf{C}) = \emptyset$.

The properties of actions

Let us now focus on how to determine which properties defined in Definition 1 apply on which actions based on the reconstructed search tree from Definition 6.

Init-applicable Let $a^{\triangleright} \in \mathcal{O}^{\triangleright}$such that there exists $s \in$ ap-res$(a^{\triangleright})$ for which i-par$(s^{\triangleright}) = s_I^{\triangleright}$, or i-par$(s^{\triangleright}) = s'^{\triangleright}$, such that $s'^{\triangleright} = \pi \circ s_I^{\triangleright}$ where π is a sequence of actions from $\mathcal{O}^{\mathsf{priv}_i}$. Then $a^{\triangleright}$ is init-applicable, i.e., prop$(\{a^{\triangleright}\}) = \{\mathsf{ia}\}$. Conversely, if there is no such $s^{\triangleright} \in$ ap-res$(a^{\triangleright})$, $a^{\triangleright}$ is not-init-applicable.

Privately-independent Among all edges in ST a particular action is responsible for, we need to find those where the edge originates in two PEBD states. We use the following algorithm:

1. Let $\mathcal{O}_s^{\triangleright} = \emptyset$ for each $s^{\triangleright} \in S_{\mathsf{rec}}$ denote the subset of actions responsible for the edge from i-par$(s^{\triangleright})$ to $s^{\triangleright}$ identified to be privately-independent.
2. For each $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ and for each $s_1^{\triangleright}, s_2^{\triangleright} \in$ ap-res$(a^{\triangleright})$ such that $s_1^{\triangleright} \neq s_2^{\triangleright}$
   (a) $s_1'^{\triangleright} = i$-par$(s_1^{\triangleright}), s_2'^{\triangleright} = i$-par$(s_2^{\triangleright})$
   (b) If $(s_1'^{\triangleright}, s_2'^{\triangleright}) \in E_{\mathsf{sen}}$, add the action $a^{\triangleright}$ to both $\mathcal{O}_{s_1}^{\triangleright}, \mathcal{O}_{s_2}^{\triangleright}$.
3. For each $s^{\triangleright} \in S_{\mathsf{rec}}$, at least one action $a^{\triangleright} \in \mathcal{O}_s^{\triangleright}$ is privately-independent, i.e., prop$(\{\mathcal{O}_s^{\triangleright}\}) = \{\mathsf{pi}\}$.

Privately-dependent For each edge from $s^{\triangleright}$ to $s'^{\triangleright}$ in ST a particular action $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ is responsible for, we need to determine whether $a^{\triangleright}$ is responsible also for all transitions from states $s''^{\triangleright}$ such that $(s^{\triangleright}, s''^{\triangleright}) \in E_{\mathsf{sen}}$. We use the following algorithm:

1. Let $\mathcal{O}_s^{\triangleright} = \emptyset$ for each $s^{\triangleright} \in S_{\mathsf{rec}}$ denote the subset of actions responsible for the edge from i-par$(s^{\triangleright})$ to $s^{\triangleright}$ identified to be privately-dependent.
2. For each $a^{\triangleright} \in \mathcal{O}^{\triangleright}$, for each $s_1^{\triangleright} \in$ ap-res$(a^{\triangleright})$ and for each PBED state $s_2^{\triangleright}$ s.t. $(i$-par$(s_1^{\triangleright}), s_2^{\triangleright}) \in E_{\mathsf{sen}}$
   (a) If there is no $s_3^{\triangleright} \in$ ap-res$(a^{\triangleright})$ such that i-par$(s_3^{\triangleright}) = s_2^{\triangleright}$, add action $a^{\triangleright}$ to $\mathcal{O}_{s_1}^{\triangleright}$.
3. For each $s^{\triangleright} \in S_{\mathsf{rec}}$, at least one action of $a^{\triangleright} \in \mathcal{O}_s^{\triangleright}$ is privately-dependent, i.e., prop$(\{\mathcal{O}_s^{\triangleright}\}) = \{\mathsf{pd}\}$.

Privately-nondeterministic Among all edges in ST a particular action is responsible for, we need to find those where the edges originate in the same state but results in two PEBD states. We use the following algorithm:

1. Let $\mathcal{O}_s^{\triangleright} = \emptyset$ for each $s^{\triangleright} \in S_{\mathsf{sen}}$ denote the subset of actions responsible for the edge from i-par$(s^{\triangleright})$ to $s^{\triangleright}$ identified to be privately-nondeterministic.
2. For each $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ and for each $s_1^{\triangleright}, s_2^{\triangleright} \in$ ap-res$(a^{\triangleright})$ such that $(s_1^{\triangleright}, s_2^{\triangleright}) \in E_{\mathsf{rec}}$ and $s'^{\triangleright} = i$-par$(s_1^{\triangleright}) = i$-par$(s_2^{\triangleright})$,
   (a) add action $a^{\triangleright}$ to $\mathcal{O}_{s'}^{\triangleright}$.
3. For each $s^{\triangleright} \in S_{\mathsf{sen}}$, at least one action of $a^{\triangleright} \in \mathcal{O}_s^{\triangleright}$ is privately-nondeterministic, i.e., prop$(\{\mathcal{O}_s^{\triangleright}\}) = \{\mathsf{pn}\}$.

Example. The above algorithms can be used to construct prop based on the search tree in Figure 3. Because i-par$(s_2) = s_1$, s_1 is a direct successor of s_I (not shown in the figure), and $s_2 \in$ ap-res$(\mathsf{SL}^{\triangleright})$, $\mathsf{SL}^{\triangleright}$ is init-applicable, whereas $\mathsf{C}^{\triangleright}$ is not-inint-applicable. Because ap-res$(\mathsf{SL}) = \{s_2, s_3, s_6\}$, i-par$(s_2) = s_1$ and i-par$(s_6) = s_4$, and $(s_1, s_4) \in E_{\mathsf{sen}}$, $\mathsf{SL}^{\triangleright}$ is privately-independent. Finally, because $(s_2, s_3) \in E_{\mathsf{rec}}$ and i-par$(s_2) = i$-par(s_3), $\mathsf{SL}^{\triangleright}$ is privately non-deterministic. Thus we can construct prop$(\{\mathsf{SL}^{\triangleright}\}) = \{\mathsf{ia}, \mathsf{pi}, \mathsf{pn}\}$ and prop$(\{\mathsf{C}^{\triangleright}\}) = \{\mathsf{nia}\}$.

Computing the Privacy Leakage

Once the properties of actions are determined and the prop function is constructed, we can proceed to compute the actual privacy leakage. To do so, we first need to compute the number of possible transition systems an action represents given its properties. Then we need to combine the information about each action to compute the total number of possible transition systems based on the Equation 1 and finally to compute the actual privacy leakage.

Transition Systems of an Action

To account for all possible combinations of the properties for arbitrary values of p and d, we introduce a novel general approach to compute the number of transition systems represented by a given set $X \in 2^{\{\mathsf{ia},\mathsf{nia},\mathsf{pi},\mathsf{pd},\mathsf{pn}\}}$ of properties. We construct a first-order logic formula for each X to describe the properties of actions related to privacy leakage. For each $V \in \mathcal{V}^{\mathsf{priv}}$, and each $v \in \mathsf{dom}(V)$, we define first-order logic propositions $\mathsf{pre}_{V=v}$ and $\mathsf{eff}_{V=v}$. The formulas are independent of particular actions and depend only on the set X of properties. The semantics of $\mathsf{pre}_{V=v} = \mathsf{true}$ is that for some $a \in \mathcal{O}_a^{\triangleright}$, $\mathsf{pre}(a)[V] = v$ and of $\mathsf{pre}_{V=v} = \mathsf{false}$ is that for all $a \in \mathcal{O}_a^{\triangleright}$, $\mathsf{pre}(a)[V] \neq v$, analogously $\mathsf{eff}_{V=v}$ for effects, where $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ is any action which has the properties X.

Let $a^{\triangleright} \in \mathcal{O}^{\triangleright}$, $V \in \mathcal{V}^{\mathsf{priv}}$, and $v_0 \in \mathsf{dom}(V)$ is some fixed value of V, then

$$\mathsf{ia} = \bigwedge_{V \in \mathcal{V}^{\mathsf{priv}}} \mathsf{pre}_{V=v_0}$$

$$\mathsf{nia} = \bigvee_{V \in \mathcal{V}^{\mathsf{priv}}} \neg\mathsf{pre}_{V=v_0}$$

$$\mathsf{pi} = \bigvee_{V \in \mathcal{V}^{\mathsf{priv}}} \Big(\bigwedge_{v \in \mathsf{dom}(V)} \mathsf{pre}_{V=v} \Big)$$

$$\mathsf{pd} = \bigvee_{V \in \mathcal{V}^{\mathsf{priv}}} \Big(\bigvee_{v \in \mathsf{dom}(V)} \mathsf{pre}_{V=v} \wedge \bigvee_{v \in \mathsf{dom}(V)} \neg\mathsf{pre}_{V=v} \Big)$$

$$\mathsf{pn} = \bigvee_{V \in \mathcal{V}^{\mathsf{priv}}} \Big(\bigvee_{v \neq v' \in \mathsf{dom}(V)} (\mathsf{eff}_{V=v} \wedge \mathsf{eff}_{V=v'}) \Big)$$

Combinations of properties can be obtained by conjunction.

Example. For the example action $\mathsf{SL}^{\triangleright}$, for which $\mathsf{prop}(\{\mathsf{SL}^{\triangleright}\}) = \{\mathsf{ia}, \mathsf{pi}, \mathsf{pn}\}$, the formula is $\mathsf{ia} \wedge \mathsf{pi} \wedge \mathsf{pn}$. There are two private binary variables $\mathsf{l1}, \mathsf{l2}$ (their identities are not known to the adversary but we use them here for clarity). The complete formula is

$$\begin{aligned} &(\mathsf{pre}_{\mathsf{l1}=\mathsf{true}} \wedge \mathsf{pre}_{\mathsf{l2}=\mathsf{true}}) \wedge \\ &((\mathsf{pre}_{\mathsf{l1}=\mathsf{true}} \wedge \mathsf{pre}_{\mathsf{l1}=\mathsf{false}}) \vee (\mathsf{pre}_{\mathsf{l2}=\mathsf{true}} \wedge \mathsf{pre}_{\mathsf{l2}=\mathsf{false}})) \wedge \\ &((\mathsf{eff}_{\mathsf{l1}=\mathsf{true}} \wedge \mathsf{eff}_{\mathsf{l1}=\mathsf{false}}) \vee (\mathsf{eff}_{\mathsf{l2}=\mathsf{true}} \wedge \mathsf{eff}_{\mathsf{l2}=\mathsf{false}})) \end{aligned} \quad (3)$$

The next step is to compute the number of possible transition systems $\tau(X)$ w.r.t. the set of properties X. Let $\bar{\lambda}_X = \bigwedge_{x \in X} x$ denote the conjunction of the formulas respective to the properties in X. Let M be the set of all models of the formula $\bar{\lambda}_X$. For each $V \in \mathcal{V}^{\mathsf{priv}}$ and $v \in V$, the model $m \in M$ either assigns $\mathsf{pre}_{V=v} = \mathsf{true}$, $\mathsf{pre}_{V=v} = \mathsf{false}$, or $\mathsf{pre}_{V=v}$ is not in $\bar{\lambda}_X$ and therefore m assigns no value to $\mathsf{pre}_{V=v}$. The same holds for $\mathsf{eff}_{V=v}$. For each variable $V \in \mathcal{V}^{\mathsf{priv}}$, we define the set $P_V^+ = \{\mathsf{pre}_{V=v} | v \in \mathsf{dom}(V), \mathsf{pre}_{V=v} = \mathsf{true}\}$ of positive propositions and P_V^{free} the set of unassigned propositions (analogously E_V^+ , and E_V^{free} for effects). Let $p^{\mathsf{free}} = \sum_{V \in \mathcal{V}^{\mathsf{priv}}} |P_V^{\mathsf{free}}|, p^+ = \sum_{V \in \mathcal{V}^{\mathsf{priv}}} |P_V^+|$ and e^{free}, e^+ analogously.

Example. Let us consider the formula in Equation 3. One model of the formula assigns true to $\mathsf{pre}_{\mathsf{l1}=\mathsf{true}}$, $\mathsf{pre}_{\mathsf{l2}=\mathsf{true}}$, $\mathsf{pre}_{\mathsf{l1}=\mathsf{false}}$, $\mathsf{eff}_{\mathsf{l1}=\mathsf{true}}$, $\mathsf{eff}_{\mathsf{l1}=\mathsf{false}}$, $\mathsf{eff}_{\mathsf{l2}=\mathsf{false}}$ and false to $\mathsf{pre}_{\mathsf{l2}=\mathsf{false}}$, $\mathsf{eff}_{\mathsf{l2}=\mathsf{true}}$. Based on the semantics, this model describes a transition system where there are all transitions $\mathsf{SL}^{\triangleright}$ for $\mathsf{l1}$, but only $\mathsf{false} \rightarrow \mathsf{true}$ for $\mathsf{l2}$ as shown in Figure 4. There are no unassigned propositions and thus $p^{\mathsf{free}} = e^{\mathsf{free}} = 0$, $p^+ = 3$, and $e^+ = 3$.

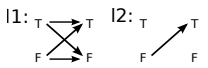


Figure 4: Transition system of the $\mathsf{SL}^{\triangleright}$ action based on one possible model of formula in Equation 3.

If $p^+ = 0$ and $e^+ = 0$ then $t^{\emptyset} = (2^{d^2} - 1)^p$ as in STK. Otherwise, we need to evaluate all possible combinations of free propositions. To do so, for each $k' \leq p^{\mathsf{free}}$, $l' \leq e^{\mathsf{free}}$ let us fix a subset of k' precondition propositions and a subset of l' effect propositions to be true, and all other free propositions to be false.

This way we arrive at a fixed number of true preconditions and true effects, in particular $k = p^+ + k'$ and $l = e^+ + l'$. The number of possible transition systems for such configuration is equal to the number of bipartite graphs with partitions of size k and l such that each of the nodes has a degree ≥ 1 (all nodes are connected). There is $(2^k - 1)^l$ of all bipartite graphs and $(2^{k-j} - 1)^l$ bipartite graphs which do not use j particular fixed nodes. For all values of j, we use this number in the inclusion-exclusion principle and arrive at

$$ts(k, l) = \sum_{j=0}^{k} (-1)^k \binom{k}{j} (2^{k-j} - 1)^l$$

Moreover, we care not only about the number of true preconditions and effects, but also about their identity, therefore we need to consider all possible choices of k' and l' propositions. The final number of transition systems for action for which the set X of properties holds is

$$\tau(X) = \sum_{k'=0}^{k'=p^{\mathsf{free}}} \sum_{l'=0}^{l'=e^{\mathsf{free}}} \binom{p^{\mathsf{free}}}{k'} \binom{e^{\mathsf{free}}}{l'} ts(p^+ + k', e^+ + l')$$

In order to exclude the empty transition system, if $p^+ = 0$ then k' starts from 1 instead of 0 and if $e^+ = 0$ then l' starts from 1 instead of 0.

Example. For the example action $\mathsf{SL}^{\triangleright}$, we need to compute $t^{\mathsf{ia},\mathsf{pi},\mathsf{pn}}$, since there are no free propositions, $t^{\mathsf{ia},\mathsf{pi},\mathsf{pn}} = ts(p^+, e^+) = ts(3, 3) = 49$ possible transition systems. Note that this value holds for any $\mathsf{ia}, \mathsf{pi}, \mathsf{pn}$ action which is in a problem with two binary private variables.

Total Number of Transition Systems

For each $a^{\triangleright} \in \mathcal{O}^{\triangleright}$, let the number of transition systems represented by $a^{\triangleright}$ before the algorithm is executed be denoted by $\tau_{\mathsf{apriori}}(a^{\triangleright})$ and let the number of transition systems represented by $a^{\triangleright}$ after the algorithm is terminated be denoted by $\tau_{\mathsf{post}}(a^{\triangleright})$. In this section, we focus on the computation of $\tau_{\mathsf{post}}(a^{\triangleright})$, the computation of $\tau_{\mathsf{apriori}}(a^{\triangleright})$ is analogous.

Based on the search tree reconstruction, let X be a set of properties s.t. $\mathsf{prop}(\mathcal{O}^X)$ for some $\mathcal{O}^X \subseteq \mathcal{O}^{\triangleright}$. We know

that for at least one $a^{\triangleright} \in \mathcal{O}^X$, X holds, but we do not know which one it is. Consider an action $a^{\triangleright}$ such that $a^{\triangleright} \in \mathcal{O}^X$ and $a^{\triangleright} \in \mathcal{O}^{X'}$ where $X \neq X'$, how do we best (optimally) determine the number of transition systems $\tau_{\mathsf{post}}(a^{\triangleright})$ in the presence of such information?

We formulate the problem as a combinatorial optimization problem and solve it using LP. We define an LP variable $\bar{a}$ for each action $a^{\triangleright} \in \mathcal{O}^{\triangleright}$. The variable $\bar{a}$ represents $\bar{a} = \log \tau_{\mathsf{post}}(a^{\triangleright})$. Logarithm can be applied to both sides of an inequality as we use logarithm with base > 1. This way, we can use a sum variant of Equation 1, $\sum_{a^{\triangleright} \in \mathcal{O}^{\triangleright}} \tau_{\mathsf{post}}(a^{\triangleright})$. Each LP variable $\bar{a}$ is bounded by $1 \leq \bar{a} \leq \log t^{\emptyset}$. We formulate the following objective function to be maximized:

$$\mathsf{Maximize} \sum_{a^{\triangleright} \in \mathcal{O}^{\triangleright}} \bar{a}$$

resulting in the logarithm of Equation 1. For each $\mathcal{O}^X \subseteq \mathcal{O}^{\triangleright}$ we formulate a disjunctive constraint:

$$\bigvee_{a^{\triangleright} \in \mathcal{O}^{X'}} (\bar{a} \leq \log \tau(X))$$

For example if there is a set of actions $\{a_1^{\triangleright}, a_2^{\triangleright}, a_3^{\triangleright}\}$ and we know that at least one of the actions is privately-independent the disjunctive constraint is:

$$\bar{a}_1 \leq \log t^{\{\mathsf{pi}\}} \vee \bar{a}_2 \leq \log t^{\{\mathsf{pi}\}} \vee \bar{a}_3 \leq \log t^{\{\mathsf{pi}\}}$$

Such a disjunctive formulation can either be directly solved by a branch-and-bound algorithm or transformed to mixed-integer linear program (MILP) by using the Big-M reformulation. In order to do so, for each constraint in the disjunction we define a binary variable, e.g., $y_1, y_2, y_3 \in \{0, 1\}$, and enforce that only one of them holds true by

$$y_1 + y_2 + y_3 = 1$$

Then we define a large enough constant M and reformulate the above disjunction as

$$\begin{aligned}
\bar{a}_1 &\leq \log t^{\{\mathsf{pi}\}} + M(1 - y_1) \\
\bar{a}_2 &\leq \log t^{\{\mathsf{pi}\}} + M(1 - y_2) \\
\bar{a}_3 &\leq \log t^{\{\mathsf{pi}\}} + M(1 - y_3)
\end{aligned}$$

which significantly expands all but one of the bounds, thus rendering them ineffective. Such transformation then allows to use a standard solver with MILP capabilities, such as the IBM CPLEX.

Example. Formulating the LP for the example problem is straightforward, especially because $|\mathcal{O}^X| = 1$ for all transitions and thus we do not need to use disjunctive constraints. The objective value is 12.98 which represents the logarithm of the number of possible transition systems given the received information, that is, $\log \tau(I_{\mathsf{post}}) = 12.98$. By considering only information learned from the final plan, we arrive at $\log \tau(I_{\mathsf{apriori}}) = 14.98$ and thus the total private information leakage of the example problem is $\log t_{\mathsf{apriori}} - \log t_{\mathsf{post}} = 2$ bits.

Handling Complexity

Although general enough, the described computation of leakage is not tractable for practical use on realistically large problems, such as the CoDMAP benchmarks. The most demanding operations are the finding of PEBD states and determining the number of transition systems $\tau(X)$ based on the set X of properties.

The PEBD states computation is $O(|S_{\mathsf{rec}}|^2)$, i.e., quadratic in the number of received states, as we need to test each state against each other state. Moreover, the number of received states can be up to the number of states expanded by the agent α, as in the case where all actions are public, every expanded state is also sent to all other agents.

The computation of $\tau(X)$ is even worse as we need to find all models of the formula $\bar{\lambda}_X$. Let $\mathsf{vars}(\bar{\lambda}_X)$ denote the number of propositional variables in $\bar{\lambda}_X$, then finding all valuations of $\bar{\lambda}_X$ is $O(2^{\mathsf{vars}(\bar{\lambda}_X)})$ and $\mathsf{vars}(\bar{\lambda}_X)$ can be as large as $p{\cdot}d$, therefore the complexity is $O(2^{pd})$. That said, the values are problem-independent and thus can be pre-computed just once and stored in a table. Still, the complexity is forbidding for larger values of p and d.

In order to be able to run the analysis on the actual CoDMAP domains, we need to simplify the privacy leakage estimate computation. First, notice that the full set E_{rec} of PEBD states over the received states is used only in the computation of privately-dependent actions. Therefore, we remove the computation of privately-dependent actions from the analysis and thus we can skip the pre-computation of all PEBD states. Instead, we test the states on PEBD equivalence only when needed.

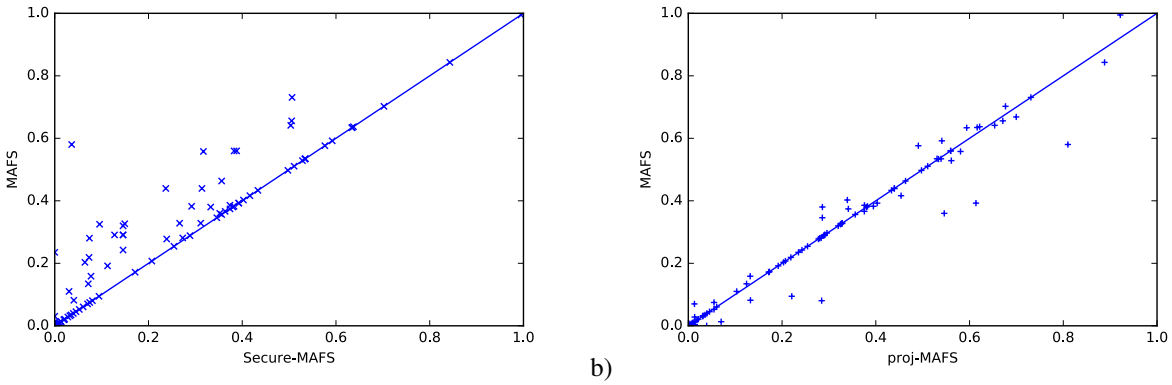
The computation of $\tau(X)$ is a bigger challenge. We use the pre-computed values for small values of p and d which are stored in the form of $\log \tau(X)$. Notice, that for $\tau(\emptyset) = (2^d - 1)^p$ the value $\log \tau(\emptyset) = p(2^d - 1)$ can be easily computed for any (positive) p, d. For a simple linear extrapolation, we define a coefficient k_X for each set X of properties as an average value of

$$\frac{\log \tau(X)}{p(2^d - 1)}$$

over all values of p and d for which we were able to compute $\log \tau(X)$ exactly. For higher values of p, d we extrapolate as $\log \tau(X) = k_X \log \tau(\emptyset) = k_X p(2^d - 1)$. Clearly, using an average is a very simple technique for approximation, nevertheless, the standard deviation is on average (over all X) only 10% and as the leakage is computed as a sum of $\log \tau(X)$, the overall standard deviation can only decrease. We assume such approximation to be reasonably precise and leave better approximations for future work.

Evaluation

We demonstrate the ability of the proposed leakage quantification metric to be used to compare privacy-preserving planners. In order to compare various planners on various domains we do not compare the absolute value of the leakage (that is, the leaked bits of information), but a ratio of the leakage of the particular algorithm to the ground truth leakage. The ground truth leakage is computed the same way



a) b)

Figure 5: Detailed leakage ratio comparison, MAFS vs. Secure-MAFS (a) and MAFS vs. MAFS with projected heuristic (b).

as the leakage of the algorithm, but with action properties derived directly from the private information of the agent. Similarly, the upper bounds on $p = |\mathcal{V}^{\mathsf{priv}}|$ and d are computed directly from the private problem description, where $d = \max_{V \in \mathcal{V}^{\mathsf{priv}}} |\mathsf{dom}(V)|$.

Implementation

We have modified the MAPlan planner to a) print out sent and received states, b) enable the ϵ-MAFS search. The leakage analysis tool[1] is implemented in Python using SymPy and PuLP[2]. The upper bounds on $p = |\mathcal{V}^{\mathsf{priv}}|$ and d necessary for the computation are computed directly from the private problem description, where $d = \max_{V \in \mathcal{V}^{\mathsf{priv}}} |\mathsf{dom}(V)|$. We have used the pre-computed values of $\log t^X$ for small values of p, d and simplified extrapolations for large values, as described in the previous section.

We have evaluated the following planner configurations:

MAFS Plain MAFS as implemented in MAPlan, using A* search and MA-LM-Cut distributed heuristic (Štolba, Fišer, and Komenda 2015) **(dist)** and projected LM-Cut heuristic (Helmert and Domshlak 2009) **(proj)**.

Secure-MAFS As Secure-MAFS is not implemented in MAPlan, we have simulated its execution post-hoc by removing the states which would not be sent. We evaluate variants with the distributed **(dist)** and projected **(proj)** LM-Cut heuristic.

We have set the planning memory limit to 512 MB and time limit to 15 minutes. Note that it is not necessary for the planner to find a plan in order to perform the leakage evaluation (only the final plan $\pi^{\triangleright}$ cannot be used in that case). As the leakage evaluation runtime and memory consumption is relatively high w.r.t. the number of received states such limits are reasonable.

Benchmarks

The evaluation is based on the planning domains used in the CoDMAP (Komenda, Stolba, and Kovacs 2016) competition. Moreover, we have implemented a domain named UAVs which was used by STK as a running example. The simplest problem is shown in Figure 1. The planning task is to visit the locations while refueling at a base where the UAV operator and base are agents and the private information is the visited location and state of supplies respectively. We have constructed 5 problems with an increasing number of locations and UAVs.

[1] http://github.com/stolba/privacy-analysis

[2] http://www.sympy.org; http://pythonhosted.org/PuLP/

Comparison of the Search Algorithms

search	MAFS		Secure-MAFS	
heuristic	dist.	proj.	dist.	proj.
blocksworld	*5.18*	**4.85**	*5.18*	**4.85**
depot	**0.08**	*0.13*	**0.08**	*0.13*
driverlog	0.17	0.17	0.17	0.17
elevators	3.37	*3.40*	**3.35**	3.38
logistics	5.82	*6.14*	**5.44**	5.75
rovers	*3.76*	3.72	2.78	**2.76**
satellites	*3.10*	*3.10*	**1.96**	**1.96**
sokoban	**0.05**	*0.06*	**0.05**	*0.06*
woodwork.	0.00	0.00	0.00	0.00
taxi	0.00	0.00	0.00	0.00
wireless	0.19	0.19	0.19	0.19
zenotravel	3.04	*3.16*	**1.87**	2.03
uav	2.92	*3.14*	**2.30**	2.52
sum	27.69	*28.04*	**23.38**	23.78

Table 1: Comparison of search algorithms. Minimal leakage in bold, maximal in italics.

Table 1 shows an evaluation of the leakage ratio, where the leakage ratio is summed over all problems in the domain for each planner configuration. The table clearly shows that the use of Secure-MAFS decreases privacy leakage, which has already been suggested by STK, but here, we present a quantified result. Moreover, the results show that the use of projected heuristic significantly increases privacy leakage in some domains as Proposition 5 can be used to discern PEBD states. For comparison, the Figure 6 shows the absolute values of leaked private information in bits averaged per domain for each planner configuration.

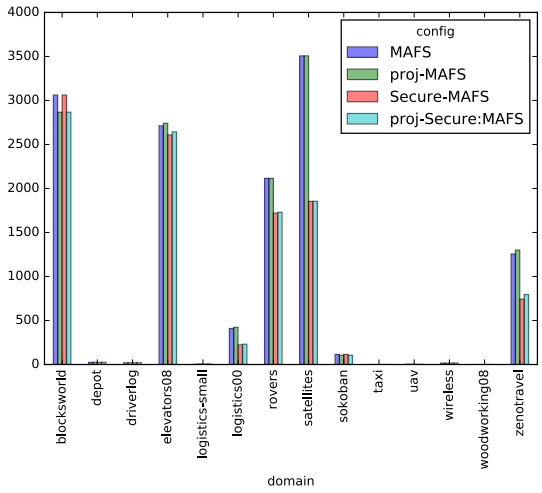


Figure 6: Absolute values of privacy leakage in bits of information averaged per domain.

Figure 5 provides a more detailed comparison of the leakage ratios per problem. Again, in some problems, the proposed privacy leakage metric was able to show that Secure-MAFS (above) leaks less private information and MAFS with projected heuristic (below) leaks more private information.

Conclusion

In this work, we have proposed a generalization of the privacy leakage quantification approach of (Štolba, Tožička, and Komenda 2017) to arbitrary numbers of private variables and sizes of private domains. As the original work was only theoretical, we have proposed a number of algorithms and computational techniques to arrive at the privacy leakage metric for actual MAP planners and domains.

In order to be tractable on real benchmark problems, we have proposed an approximation of the leakage quantification metric. This enabled us to evaluate the MAFS and Secure-MAFS algorithms on the CoDMAP domains using distributed and projected heuristics. In addition to confirming the known fact that Secure-MAFS leaks less information than MAFS, we were also able to show a novel result stating that the MAFS algorithm actually leaks more private information when using the projected heuristic then when using the distributed one (assuming that the distributed computation is secure).

Acknowledgments This research was supported by the Czech Science Foundation (grant no. 18-24965Y). The authors acknowledge the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

References

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, 28–35.

Brafman, R. I. 2015. A privacy preserving algorithm for multi-agent planning and search. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, (IJCAI'15)*, 1530–1536.

Durfee, E. H. 1999. Distributed problem solving and planning. In Weiß, G., ed., *A Modern Approach to Distributed Artificial Intelligence*. San Francisco, CA: The MIT Press. chapter 3.

Fišer, D.; Štolba, M.; and Komenda, A. 2015. MAPlan. In *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP'15)*, 8–10.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, 162–169.

Komenda, A.; Stolba, M.; and Kovacs, D. L. 2016. The international competition of distributed and multiagent planners (CoDMAP). *AI Magazine* 37(3):109–115.

Nissim, R., and Brafman, R. I. 2014. Distributed heuristic forward search for multi-agent planning. *Journal of Artificial Intelligence Research* 51:293–332.

Oliehoek, F. A., and Amato, C. 2016. *A concise introduction to decentralized POMDPs*. Springer.

Smith, G. 2009. On the foundations of quantitative information flow. In *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures*, 288–302.

Štolba, M.; Fišer, D.; and Komenda, A. 2015. Admissible landmark heuristic for multi-agent planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 211–219.

Štolba, M.; Fišer, D.; and Komenda, A. 2016. Potential heuristics for multi-agent planning. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling, ICAPS'16*, 308–316.

Štolba, M.; Komenda, A.; and Kovacs, D. 2016. Competition of distributed and multiagent planners (CoDMAP). In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4343–4345.

Štolba, M.; Tožička, J.; and Komenda, A. 2017. Quantifying privacy leakage in multi-agent planning. *Transactions on Internet Technology (TOIT)*.

Torreño, A.; Onaindia, E.; Komenda, A.; and Štolba, M. 2017. Cooperative multi-agent planning: a survey. *ACM Computing Surveys (CSUR)* 50(6):84.

Tožička, J.; Štolba, M.; and Komenda, A. 2017. The limits of strong privacy preserving multi-agent planning. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*.

Van Der Krogt, R. 2009. Quantifying privacy in multiagent planning. *Multiagent and Grid Systems* 5(4):451–469.